



**FACULTAD DE INGENIERÍA, ARQUITECTURA Y
URBANISMO**

**ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA
DE SISTEMAS**

TESIS

**ANÁLISIS COMPARATIVO DE FRAMEWORKS
OPEN SOURCE PARA EL DESARROLLO DE
APLICACIONES MÓVILES ANDROID BASADAS EN
TECNOLOGÍA WEB**

**PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO
DE SISTEMAS**

Autor:

Bach. Inoñan Coronado Cristian Wilfredo

ORCID:

<https://orcid.org/0000-0002-2250-7174>

Asesor:

Mg. Chavarry Chankay Mariana

ORCID:

<https://orcid.org/0000-0001-5136-7177>

Linea de Investigación:

Infraestructura, Tecnología y Medio Ambiente

Pimentel - Perú 2018

APROBACIÓN DEL JURADO

ANÁLISIS COMPARATIVO DE FRAMEWORKS OPEN SOURCE PARA EL DESARROLLO DE APLICACIONES MÓVILES ANDROID BASADAS EN TECNOLOGÍA WEB

Bach. Inoñan Coronado Cristian Wilfredo
Autor

Mg. Chavarry Chankay Mariana
Asesor

Mg. Maquen Niño Enver Jose Carlos
Presidente de Jurado

Ing. Mejía Cabrera Heber Iván
Secretario de Jurado

Mg. Bravo Ruiz Jaime Arturo
Vocal de Jurado

Dedicatorias

A mi madre Marleny Coronado Ventura por ser mi motor y motivo de salir adelante, inculcándome siempre valores y que con su gran esfuerzo que realizó y realiza para que seamos mejores personas, por ser luchadora y darme esas ganas de superación. A mi tío Lorenzo Coronado Ventura por su apoyo incondicional, quien me motivo a seguir estudiando. A mis hermanos por su apoyo. A mi padre Ceferino Inoñan Santisteban por el apoyo que me brindó. Y a todos los que de alguna u otra manera intervinieron para poder cumplir este objetivo. ¡Gracias Totales!

Agradecimientos

Agradezco primeramente a Dios por darme fuerzas para seguir en la lucha y cumplir este objetivo, por la salud, sabiduría y paciencia.

A mi madre Marleny Coronado Ventura por su lucha y esfuerzo de superación, por ser una madre que permitió y logró llevar a sus hijos por un buen camino.

A mis tíos, mis hermanos y a mi padre que estuvieron a mi lado apoyándome de alguna manera en el proceso de formación personal y profesional.

A la Universidad Señor de Sipan y los distintos docentes que en el transcurso de mi carrera profesional aportaron conocimientos, experiencias y buenas prácticas para lograr ser un buen profesional.

A los amigos que estuvieron pendientes en el transcurso de mi formación profesional y por su apoyo moral incondicional.

Resumen

El presente proyecto realiza un análisis comparativo de frameworks para el desarrollo de aplicaciones móviles utilizando tecnología web, para el análisis comparativo se utiliza un modelo de calidad que provee características para realizar la evaluación del software.

Para seleccionar los frameworks para su futura evaluación, se realizó la búsqueda en diferentes sitios web para obtener una lista de frameworks que permitan desarrollar aplicaciones para dispositivos móviles utilizando tecnología web, se consultó a Google para obtener la cantidad de resultados que este arroja de cada uno de los frameworks encontrados, ordenándolos de mayor a menor, posteriormente se realizó una pre selección de frameworks que se realizó teniendo en cuenta la cantidad de referencias que tiene en distintos sitios web, los cuales obviamente son los más conocidos y/o referenciados por la comunidad, teniendo mejores resultados los frameworks Ionic y jQuery Mobile. Se utiliza el método The Qualification and Selection of Open Source software (QSOS) para realizar un análisis teórico de los frameworks seleccionados con el propósito de definir las características comparativas del software de código abierto.

Se implementó una aplicación de apuestas de fútbol (Golwin), la cual se dividió en dos partes, lado del cliente y lado del servidor, en el lado del servidor de desarrollo una API RESTFul la cual se implementó con Spring MVC, Hibernate y Spring Security Oauth2, utilizando JSON como formato de intercambio de información. Para el lado del cliente se desarrolló una aplicación con cada uno de los frameworks seleccionados, los cuales consumen los servicios que brinda la API RESTFul.

Para la creación del modelo de calidad se utiliza el método IQMC (Modelo de Construcción de Calidad Individual) en conjunto con la norma ISO 25010 para establecer características, subcaracterísticas, atributos derivados y atributos básicos que sean apropiados para realizar la evaluación del software.

Palabras clave: Frameworks, Ionic, jQuery Mobile, IQMC, ISO 25010.

Abstract

The present project performs a comparative analysis of frameworks for the development of mobile applications using web technology, for the comparative analysis a quality model is used that provides characteristics to carry out the evaluation of the software.

In order to select the frameworks for future evaluation, a search was made on different websites to obtain a list of frameworks that allow to develop applications for mobile devices using web technology, Google was consulted to obtain the amount of results that it throws from each one of the found frameworks, ordering them from highest to lowest, there was a pre selection of frameworks that was done taking into account the number of references that has in different websites, which obviously are the most known and/or referenced by the community , With better results the frameworks Ionic and jQuery Mobile. The method The Qualification and Selection of Open Source software (QSOS) is used to perform a theoretical analysis of the selected frameworks with the purpose of defining the comparative characteristics of open source software.

A football bets application (Golwin) was implemented, which was divided into two parts, client side and server side, on the server side of development a API RESTFul which was implemented with Spring MVC, Hibernate and Spring Security Oauth2, using JSON as an information exchange format. For the client side, an application was developed with each of the selected frameworks, which consume the services provided by the API RESTFul.

For the creation of the quality model, the IQMC (Individual Quality Construction Model) method is used in conjunction with the ISO 25010 standard to establish characteristics, subcharacteristics, derived attributes and basic attributes that are appropriate for evaluating the software.

Keywords: Frameworks, Ionic, jQuery Mobile, IQMC, ISO 25010.

ÍNDICE

I. INTRODUCCIÓN	8
1.1. Realidad Problemática.....	8
1.2. Antecedentes de estudio.....	10
1.3. Teorías relacionadas al tema	13
1.4. Formulación del Problema.	23
1.5. Justificación e Importancia del estudio.	23
1.6. Hipótesis.....	24
1.7. Objetivos.....	25
1.7.1. Objetivo general.....	25
1.7.2. Objetivos específicos	25
II. MATERIAL Y MÉTODO	25
2.1. Tipo y Diseño de investigación.....	25
2.2. Población y muestra	25
2.3. Variables, operacionalización.	26
2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.....	28
2.5. Procedimiento análisis de datos.....	29
2.6. Criterios éticos.	31
2.7. Criterios de rigor científico.	31
III. RESULTADOS	32
3.1. Resultados en tablas y gráficos.....	32
3.2. Discusión de resultados.....	51
3.3. Aporte práctico.....	52
IV. CONCLUSIONES Y RECOMENDACIONES	224
4.1. Conclusiones.....	224
4.2. Recomendaciones.....	226
REFERENCIAS.....	227
ANEXOS	230

I. INTRODUCCIÓN

Existen en la actualidad gran cantidad de frameworks que permiten agilizar la creación de aplicaciones para dispositivos móviles con interfaces amigables y atractivas para el usuario, muchos de estos frameworks son open source y otros son de pago. La cantidad de frameworks actuales es muy numerosa, y el desarrollador de aplicaciones móviles observa una amplia gama de opciones, estas opciones son muy diversas. Dentro de esta gran cantidad existen frameworks que permiten crear aplicaciones para dispositivos móviles utilizando tecnología web.

Por ende el presente proyecto tiene como objetivo principal, realizar un análisis comparativo de frameworks open source para el desarrollo de aplicaciones móviles utilizando tecnología web, basándose en un modelo de calidad que permitió establecer características, subcaracterísticas, atributos derivados y atributos básicos para realizar la evaluación de los frameworks, para de esta manera obtener software de calidad, utilizando menos tiempo en el desarrollo, para así mejorar la calidad del software y reducir los costos que implica el desarrollo del mismo.

Este análisis comparativo de frameworks open source para el desarrollo de aplicaciones móviles basados en tecnología web tiene por finalidad permitirles a los desarrolladores de aplicaciones móviles identificar frameworks de calidad, logrando de esta manera tomar una buena decisión al elegir un framework para el proceso de desarrollo, y así lograr obtener un software de calidad en mucho menos tiempo y reduciendo los costos que implican el proceso de desarrollo de la aplicación.

1.1. Realidad Problemática.

Los datos de análisis muestran que los dispositivos móviles representan aproximadamente el 25% de todos los puntos de vista de las páginas web en los EE.UU., frente al 10% de hace dos años. Esto ha dado lugar a que los desarrolladores adapten el contenido web a dispositivos móviles para de esta manera contextualizar aplicaciones web con capacidades móviles como la

geolocalización, acelerómetro, acceder a la cámara, entre otras. (Hale & Hanson, 2015).

Con la popularidad de los dispositivos móviles portátiles como teléfonos inteligentes y tabletas, cada vez más aplicaciones están migrando del escritorio a las plataformas móviles, el soporte de las aplicaciones web para tales plataformas es un requisito crucial para el negocio. Para la parte de la interfaz de usuario es compatible con varios dispositivos, siendo solo una parte para el éxito de aplicaciones móviles que son creadas con tecnología web. (Marenkov, Robal, & Kalja, 2015).

Desarrollar aplicaciones móviles utilizando HTML, CSS y JavaScript proporciona a los desarrolladores construir una aplicación con estas tecnologías, esto permite a los desarrolladores crear aplicaciones ricas y de alto rendimiento que los usuarios móviles esperan, al tiempo que alivia la necesidad de desarrollar y mantener varias aplicaciones nativas autónomas en lenguajes específicos de la plataforma como IOS o Android. (Hale & Hanson, 2015).

En la actualidad los usuarios acceden a internet con mayor comodidad con los dispositivos móviles, sin la sobrecarga de un ordenador portátil o personal, uno de los retos es proporcionar experiencia basada en la web en una plataforma móvil.

El reto para el desarrollo de aplicaciones móviles es la creación de software que funcione en muchos sistemas operativos móviles, pero existe una amplia variedad de estos sistemas operativos móviles y dispositivos incompatibles que cada plataforma requiere diferentes habilidades o métodos para el desarrollo de software, teniendo como problema un mayor esfuerzo y costo para el desarrollo del software, una alternativa a esto es utilizar software desarrollado con tecnología web (HTML, CSS, JavaScript). (Katzmaier & Hanneghan, 2013).

Debido a la gran cantidad de opciones de framework para el desarrollo, la selección es un proceso complejo debido a múltiples factores que se tienen que

considerar para el desarrollo: Tiempo de desarrollo, integración de hardware, rendimiento, costos de desarrollo entre otros; ya que no existe una comparación entre frameworks bajo un modelo de calidad que permita proveer métricas para especificar los requerimientos de tal manera que se pueda obtener software de calidad.

Debido a esto, esta investigación se centrará en la comparación de algunos de los frameworks open source para el desarrollo de aplicaciones móviles Android basadas en tecnología web, teniendo en cuenta algunas características y/o criterios para su evaluación utilizando un modelo de evaluación para obtener software de calidad. Obtenido el resultado del análisis comparativo se podrá saber y dar a conocer cuál de los frameworks es mejor.

1.2. Antecedentes de estudio.

(Iskander Morine, 2013), en su investigación, ***“Estudio comparativo de alternativas y frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android”***, Se comparan las ventajas y desventajas de utilizar herramientas de desarrollo teniendo en cuenta el conocimiento del desarrollador y el tiempo que puede durar la construcción del producto final, teniendo en cuenta que son desarrolladas en lenguajes diferentes al nativo, evaluando así el funcionamiento de los frameworks para desarrollar aplicaciones móviles, tomando como muestra nueve alternativas que permiten crear aplicaciones para dispositivos móviles Android. Se seleccionaron cuatro frameworks para determinar que elementos y componentes pueden acceder cada uno.

(Bermeo Rodríguez, 2014), en su investigación, ***“Análisis comparativo de frameworks JavaScript jQuery MooTools, para la implementación de aplicaciones Web en la empresa SOFYA aplicación a un caso de estudio”***. Rodríguez desarrolló un modelo de evaluación compuesto por factores de calidad (flexibilidad, escalabilidad y reutilización), basándose en la norma ISO/IEC 25000 y en las métricas de evaluación de cumplimiento de los frameworks, para seleccionar un framework (jQuery o MooTools) que brinde las

mejores características técnicas, la cual permita desarrollar aplicaciones web para la empresa SFYA SYSTEMS.

(Carlos Sánchez Acosta, 2015), en su investigación, **“Análisis comparativo de frameworks para el desarrollo de aplicaciones web en java”**. Se realizó un análisis comparativo de frameworks que permiten el desarrollo de aplicaciones web, donde se aplicó una matriz como modelo de evaluación. Para el análisis y selección de los frameworks se basó en criterios de madurez y documentación. Spring, Struts, JSF, Angular JS fueron los frameworks utilizados para la evaluación. Se realizó un análisis teórico de los frameworks mediante el método QSOS, el cual define características comparativas del software de código libre. Se implementó un sistema de votación con un caso práctico para entender la implementación y codificación en la práctica de los frameworks evaluados. Se establecieron características, subcaracterísticas, atributos y métricas basándose en la norma ISO 25010 y en un modelo de evaluación basado en el modelo IQMC. Se empleó la técnica AHP (Analytic Hierarchy Process) para asignar porcentajes de valor a características y subcaracterísticas. Los resultados obtenidos permiten determinar el framework adecuado para desarrollar aplicaciones web teniendo en cuenta la funcionalidad, fiabilidad, rendimiento, usabilidad, mantenibilidad, portabilidad y compatibilidad.

Estado del arte

(Iskander Morine, 2013), en su investigación “Estudio comparativo de alternativas y frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android”, realizó una comparación de diferentes frameworks los cuales permiten desarrollar aplicaciones móviles para Android, para brindar una guía de referencia al momento de elegir que alternativa se adapta mejor a sus requerimientos, teniendo en cuenta los alcances y limitaciones de cada framework.

Primeramente, realizaron un análisis general de nueve alternativas, teniendo en cuenta las facilidades que ofrece para desarrollar aplicaciones y que lenguaje de programación utilizan, además, de las versiones y costes.

Posterior al análisis, se seleccionaron cuatro frameworks (librerías nativas, jQuery Mobile, PhoneGap y App Inventor), en la cual se analizó que frameworks puede acceder a elementos y componentes, definiendo así cuál se recomienda para el desarrollo de la aplicación que se desea realizar dependiendo al enfoque que se le dará, teniendo en cuenta los conocimientos del desarrollador, para garantizar el mejor resultado con el menor esfuerzo posible.

Como conclusiones: se observó que existen muchas alternativas y frameworks los cuales permiten obtener resultados similares, pero siguiendo otras lógicas y que utilizan lenguajes de programación diferente a la programación nativa que ofrece Android.

Dojo Mobile, jQuery Mobile, Sencha Mobile permiten desarrollar aplicaciones utilizando JavaScript y CSS, para desarrollar aplicaciones personalizables y con una interfaz de usuario agradable e intuitiva con mucho menor esfuerzo y tiempo. Estas alternativas son recomendadas cuando no se requiere utilizar componentes y herramientas del sistema operativo o del teléfono móvil.

AppInventor fue una de las opciones de mayor interés, puesto que permite desarrollar aplicaciones para "Android" utilizando bloques visuales sin la necesidad de realizar programación, pero con la desventaja de estar limitado a las funciones del teléfono a la cual se puede acceder.

(Bermeo Rodríguez, 2014), en su investigación "Análisis comparativo de frameworks javascript: jquery y mootools, para la implementación de aplicaciones web en la empresa sofya. Aplicación a un caso de estudio", se enfocan en la selección de un framework (jQuery o MooTools) para la implementación de una aplicación web. Se basó en la norma ISO/IEC 25000 para desarrollar un modelo de evaluación compuesto por factores de calidad, para la evaluación del software utilizó el método IQMC como guía para identificar características, subcaracterísticas, atributos derivados y atributos básicos.

Se desarrolló el modelo de evaluación haciendo énfasis en la flexibilidad, escalabilidad y reutilización, teniendo como finalidad de que el trabajo de investigación sirva para futuras evaluaciones de software.

Se desarrollaron algunos módulos del sistema SCPC (Sistema de Comercialización y Producción de Concreto), para verificar los beneficios del framework seleccionado.

Mediante la aplicación del modelo de calidad ISO/IEC 25000 y siguiendo las directrices y técnicas del método IQMC, se llegó a la conclusión y a la vez obteniendo resultados, se observó que jQuery supera a MooTools en la mayoría de características, además de que cuenta con una gran comunidad que respalda su evolución, contando así con una extensa cantidad de documentación, foros, tutoriales y componentes prefabricados que hacen que jQuery sea fácil de aprender y usar.

Se concluyó que jQuery es una herramienta concreta y completa ya que dependiendo de los requerimientos, la escala, dificultad y enfoque del proyecto propuesto se ajusta a las necesidades presentadas durante el desarrollo, esto se observó después de desarrollar la aplicación SCPC.

1.3. Teorías relacionadas al tema

1.3.1. Framework

Son módulos concretos que facilitan la organización y desarrollo de software que ayudan a desarrollar y unificar los diferentes componentes de un proyecto, para facilitar y agilizar el desarrollo de software. Permiten a los diseñadores, analistas y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel que debe proveer un sistema funcional. (Osmani, 2012).

1.3.2. Frameworks en el lado del cliente

1.3.2.1. Framework Ionic

Es un SDK de código abierto, lanzado bajo la licencia permisiva MIT, permite a los desarrolladores crear aplicaciones móviles de alta calidad utilizando

tecnologías web (HTML, CSS y JavaScript). Creado por Ben Sperry, Adam Bradley y Max Lynch en 2013.

El objetivo de Ionic es centrarse principalmente en el aspecto y la interacción de interfaz de usuario de una aplicación. Con la finalidad de simplificar una gran parte del proceso de desarrollo de aplicaciones (Front-End).

Los componentes de Ionic son elementos de interfaz de usuario reutilizables que sirven como bloques de construcción de la aplicación para dispositivos móviles. Estos componentes están formados por HTML, CSS y a veces JavaScript. Cada componente se adapta a la plataforma en la que se ejecuta la aplicación.

La navegación funciona como una pila, empujar (push) una página a la pila para navegar hasta ella, y hacer estallar (pop) para volver atrás.

Actualmente Ionic utiliza Angular y Apache Córdoba para trabajar con todo su potencial. Si bien todavía se puede utilizar la parte CSS del Framework, se perderá poderosas interacciones de interfaz de usuario, gestos, animaciones entre otras cosas. Angular es el framework que potencia a Ionic, que es el responsable del componente API, que es el componente básico de Ionic. ("Ionic Framework - Concepts," 2017).

1.3.2.2. JQuery Mobile

JQuery Mobile es un sistema de interfaz de usuario construida con jQuery y jQuery UI basado en HTML5, diseñado para crear sitios web y aplicaciones que sean accesibles en todos los dispositivos de teléfonos inteligentes, tabletas y equipos de escritorio.

Compatible con la mayoría de dispositivos y plataformas: jQuery mobile permite implementar una sola aplicación altamente calificado y personalizado con HTML5 que funciona en los teléfonos y plataformas inteligentes más populares del mercado.

Además, provee herramientas para construir interfaces táctiles dinámicas que se adaptaran a un gran rango de dispositivos móviles. (“jQuery Mobile - About,” 2017).

1.3.3. Frameworks en el lado del servidor

1.3.3.1. Spring Framework MVC

Es un framework completo de programación y configuración que permite crear aplicaciones empresariales modernas y que se implementa en cualquier tipo de plataforma puesto que se basa en Java.

Características

- Inyección de dependencias.
- Programación orientada a aspectos (AOP).
- Implementa el patrón MVC para la creación de aplicaciones mediante Spring MVC.
- Permite la creación de servicios web RESTful
- Soporte básico para JDBC, JPA, JMS.
- Permite integración con otros frameworks.
- Muchos mas
(“Spring Framework Reference Documentation,” n.d.)

1.3.3.2. Spring Security

Spring security ofrece autenticación y autorización a las aplicaciones Java. Tiene la facilidad que puede extenderse para cumplir con los requisitos personalizados. Cuenta con Soporte y extensible tanto para autenticación como para autorización; implementa protección contra ataques como la fijación de sesión, clickjacking, falsificación de solicitud de cross site, etc; integración API Servlet; integración con Spring Web MVC; entre muchas más. (“Spring Security,” n.d.).

Spring Security proporciona implementación de OAuth, que apoya la implementación de los proveedores y consumidores de OAuth, pudiendo implementar Oauth 1 (1a) y Oauth 2. (“Spring Security OAuth,” n.d.)

1.3.3.3. Hibernate

Como framework de mapeo de Object/Relational (ORM), Hibernate se ocupa de la persistencia de los datos, ya que se aplica a las bases de datos relacionales a través de JDBC.

Además de su propia API "nativa", Hibernate también es una implementación de la especificación Java Persistence API (JPA). Como tal, se puede utilizar fácilmente en cualquier entorno que soporte JPA incluyendo aplicaciones Java SE, servidores de aplicaciones Java EE, contenedores Enterprise OSGi, etc.

Hibernate soporta la lazy initialization (carga perezosa), numerosas estrategias de búsqueda. Hibernate no requiere tablas o campos de base de datos especiales, genera gran parte del SQL en el momento de inicialización del sistema en lugar de en tiempo de ejecución.

(“Hibernate ORM,” n.d.)

1.3.4. La familia de normas ISO/IEC 25000

ISO/IEC 25000, proporciona una guía para el uso de las series de estándares internacionales llamados Requisitos y Evaluación de Calidad de Productos Software conocida como SQuaRE, que tiene como objetivo general crear, enriquecer y unificar las series que cubren dos procesos principales: La especificación de requerimientos de calidad del software y la evaluación de la calidad del software, soportada por el proceso de medición de calidad del software.

La norma ISO/IEC 25000 se encuentra compuesta por varias partes o divisiones.
(“NORMAS ISO 25000,” 2017)



Figura 1. Divisiones de ISO 25000. Fuente: (“NORMAS ISO 25000,” 2017).

1.3.4.1. ISO/IEC 25010

Fuente: (“ISO 25010,” n.d.)

Determina las características de calidad del producto software que se pueden evaluar, y define varios modelos de calidad.

Requisitos como la funcionalidad, rendimiento, seguridad, mantenibilidad, etc., son los que se encuentran representados en el modelo de calidad, el cual realiza una jerarquía de la calidad del producto en características y subcaracterísticas.

El modelo de calidad del producto está compuesto por ocho características de calidad que se muestran en la siguiente figura:



Figura 2. Características ISO 25010. Fuente: (“ISO 25010,” n.d.)

1.3.5. Metodo IQMC

Fuente: (Coral Calero, 2010)

El método IQMC ofrece varias guías y técnicas para identificar los factores de calidad apropiados que se deben tener en cuenta para permitir analizar la calidad

de componentes pertenecientes a un cierto dominio de software. Para empezar a construir el modelo, IQMC se basa en el catálogo de factores del estándar ISO/IEC 25000.

El desarrollo de IQMC consiste en 7 pasos, que pueden ser paralelos y/o iterados si se considera oportuno. En el primer paso, el ámbito de calidad es explorado en profundidad y en los pasos restantes se establece la construcción del modelo partiendo de las características de calidad y la descomposición en subcaracterísticas del catálogo ISO/IEC 2501n.

Etapas o pasos del método IQMC

- a. Paso 0: Estudio del ámbito del software
- b. Paso 1: Determinación de características de calidad
- c. Paso 2: Refinamiento de la jerarquía de subcaracterísticas
- d. Paso 3: Refinamiento de subcaracterísticas en atributos
- e. Paso 4: Refinamiento de atributos derivados en básicos
- f. Paso 5: Establecimiento de relaciones entre factores de calidad
- g. Paso 6: Determinación de métricas para los atributos

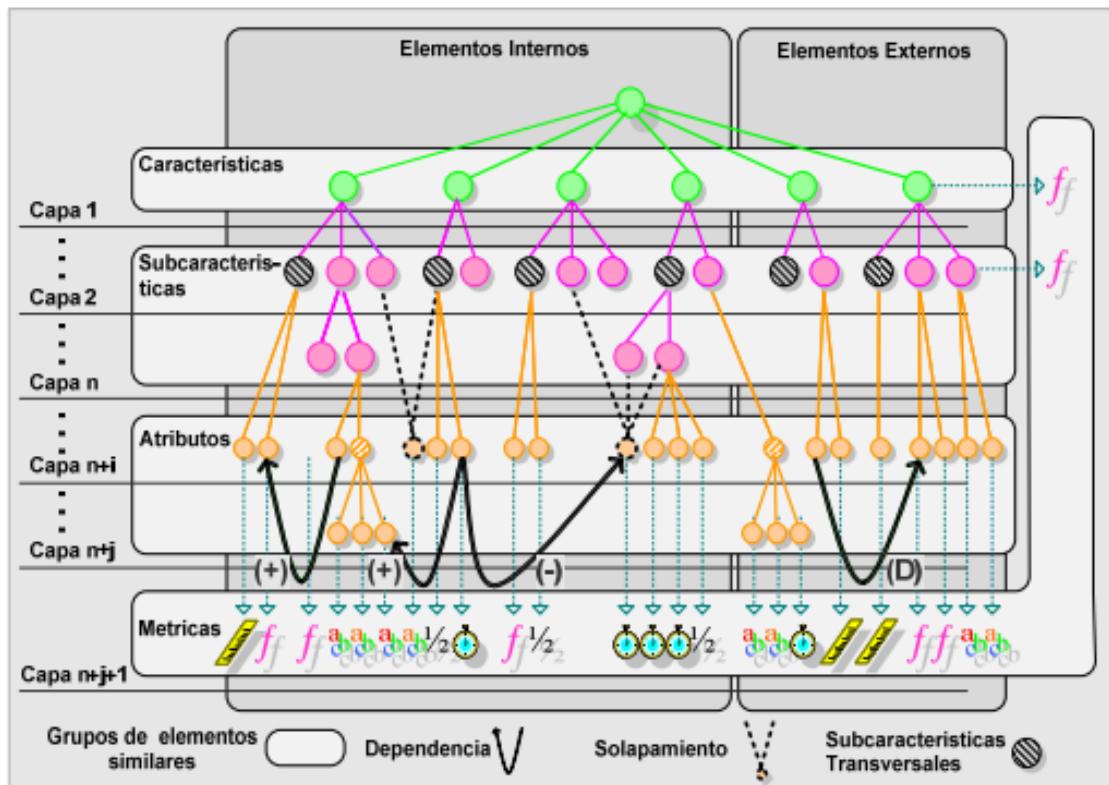


Figura 3. Descripción gráfica del método IQMC. Fuente: (Calero, Moraga, & Piattini, 2010).

1.3.6. Metodo QSOS

La calificación y selección de software de código abierto (QSOS) es una metodología para evaluar el Software Open Source Free/Libre. Esta metodología se distribuye bajo la licencia GFDL. (“QSOS - Wikipedia,” n.d.)

El proceso general diseñado se compone de cuatro etapas definidos en ciclos iterativos.

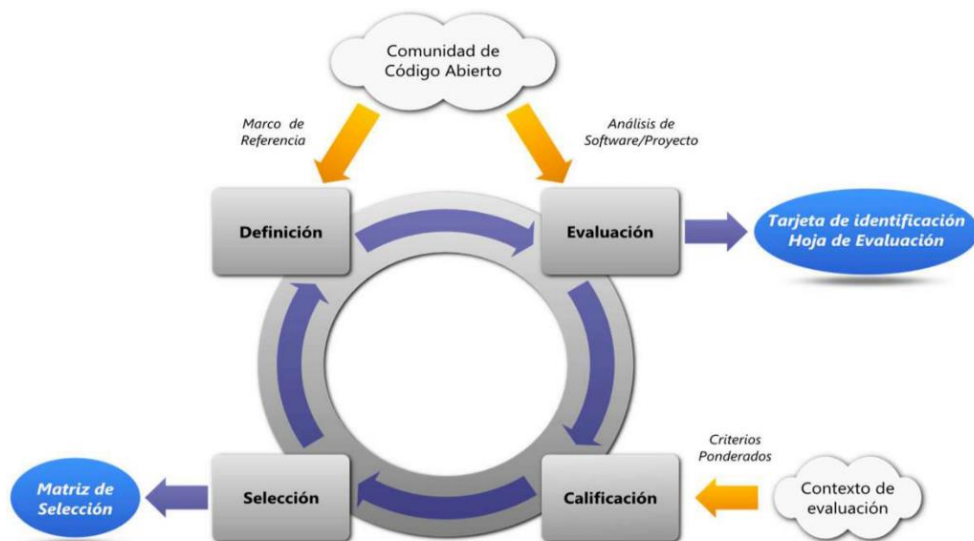


Figura 4. Etapas de QSOS. Fuente: (Ramos & Páez, 2011).

QSOS posibilita establecer el nivel de detalle del proceso en función de los requerimientos individuales, así como el avance del proceso en ciclos iterativos para ir perfeccionando las cuatro etapas definidas al aplicarse con diferentes niveles de granularidad. (Ramos & Páez, 2011).

1.3.7. Scrum

Scrum permite entregar producto del máximo valor posible de manera productiva y creativa, ya que es un marco de trabajo por el cual las personas pueden acometer problemas complejos adaptativos. (Schwaber & Sutherland, 2013)

Scrum es:

- Ligero
- Fácil de entender
- Extremadamente difícil de llegar a dominar

Scrum tiene un enfoque en el conocimiento, ya que esta metodología procede de la experiencia para la toma de decisiones, usando una perspectiva recurrente e incremental para mejorar la predictibilidad y el control del riesgo.

Esta metodología tiene una constante colaboración con el cliente dando características como: (Schwaber & Sutherland, 2013).

- Predisposición y respuesta al cambio.
- Desarrollo iterativo e incremental con entregas frecuentes de funcionalidad del proyecto.
- Comunicación verbal directa.
- Simplicidad, sólo los artefactos necesarios.
- Motivación, compromiso y responsabilidad del equipo para la autogestión y auto organización.

El cliente al no visualizar las necesidades de usuarios objetivos en su marco de trabajo, centrarse en la comunicación directa puede generar problemas de diseño e iteración.

Sprint

Es el ciclo de trabajo que se tiene para crear un entregable o incremento de software finalizado, disponible y potencialmente desplegable, que aporte valor al cliente, el cual inicia al finalizar el sprint anterior.

Está compuesto por los siguientes eventos: Reunión de Planificación del Sprint, Scrum Diario, Revisión del Sprint, Retrospectiva del Sprint. (Schwaber & Sutherland, 2013).

Objetivo del sprint

El objetivo del sprint es una meta que se debe lograr mediante la asignación de la “Lista de Producto”, ofreciendo orientación para lograr un incremento de valor en el producto que se está construyendo. (Schwaber & Sutherland, 2013).

El equipo SCRUM (SCRUM Team)

Está integrado por: el dueño del producto (Product Owner), el equipo de desarrollo (Development Team) y un SCRUM Master los cuales son auto organizados y multifuncionales. El equipo SCRUM tiene que ser totalmente autónomo, flexible, creativo y productivo para generar versiones útiles y funcionales del producto. El equipo SCRUM debe tener conocimientos sólidos al construir un producto dada la variedad de dispositivos móviles y las múltiples plataformas existentes. (Schwaber & Sutherland, 2013).

1.3.8. Definición de la terminología

Análisis

Es la etapa donde se recopilan, identifican, clasifican, y documentan los requerimientos del sistema informático, para evaluar los múltiples escenarios o tipos de interacción de los usuarios con el sistema para obtener como resultado el modelo del sistema. (“análisis - Wikcionario,” n.d.).

Open source

Open Source (Código abierto) expresión con la que se conoce al software distribuido y desarrollado libremente. Promueve el acceso al código fuente de un software que no requiere licencia para su uso. (“Open Source | Open Source Initiative,” n.d.).

Framework

Los Frameworks son componentes personalizables e intercambiables para el desarrollo de software que ayudan a desarrollar y a unir los diferentes componentes de un proyecto. (Osmani, 2012).

“Tiene como objetivos principales: acelerar el proceso de desarrollo, reutilizar código ya existente y promover las buenas prácticas de desarrollo.” (Gutiérrez, 2014).

Android

Es un sistema operativo basado en Linux para dispositivos móviles como teléfonos inteligentes y tabletas, gracias a que se encuentra por encima del Kernel puede gestionar y acceder a los recursos (controladores de pantalla, cámara, audio, memoria flash, etc). (Gironés, 2011).

Calidad

Es el grado en el que un conjunto de características inherentes (característica intrínseca permanente) cumple con los requisitos. (ISO 2005a).

Modelo

“Define características del software que pueden ser evaluadas y que influyen en la calidad del mismo. Es importante remarcar que el modelo establece que características de calidad evaluar, pero no el cómo evaluarlas.” (Moisés Rodríguez, Óscar Pedreira y Carlos M. Fernández. 2015. Certificación de la Mantenibilidad del Producto Software: Un Caso Práctico Revista Latinoamericana de Ingeniería de Software, 3(3): 127-134, ISSN 2314-2642. p 21.).

Modelo de calidad

Conjunto de factores y relaciones que proporciona una base para la especificación de requisitos y evaluación de la calidad de los componentes software que se usan como base para definir un framework de calidad común y ser adaptados a contextos específicos. (Coral Calero, 2010).

Modelo de calidad de software

“Conjunto de características y las relaciones entre las mismas, que proveen la base para especificar requisitos de calidad y evaluar la calidad.” (NC-ISO/IEC 9126-1, 2005.).

Proceso de evaluación

“Su objetivo es definir actividades a realizar para poder realizar la evaluación de la calidad del producto software, para generar un informe con los resultados obtenidos, asegurando un informe completo, repetible y legible para el público objetivo del mismo.” (Moisés Rodríguez, Óscar Pedreira y Carlos M. Fernández. 2015. Certificación de la Mantenibilidad del Producto Software: Un Caso Práctico Revista Latinoamericana de Ingeniería de Software, 3(3): 127-134, ISSN 2314-2642. p 21.).

1.4. Formulación del Problema.

¿Mediante un análisis comparativo, cuál de los frameworks elegidos es mejor para el desarrollo de aplicaciones móviles Android basadas en tecnología web?

Delimitación de la Investigación

Las delimitaciones consideradas para la investigación son:

- a. Los frameworks elegidos serán para el lado del cliente (Front-End)
- b. Los frameworks elegidos se implementan en base a html, css, javascript.
- c. La aplicación que se desarrollará para cada framework se aplicará a un sistema transaccional, la aplicación se llama Golwin donde se realizan apuestas de fútbol.
- d. Se aplicarán, utilizarán solo herramientas y/o tecnologías de los propios frameworks, mas no complementos externos.
- e. Por ser frameworks Front-end del lado del cliente se tendrá que realizar la aplicación para el back-end del lado del servidor.

1.5. Justificación e Importancia del estudio.

La variedad de dispositivos plantea un nuevo desafío importante en el diseño de interfaz de usuario para establecer aplicaciones en las que los usuarios en su mayoría no sentirían ninguna diferencia entre varios dispositivos. Independientemente de un sistema, su interfaz de usuario debe ser suave e intuitiva en todas las plataformas para todos los dispositivos. (Marenkov et al., 2015).

Por el rápido crecimiento y flujo de los negocios actuales, se propone buscar una recopilación de frameworks que permitan el desarrollo de aplicaciones para dispositivos móviles Android utilizando tecnología web.

Este proyecto tiene como objetivo realizar un análisis comparativo sobre los frameworks como lo son JQuery Mobile e Ionic, que son utilizados por la comunidad para el desarrollo de aplicaciones en dispositivos móviles Android utilizando tecnología web, dicho análisis comparativo va a permitir tener un criterio para crear aplicaciones que sea eficientes, estables, portables, escalables, entre otros; permitiendo ahorrar tiempo y dinero en el desarrollo. Y obteniendo de esta manera un software de calidad mediante un modelo de evaluación.

Limitaciones de la Investigación

Esta investigación tiene las siguientes limitaciones:

La cantidad de frameworks para el desarrollo de aplicaciones para entorno móvil es muy extensa y no calculable, por lo que se dificultará tener una lista exacta de todos ellos, lo que impedirá tener una población aproximada o exacta a la realidad.

Al ser la lista no calculada de frameworks, y cada vez surgen más, no se pudo encontrar un ranking de los más usados.

Al no tener un sistema transaccional en el lado del servidor con el cual poder realizar las implementaciones para los frameworks que son para el lado del cliente se tendrá que implementar un sistema para realizar el caso de estudio, lo que implicará en la demora en el desarrollo de la investigación.

1.6. Hipótesis.

¿El análisis comparativo de frameworks open source para el desarrollo de aplicaciones móviles Android basadas en tecnología web, aplicando un modelo de evaluación podrá determinar cuál es el mejor framework basados en métricas de calidad?

1.7. Objetivos

1.7.1. Objetivo general

Realizar un análisis comparativo de Frameworks Open Source para el desarrollo de aplicaciones móviles Android basadas en tecnología web.

1.7.2. Objetivos específicos

- a. Seleccionar framework para su análisis.
- b. Seleccionar modelo de evaluación para realizar la comparación entre frameworks.
- c. Desarrollar una aplicación como caso de estudio para cada framework.
- d. Evaluar los resultados.

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de investigación

2.1.1. Tipo de la investigación

Este proyecto corresponde al tipo de investigación aplicada; porque los conocimientos obtenidos en las investigaciones se plasmarán en la práctica para de esta manera alcanzar los objetivos, y con ello traer beneficios a la sociedad.

2.1.2. Diseño de la investigación

El diseño de la investigación es Cuasi-Experimental, porque la población y la muestra son establecidos por el investigador.

2.2. Población y muestra

2.2.1. Población

La población está determinada por la cantidad de frameworks basados en tecnología web que sirven para el desarrollo de aplicaciones móviles, los cuales se detallan en la Tabla N° 17, siendo 49 frameworks encontrados en diferentes sitios web.

2.2.2. Muestra

La muestra está determinada según los resultados de los frameworks en la Tabla N° 18, los cuales son los más mencionados o conocidos por la comunidad de

desarrolladores. De estos se elegirán a los dos con más puntuación y teniendo en cuenta investigaciones pasadas.

2.3. Variables, operacionalización.

2.3.1. Variable Independiente

Análisis comparativo de los frameworks.

2.3.2. Variable dependiente

Frameworks basados en métricas de calidad.

2.3.3. Operacionalización

En esta parte se utilizará el método QSOS para un análisis teórico de los frameworks para comparar características de software libre. También se utilizará el Modelo de Construcción de Calidad IQMC en conjunto con la División para el modelo de Calidad ISO/IEC 25010 de la familia de normas ISO/IEC 25000, la cual tiene características que van a permitir establecer subcaracterísticas, atributos y métricas para la evaluación de los framework seleccionados.

Tabla 1.

Operacionalización de variables con sus indicadores y diferentes métodos, técnicas e instrumentos para la recolección de datos

Variables	Dimensiones	Indicadores	Método, Técnicas e instrumentos de recolección de datos
Dependiente: Frameworks basados en métricas de calidad	Adecuación funcional	<ul style="list-style-type: none"> ➤ Completitud funcional ➤ Corrección funcional ➤ Pertinencia funcional 	Método IQMC, Observación, ficha de registro del Modelo de

Eficiencia de desempeño	<ul style="list-style-type: none"> ➤ Comportamiento en el Tiempo ➤ Utilización de recursos 	Calidad ISO/IEC 25010
Compatibilidad	<ul style="list-style-type: none"> ➤ Coexistencia ➤ Interoperabilidad 	
Usabilidad	<ul style="list-style-type: none"> ➤ Capacidad para reconocer su adecuación. ➤ Capacidad de aprendizaje. ➤ Capacidad para ser usado ➤ Protección frente a errores de usuario ➤ Accesibilidad 	
Fiabilidad	<ul style="list-style-type: none"> ➤ Madurez. ➤ Disponibilidad ➤ Tolerancia a fallos ➤ Capacidad de recuperación. 	
Seguridad	<ul style="list-style-type: none"> ➤ Confidencialidad ➤ Integridad ➤ No repudio ➤ Autenticidad ➤ responsabilidad 	
Mantenibilidad	<ul style="list-style-type: none"> ➤ Modularidad ➤ Reusabilidad ➤ Capacidad para ser modificado 	
Portabilidad.	<ul style="list-style-type: none"> ➤ Adaptabilidad. 	

- Capacidad de ser instalado.
- Capacidad de ser reemplazado.

Nota: Se muestra el indicador dependiente con sus dimensiones e indicadores y los respectivos métodos, técnicas e instrumentos para la recolección de datos.
Fuente: Elaboración propia

2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.

Técnicas de recolección de datos

- a. Observación: esta técnica va a ayudar a verificar los resultados de las características, subcaracterísticas a ser evaluados por el modelo de calidad.

- b. Análisis documental: Esta técnica permite obtener información de las distintas documentaciones (manuales, tutoriales, artículos, etc.) de los frameworks para ser implementados, para luego dar solución al problema planteado y así obtener resultados.

Instrumentos de recolección de datos

El instrumento que se usará en esta investigación será una ficha de registro donde evaluaremos los framework según las características, subcaracterísticas y atributos del modelo de evaluación del ISO/IEC 25010, obteniendo puntajes de cada framework y de esta manera realizar el análisis comparativo entre los frameworks.

Validez y confiabilidad

Mediante los siguientes métodos se medirán la validez y confiabilidad, los cuales son adecuados para el desarrollo de la investigación:

- a. Método IQMC: Este método nos ayudara a la selección de los framework, la cual consta de una serie de pasos en las cuales incluiremos a la norma ISO/IEC 25010 para aplicar el modelo de evaluación para los frameworks.

- b. Observación: Este método ayudará al análisis de los diferentes frameworks, la cual se aplicará en el inicio, en el proceso y en los resultados de la investigación de los frameworks.
- c. Experimental: Este método nos ayudara a evaluar los diferentes frameworks según el modelo de calidad.
- d. Medición: Este método ayudará a obtener la información numérica de las características del modelo de calidad que se evaluarán, para luego comparar dichas características medibles para cada framework.

2.5. Procedimiento análisis de datos.

- a. **Método IQMC:** Para la recolección de datos mediante el método IQMC en la presente investigación se siguió el siguiente proceso.
 - i. Estudiar el ámbito del software para saber cuáles son los componentes, funcionalidades y servicios fundamentales que debe proporcionar un framework front-end para móviles. El framework debe tener compatibilidad de sus componentes para ser interpretado por los navegadores web al momento de su desarrollo; debe de contar con comunicación asíncrona (AJAX); debe permitir manipular y recorrer el Modelo de Objetos del Documento (DOM), permitir validar formularios, contar con efectos visuales, manejar eventos, tener un buen soporte y contar con una buena documentación. Se desarrolló una aplicación de apuestas de fútbol (Golwin) con cada framework para corroborar de que contaba con las características y funcionalidades que se mencionaron.
 - ii. Determinar las características de calidad según la norma ISO/IEC 25010, en las cuales se elegirán características a conveniencia, ya que son aplicables a framework front-end (lado del cliente), las cuales son: funcionalidad, rendimiento, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad y portabilidad.

- iii. Jerarquizar las características de calidad en subcaracterísticas, las cuales se describirán y nos ayudarán a crear el modelo de evaluación de calidad detallado para realizar la evaluación de cada framework.
 - iv. Jerarquizar las subcaracterísticas en atributos, que nos van a permitir crear un modelo de evaluación de calidad, las cuales nos servirán para realizar una buena evaluación de cada framework.
 - v. Derivar los atributos en atributos básicos, los cuales nos permiten tener atributos de evaluación más detallados, para así poder crear el modelo de evaluación de calidad, siendo así poder realizar una mejor evaluación de cada framework.
 - vi. Establecer relaciones entre factores de calidad, esto permite crear las relaciones jerárquicas de las características, subcaracterísticas, atributos y atributos básicos.
 - vii. Determinar métricas para los atributos, las cuales nos permiten evaluar las características de calidad utilizando la norma ISO/IEC 25010 de un framework para así seleccionar el que tenga mejores resultados. Utilizando métricas de cumplimiento y métricas de cumplimiento por rangos.
- b. **Observación:** Observar y documentar el funcionamiento y comportamiento de la aplicación desarrollada con cada uno de los frameworks, y de esta manera poder llenar la ficha de registro con las características del modelo de evaluación de la norma ISO/IEC 25010.
 - c. **Ficha de registro:** Después de aplicar la técnica de observación debemos realizar la evaluación del framework según el modelo de evaluación que se desarrolló y/o aplicó, para posteriormente obtener los resultados.

Análisis Estadístico e interpretación de los datos

Se realizará mediante métricas booleanas y por rangos para la asignación en las diferentes características y subcaracterísticas y atributos de la matriz de calidad

y de esta manera comparar los resultados obtenidos con las métricas aplicadas para la comparación de los frameworks seleccionados.

- a. Métricas booleanas: si cumple se asignará un valor de 1, caso contrario de asignará un 0.
- b. Métricas por rangos: la asignación de valores se realiza para un valor mínimo se asignará 0, para un valor máximo se asignará 4, se califica según se cumpla en su totalidad o nulo la característica evaluada, asignando el más alto puntaje o caso contrario el más mínimo respectivamente.

2.6. Criterios éticos.

Los principios éticos que se respetan en el presente proyecto es el Código Deontológico del Colegio de Ingenieros del Perú en su Capítulo III “Faltas Contra la Ética Profesional y Sanciones” y su Sub Capítulo II “De la Relación con El Público” en su Artículo 106 expresa:

El presente proyecto de investigación plasmará lo más claro y conciso su contenido con la finalidad de generar un aporte a los desarrolladores de aplicaciones para Android utilizando tecnología web que deseen acelerar el tiempo de desarrollo y ahorrar costos de producción, estando disponible para el público en general el presente proyecto.

2.7. Criterios de rigor científico.

Validez: según el objetivo general de la investigación, están escritas clara y precisamente en la operacionalización de variables y en la definición de las dimensiones. Para poder validar los resultados obtenidos se realizará la evaluación de las dimensiones según los indicadores establecidos y las técnicas de recolección de datos definidas.

Replicabilidad: Dado que los métodos a aplicar están establecidos por los autores de los artículos de investigación consultados, los resultados de esta investigación no se pueden contradecir en caso se vuelva a repetir, puesto que

se pueden existir diferencias en los resultados en caso se tome un enfoque diferente.

III. RESULTADOS

3.1. Resultados en tablas y gráficos.

A continuación, se mostrarán los resultados obtenidos de la evaluación, aplicando la matriz de calidad del modelo ISO 25010 a la variable dependiente, para así poder aplicar las métricas correspondientes a los indicadores que se seleccionaron.

3.1.1. Adecuación funcional

a. Para el indicador de Completitud funcional

La tabla 2 representa las funcionalidades que un framework brinda al usuario para desarrollar aplicaciones amigables e interactivas, cubriendo las tareas y objetivos planteados por el usuario, además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 2.

Resultados obtenidos para el indicador de Completitud funcional entre el framework Ionic y jQuery Mobile.

Completitud funcional	Métrica	Ionic	jQuery Mobile
Creación de aplicaciones interactivas	1,0	1	1
Creación de aplicaciones amigables	1,0	1	1
Creación de aplicaciones robustas	1,0	1	1
Creación de aplicaciones orientadas a móviles	1,0	1	1
Puntaje		4	4

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010

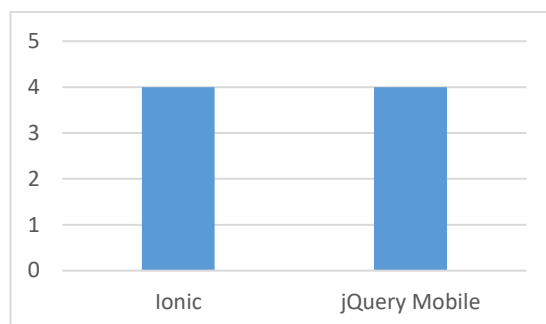


Figura 5. Puntaje del indicador de Completitud funcional de cada framework.
Fuente: Matriz de calidad ISO 25010.

En la tabla 2 y Figura 5 de puntaje del indicador de integridad de cada framework, se observa que existe un empate entre ambos, contando un total de 4 puntos para ambos frameworks, que indica que ambos tienen funcionalidades para realizar tareas y objetivos planteados por el usuario para el desarrollo de aplicaciones.

b. Para el indicador de Corrección funcional

La tabla 3 representa la capacidad del framework para proporcionar los resultados o efectos correctos, además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 3.

Resultados obtenidos del indicador de Corrección funcional.

Corrección funcional	Métrica	Ionic	jQuery Mobile
Efectividad en la creación de efectos y animaciones			
Configuración de "fps" (Frames por segundo)	1,0	1	0
Configuración de "unit" (Unidad de medida)	1,0	1	0
Configuración de tiempo	1,0	1	0
Adaptación automática	1,0	1	1
Efectividad en la manipulación del DOM			
Selectores "id" y "class"	1,0	1	1

Selectores CSS	1,0	1	1
Selectores por tipo de Elemento	1,0	1	1
Selectores personalizados	1,0	1	1
Puntaje		8	5

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010

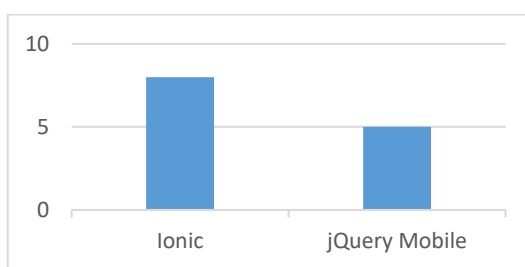


Figura 6. Puntaje del indicador de Corrección funcional de cada framework.

Fuente: Matriz de calidad ISO 25010.

En la tabla 3 y figura 6, se observa que el framework Ionic cumple con todos los atributos del indicador de precisión, superando así a jQuery Mobile. Esto indica que Ionic proporciona los resultados o efectos correctos al momento de desarrollar una aplicación.

c. Para el indicador de Pertinencia funcional

La tabla 4 representa las funcionalidades que posee el framework para la comunicación con el servidor (Ajax), gestión de efectos y eventos, manipulación y recorrido del DOM, validación de formularios, entre otros, además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 4.

Resultados obtenidos del indicador de Pertinencia funcional.

Pertinencia funcional	Métrica	Ionic	jQuery Mobile
Método Ajax	1,0	1	1
Métodos para manipular el DOM	1,0	1	1

Elementos interactivos	1,0	1	1
Validación de formularios	1,0	1	1
Efectos visuales	1,0	1	1
Gestión de tablas o grids	1,0	1	1
Puntaje		6	6

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

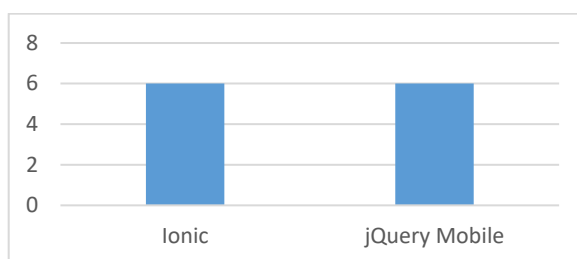


Figura 7. Puntaje del indicador de Pertinencia funcional para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 4 y figura 7 se observa que existe un empate entre ambos frameworks, ambos cumplen con todos los atributos del indicador de idoneidad, lo cual indica que ambos poseen funcionalidades para la comunicación con el servidor, gestión de efectos y eventos, manipulación y recorrido del DOM, entre otros; lo cual permite crear aplicaciones con todas estas posibilidades.

3.1.2. Rendimiento

a. Para el indicador de comportamiento en el tiempo

Para esta característica se eligió por conveniencia este indicador con los atributos que se presentan en la tabla 5, por ser atributos que se pueden comprobar en un framework del lado del cliente; además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 5.

Resultados obtenidos del indicador de comportamiento en el tiempo.

Comportamiento en el Tiempo	Métrica	Ionic	jQuery Mobile
Configuración del tiempo de espera en una comunicación Cliente – Servidor	1, 0	1	1
Comunicación Síncrona y Asíncrona	1, 0	1	1
Tiempo de carga en prueba local	R T(ms)		
	5 100	4.75	4.72
	4 200	(125	(128
	3 300	ms)	ms)
	2 500		
	1 1000+		
Puntaje		6.75	6.72

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

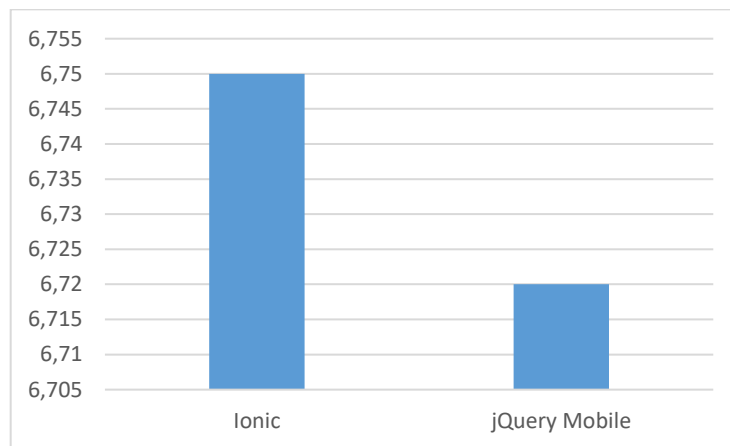


Figura 8. Puntaje del indicador de comportamiento en el tiempo de cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 5 y figura 8 se observa que existe un empate entre ambos frameworks, ya que ambos tienen la capacidad de establecer tiempos de espera entre una comunicación cliente – servidor, y también se pueden realizar comunicaciones síncrona y asíncrona.

b. Para el indicador de Utilización de recursos

Tabla 6.

Resultados obtenidos del indicador de utilización de recursos.

Utilización de recursos	Métrica	Ionic	jQuery Mobile
Necesita librerías	1, 0	1	1
JavaScript	1, 0	1	1
Incorporación de AJAX	1, 0	1	1
Iconos propios	1, 0	1	1
Puntaje		4	4

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

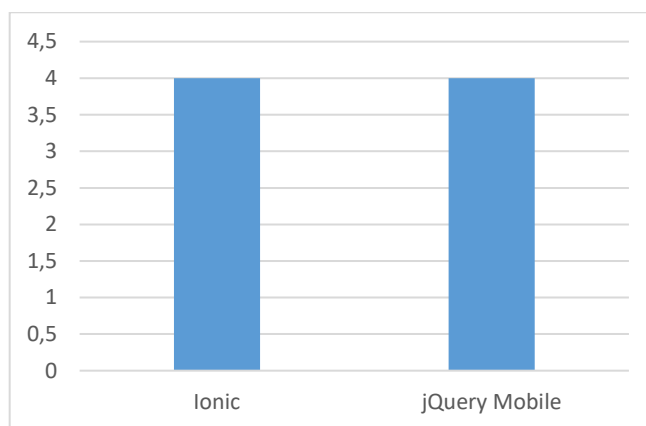


Figura 9. Puntaje del indicador de utilización de recursos para cada framework. Fuente: Matriz de calidad ISO 25010.

3.1.2.1. Compatibilidad

a. Para el indicador de coexistencia

La tabla 7 representa la capacidad que tiene un framework para coexistir con otro en un entorno común, compartiendo recursos comunes; además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 7.

Resultados obtenidos del indicador de coexistencia.

Coexistencia	Métrica	Ionic	JQuery Mobile
Coexistencia con HTML5	1,0	1	1
Coexistencia con CSS3	1,0	1	1
Coexistencia con JavaScript	1,0	1	1
Puntaje		3	3

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

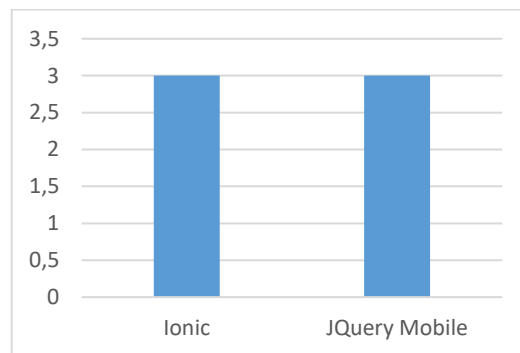


Figura 10. Puntaje del indicador de coexistencia para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 7 y figura 10 se observa que existe un empate entre ambos frameworks, lo que indica que ambos pueden coexistir en un entorno común, compartiendo recursos comunes, según los atributos establecidos por la matriz de calidad ISO 25010.

b. Para el indicador de interoperabilidad

En la tabla 8 representa la capacidad de dos o más frameworks para intercambiar información y utilizar la información intercambiada, además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 8.

Resultados obtenidos del indicar de interoperabilidad.

Interoperabilidad	Métricas	Ionic	jQuery Mobile
Intercambio de información con PHP	1,0	1	1
Intercambio de información con Java EE	1,0	1	1
Intercambio de información con C# (ASP.net)	1,0	1	1
Intercambio de información con Pyton	1,0	1	1
Intercambio de información con Ruby	1,0	1	1
Puntaje		5	5

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

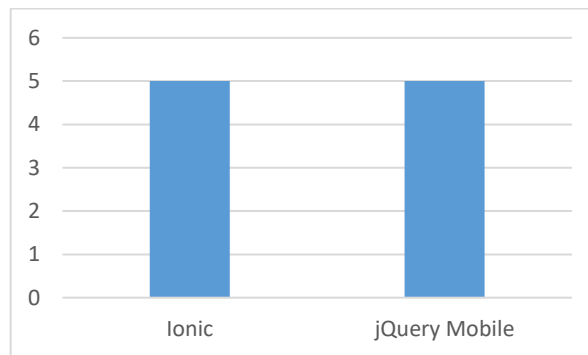


Figura 11. Puntaje del indicador de interoperabilidad de cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 8 y figura 11 observamos que existe un empate entre ambos frameworks para el indicador de interoperabilidad, indicando así que ambos tienen la capacidad de intercambiar y utilizar información entre uno o más frameworks, y con diferentes lenguajes de programación.

3.1.3. Usabilidad

a. Para el indicador de Capacidad para reconocer su adecuación

La tabla 9 representa la capacidad de los desarrolladores del framework para permitirle al usuario identificar si sus funcionalidades son adecuadas para sus necesidades, teniendo como único atributo a principios bien definidos, que se

fue elegido por conveniencia; además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 9.

Resultados obtenidos del indicador de Capacidad para reconocer su adecuación.

Capacidad para reconocer su adecuación	Métrica	Ionic	jQuery Mobile
Información real en el sitio web	1,0	1	1
Actualización de contenido	1,0	1	1
Puntaje		2	2

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

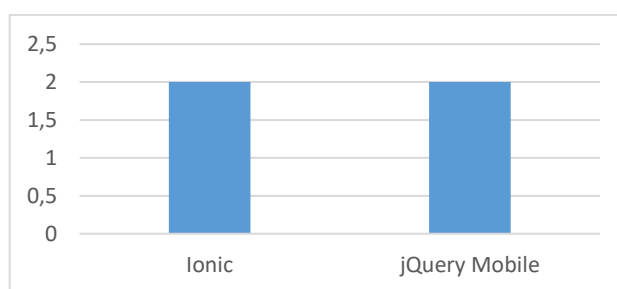


Figura 12. Puntaje del indicador de Capacidad para reconocer su adecuación para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 9 y figura 12 observamos que el framework Ionic tiene un mejor puntaje frente a jQuery Mobile, siendo este el que tenga mejor definido sus principios de sus funcionalidades, permitiendo así identificar si estas cumplirán con sus necesidades.

b. Para el indicador de Capacidad de aprendizaje

La tabla 10 representa la capacidad de la comunidad que tiene el framework para permitirle al usuario aprender los diferentes atributos, métodos y componentes que este posee; además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 10.

Resultados obtenidos del indicador de Capacidad de aprendizaje.

Capacidad de Aprendizaje	Métrica	Ionic	jQuery Mobile
Manual técnico	0,4	4	3
Foros	0,4	4	4
Tutoriales	0,4	4	4
Cursos on-line	0,4	4	4
Estudios e investigaciones	0,4	4	3
Documentación	0,4	4	4
Puntaje		6	5.25

Nota: Se evalúa según la métrica (por rango de 0 a 4). Fuente: Matriz de calidad ISO 25010.

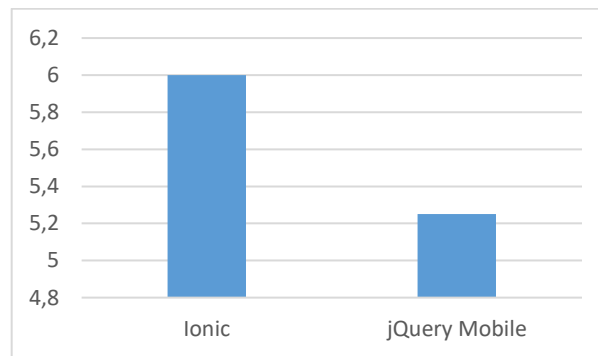


Figura 13. Puntaje del indicador de Capacidad de aprendizaje para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 10 y figura 13 observamos que el framework Ionic cuenta con muchos más recursos que jQuery Mobile, además, el framework Ionic tiene una mejor comunidad la cual permite que el usuario aprenda los diferentes atributos, métodos y componentes que posee el framework

c. Para el indicador de Capacidad para ser usado

La tabla 11 representa la capacidad del framework para permitirle al usuario operarlo y controlarlo con facilidad; además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 11.

Resultados obtenidos del indicador de Capacidad para ser usado.

Capacidad para ser usado	Métrica	Ionic	jQuery Mobile
POO. basada en Clases.	1,0	1	0
Reconocimiento de entorno	1,0	1	1
Licencia	1,0	1	1
Puntaje		3	2

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

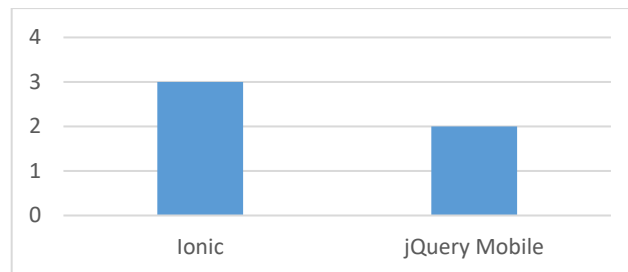


Figura 14. Puntaje del indicador de Capacidad para ser usado para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 11 y figura 14 podemos observar que el framework Ionic cuenta con una programación basada en clases, lo que permite tener ordenado el código para su fácil manipulación y modificación. El framework Ionic permite separar la capa visual de la lógica de negocio, esta estructura es propia del framework.

3.1.4. Fiabilidad

a. Para el indicador de madurez

La tabla 12 representa la capacidad con la cuenta el framework para evitar algunas fallas en el resultado del producto, ya que este tiene un buen tiempo en el mercado y es utilizado por la comunidad para el desarrollo de aplicaciones, además de contar con actualizaciones disponibles y constante evolución y aceptación por parte de la comunidad.

Tabla 12.

Resultados obtenidos del indicador de madurez.

Madurez	Métrica	Ionic	jQuery Mobile
Tiempo en el mercado		4	4
Actualizaciones disponibles		4	1
Posee una base del conocimiento software		1	1
Popularidad en GitHub	R #Star		
	5 100000+		
	4 50000	3.4	2.1
	3 25000		
	2 10000		
	1 1000+		
Aplicaciones realizadas		1	1
Puntaje		7.4	5.26

Nota: Se evalúa según la métrica (por rango). Fuente: Matriz de calidad ISO 25010.

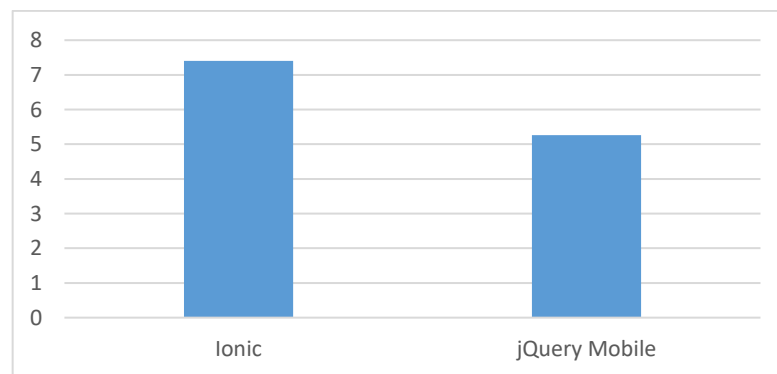


Figura 15. Puntajes del indicador de madurez para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 12 y figura 15 podemos observar que el framework Ionic tiene un mejor resultado para el indicador de madurez, siendo este el que mejor cumple con la evaluación de los atributos establecidos en la matriz de calidad ISO 25010.

b. Para el indicador de disponibilidad

La tabla 13 representa si un framework tiene facilidades de accesibilidad y uso; además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 13.

Resultados obtenidos del indicador de disponibilidad.

Disponibilidad	Métrica	Ionic	jQuery Mobile
Descarga del Framework			
Descarga con compresión	1,0	1	1
Descarga sin compresión	1,0	1	1
Descarga desde el sitio web oficial	1,0	1	1
Descarga desde sitios web no oficiales	1,0	0	1
Referenciar el Framework			
Referencia a través del sitio web oficial	1,0	1	1
Referencia a través de Google	1,0	1	1
Referencia a través de sitios web no oficiales	1,0	1	1
Puntaje		6	7

Nota: Se evalúa según métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

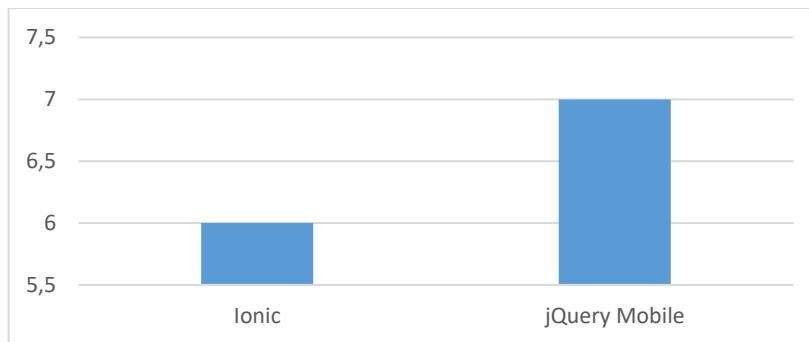


Figura 16. Puntaje del indicador de disponibilidad para cada framework.

Fuente: Matriz de calidad ISO 25010.

En la tabla 13 y figura 16 se puede observar que el framework jQuery mobile tiene una mejor disponibilidad, ya que se puede descargar y referenciar de distintos sitios, sean o no oficiales, siendo esto posiblemente una mala opción ya que al descargar de sitios no oficiales se corre el riesgo de no obtener un framework actualizado y confiable. Por el contrario, el framework Ionic solo permite descargarse de su propia página web o repositorio, lo que va a permitir tener un software confiable y actualizado.

3.1.5. Seguridad

Tabla 14.

Resultados para los indicadores de seguridad.

Seguridad	Métrica	Ionic	jQuery Mobile
Confidencialidad			
Protección de datos	1, 0	0	0
Acceso solo a usuarios específicos	1, 0	0	0
Es seguro frente a internet	1, 0	0	0
Cifrado en la información	1, 0	0	0
Integridad			
Información correcta en la base de datos	1, 0	0	0
No permite modificación de datos	1, 0	0	0
No repudio			
Comunicación confiable cliente-servidor	1, 0	0	0
Autenticidad			
Evita la suplantación de identidad	1, 0	0	0
Autenticidad generada por el usuario	1, 0	1	1
Responsabilidad			
Se hace responsable de la seguridad	1, 0	0	0
Puntaje		1	1

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

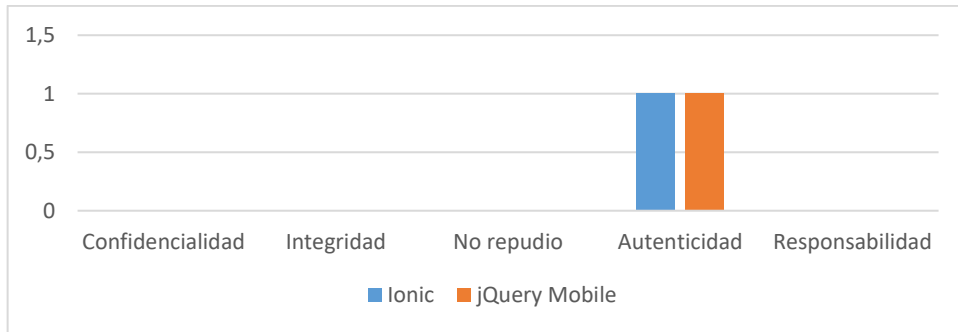


Figura 17. Puntajes los indicadores de seguridad para cada framework. Fuente: Matriz de calidad ISO 25010.

3.1.6. Mantenibilidad

a. Para el indicador de modularidad

Tabla 15.

Resultados para el indicador de modularidad para cada framework.

Modularidad	Métrica	Ionic	jQuery Mobile
Es modular	1, 0	1	1
Crear componentes	1, 0	1	1
Eliminar componentes	1, 0	1	1
Modificar componentes	1, 0	1	1
Puntaje		4	4

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

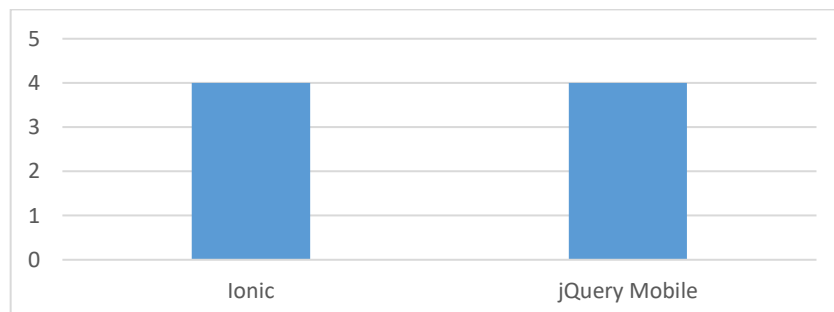


Figura 18. Puntajes de modularidad para cada framework. Fuente: Matriz de calidad ISO 25010.

b. Para el indicador de reusabilidad

Tabla 16.

Resultado para el indicador de reusabilidad para cada framework.

Reusabilidad	Métrica	Ionic	jQuery Mobile
Componentes re-usables	1, 0	1	1
Puntaje		1	1

Nota: Se evalúa según la métrica (1 si cumple, 0 no cumple). Fuente: Matriz de calidad ISO 25010.

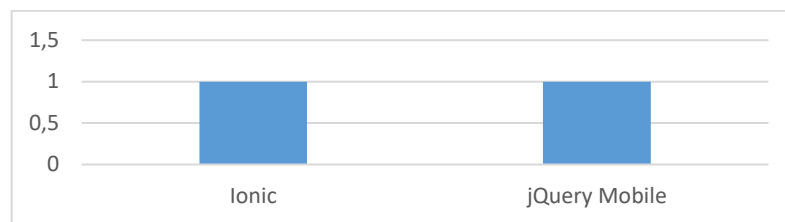


Figura 19. Puntajes de reusabilidad para cada framework. Fuente: Matriz de calidad ISO 25010.

3.1.7. Portabilidad

a. Para el indicador de adaptabilidad

En la tabla 17 representa la capacidad que tiene el framework para ser adaptado de forma efectiva y eficiente a diferentes navegadores de distintos dispositivos para ser probado o utilizado; además contiene datos obtenidos de la matriz de calidad del ISO 25010.

Tabla 17.

Resultados obtenidos del indicador de adaptabilidad.

Adaptabilidad	Métrica	Ionic	jQuery Mobile
Adaptación de sintaxis	1,0	1	0
Funciona en Internet Explorer	1,0	1	1
Funciona en Safari	1,0	1	1
Funciona en Mozilla Firefox	1,0	1	1
Funciona en Google Chrome	1,0	1	1

Funciona en Opera	1,0	1	1
Funciona en navegadores web para móviles	1,0	1	1
Puntaje		7	6

Nota: Se evalúa según métrica (1 si cumple, 2 no cumple). Fuente: Matriz de calidad ISO 25010.

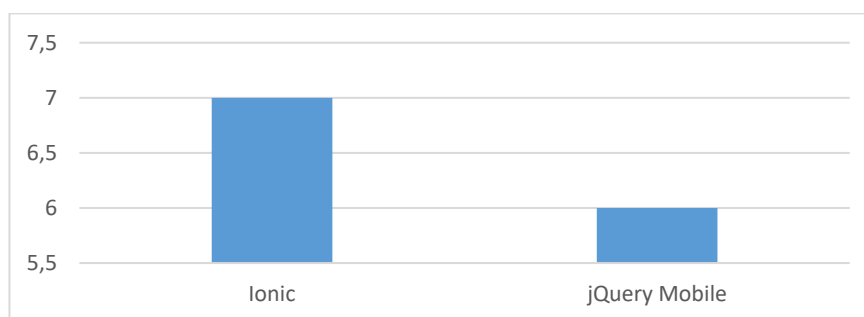


Figura 20. Puntaje del indicador de adaptabilidad para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 17 y figura 20, se observa que el framework Ionic está por encima de jQuery Mobile para el indicador de adaptabilidad, ya que es el que mejor permite que la aplicación desarrollada con este framework tenga mayores posibilidades de ser probada y utilizada en diferentes entornos, lo que permite a la aplicación llegar a más entornos para su funcionamiento.

b. Para el indicador de facilidad de instalación

La tabla 18 representa la facilidad que cuenta un framework para ser instalado y/o desinstalado de forma exitosa para el desarrollo de una aplicación.

Tabla 18.

Resultados obtenidos del indicador de facilidad de instalación.

Facilidad de instalación	Métrica	Ionic	jQuery Mobile
Manuales de instalación	1,0	1	1
Ayuda en línea	0,4	4	4
Tiempo de instalación	0,4	0	1

Nota: Se evalúa según la métrica (booleano y por rango). Fuente: Matriz de calidad ISO 25010.

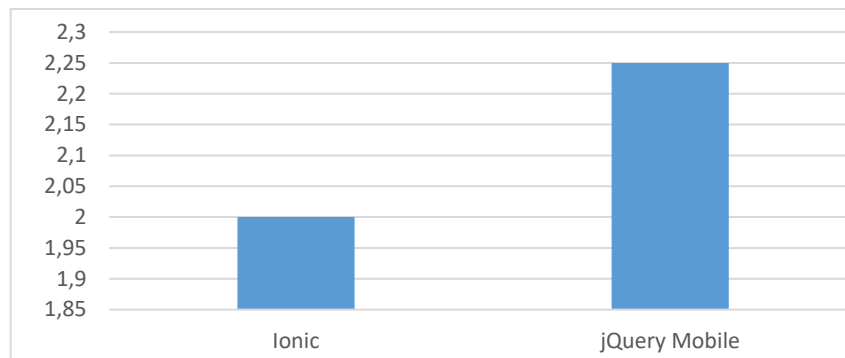


Figura 21. Puntaje del indicador de facilidad de instalación para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla N° 18 y grafico N° 17 observamos que jQuery Mobile tiene un mayor puntaje de evaluación en cuanto al indicador de facilidad de instalación, ya que este se descarga e instala a la aplicación de forma manual; en cambio el framework Ionic se descarga e instala a la aplicación mediante comandos de instalación, lo que hace que la descarga e instalación no tenga interacción directa con el usuario.

c. Para el indicador de capacidad de ser reemplazado

La tabla 19 representa la capacidad que tiene un framework para ser utilizado y/o reemplazado en lugar de otro, para obtener el mismo propósito y funcionando en el mismo entorno.

Tabla 19.

Resultados obtenidos del indicador de capacidad de ser reemplazado.

Capacidad de ser reemplazado	Métrica	Ionic	jQuery Mobile
Reemplazo de versiones (con compresión y sin compresión)	0,4	4	1

Cambio de versiones (actualización de Framework)	0,4	3	1
Reemplazo de Framework	0,4	4	4
Puntaje		2.75	1.50

Nota: Se evalúa según la métrica (por rango). Fuente: Matriz de calidad ISO 25010.

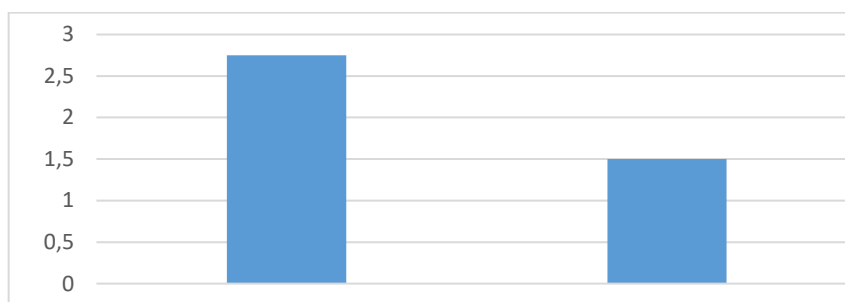


Figura 22. Puntaje del indicador de capacidad de ser reemplazado para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 19 y figura 22, podemos observar que el framework Ionic tiene un mayor puntaje para el indicador de capacidad de ser reemplazado, ya que este permite realizar actualizaciones del framework en la aplicación mediante la consola de comando CLI, realizando todo el trabajo el propio framework y el usuario no tiene mucha intervención. Por el contrario, jQuery Mobile no cuenta con estas características, ya que el usuario es el encargado de realizar la actualización manualmente, desde la descarga hasta la actualización/instalación de una nueva versión.

3.1.8. Resultados de la matriz de calidad

En la siguiente tabla se muestra el resumen de los resultados obtenidos de los frameworks expuestos en la matriz del modelo de calidad:

Tabla 20.

Resultados globales de cada uno de los frameworks de la matriz de calidad

características	Ionic	jQuery Mobile
-----------------	-------	---------------

Adecuación funcional	18	15
Rendimiento	10.75	10.72
Compatibilidad	8	8
Usabilidad	11	9.25
Fiabilidad	13.4	12.26
Seguridad	1	1
Mantenibilidad	5	5
Portabilidad	11.75	9.75
Total	78.9	70.98

Nota: Se realiza la suma de los resultados de los indicadores y se saca un promedio. Fuente: Elaboración propia.

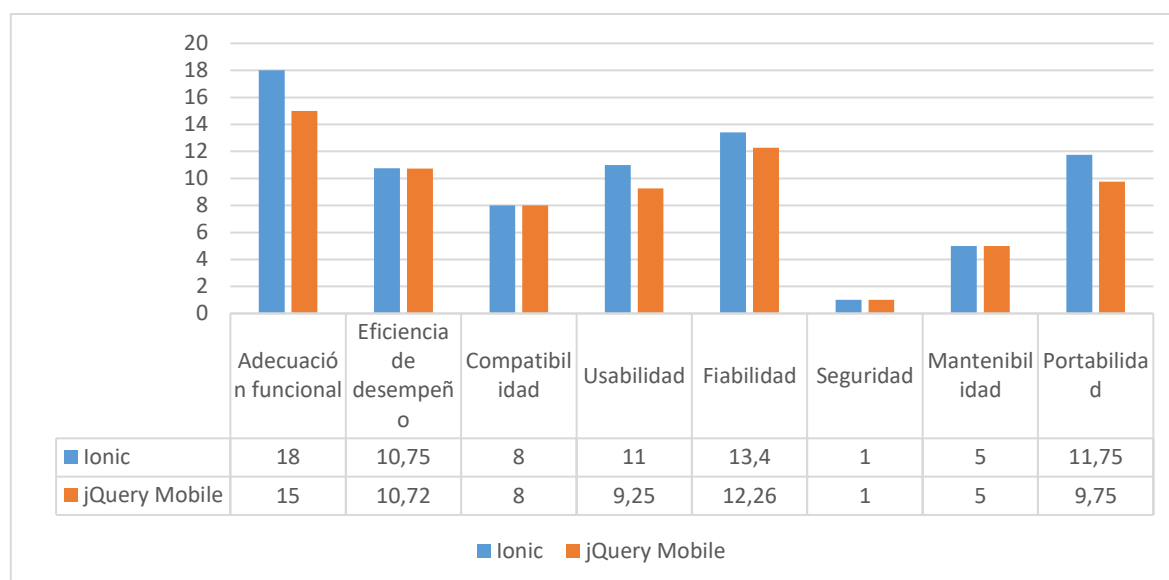


Figura 23. Puntaje del resultado global para cada framework. Fuente: Matriz de calidad ISO 25010.

En la tabla 20 y figura 23 se observa que el framework Ionic cumple con la mayoría de las características del modelo de evaluación ISO 25010, con una puntuación de 78.9, sobre 49.25 de jQuery mobile.

3.2. Discusión de resultados.

Como se puede observar en la tabla N° 20 y grafico N° 19 de los resultados de la matriz de calidad, el framework Ionic es el que mejor cumple con las

características del modelo de calidad, siendo así uno de los frameworks más populares y utilizados por la comunidad de desarrolladores.

Ionic framework permite desarrollar aplicaciones para el lado del cliente de manera rápida, lo que permite ahorrar tiempo de desarrollo, además que cuenta con una estructura de clases para la programación lo que permite tener un código ordenado y legible que permite su fácil entendimiento, modificación, implementación, entre otras cosas más.

Ionic Framework se basa en angular y trae consigo integrado Córdova, lo que nos va a permitir crear las aplicaciones para dispositivos móviles Android, sin la necesidad de realizar mucho esfuerzo ni tiempo, basta con ejecutar unos comandos desde la consola para generar nuestra aplicación para Android.

Así mismo jQuery Mobile no se queda tan atrás, ya que permite realizar muchas cosas con la potencia de jQuery que es el core para el correcto y potente funcionalidad de jQuery Mobile. No obstante, jQuery Mobile no cuenta con Apache Cordova (PhoneGap), si necesita generar una aplicación para Android se tendría que instalar Apache Cordova (PhoneGap) y así en conjunto con esta herramienta poder realizar lo que necesitamos.

3.3. Aporte práctico.

3.3.1. Seleccionar framework para su análisis

3.3.1.1. Framework web móviles existentes

Para empezar con el análisis de los frameworks web para móviles, se tiene que identificar cuales existen en el mercado, cuales son los más conocidos y usados por la comunidad de desarrolladores para seleccionarlos según algunos criterios que se establecerán.

Consultado en diferentes paginas hasta mayo de 2017, existen muchos frameworks web que permiten desarrollar aplicaciones para dispositivos móviles, en la siguiente tabla se muestran los frameworks que se encontraron consultando diferentes páginas web, las cuales hacen referencia a frameworks

web para móviles, se ordenaron según la cantidad de resultados de búsqueda en google.

Tabla 21.

Frameworks web móviles consultados en diferentes páginas web y cantidad de resultados en Google.

Framework	Búsqueda en google	Número de búsquedas
ionic	ionic framework	19400000
React Native	framework React Native	7710000
Native Script	framework Native Script	5620000
Phoneygap	Phoneygap	5230000
jquery mobile	jquery mobile	4750000
Agate Mobile Application	framework Agate Mobile Application	3110000
Mobile Angular UI	framework Mobile Angular UI	2790000
sencha tocuh	framework Sencha Touch 2	1550000
RhoMobile	framework RhoMobile	1020000
WebApp.Net	framework WebApp.Net	735000
Famo.us	framework Famo.us js	620000
Xamarin	framework Xamarin	560000
Junior	framework junior.js	550000
iWebKit	framework iWebKit	544000
Meteor	framework Meteor	513000
Rad.js	framework RAD.js	439000
LungoJS	framework LungoJS	432000
Framework7	Framework7	430000
kendo ui	framework Kendo UI	408000
Ratchet	framework Ratchet	403000
Handheld Designer	framework Handheld Designer	397000
Apache Flex	framework Apache Flex	393000
appMobi	framework appMobi	352000
Appcelerator Titanium	framework appcelerator titanium	344000

EMY	framework EMY	318000
TopCoat	framework TopCoat	294000
Marmalade	framework Marmalade	264000
Corona Labs	framework Corona Labs	208000
jQTouch	framework jQTouch	193000
Monaca	framework Monaca	127000
TouchStone JS	framework TouchStone JS	109000
qooxdoo	framework qooxdoo	89600
Enyo	framework Enyo	88900
Kivy	framework Kivy	84200
Moai	framework Moai	81200
Intel XDK	framework Intel XDK	78800
Mobify.js	framework Mobify.js	63400
ChocolateChip-UI	framework ChocolateChip-UI	58400
Gideros	framework Gideros	46600
NeoMAD	framework NeoMAD	44200
Onsen UI	framework Onsen UI	43500
Intel's App Framework	Intel's App Framework	31500
DHTMLX Touch	framework DHTMLX Touch	23500
Tabris.js	framework Tabris.js	15400
Fastshell	framework Fastshell	11700
NimbleKit	framework NimbleKit	6660
ViziApps	framework ViziApps	6310
AppGyver Steroids	framework AppGyver Steroids	4090
eMobic	framework eMobic	2410

Nota: Se listan según la cantidad de resultados al realizar una búsqueda. Fuente: Elaboración propia.

Además, se hizo una selección de los framework que más se mencionan en las diferentes páginas web, se seleccionó 20 páginas web en las que los siguientes frameworks fueron los más referenciados.

Tabla 22.

Paginas donde se mencionan los frameworks más utilizados o conocidos por la comunidad.

Página	Framework	Ionic	Jquery Mobile	Sencha Touch	Appcelerator Titanium	Framework7	Mobile Angular UI	Cordova/ PhoneGap	React Native	Kendo UI
http://codecondo.com/top-10-frameworks-for-mobile-development/		1	1	1				1		
https://iprodev.com/frameworks-build-mobile-application-html-css-javascript/		1	1	1	1	1	1	1		
http://www.insigniawm.com/mobile/10-best-hybrid-mobile-app-ui-frameworks-html5-css-and-js/		1		1		1	1		1	1
https://www.freelancinggig.com/blog/2017/02/01/top-10-hybrid-mobile-app-frameworks/		1	1	1		1	1	1		1
https://dzone.com/articles/10-leading-html5-frameworks		1		1	1	1	1			1
http://www.techpluto.com/5-best-frameworks-building-mobile-apps/		1						1	1	

http://sourcefreeze.com/best-mobile-ui-frameworks-using-html5-css-and-js/	1				1					1
http://www.cssauthor.com/html5-mobile-ui-framework/	1	1	1							1
http://www.126kr.com/article/3yeh25zi25u		1	1	1			1	1	1	
https://blog.jscrambler.com/10-frameworks-for-mobile-hybrid-apps/?utm_medium=referral&utm_source=news.js.org	1		1	1		1	1	1	1	1
https://www.singsys.com/blog/mobile-application-frameworks-to-build-with-html-css-javascript/	1	1	1	1		1	1	1		1
https://www.kickstarter.com/projects/1311831077/learn-5-best-mobile-development-frameworks/description	1	1						1	1	
http://stateofjs.com/2016/mobile/	1	1						1	1	
https://www.g2crowd.com/categories/mobile-development-frameworks	1	1		1		1		1	1	1
http://www.valuecoders.com/blog/technology-and-		1	1	1			1	1	1	

apps/top-javascript-frameworks-for-mobile-app-development/ http://www.pixelcrayons.com/blog/mobile/7-best-hybrid-app-development-frameworks-for-2017/	1	1			1			1	
http://blog.aulaformativa.com/framework-moviles-crear-aplicaciones-uso-html-css-javascript/ http://besttoppers.com/mobile-development-framework/	1	1	1		1	1	1		1
http://blogs.perceptionssystem.com/javascript-frameworks-for-mobile-app-development/	1	1	1	1	1		1	1	1
https://jaxenter.com/top-frameworks-for-mobile-app-development-130689.html	1	1					1		
Resultados	18	15	12	9	11	10	15	10	10

Nota: El resultado se obtuvo al realizar una búsqueda en diferentes sitios web y dentro de ellas se obtuvieron las más mencionadas. Fuente: Elaboración propia.

De la tabla anterior podemos concluir que los framework Ionic, jQuery Mobile, PhoneGap y Sencha touch son los más referenciados por las distintas páginas web que publicaron acerca de frameworks para el desarrollo de aplicaciones para dispositivos móviles utilizando tecnología web; también teniendo en cuenta la cantidad de búsquedas de Google dichos frameworks tienen altas cantidades de resultados en la búsqueda.

El framework Ionic ha sido utilizado para el desarrollo de una aplicación para consumir servicios de una API REST que brinda servicios para la venta de ropa. “*Desarrollo de una API REST con sus aplicaciones web y móvil para la venta de ropa online de la empresa Rossman*” (Santos Hernández & Serrano Parreño, 2017).

En la investigación “*Tendencias en desarrollo móvil bajo las tecnologías Android e IOS*” (Salazar Cardona, Angarrita, & Montoya, 2016), en la cual busca contextualizar las tendencias de desarrollo móvil para las plataformas más representativas del mercado, en el cual mencionan al desarrollo nativo DART, GO, Swift, mencionando como tendencias a frameworks como jQuery Mobile e Ionic.

La investigación “*Desarrollo de aplicaciones móviles multiplataforma*” (Lisandro Nahuel, 2017), en la cual realizan un análisis comparativo de diferentes enfoques que predominan y realizan el desarrollo de aplicaciones móviles, tales como, desarrollo nativo para Android e IOS, desarrollo con PhoneGap y jQuery Mobile, Sencha Touch, Appcelerator Titanium 3, Xamarin, Delphi 10 Seattle, en los cuales tiene como trabajo futuro profundizar en un estudio y análisis de herramientas como Ionic, NativeScript, React Native, entre otros.

Se trabajará con los frameworks Ionic y jQuery mobile, teniendo en cuenta la información de las tablas mostradas anteriormente y las investigaciones mencionadas anteriormente.

3.3.1.2. Evaluación según modelo QSOS

El modelo QSOS se utiliza para la calificación y selección de software de código abierto, fue creado para facilitar la evaluación de software open-source. QSOS identifica los siguientes criterios relativos:

- a. Requerimientos técnicos actuales y planificados.
- b. Requerimientos funcionales actuales y planificados.
- c. Sostenibilidad del software.
- d. Nivel de estabilidad del software.
- e. Gestión de los fallos de funcionamiento del software.
- f. Nivel de soporte disponible y previsto para el software
- g. Influencia en el desarrollo del software.
- h. Prospecciones funcionales del software evaluado.

El proceso general diseñado consta de cuatro etapas para la ejecución del modelo: Definición, Evaluación, Calificación, Selección. En la siguiente tabla se definen cada uno de los pasos.

Tabla 23.

Definición de etapas de QSOS.

Etapas	Descripción
1. Definición	Se constuye los marcos de referencia que se utilizarán en las siguientes etapas.
2. Evaluación	Se realiza la evaluación criterios siguientes: cobertura funcional, riesgos para el usuario y riesgos para el proveedor de servicios.
3. Calificación	Ponderación de criterios divididos en los tres ejes, según los requerimientos de los usuarios y/o estrategias establecidas por el proveedor de servicios.
4. Selección	Se aplica el filtro que se creó en la Etapa 3, en datos proporcionados por las etapas 1 y 2, para continuar las consultas, comparaciones y selecciones de productos.

Nota: Etapas del metodo QSOS. Fuente: (Ramos & Páez, 2011).

Para esta investigación no se tomarán en cuenta los criterios de clasificación y selección, debido a que estos se basan en la experiencia y necesidades del usuario. Solo se realizará un análisis de evaluación de los frameworks que se están estudiando.

3.3.1.2.1. Definición

Punto de referencia para los siguiente tres pasos, define varios elementos para ser usados por el resto del proceso.

Los marcos de referencia son:

- a. Familias de software:** para este caso la familia de software se define como: “Frameworks web para el desarrollo de aplicaciones móviles”.
- b. Tipos de licencia:** Existen muchos tipos de licencias utilizadas para el software libre y de código abierto. Las características elegidas para describir los tipos de licencia son:
 - i. Licencia propietaria:** define si el código debe ser privado o permanecer libre.
 - ii. Viralidad de las licencias:** analiza si el código fuente del producto está ligado con otros módulos, con la finalidad de que permanezcan bajo la misma licencia.
 - iii. Herencia de la licencia:** El código derivado hereda inevitablemente la misma licencia, pudiéndosele aplicar restricciones adicionales.

Tabla 24.

Licenciamiento Frameworks.

Framework	Licencia	Propietaria	Viralidad	Herencia
Ionic	MIT	Si	No	No
jQuery Mobile	MIT	Si	No	No

Nota: Características elegidas para describir los tipos de licencia. Fuente: (Ramos & Páez, 2011).

- c. Tipos de comunidades:** los tipos de comunidades identificadas de software libre se clasifican de la siguiente manera:

- i. **Desarrollador aislado:** una misma persona desarrolla y gestiona el software.
- ii. **Grupo de desarrolladores:** varias personas colaboran al desarrollo de software de manera informal.
- iii. **Organización de desarrolladores:** se gestiona el ciclo de vida del software de una manera formal en un determinado grupo de desarrolladores, basados en la asignación de funciones (desarrollador, tester, administrador de entrega, etc) y la meritocracia.
- iv. **Entidad legal:** una persona jurídica sin fines de lucro que dirige la comunidad, este posee derechos de autor y también gestiona el patrocinio y subvenciones vinculadas.
- v. **Entidad comercial:** remunera a los desarrolladores principales del proyecto mediante la venta de servicios o de versiones comerciales del software.

Los frameworks estudiados están en el último tipo de comunidad, ya que Ionic es desarrollada por Drifty Co, que es una entidad comercial; y jQuery Mobile es desarrollada por jQuery Foundation, que también es una entidad comercial.

3.3.1.2.2. Evaluación

La evaluación del framework se realiza en esta etapa, utilizando información de la comunidad de código abierto cuyo objetivo es:

a. Generar la tarjeta de identificación del software

Constituye datos directivos y objetivos relevantes en el proceso de registro.

Una tarjeta de identificación consta de lo siguiente:

Información General

- Nombre del software.
- Referencia, fecha de creación, fecha de publicación de la tarjeta de identificación.
- Autor
- Tipo de software
- Breve descripción del software

- Licencias a las cuales se encuentra sujeto el software
- URL del proyecto y sitio de demostración
- Sistemas operativos compatibles
- Origen
- Servicios existentes
- Documentación
- Cursos de formación
- Ofertas de consultoría
- Aspectos funcionales y técnicos
- Tecnologías de implementación
- Requisitos técnicos

b. Generar la hoja de evaluación del software

Califica al software con un puntaje de 0, 1 o 2 según el cumplimiento del criterio.

La hoja de evaluación del software tiene cinco criterios fundamentales:

- i. **Durabilidad intrínseca:** Edad y estabilidad del software.
- ii. **Industrialización del desarrollo:** Popularidad del software y tamaño de la comunidad de desarrolladores y usabilidad.
- iii. **Disponibilidad de plataformas:** Señala el soporte que tiene para los diferentes sistemas operativos.
- iv. **Adaptabilidad técnica:** indica la complejidad de ampliar el software.
- v. **Estrategia:** muestra la licencia y el marco formal para el desarrollo del software.

Cobertura funcional

Es determinado por la familia del software y el producto en el marco de referencia de la Etapa 1 (Definición). La norma de puntuación se muestra en la siguiente tabla.

Tabla 25.

Norma de puntuación

Funcionalidad	Puntuación
No cubierta	0
Parcialmente cubierta	1
Completamente cubierta	2

Nota: Valores para la puntuación de los indicadores. Fuente: (Ramos & Páez, 2011).

3.3.1.3. Resultados evaluación y análisis.

La evaluación se realizó según los criterios mencionados en el anexo 1, obteniendo las siguientes tablas de resultados.

Tabla 26.

Resultados de Durabilidad intrínseca de cada framework.

	Durabilidad intrínseca	Puntuación del Framework	
		Ionic	jQuery Mobile
Madurez	Edad	2	2
	Estabilidad	2	2
	Historial de problemas	2	2
	Probabilidad de bifurcación	2	2
	Popularidad	2	2
	Referencias	2	2
Adopción	Calidad de la comunidad	2	1
	Libros	2	2
	Manuales y tutoriales	2	2
Estilo de liderazgo	Tamaño del equipo de desarrollo	2	1
	Estilo de la administración del equipo	2	2

	Numero de desarrolladores	2	1
	Actividad en errores	2	2
Actividad	Actividad en funcionalidad	2	2
	Actividad en liberaciones de nuevas versiones	1	1
Totales		1.9	1.7

Nota: sumatoria de los resultados de los indicadores. Fuente: Elaboración propia

Tabla 27.

Resultados de Solución industrializada de cada framework.

Solución industrializada		Puntuación del Framework	
		Ionic	jQuery Mobile
	Formación	2	2
Servicios	Soporte	2	2
	Consultoría	1	1
Documentación	Documentación	2	2
Método de calidad	Aseguramiento de calidad	1	1
	Herramientas	1	1
Totales		1.5	1.5

Nota: Puntaje para solución industrializada. Fuente: Elaboración propia

Tabla 28.

Resultados de disponibilidad de plataformas de cada framework.

Disponibilidad de plataformas		Puntuación del Framework	
		Ionic	jQuery Mobile
Paquetes	Windows	2	2
	Debían/Ubuntu	2	2

	Red Hat/Fedora	2	2
	Suse	2	2
	Mandriva	2	2
	Mac OS X	2	2
	Solaris	2	2
Totales		2	2

Nota: Puntaje para disponibilidad de plataformas. Fuente: Elaboración propia

Tabla 29.

Resultados de adaptabilidad técnica de cada framework.

Adaptabilidad técnica		Puntuación del Framework	
		Ionic	jQuery Mobile
	Modularidad	2	2
Modularidad	Facilidad de extensión de código	2	2
Totales		2	2

Nota: Puntaje para adaptabilidad técnica. Fuente: Elaboración propia.

Tabla 30.

Resultados de estrategia de cada framework.

Estrategia		Puntuación del Framework	
		Ionic	jQuery Mobile
	Permisividad	2	2
Licencia	Protección contra extensiones del framework	0	0
Copyright	Derechos de Copyright	2	2
Modificación del código fuente	Modificación del código fuente	2	2

RoadMap	RoadMap (mapa de versiones y liberaciones)	2	2
Patrocinador	Patrocinador	2	2
Independencia estratégica	Independencia estratégica	2	2
Totales		1.7	1.7

Nota: Puntaje para la de estrategia del framework en el mercado. Fuente: Elaboración propia.

Tabla 31.

Resultados de industrialización de desarrollo de cada framework.

Industrialización de desarrollo		Puntuación del Framework	
		Ionic	jQuery Mobile
	IDE	2	2
	Herramientas de construcción	2	2
Herramientas	Herramientas de pruebas	1	1
	Herramientas gráficas para las interfaces	2	2
Totales		1.8	1.8

Nota: Puntaje del framework para el indicador de industrialización de desarrollo. Fuente: Elaboración propia.

En la siguiente tabla se obtiene la tabulación de los resultados. Además, también lo podemos observar gráficamente.

Tabla 32.

Resumen de resultados QSOS de cada framework.

	Framework	
	Ionic	jQuery Mobile

Durabilidad intrínseca	1.9	1.7
Solución industrializada	1.5	1.5
Disponibilidad de plataformas	2	2
Adaptabilidad técnica	2	2
Estrategia	1.7	1.7
Industrialización del desarrollo	1.8	1.8

Nota: Promedio de resultados al aplicar el método QSOS en cada framework.

Fuente: Elaboración propia.

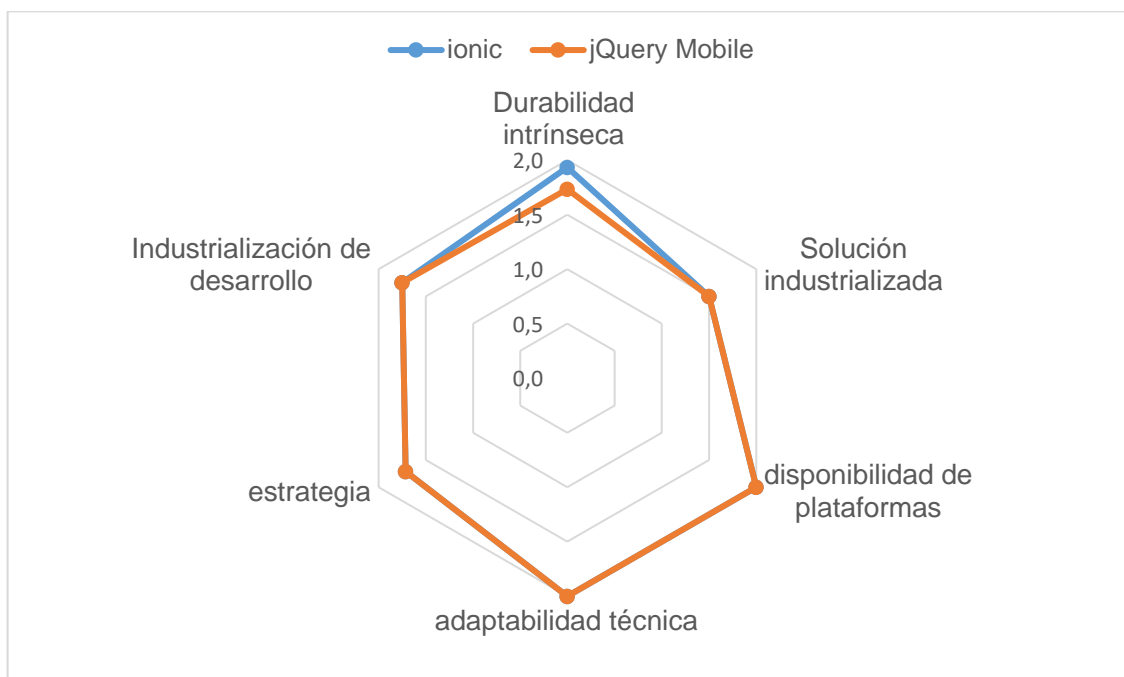


Figura 24. Resultado de la evaluación de los frameworks. Fuente: Elaboración propia.

En la figura anterior se observa como los frameworks en algunos criterios tienen valores similares y en otro los valores son diferentes, con esta información se realiza el análisis siguiente:

- a. **Durabilidad intrínseca:** se observa que el framework ionic tiene más posibilidad de permanecer en el mercado que jQuery mobile.
- b. **Industrialización del desarrollo:** existe un empate ya que ambos no cuentan con un IDE de desarrollo propio.

- c. Estrategia:** ambos framework tienen un empate en cuanto a una estrategia similar por pertenecer a la misma entidad.
- d. Solución industrializada:** ambos frameworks cuentan con documentación, libros, cursos de formación que los respaldan.
- e. Disponibilidad de plataformas:** los frameworks al ser implementados por html, Css, JavaScript, que son contenido para la web, están disponibles para cualquier plataforma.
- f. Adaptabilidad técnica:** observamos un empate entre los frameworks ya que están compuestos de módulos bien definidos.

3.3.2. Seleccionar modelo de evaluación para realizar la comparación entre frameworks

El modelo de evaluación será ISO/IEC 20510, que nos permitirá crear el modelo de calidad y evaluación de los frameworks gracias a sus características y subcaracterísticas, así mismo siguiendo los lineamientos de la metodología IQMC que determinará varios atributos medibles para la calidad consiguiendo así un análisis más detallado y específico. El modelo será una base de medición flexible para evaluar los frameworks Ionic y jQuery Mobile.

3.3.2.1. Construcción del modelo de evaluación de calidad.

3.3.2.1.1. Aplicación del método IQMC

El método IQMC propone las siguientes etapas:

i. Paso 0: Estudio del ámbito del software

Los componentes, funcionalidades y servicios fundamentales que un framework Front-End debe proporcionar son los siguientes:

Compatibilidad: Todos los frameworks deben tener componentes que puedan ser interpretados por los navegadores más usados en la actualidad (Google Chrome, Mozilla Firefox, Internet Explorer, Safari y Opera) y también por la mayoría de dispositivos android.

Comunicación asíncrona (Ajax): Usando esta técnica de programación, es fácil para el desarrollador usar XMLHttpRequest para gestionar y manipular datos sin

la necesidad de recargar las vistas, aumentando la interactividad y experiencia del usuario.

Manipulación y recorrido del DOM: Las vistas creadas solo con HTML carecen de dinamismo, por lo tanto, brindan interacción al usuario final, JavaScript es un lenguaje dinámico que posee métodos para acceder, recorrer y manipular el DOM.

A medida que las aplicaciones carecen, el código JavaScript se hace difícil de mantener, entorpeciendo la reutilización y la manipulación de elementos de páginas web, debido a esto, los frameworks son creados para aportar métodos que simplifican líneas de código, brindando al usuario la facilidad de agregar, editar, cambiar y eliminar elementos de una página HTML de manera simple y dinámica.

Validación de formularios: Los frameworks Front-End permiten de una manera fácil validar campos dentro de un formulario. Esto al desarrollador le permite simplificar y reducir código para procesar dicho formulario, ya que los datos llegan previamente validados, reduciendo los errores de tipos de datos.

Efectos visuales: La idea de emplear efectos visuales en una aplicación móvil, es que la experiencia de navegación del usuario final sea fácil, amigable e intuitiva. El aporte de los frameworks web para móviles se destaca en la facilidad que brindan a los desarrolladores para implementar efectos y animaciones, cuanto menos se demore, mayor es el aporte de la herramienta.

Manejo de eventos: Los eventos son uno de los elementos más importantes en el desarrollo de aplicaciones interactivas para el usuario final, ya que sirven para realizar acciones en las vistas, acciones que están disponibles y que el usuario necesita. Los eventos son la base para la interacción con el usuario final.

Soporte: El equipo responsable de un framework es el encargado de dar mantenimiento, publicar nuevas actualizaciones, corregir errores y proporcionar ayuda a través de foros y otros medios. Si el framework no tiene un equipo

organizado, responsable y comprometido que garantice calidad en el producto, no es recomendable para una empresa invertir recursos para el aprendizaje del mismo.

Documentación: La documentación de los frameworks o cualquier software en general es importante, ya que ayuda a identificar fácilmente los aspectos y características que forman parte de los mismos. Una documentación legible, organizada y actualizada proporciona identidad y popularidad a un framework.

ii. Paso 1: Determinación de características de calidad

Las características de calidad se toman de la norma ISO/IEC 25010, las cuales se nombran en la siguiente tabla.

Tabla 33.

Características de calidad.

Características de calidad		
N°	Nombre	Descripción
1	Adecuación funcional	En un framework para el desarrollo de aplicaciones móviles, representa las funcionalidades que este brinda al usuario para desarrollar aplicaciones móviles interactivas y amigables.
2	Eficiencia de desempeño	Representa los recursos como el tiempo y el número de funciones que un framework utiliza bajo ciertas condiciones.
3	Compatibilidad	Es la capacidad que un framework tiene para llevar a cabo sus funciones sin importar el navegador web o dispositivo en el que se esté ejecutando.
4	Usabilidad	Este factor evalúa la facilidad con que los usuarios pueden utilizar los diferentes métodos que posee un framework con el fin de alcanzar un objetivo concreto.
6	Fiabilidad	Permite determinar la probabilidad de que un framework desarrolle una cierta función, durante un

		tiempo establecido y bajo un conjunto de condiciones definidas.
6	Seguridad	Evalua si un framework posee componentes para la protección de datos en el desarrollo de aplicaciones.
7	Mantenibilidad	Permite determinar la capacidad que tiene un framework para ser modificado efectivamente y eficientemente, para mejora continua del mismo.
8	Portabilidad	Determina la facilidad que posee un framework para ser instalado, adaptado o reemplazado de forma efectiva y eficiente en diferentes ambientes.

Nota: Descripción de las características de calidad. Fuente: ISO 25010.

iii. Paso 2: refinamiento de la jerarquía de subcaracterísticas de calidad.

Tabla 34.

Subcaracterísticas de Adecuación funcional.

Característica 1		Adecuación funcional
Subcaracterísticas		
N°	Nombre	Descripción
1.1	Complejidad funcional	Grado en el cual el conjunto de funcionalidades que posee un framework cubre todas las tareas y los objetivos planteados por el usuario
1.2	Corrección funcional	Capacidad del framework para proporcionar los resultados o efectos correctos y con el grado de precisión acordado.
1.3	Pertinencia funcional	El framework posee funcionalidades para la comunicación con el servidor, gestión de efectos y eventos, manipulación y recorrido del DOM, validación de formulación.

Nota: Descripción de las subcaracterísticas de adecuación funcional. Fuente: ISO 25010.

Tabla 35.

Subcaracterísticas de eficiencia de desempeño.

Característica 2		Eficiencia de desempeño
Subcaracterísticas		
N°	Nombre	Descripción
2.1	Comportamiento en el tiempo	Capacidad de un framework para proporcionar tiempos, procesamiento e índices de respuesta al realizar una funcionalidad específica bajo ciertas condiciones
2.2	Utilización de recursos	Determina las cantidades y tipos de recursos que utiliza un framework cuando ejecuta una función bajo condiciones específicas.

Nota: Descripción de las subcaracterísticas de eficiencia de desempeño. Fuente: ISO 25010.

Tabla 36.

Subcaracterísticas de Compatibilidad.

Característica 3		Compatibilidad
Subcaracterísticas		
N°	Nombre	Descripción
3.1	Coexistencia	Capacidad de un framework para coexistir con otro, compartiendo recursos comunes.
3.2	Interoperabilidad	Capacidad de dos o más frameworks para intercambiar y utilizar la información.

Nota: Descripción de las subcaracterísticas de compatibilidad. Fuente: ISO 25010.

Tabla 37.

Subcaracterísticas de Usabilidad.

Característica 4		Usabilidad
Subcaracterísticas		
N°	Nombre	Descripción

4.1	Capacidad para reconocer su adecuación	Capacidad de los proveedores del framework para permitir al usuario identificar si sus funcionalidades son adecuadas para sus necesidades.
4.2	Capacidad de aprendizaje	Capacidad de la comunidad del framework para permitirle al usuario aprender los diferentes atributos, métodos y componentes que posee.
4.3	Capacidad para ser usado	Capacidad del framework para ser tratado y controlado con facilidad por el usuario.
4.4	Protección frente a errores de usuario	Protección de errores de usuario: capacidad del sistema para proteger a los usuarios de hacer errores. Rendimiento: No aplica Motivo: Los frameworks Front-End no aportan facilidades para detectar y proteger a los usuarios desarrolladores de sus errores.
4.5	Accesibilidad	Capacidad del framework para ser empleado por usuarios con determinadas capacidades. Refinamiento: No aplica. Motivo: Los frameworks Front-End no poseen funcionalidades o facilidades de esta categoría.

Nota: Descripción de las subcaracterísticas de usabilidad. Fuente: ISO 25010

Tabla 38.

Subcaracterísticas de Fiabilidad.

Característica 5		Fiabilidad
Subcaracterísticas		
N°	Nombre	Descripción
5.1	Madurez	Capacidad del framework para prevenir errores en el producto.
5.2	Disponibilidad	Determina si un framework tiene capacidades de accesibilidad y uso
5.3	Tolerancia a fallos	Capacidad de un framework para ejecutarse en presencia de fallos de software. Refinamiento: No aplica.

Motivo: Los frameworks Front-End no poseen funcionalidades de esta categoría.

5.4 Capacidad de recuperación Capacidad del framework en caso de interrupción recuperar datos afectados y restablecerlos al estado deseado de la ejecución.

Refinamiento: No aplica.

Motivo: Los framework Front-End no poseen funcionalidades para realizar las tareas mencionadas.

Nota: Descripción de las subcaracterísticas de fiabilidad. Fuente: ISO 25010.

Tabla 39.

Subcaracterísticas de seguridad.

Característica 6		Seguridad
Subcaracterísticas		
N°	Nombre	Descripción
6.1	Confidencialidad	Capacidad que tiene el framework para proteger el acceso de datos e información no autorizados accidental o deliberadamente.
6.2	Integridad	Capacidad del framework para prevenir accesos o modificaciones no autorizados a datos o programas.
6.3	No repudio	Capacidad del framework para demostrar que las acciones o eventos no puedan ser repudiadas posteriormente.
6.4	Autenticidad	Capacidad del framework para demostrar la identidad de un sujeto o un recurso.
6.5	Responsabilidad	Capacidad del framework para rastrear de manera inequívoca las acciones de una entidad.

Nota: Descripción de las subcaracterísticas de seguridad. Fuente: ISO 25010.

Tabla 40.

Subcaracterísticas de Mantenibilidad.

Característica 7	Mantenibilidad
-------------------------	-----------------------

Subcaracterísticas

N°	Nombre	Descripción
7.1	Modularidad	Capacidad del framework para permitir que un cambio tenga un impacto mínimo en los demás componentes.
7.2	Reusabilidad	Capacidad que tiene el framework para permitir que un activo sea reutilizado en más de un sistema software o en la construcción de otros activos.
7.3	Capacidad de ser modificado	Capacidad del framework para permitir ser modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.

Nota: Descripción de las subcaracterísticas de mantenibilidad. *Fuente:* ISO 25010.

Tabla 41.

Subcaracterísticas de Portabilidad.

Característica 8		Portabilidad
Subcaracterísticas		
N°	Nombre	Descripción
8.1	Adaptabilidad	Capacidad del framework para ser adaptado de forma efectiva y eficiente a diferentes navegadores web o dispositivos.
8.2	Facilidad de instalación	Facilidad con la que un framework se puede instalar y/o desinstalar de forma exitosa en una aplicación.
8.3	Capacidad de ser reemplazado	Capacidad de un framework para ser utilizado en lugar de otro, con el mismo propósito y en el mismo navegador web o dispositivo.

Nota: Descripción de las subcaracterísticas de portabilidad. *Fuente:* ISO 25010.

iv. Paso 3: Refinamiento de subcaracterísticas en atributos

Tabla 42.

Atributos de completitud funcional.

Característica 1	Adecuación funcional
Subcaracterística 1.1	Completitud funcional

Atributos

N°	Nombre	Descripción
1.1.1	Creación de aplicaciones interactivas	Permite la creación de aplicaciones donde el usuario final tiene una amplia capacidad de control sobre las mismas.
1.1.2	Creación de aplicaciones amigables	Permite la creación de aplicaciones visualmente atractivas y fáciles de usar
1.1.3	Creación de aplicaciones robustas	Posee funcionalidades para desarrollar aplicaciones robustas (estables, seguras, alto nivel de procesamiento, etc.)
1.1.4	Creación de aplicaciones orientadas a móviles	Permite la creación de interfaces diseñadas para que las aplicaciones web puedan ser accedidas desde un dispositivo móvil.

Nota: Descripción de los atributos de completitud funcional. Fuente: ISO 25010.

Tabla 43.

Atributos de corrección funcional.

Característica 1		Adecuación funcional
Subcaracterística 1.2		Corrección funcional
Atributos		
N°	Nombre	Descripción
1.2.1	Efectividad en la creación de efectos y animaciones	Posee métodos para monitorear y verificar que los resultados de los efectos y animaciones de una aplicación sean correctos
1.2.2	Efectividad en la manipulación del DOM	Proporciona los suficientes mecanismos para manipular el DOM de manera efectiva

Nota: Descripción de los atributos de corrección funcional. Fuente: ISO 25010.

Tabla 44.

Atributos de pertinencia funcional.

Característica 1		Adecuación funcional
Subcaracterística 1.3		Pertinencia Funcional
Atributos		
N°	Nombre	Descripción
1.3.1	Método Ajax	Posee el mecanismo Ajax para realizar una comunicación Cliente – Servidor sin recargar la vista.
1.3.2	Métodos para manipular el DOM	Proporciona funcionalidades para gestionar los elementos de un documento HTML XHTML.
1.3.3	Elementos interactivos	Proporciona un conjunto de elementos interactivos configurables, listos para ser utilizados en las vistas.
1.3.4	Validación de formularios	Posee librerías, componentes y funcionalidades específicas para la validación de formularios.
1.3.5	Efectos visuales	Posee un conjunto establecido de efectos y la posibilidad de crear nuevos, con propiedades personalizadas por el desarrollador.
1.3.6	Gestión de tablas o Grids	Posee librerías y componentes que ayudan a gestionar datos en una tabla.

Nota: Descripción de los atributos de pertinencia funcional. Fuente: ISO 25010.

Tabla 45.

Atributos de comportamiento en el tiempo.

Característica 2		Eficiencia de desempeño
Subcaracterística 2.1		Comportamiento en el tiempo
Atributos		
N°	Nombre	Descripción

2.1.1	Configuración del tiempo de espera en una comunicación Cliente – Servidor	Posee métodos para configurar el tiempo de espera de una solicitud realizada desde el cliente hacia el servidor.
2.1.2	Comunicación síncrona y asíncrona	Gestiona el tipo de petición (síncrona o asíncrona) que se realiza desde el cliente hasta el servidor, modificando el tiempo de respuesta en una aplicación.
2.1.3	Tiempo de carga en prueba local	Tiempo que se demora en recibir la respuesta del servidor.

Nota: Descripción de los atributos de comportamiento en el tiempo. Fuente: ISO 25010.

Tabla 46.

Atributos de utilización de recursos.

Característica 2		Eficiencia de desempeño
Subcaracterística 2.2		Utilización de recursos
Atributos		
N°	Nombre	Descripción
2.2.1	Necesita librerías	Si el framework necesita alguna librería para satisfacer las necesidades de un desarrollador.
2.2.2	JavaScript	Si el framework trabaja bajo javascript para su funcionamiento
2.2.3	Incorporación de AJAX	Si el framework incorpora Ajax para satisfacer las necesidades del desarrollador
2.2.4	Iconos propios	Si el framework incorpora iconos para satisfacer las necesidades del desarrollador.

Nota: Descripción de los atributos de utilización de recursos. Fuente: ISO 25010.

Tabla 47.

Atributos de coexistencia.

Característica 3	Compatibilidad
-------------------------	-----------------------

Subcaracterística 3.1 Coexistencia**Atributos**

N°	Nombre	Descripción
3.1.1	Coexistencia con HTML5	Capacidad del framework para coexistir con HTML5
3.1.2	Coexistencia con CSS3	Capacidad del framework para coexistir con CSS3
3.1.3	Coexistencia con JavaScript	Capacidad del framework para coexistir con JavaScript

Nota: Descripción de los atributos de coexistencia. Fuente: ISO 25010.

Tabla 48.

Atributos de interoperabilidad.

Característica 3		Compatibilidad
Subcaracterística 3.2		interoperabilidad
Atributos		
N°	Nombre	Descripción
3.2.1	Intercambio de información con PHP	Capacidad del framework para intercambiar información con PHP
3.2.2	Intercambio de información con Java EE	Capacidad del framework para intercambiar información con Java EE.
3.2.3	Intercambio de información con C# (ASP.net)	Capacidad del framework para intercambiar información con C#.
3.2.4	Intercambios de información con Python.	Capacidad del framework para intercambiar información con Python.
3.2.5	Intercambios de información con Ruby.	Capacidad del framework para intercambiar información con Ruby.

Nota: Descripción de los atributos de interoperabilidad. Fuente: ISO 25010.

Tabla 49.

Atributos de Capacidad para reconocer su adecuación.

Característica 4		Usabilidad
Subcaracterística 4.1		Capacidad para reconocer su adecuación
Atributos		
N°	Nombre	Descripción
4.1.1	Información real en sitio web	Claridad en la definición del framework expuesta en el sitio web y de las funcionalidades integradas en el mismo.
4.1.2	Actualización de contenido	Actualización del contenido del framework si sale una nueva versión.

Nota: Descripción de los atributos de capacidad para reconocer su adecuación.

Fuente: ISO 25010.

Tabla 50.

Atributos de capacidad de aprendizaje.

Característica 4		Usabilidad
Subcaracterística 4.2		capacidad de aprendizaje
Atributos		
N°	Nombre	Descripción
4.2.1	Manual técnico	Disponibilidad de manuales técnicos
4.2.2	Foros	Disponibilidad de foros para el soporte a usuarios
4.2.3	Tutoriales	Disponibilidad de tutoriales oficiales y no oficiales
4.2.4	Cursos on-line	Disponibilidad de cursos on-line oficiales y no oficiales.
4.2.5	Estudios e investigaciones	Se han realizado publicaciones técnicas de las capacidades de la herramienta.
4.2.6	Documentación	Disponibilidad de la documentación del framework.

Nota: Descripción de los atributos de capacidad de aprendizaje. Fuente: ISO 25010.

Tabla 51.

Atributos de Capacidad para ser usado.

Característica 4		Usabilidad
Subcaracterística 4.3		Capacidad para ser usado
Atributos		
N°	Nombre	Descripción
4.3.1	POO basada en clases	Se basa en la programación orientada a objetos en clases para el desarrollo de aplicaciones.
4.3.2	Reconocimiento de entorno	Posee funcionalidades que identifican y advierten si un navegador o dispositivo no soporta la versión del framework.
4.3.3	Licencia	Posee alguna licencia de distribución.

Nota: Descripción de los atributos de capacidad para ser usado. Fuente: ISO 25010.

Tabla 52.

Atributos de protección contra errores de usuario.

Característica 4		Usabilidad
Subcaracterística 4.4		Protección contra errores de usuario
Atributos		
N°	Nombre	Descripción
4.4.1	Modificación por el usuario	Permite que el usuario pueda relizar alguna acción que muestre el error (mensajes de alerta)

Nota: Descripción de los atributos de protección contra errores de usuario. Fuente: ISO 25010.

Tabla 53.

Atributos de accesibilidad.

Característica 4		Usabilidad
-------------------------	--	-------------------

Subcaracterística 4.6 Accesibilidad (No Aplica)**Atributos (No Aplica)**

Nota: No aplica atributos esta subcaracterística para evaluación de los frameworks. Fuente: ISO 25010.

Tabla 54.

Atributos de Madurez.

Característica 5		Fiabilidad
Subcaracterística 5.1		Madurez
Atributos		
N°	Nombre	Descripción
5.1.1	Tiempo en el mercado	Tiempo que tiene el framework en el mercado y su historial
5.1.2	Actualizaciones disponibles	Disponibilidad de actualizaciones al año del producto.
5.1.3	Posee una base del conocimiento software	Mantiene una base del conocimiento de los errores detectados y las soluciones óptimas.
5.1.4	Popularidad en GitHub	Cuenta con un gran número de estrellas en GitHub.
5.1.5	Aplicaciones realizadas	Se ha utilizado el framework para desarrollar aplicaciones.

Nota: Descripción de los atributos de madurez. Fuente: ISO 25010.

Tabla 55.

Atributos de Disponibilidad.

Característica 5		Fiabilidad
Subcaracterística 5.2		Disponibilidad
Atributos		
N°	Nombre	Descripción
5.2.1	Descarga del framework	Posibilidad de descargar el framework para usarlo de forma local.

5.2.2	Referenciar el framework	Posibilidad de referencias el framework para usarlo de forma remota.
5.2.3	Repositorio en GitHub	Cuenta con su propio repositorio en GitHub

Nota: Descripción de los atributos de disponibilidad. Fuente: ISO 25010.

Tabla 56.

Atributos de tolerancia a fallos.

Característica 5		Fiabilidad
Subcaracterística 5.3		Tolerancia a fallos
Atributos		
N°	Nombre	Descripción
5.3.1	Permite acceder a la información con fallos en el sistema	Si el framework da la opción de acceder al registro de fallos en el sistema.
5.3.2	Realiza respaldos	Si el framework realiza respaldos.

Nota: Descripción de los atributos de tolerancia a fallos. Fuente: ISO 25010.

Tabla 57.

Atributos de capacidad de recuperación.

Característica 5		Fiabilidad
Subcaracterística 5.4		Capacidad de recuperación
Atributos		
N°	Nombre	Descripción
5.4.1	Capacidad de recuperar información	Si el framework permite recuperar información al presentarse alguna pérdida de esta.

Nota: Descripción de los atributos de capacidad de recuperación. Fuente: ISO 25010

Tabla 58.

Atributos de Confidencialidad.

Característica 6		Seguridad
-------------------------	--	------------------

Subcaracterística 6.1 Confidencialidad

Atributos

N°	Nombre	Descripción
6.1.1	Protección de datos	El framework permite proteger los datos
6.1.2	Acceso solo a usuarios especificados	Permite restringir el acceso solo a usuarios específicos
6.1.3	Es seguro frente a internet	Brinda alguna manera de navegación segura.
6.1.4	Cifrado en la información	Permite cifrar la información al navegar por internet.

Nota: Descripción de los atributos de confidencialidad. Fuente: ISO 25010.

Tabla 59.

Atributos de Integridad.

Característica 6		Seguridad
Subcaracterística 6.2		Integridad
Atributos		
N°	Nombre	Descripción
6.2.1	Información correcta en la base de datos	El framework permite mantener una correcta información en la base de datos
6.2.2	No permite modificación de datos	El framework no permite la modificación de datos

Nota: Descripción de los atributos de integridad. Fuente: ISO 25010.

Tabla 60.

Atributos de No repudio.

Característica 6		Seguridad
Subcaracterística 6.3		No repudio
Atributos		
N°	Nombre	Descripción
6.3.1	Comunicación confiable cliente-servidor	El framework permite una comunicación confiable cliente-servidor

Nota: Descripción de los atributos de no repudio. Fuente: ISO 25010.

Tabla 61.

Atributos de Autenticidad.

Característica 6		Seguridad
Subcaracterística 6.4		Autenticidad
Atributos		
N°	Nombre	Descripción
6.4.1	Evita la suplantación de identidad	El framework evita la suplantación de identidad de usuario.
6.4.2	Autenticidad generada por el usuario	El framework permite que el usuario pueda generar una autenticación.

Nota: Descripción de los atributos de autenticidad. Fuente: ISO 25010.

Tabla 62.

Atributos de Respnsabilidad.

Característica 6		Seguridad
Subcaracterística 6.5		Responsabilidad
Atributos		
N°	Nombre	Descripción
6.5.1	Se hace responsable de la seguridad	El framework es responsable de la seguridad de la informacion.

Nota: Descripción de los atributos de responsabilidad. Fuente: ISO 25010.

Tabla 63.

Atributos de Modularidad.

Característica 7		Mantenibilidad
Subcaracterística 7.1		Modularidad
Atributos		
N°	Nombre	Descripción

7.1.1	Es modular	Capacidad del framework para ser estudiado, entendido y que sus partes interactúan entre sí para cumplir su función.
7.1.2	Crear componentes	Capacidad del framework para permitir crear componentes con poco esfuerzo.
7.1.3	Eliminar componentes	Capacidad del framework para eliminar componentes no requeridos.
7.1.4	Modificar componentes	Capacidad del framework para permitir modificar un componente y que el usuario lo adapte a sus necesidades.

Nota: Descripción de atributos de modularidad. Fuente: ISO 25010.

Tabla 64.

Atributos de Reusabilidad.

Característica 7		Mantenibilidad
Subcaracterística 7.2		Reusabilidad
Atributos		
N°	Nombre	Descripción
7.2.1	Componentes re-usables	Capacidad del framework para permitir reutilizar los componentes.

Nota: Descripción de los atributos de reusabilidad. Fuente: ISO 25010.

Tabla 65.

Atributos de adaptabilidad.

Característica 8		Portabilidad
Subcaracterística 8.1		Adaptabilidad
Atributos		
N°	Nombre	Descripción
8.1.1	Adaptación de sintaxis	Permite usar las funcionalidades de un framework adaptando su sintaxis a la de otro.
8.1.2	Funciona en Internet Explorer	Capacidad del framework para operar en el navegador web internet explorer

8.1.3	Funciona en Safari	Capacidad del framework para operar en el navegador web Safari
8.1.4	Funcionan en Mozilla Firefox	Capacidad del framework para operar en el navegador web Mozilla Firefox
8.1.5	Funciona en Google Chrome	Capacidad del framework para operar en el navegador web Google Chrome
8.1.6	Funciona en Opera	Capacidad del Framework para operar en el navegador web Opera.
8.1.7	Funciona en navegadores web para móviles	Posee funcionalidades para crear interfaces orientadas a los dispositivos móviles.

Nota: Descripción de los atributos de adaptabilidad. Fuente: ISO 20510.

Tabla 66.

Atributos que Capacidad para ser instalado.

Característica 8		Portabilidad
Subcaracterística 8.2		Capacidad para ser instalado
Atributos		
N°	Nombre	Descripción
8.2.1	Manuales de instalación	Provee manuales de instalación
8.2.2	Ayuda en línea	Los proveedores o los usuarios del framework a través de foros o archivos multimedia proveen ayudas para la instalación.
8.2.3	Tiempo de instalación	Tiempo que toma a los usuarios instalar el framework.

Nota: Descripción de los atributos de capacidad para ser instalado. Fuente: ISO 25010

Tabla 67.

Atributos de Capacidad para ser reemplazado.

Característica 8		Portabilidad
Subcaracterística 8.3		Capacidad para ser reemplazado

Atributos

N°	Nombre	Descripción
8.3.1	Reemplazo de versiones (con comprensión y sin compresión)	Facilidad para cambiar un framework descomprimido por uno comprimido en una aplicación determinada. Framework versión comprimida: Disminuye el peso de los archivos en una aplicación. Código fuente ofuscado. Framework versión descomprimida: Aumenta el peso de los archivos en una aplicación. Código fuente sin ofuscar.
8.3.2	Cambio de versiones (actualización del framework)	Facilidad para reemplazar archivos de un framework antiguo por uno nuevo.
8.3.3	Reemplazo de framework	Facilidad para reemplazar un framework por otro de diferente proveedor.

Nota: Descripción de los atributos de capacidad para ser reemplazado. Fuente: ISO 25010.

v. Paso 4: Refinamiento de atributos derivados en básicos.

Tabla 68.

Atributos básicos de efectividad en la creación de efectos y animaciones.

Característica 1		Funcionalidad
Subcaracterística 1.2		Precisión
Atributo derivado 1.2.1		Efectividad en la creación de efectos y animación
Atributos básicos		
N°	Nombre	Descripción
1.2.1.1	Configuración de fps (frames por segundo)	Permite crear animaciones con el nivel de fluidez requerido

1.2.1.2	Configuración unit (unidad de medida)	Permite definir la unidad de medida para la ubicación y propagación apropiada de animaciones y efectos.
1.2.1.3	Configuración de tiempo	Permite crear efectos y animaciones con tiempos de ejecución configurables.
1.2.1.4	Adaptación automática	Permite identificar de forma automática las dimensiones de un entorno específico, para crear elementos, efectos y animaciones que se ajusten a este.

Nota: Descripción de atributos básicos. Fuente: ISO 25010.

Tabla 69.

Atributos básicos de efectividad en la manipulación del DOM.

Característica 1		Funcionalidad
Subcaracterística 1.2		Precisión
Atributo derivado 1.2.1		Efectividad en la manipulación del DOM
Atributos básicos		
N°	Nombre	Descripción
1.2.2.1	Selectores “id” y “class”	Contienen métodos para seleccionar elementos del DOM a través de las propiedades “id” y “class”.
1.2.2.2	Selectores CSS.	Provee métodos para seleccionar elementos del DOM de manera similar a la que se usa en las hojas de estilo
1.2.2.3	Selectores por tipo de elemento.	Permite seleccionar elementos del DOM haciendo uso del tipo, ejemplo: “table”, “div”, “ul”, etc.
1.2.2.4	Selectores personalizados	Provee un conjunto de métodos que integran operadores aritméticos y de comparación para facilitar las selecciones.

Nota: Descripción de atributos básicos. Fuente: ISO 25010

Tabla 70.

Atributos básicos de descarga del framework.

Característica 5		Usabilidad
Subcaracterística 5.2		Disponibilidad
Atributo derivado 5.2.1		Descarga del framework
Atributos básicos		
N°	Nombre	Descripción
5.2.1.1	Descarga con compresión	Permite descargar el framework con compresión para usar en aplicaciones en producción.
5.2.1.2	Descarga sin compresión	Permite descargar el framework sin compresión para usar en aplicaciones en desarrollo
5.2.1.3	Descarga desde el sitio oficial	Permite descargar el framework con o sin compresión desde el sitio web oficial
5.2.1.4	Descarga desde sitios web no oficiales	Permite descargar el framework con o sin compresión desde sitios web no oficiales.

Nota: Descripción de los atributos básicos. Fuente: ISO 25010.

Tabla 71.

Atributos básicos de referenciar el framework.

Característica 5		Usabilidad
Subcaracterística 5.2		Disponibilidad
Atributo derivado 5.2.2		Referenciar el framework
Atributos básicos		
N°	Nombre	Descripción
5.2.2.1	Referencia a través del sitio web oficial	Permite referenciar el framework a través del sitio web oficial.
5.2.2.2	Referencia a través de Google	Permite referenciar el framework a través de Google.

5.2.2.3 Referencia a través de sitios web no oficiales Permite referenciar el framework a través de sitios web no oficiales.

Nota: Descripción de los atributos básicos. Fuente: ISO 25010.

vi. Paso 5: Establecimiento de relaciones entre factores de calidad.

Para desarrollar el modelo de calidad, se mantienen las relaciones jerárquicas de profundidad que propone la norma ISO/IEC 25000.

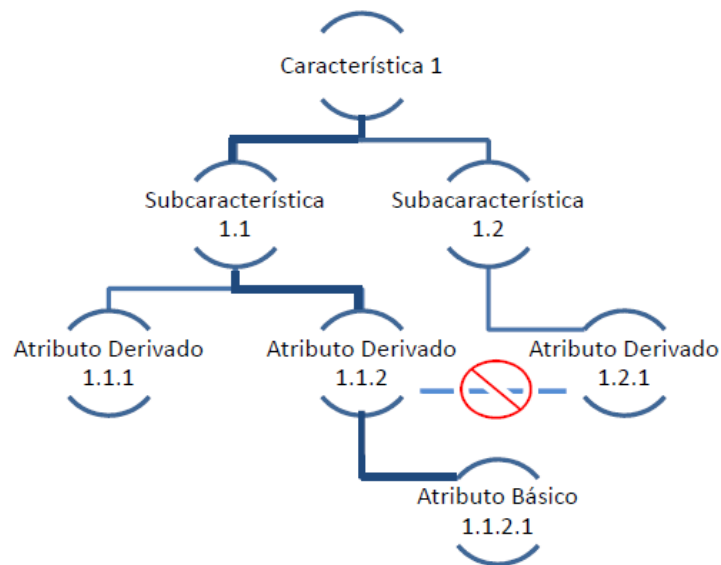


Figura 25. Relaciones entre factores de calidad. Fuente: (Bermeo Rodríguez, 2014).

vii. Paso 6: Determinación de métricas para los atributos.

Para evaluar las características de calidad utilizando la norma ISO/IEC 25010 de un framework y seleccionar el más adecuado para el desarrollo de aplicaciones para dispositivos móviles, se hará uso de métricas objetivas (medidas directas), que determinaran el nivel de cumplimiento que cada framework posee frente a los atributos de calidad planteados en los pasos 3 y 4 del método IQMC.

Las siguientes métricas se utilizarán en el modelo de evaluación:

- Evaluación del cumplimiento de las características.

Tabla 72.

Métricas de cumplimiento.

N°	Cumplimiento	Equivalente contable
1	Si	1
2	No	0

Nota: Métricas de evaluación (1 si cumple, 0 no cumple).

➤ Evaluación del cumplimiento de las características por rango:

Tabla 73.

Métrica de cumplimiento por rangos.

N°	Intervalos	Equivalente contable	Cumplimiento
1	0	0	Nulo
2	1	0.25	Bajo
3	2	0.50	Medio
4	3	0.75	Alto
5	4	1	completo

Nota: Métricas de evaluación por rangos.

3.3.2.1.2. Modelo integrado

La siguiente tabla se divide en las siguientes categorías:

Primero están las *características* que son del primer nivel. Luego están las *subcaracterísticas* que se derivan de las características, segundo nivel. Posteriormente siguen los *atributos derivados*, tercer nivel. Como cuarto nivel están los *atributos básicos* de evaluación.

Tabla 74.

Construcción del Modelo de calidad.

1	Adecuación funcional		
1.1	Completitud funcional		Métrica
1.1.1	Creación de aplicaciones interactivas		Si = 1, No = 0
1.1.2	Creación de aplicaciones amigables		Si = 1, No = 0
1.1.3	Creación de aplicaciones robustas		Si = 1, No = 0

1.1.4	Creación de aplicaciones orientadas a móviles	Si = 1, No = 0
1.2	Corrección funcional	
1.2.1	Efectividad en la creación de efectos y animaciones	Métrica
1.2.1.1	Configuración de "fps" (Frames por segundo)	Si = 1, No = 0
1.2.1.2	Configuración de "unit" (Unidad de medida)	Si = 1, No = 0
1.2.1.3	Configuración de tiempo	Si = 1, No = 0
1.2.1.4	Adaptación automática	Si = 1, No = 0
1.2.2	Efectividad en la manipulación del DOM	Métrica
1.2.2.1	Selectores "id" y "class"	Si = 1, No = 0
1.2.2.2	Selectores CSS	Si = 1, No = 0
1.2.2.3	Selectores por tipo de Elemento	Si = 1, No = 0
1.2.2.4	Selectores personalizados	Si = 1, No = 0
1.3	Pertinencia funcional	Métrica
1.3.1	Método Ajax.	Si = 1, No = 0
1.3.2	Métodos para manipular el DOM	Si = 1, No = 0
1.3.3	Elementos interactivos	Si = 1, No = 0
1.3.4	Validación de formularios	Si = 1, No = 0
1.3.5	Efectos visuales	Si = 1, No = 0
1.3.6	Gestión de tablas o grids	Si = 1, No = 0
2	Eficiencia de desempeño	
2.1	Comportamiento en el Tiempo	Métrica
2.1.1	Configuración del tiempo de espera en una comunicación Cliente – Servidor	Si = 1, No = 0
2.1.2	Comunicación Síncrona y Asíncrona	Si = 1, No = 0
2.1.3	Tiempo de carga en prueba local	Rango Tiempo(ms) 5 100

		4	200
		3	300
		2	500
		1	1000+
2.2	Utilización de recursos	Métrica	
2.2.1	Necesita librerías	SI = 1, No = 0	
2.2.2	JavaScript	SI = 1, No = 0	
2.2.3	Incorporación de AJAX	SI = 1, No = 0	
2.2.4	Iconos propios	SI = 1, No = 0	
3	Compatibilidad		
3.1	Coexistencia	Métrica	
3.1.1	Coexistencia con HTML5	Si = 1, No = 0	
3.1.2	Coexistencia con CSS3	Si = 1, No = 0	
3.1.3	Coexistencia con JavaScript	Si = 1, No = 0	
3.2	Interoperabilidad	Métrica	
3.2.1	Intercambio de información con PHP	Si = 1, No = 0	
3.2.2	Intercambio de información con Java EE	Si = 1, No = 0	
3.2.3	Intercambio de información con C# (ASP.net)	Si = 1, No = 0	
3.2.4	Intercambio de información con Python	Si = 1, No = 0	
3.2.5	Intercambio de información con Ruby	Si = 1, No = 0	
4	Usabilidad		
4.1	Capacidad para reconocer su adecuación	Métrica	
4.1.1	Información real en el sitio web	Si = 1, No = 0	
4.1.2	Actualización de contenido	Si = 1, No = 0	
4.2	Capacidad de aprendizaje	Métrica	Evaluación
4.2.1	Manual técnico		
4.2.2	Foros		
4.2.3	Tutoriales	[0:4]	
4.2.4	Cursos on-line		
			A mayor recursos disponibles, mayor calificación.

4.2.5	Estudios e investigaciones		
4.2.6	Documentación		
4.3	Capacidad para ser usado	Métrica	
4.3.1	POO. basada en Clases		Si = 1, No = 0
4.3.2	Reconocimiento de entorno		Si = 1, No = 0
4.3.3	Licencia		Si = 1, No = 0
4.4	Protección frente a errores de usuario		
4.4.1	Modificación por el usuario		Si = 1, No = 0
4.5	Accesibilidad (N/A)		
5	Fiabilidad		
5.1	Madurez	Métrica	Evaluación
5.1.1	Tiempo en el mercado	[0:4]	A mayor tiempo, mayor calificación.
5.1.2	Actualizaciones disponibles	[0:4]	1 año: [0] 2 años: [1] 3 años: [2] 4 años: [3] > 4 años: [4]
5.1.3	Posee una base del conocimiento software		Si = 1, No = 0
5.1.4	Popularidad en GitHub	Rango	#Estrellas
		5	100000 +
		4	50000
		3	25000
		2	10000
		1	1000+
5.1.5	Aplicaciones realizadas		Si = 1, No = 0
5.2	Disponibilidad		
5.2.1	Descarga del Framework	Métrica	
5.2.1.1	Descarga con compresión		Si = 1, No = 0
5.2.1.2	Descarga sin compresión		Si = 1, No = 0

5.2.1.3	Descarga desde el sitio web oficial	Si = 1, No = 0
5.2.1.4	Descarga desde sitios web no oficiales	Si = 1, No = 0
5.2.2	Referenciar el Framework	Métrica
5.2.2.1	Referencia a través del sitio web oficial	Si = 1, No = 0
5.2.2.2	Referencia a través de Google	Si = 1, No = 0
5.2.2.3	Referencia a través de sitios web no oficiales	Si = 1, No = 0
5.2.3	Repositorio en GitHub	Si = 1, No = 0
5.3	Tolerancia a fallos	Métrica
5.3.1	Permite acceder a la información con fallos en el sistema	Si = 1, No = 0
5.3.2	Realizar respaldos	Si = 1, No = 0
5.4	Capacidad de recuperación	Métrica
5.4.1	Capacidad de recuperar información	Si = 1, No = 0
6	Seguridad	
6.1	Confidencialidad	Métrica
6.1.1	Protección de datos	Si = 1, No = 0
6.1.2	Acceso solo a usuarios específicos	Si = 1, No = 0
6.1.3	Es seguro frente a internet	Si = 1, No = 0
6.1.4	Cifrado en la información	Si = 1, No = 0
6.2	Integridad	Métrica
6.2.1	Información correcta en la base de datos	Si = 1, No = 0
6.2.2	No permite modificación de datos	Si = 1, No = 0
6.3	No repudio	Métrica
6.3.1	Comunicación confiable cliente-servidor	Si = 1, No = 0
6.4	Autenticidad	Métrica
6.4.1	Evita la suplantación de identidad	Si = 1, No = 0

6.4.2	Autenticidad generada por el usuario		Si = 1, No = 0
6.5	Responsabilidad		Métrica
6.5.1	Se hace responsable de la seguridad		Si = 1, No = 0
7	Mantenibilidad		
7.1	Modularidad		Métrica
7.1.1	Es modular		Si = 1, No = 0
7.1.2	Crear componentes		Si = 1, No = 0
7.1.3	Eliminar componentes		Si = 1, No = 0
7.1.4	Modificar componentes		Si = 1, No = 0
7.2	Reusabilidad		Métrica
7.2.1	Componentes re-usables		Si = 1, No = 0
8	Portabilidad		
8.1	Adaptabilidad		Métrica
8.1.1	Adaptación de sintaxis		Si = 1, No = 0
8.1.2	Funciona en Internet Explorer		Si = 1, No = 0
8.1.3	Funciona en Safari		Si = 1, No = 0
8.1.4	Funciona en Mozilla Firefox		Si = 1, No = 0
8.1.5	Funciona en Google Chrome		Si = 1, No = 0
8.1.6	Funciona en Opera		Si = 1, No = 0
8.1.7	Funciona en navegadores web para móviles		Si = 1, No = 0
8.2	Facilidad de instalación	Métrica	Evaluación
8.2.1	Manuales de instalación	Si = 1, No = 0	
8.2.2	Ayuda en línea	[0:4]	A mayor recursos identificados en el transcurso del desarrollo del proyecto, mayor calificación.
8.2.3	Tiempo de instalación	[0:4]	A menor tiempo, mayor calificación. 0 segundos: [4] [1-30] segundos: [3] 1 minuto: [2]

			3 minutos: [1] >3 minutos: [0]
8.3	Capacidad para ser reemplazado	Métrica	Evaluación
8.3.1	Reemplazo de versiones (con compresión y sin compresión)	[0:4]	A mayor dificultad, menor calificación
8.3.2	Cambio de versiones (actualización de Framework)		
8.3.3	Reemplazo de Framework		

Nota: Cuadro resumido de características, subcaracterísticas y atributos con su respectiva métrica de evaluación. Fuente: ISO 25010.

Nota: (N/A) = No Aplica.

3.3.2.2. Aplicación del modelo

A continuación, se aplica el modelo anteriormente desarrollado en los frameworks Ionic y jQuery Mobile.

Tabla 75.

Aplicación del modelo de evaluación de calidad.

Características, Atributos derivados y Atributos básicos	Subcaracterísticas, Atributos básicos	Métrica	Ionic	jQuery Mobile
1	Adecuación funcional			
1.1	Complejidad funcional			
1.1.1	Creación de aplicaciones interactivas	Si = 1, No = 0	1	1
1.1.2	Creación de aplicaciones amigables	Si = 1, No = 0	1	1

1.1.3	Creación de aplicaciones robustas	Si = 1, No = 0	1	1
1.1.4	Creación de aplicaciones orientadas a móviles	Si = 1, No = 0	1	1
1.2	Corrección funcional			
1.2.1	Efectividad en la creación de efectos y animaciones			
1.2.1.1	Configuración de "fps" (Frames por segundo)	Si = 1, No = 0	1	0
1.2.1.2	Configuración de "unit" (Unidad de medida)	Si = 1, No = 0	1	0
1.2.1.3	Configuración de tiempo	Si = 1, No = 0	1	0
1.2.1.4	Adaptación automática	Si = 1, No = 0	1	1
1.2.2	Efectividad en la manipulación del DOM			
1.2.2.1	Selectores "id" y "class"	Si = 1, No = 0	1	1
1.2.2.2	Selectores CSS	Si = 1, No = 0	1	1
1.2.2.3	Selectores por tipo de Elemento	Si = 1, No = 0	1	1
1.2.2.4	Selectores personalizados	Si = 1, No = 0	1	1
1.3	Pertinencia funcional			
1.3.1	Método Ajax.	Si = 1, No = 0	1	1
1.3.2	Métodos para manipular el DOM	Si = 1, No = 0	1	1
1.3.3	Elementos interactivos	Si = 1, No = 0	1	1
1.3.4	Validación de formularios	Si = 1, No = 0	1	1
1.3.5	Efectos visuales	Si = 1, No = 0	1	1
1.3.6	Gestión de tablas o grids	Si = 1, No = 0	1	1

2 Eficiencia de desempeño

2.1	Comportamiento en el Tiempo				
2.1.1	Configuración del tiempo de espera en una comunicación Cliente – Servidor	Si = 1, No = 0	1	1	
2.1.2	Comunicación Síncrona y Asíncrona	Si = 1, No = 0	1	1	
2.1.3	Tiempo de carga en prueba local	R	T(ms)		
		5	100	4.75	4.72
		4	200	(125	(128
		3	300	ms)	ms)
		2	500		
		1	1000+		
2.2	Utilización de recursos				
2.2.1	Necesita librerías	SI = 1, No = 0	1	1	
2.2.2	JavaScript	SI = 1, No = 0	1	1	
2.2.3	Incorporación de AJAX	SI = 1, No = 0	1	1	
2.2.4	Iconos propios	SI = 1, No = 0	1	1	
3	Compatibilidad				
3.1	Coexistencia				
3.1.1	Coexistencia con HTML5	Si = 1, No = 0	1	1	
3.1.2	Coexistencia con CSS3	Si = 1, No = 0	1	1	
3.1.3	Coexistencia con JavaScript	Si = 1, No = 0	1	1	
3.2	Interoperabilidad				
3.2.1	Intercambio de información con PHP	Si = 1, No = 0	1	1	
3.2.2	Intercambio de información con Java EE	Si = 1, No = 0	1	1	
3.2.3	Intercambio de información con C# (ASP.net)	Si = 1, No = 0	1	1	
3.2.4	Intercambio de información con Python	Si = 1, No = 0	1	1	
3.2.5	Intercambio de información con Ruby	Si = 1, No = 0	1	1	

4	Usabilidad				
4.1	Capacidad para reconocer su adecuación				
4.1.1	Información real en el sitio web	Si = 1, No = 0	1	1	
4.1.2	Actualización de contenido	Si = 1, No = 0	1	1	
4.2	Capacidad de aprendizaje				
4.2.1	Manual técnico	[0:4]	4	3	
4.2.2	Foros	[0:4]	4	4	
4.2.3	Tutoriales	[0:4]	4	4	
4.2.4	Cursos on-line	[0:4]	4	4	
4.2.5	Estudios e investigaciones	[0:4]	4	3	
4.2.6	Documentación	[0:4]	4	3	
4.3	Capacidad para ser usado				
4.3.1	POO. basada en Clases	Si = 1, No = 0	1	0	
4.3.2	Reconocimiento de entorno	Si = 1, No = 0	1	1	
4.3.3	Licencia	Si = 1, No = 0	1	1	
4.4	Protección frente a errores de usuario				
4.4.1	Modificación por el usuario	Si = 1, No = 0	1	1	
4.5	Accesibilidad (N/A)				
5	Fiabilidad				
5.1	Madurez				
5.1.1	Tiempo en el mercado	[0:4]	4	4	
5.1.2	Actualizaciones disponibles	[0:4]	4	1	
5.1.3	Posee una base del conocimiento software	Si = 1, No = 0	1	1	
5.1.4	Popularidad en GitHub	R #Star			
		5 100000+			
		4 50000	3.4	2.1	
		3 25000			
		2 10000			
		1 1000+			
5.1.5	Aplicaciones realizadas	Si = 1, No = 0	1	1	
5.2	Disponibilidad				

5.2.1	Descarga del Framework				
5.2.1.1	Descarga con compresión	Si = 1, No = 0	1	1	
5.2.1.2	Descarga sin compresión	Si = 1, No = 0	1	1	
5.2.1.3	Descarga desde el sitio web oficial	Si = 1, No = 0	1	1	
5.2.1.4	Descarga desde sitios web no oficiales	Si = 1, No = 0	0	1	
5.2.2	Referenciar el Framework				
5.2.2.1	Referencia a través del sitio web oficial	Si = 1, No = 0	1	1	
5.2.2.2	Referencia a través de Google	Si = 1, No = 0	1	1	
5.2.2.3	Referencia a través de sitios web no oficiales	Si = 1, No = 0	1	1	
5.2.3	Repositorio en GitHub	Si = 1, No = 0	1	1	
5.3	Tolerancia a fallos				
5.3.1	Permite acceder a la información con fallos en el sistema	Si = 1, No = 0	0	0	
5.3.2	Realizar respaldos	Si = 1, No = 0	0	0	
5.4	Capacidad de recuperación				
5.4.1	Capacidad de recuperar información	Si = 1, No = 0	0	0	
6	Seguridad				
6.1	Confidencialidad				
6.1.1	Protección de datos	Si = 1, No = 0	0	0	
6.1.2	Acceso solo a usuarios específicos	Si = 1, No = 0	0	0	
6.1.3	Es seguro frente a internet	Si = 1, No = 0	0	0	

6.1.4	Cifrado en la información	Si = 1, No = 0	0	0
6.2	Integridad			
6.2.1	Información correcta en la base de datos	Si = 1, No = 0	0	0
6.2.2	No permite modificación de datos	Si = 1, No = 0	0	0
6.3	No repudio			
6.3.1	Comunicación confiable cliente-servidor	Si = 1, No = 0	0	0
6.4	Autenticidad			
6.4.1	Evita la suplantación de identidad	Si = 1, No = 0	0	0
6.4.2	Autenticidad generada por el usuario	Si = 1, No = 0	1	1
6.5	Responsabilidad			
6.5.1	Se hace responsable de la seguridad	Si = 1, No = 0	0	0
7	Mantenibilidad			
7.1	Modularidad			
7.1.1	Es modular	Si = 1, No = 0	1	1
7.1.2	Crear componentes	Si = 1, No = 0	1	1
7.1.3	Eliminar componentes	Si = 1, No = 0	1	1
7.1.4	Modificar componentes	Si = 1, No = 0	1	1
7.2	Reusabilidad			
7.2.1	Componentes re-usables	Si = 1, No = 0	1	1
8	Portabilidad			
8.1	Adaptabilidad			
8.1.1	Adaptación de sintaxis	Si = 1, No = 0	1	0
8.1.2	Funciona en Internet Explorer	Si = 1, No = 0	1	1
8.1.3	Funciona en Safari	Si = 1, No = 0	1	1
8.1.4	Funciona en Mozilla Firefox	Si = 1, No = 0	1	1
8.1.5	Funciona en Google Chrome	Si = 1, No = 0	1	1

8.1.6	Funciona en Opera	Si = 1, No = 0	1	1
8.1.7	Funciona en navegadores web para móviles	Si = 1, No = 0	1	1
8.2	Facilidad de instalación			
8.2.1	Manuales de instalación	Si = 1, No = 0	1	1
8.2.2	Ayuda en línea	[0:4]	4	4
8.2.3	Tiempo de instalación	[0:4]	0	1
8.3	Capacidad para ser reemplazado			
8.3.1	Reemplazo de versiones (con compresión y sin compresión)	[0:4]	4	1
8.3.2	Cambio de versiones (actualización de calificación Framework)	[0:4]	3	1
8.3.3	Reemplazo de Framework	[0:4]	4	4

Nota: Resultado de la evaluación de los atributos aplicando las métricas de evaluación. Fuente: ISO 25010.

Nota: (N/A) = No Aplica

3.3.2.3. Resultado del análisis comparativo

En la siguiente tabla se describen los resultados del análisis comparativo, el cual se encuentra organizado en base a las características elegidas de la norma ISO/IEC 25000.

Tabla 76.

Resultados del análisis comparativo.

Características de calidad	N° atributos	%	Ionic		jQuery Mobile	
			puntos	%	puntos	%
Adecuación funcional	18	20.22	18	22.81	15	21.13

Rendimiento	7	7.87	10.75	13.62	10.72	15.10
Compatibilidad	8	8.99	8	10.14	8	11.27
Usabilidad	12	13.48	11	13.94	9.25	13.03
Fiabilidad	16	17.98	13.4	16.98	12.26	17.27
Seguridad	10	11.24	1	1.27	1	1.41
Mantenibilidad	5	5.62	5	6.34	5	7.04
Portabilidad	13	14.61	11.75	14.89	9.75	13.74
Total	89	100	78.9	100	70.98	100

Nota: Cuadro resumido del resultado de la evaluación. Fuente: Elaboración propia.

3.3.3. Desarrollar una aplicación como caso de estudio para cada framework.

Para poder desarrollar la aplicación de apuestas de fútbol (Golwin) se tuvo que realizar dos cosas, primero se desarrolló una API REST (lado del Servidor) y luego desarrollar la aplicación con cada uno de los frameworks antes mencionados (lado del Cliente).

3.3.3.1. Metodología de desarrollo Scrum

Se utilizará SCRUM como metodología para el desarrollo de la API RESTful y la aplicación del lado del cliente.

Para esta investigación se utiliza Scrum por ser una metodología que permite el desarrollo rápido e iterativo para desarrollo de aplicaciones móviles, así lo especifican investigaciones realizadas como:

“Desarrollo de una API REST con sus aplicaciones web y móvil para la venta de ropa online de la empresa Rossman” (Santos Hernández & Serrano Parreño, 2017).

“Aprendiendo a desarrollar aplicaciones para Android con la metodología ágil Scrum: Un caso de estudio” (Roque Hernández, Negrete Hoz, & Salinas Escandón, 2013).

“Método ágil Scrum, aplicado a la implantación de un sistema informático para el proceso de recolección masica de información con tecnología móvil” (Toapanta Chancusi, Vergara Ordoñez, & Campaña Ortega, 2014).

“Propuesta para documentar trabajos finales utilizando metodologías ágiles” (Uva, Daniele, Zorzán, Frutos, & Arsaute, 2014).

De acuerdo con la metodología Scrum tanto el dueño del producto y el equipo de desarrollo es el presente autor de esta investigación.

Para el levantamiento de los requerimientos se realizó una investigación sobre como es el funcionamiento y la lógica de negocio para un sistema de apuesta de futbol, teniendo como referencias a trabajos ya realizados como:

“Desarrollo de una aplicación móvil sobre la plataforma Android para la liga de fútbol sala de la facultad de ciencias de la universidad central de Venezuela” (Almeida Acosta & Romero Rivero, 2016).

“Diseño e implementación de una base de datos relacional para la gestión de apuestas de fútbol” (Cuenca Aznar, 2014).

“Diseño e implementación de una base de datos relacional para la gestión de apuestas de fútbol” (Albors Aznar, 2014).

“Diseño y desarrollo de un sistema denominado Ocaña Gol para realizar las apuestas del campeonato liga Postobon del fútbol colombiano” (Estrada Salazar, 2012).

De las referencias anteriores se pudo obtener la información necesaria para poder establecer los requerimientos del sistema GOLWIN. Además, se pudo definir las tablas y los atributos para la base de datos, tal como se mostrará más adelante.

Según la metodología Scrum, y ya realizada la investigación en los documentos anteriores, se procede a realizar los requerimientos técnicos y/o lógica del negocio, lo que vendría a ser el funcionamiento del sistema; lo que se conoce como un:

3.3.3.1.1. Primer Backlog en borrador.

Para la parte del sistema de gestión de la API RESTFul.

- Debe permitir almacenar los datos de usuarios (registro de usuarios)
- Debe permitir almacenar las apuestas en la base de datos que realiza un usuario registrado y autenticado.
- Debe almacenar los partidos en una base de datos.
- Debe almacenar los tipos de competiciones en la base de datos
- Debe almacenar las competiciones de cada tipo de competición en la base de datos.
- Debe almacenar en la base de datos los países para el registro de usuarios y para que estos estén disponibles para una competición.
- Debe almacenar en la base de datos la lista de equipos según una competición.
- Debe almacenar en la base de datos la temporada en la que está participando una lista de equipos de una determinada competición.
- Debe almacenar en la base de datos los equipos para agregarlos a la lista de equipos de una competición.
- Debe almacenar en la base de datos la jornada en la que se está jugando un partido.
- Debe almacenar en una base de datos el resultado de un partido para realizar la comparación con la apuesta que realizó un usuario y así dar a conocer si se acertó o no en la apuesta.
- Se debe tener una lista de diferentes pagos para cada partido, para así darle al usuario variedad al realizar una apuesta.

Para la parte de brindar servicios de la API RESTFul

- La API debe permitirle al usuario autenticarse.
- La API debe permitirle al usuario Registrarse en el sistema.

- La API debe permitirle al usuario realizar una apuesta y guardarlo en la base de datos.
- La API debe proporcionar la lista de partidos según la competición si el usuario las solicita.
- La API debe proporcionar la lista de tipo de competiciones si el usuario las solicita.
- La API debe proporcionar la lista de competiciones según tipo de competición si el usuario la solicita.
- La API debe proporcionar la lista de pagos si el usuario requiere hacer una apuesta de un determinado partido.
- La API debe proporcionar los datos de un usuario si este los solicita.

Para la parte de la aplicación para los clientes, los que van a consumir los servicios que brinda la API RestFull GOLWIN.

- Debe permitir registrar usuarios.
- Debe permitir ingresar a la aplicación.
- Debe permitir mostrar lista de competiciones según tipo de competición.
- Debe permitir mostrar lista de partidos según competición.
- Debe mostrar los pagos de cada partido, esto depende el tipo de apuesta a realizarse.
- Debe permitir agregar un monto para la apuesta.
- Debe permitir realizar una apuesta.

El sistema gestor de base de datos sobre el que se implementará la solución será PostgreSQL.

El sistema y la API se desarrollará en el lenguaje de programación Java, utilizando frameworks como Spring MVC, Hibernate, Spring Security, y el formato JSON para el intercambio de información.

Se eligió el Spring framework teniendo en cuenta la investigación “Análisis comparativo de frameworks para el desarrollo de aplicaciones web en java” (Sanchez Acosta, 2015), según la investigación presenta a Spring como el que

mejor resultados tiene para desarrollar un Sistema web y que permite desarrollar una API RESTFul bajo el lenguaje de programación Java, lo cual facilita el desarrollo para nuestro propósito, permitiendo el intercambio de datos mediante JSON, también agregaremos Spring security, lo que nos va a permitir asegurar nuestra sistema API RESTFul. Además, de integrarse muy bien con el framework Hibernate.

3.3.3.1.2. Historias de usuarios

Teniendo en cuenta los requisitos técnicos mencionados en la sección anterior se procede a realizar las historias de usuarios, las cuales nos van a permitir tener más claro el funcionamiento y/o la lógica de nuestro sistema de apuesta de futbol Golwin.

Historias de usuarios para la parte del sistema de gestión de la API RESTFull - Golwin.

API01 – Crear la base de datos

Como: administrador del sistema

Quiero: almacenar todos los datos en una base de datos

Para: poder tener acceso a los datos almacenados y poder manipularlos.

Condiciones:

- Crear la base de datos en Postgresql.
- Crear las tablas y sus atributos, y las relaciones que se crean convenientes.

API04 – Registrar los partidos

Como: administrador del sistema

Quiero: agregar partidos que se jugaran.

Para: que los usuarios puedan elegir el partido en el cual creen que pueden realizar una apuesta ganadora.

Condiciones:

- Se deben agregar dos equipos para un partido
- Los partidos pertenecen a una competición.
- Se debe de agregar la jornada del partido.
- Se debe agregar la fecha y hora del partido.

- Realizar un CRUD para realizar las operaciones pertinentes.

API05 – Registrar las competiciones

Como: administrador del sistema

Quiero: agregar competiciones a los que pertenecerá un partido a jugarse (Champions League, Copa mundial. Etc)

Para: poder separar los partidos según su competición, y así lo usuarios puedan encontrar los partidos que prefieren con mayor facilidad.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.
- Las competiciones deben de pertenecer a un tipo de competición.

API06 – Registrar los tipos de competiciones.

Como: administrador del sistema

Quiero: agregar tipos de competiciones a los que pertenecerá una competición (Copa, Liga)

Para: poder separar las competiciones según su tipo, y así lo usuarios puedan encontrar una competición que prefieren con mayor facilidad.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.

API07 – Registrar países.

Como: administrador del sistema

Quiero: agregar países del mundo

Para: brindarles a los usuarios poder registrar su país de procedencia, y también poder crear partidos, por ejemplo, en una copa mundial.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.

API08 – Registrar Equipos.

Como: administrador del sistema

Quiero: agregar equipos de futbol.

Para: añadirlos a una lista de equipos según una competición.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.

API09 – Registrar Lista de Equipos.

Como: administrador del sistema

Quiero: crear una lista de equipos de futbol.

Para: agruparlos por competiciones, y por temporadas.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.
- Se tiene que agregar la competición al que pertenece la lista de equipos.
- Se tiene que agregar la temporada a la que pertenece.
- En la lista se puede agregar tanto países como equipos según la competición, si es copa mundial se agregan países, si es Liga se agregan equipos (Champions League se agrega un equipo como FC Barcelona).

API10 – Registrar Temporadas.

Como: administrador del sistema

Quiero: añadir las temporadas de futbol.

Para: poder agregarla a una lista de equipos.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.

API11 – Registrar Jornadas.

Como: administrador del sistema

Quiero: añadir las Jornadas de futbol.

Para: poder agregarla a un partido de futbol.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.

API12 – Registrar Resultados.

Como: administrador del sistema

Quiero: añadir los resultados de los partidos de futbol.

Para: luego hacer la comparación con la apuesta que realizó un usuario.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.
- El resultado debe pertenecer a un partido que se encuentra registrado y que ya se halla jugado o finalizado.

API13 – Registrar Pagos.

Como: administrador del sistema

Quiero: añadir los pagos de los partidos de futbol.

Para: brindarle al usuario diferentes formas de pago según el tipo de apuesta que desee realizar.

Condiciones:

- Realizar un CRUD para realizar las operaciones pertinentes.
- Los pagos deben pertenecer a un partido que se encuentra registrado
- Los pagos se registran antes de finalizar el partido.
- Los pagos deben ser mostrados al usuario para que pueda seleccionar el que desee.

API21 – Preparar el ambiente para el desarrollo del sistema.

Como: administrador del sistema

Quiero: crear el proyecto.

Para: que se realice su desarrollo en Java.

Condiciones:

- Se debe desarrollar en el lenguaje Java.

- Se utilizarán los siguientes framework: Spring MVC, Spring Security, Hibernate.
- Se deben agregar las librerías necesarias para el correcto funcionamiento del sistema, que permitan realizar el intercambio de información en formato JSON.

Fuente: Elaboración propia

Historias de usuarios para brindar servicios de la API RESTFull Golwin.

API14 – Permitir la Autenticación de usuarios.

Como: administrador del sistema

Quiero: brindar el servicio de autenticación de los usuarios.

Para: que puedan interactuar con el sistema.

Condiciones:

- Para que un usuario de autentifique este debe estar previamente registrado.
- Este es un servicio que debe brindar la API RESTFul

API02 – Registro de usuarios

Como: administrador del sistema

Quiero: añadir y saber que usuarios tengo en mi base de datos

Para: tener un registro de ellos y que puedan almacenar sus datos.

Condiciones:

- Crear el sistema para que permita añadir a usuarios (CRUD)
- La API debe permitir el registro de usuarios.
- Los datos almacenados del usuario se deben brindar si este los solicita.

API03 – Permitir realizar apuestas

Como: administrador del sistema

Quiero: brindar un servicio de apuestas de futbol.

Para: que los usuarios puedan realizar apuestas de partidos que se encuentren disponibles.

Condiciones:

- Para que un usuario pueda realizar una apuesta este debe estar registrado en el sistema y autenticado en la aplicación.
- El usuario debe ingresar la cantidad que desea apostar.
- No se permite modificar la apuesta realizada.
- La apuesta debe pertenecer a un tipo según el pago del partido que se ha seleccionado. (Local, empate, visita)

API15 – Permitir el registro de usuarios.

Como: administrador del sistema

Quiero: brindar el servicio de registro de los usuarios.

Para: que puedan pertenecer al sistema.

Condiciones:

- Este es un servicio que debe brindar la API RESTFul

API16 – Proporcionar lista de partidos.

Como: administrador del sistema

Quiero: brindar la lista de partidos a los usuarios.

Para: que puedan seleccionar alguno y realizar apuestas.

Condiciones:

- Este es un servicio que debe brindar la API RESTFul
- Los partidos deben pertenecer a una determinada competición.
- Los partidos deben pertenecer a una determinada jornada.
- El usuario debe estar autenticado para visualizar la lista.

API17 – Proporcionar lista de tipo de competiciones.

Como: administrador del sistema

Quiero: brindar la lista de tipo de competiciones a los usuarios.

Para: que puedan seleccionar alguno y visualizar las competiciones.

Condiciones:

- Este es un servicio que debe brindar la API RESTFul.
- El usuario debe estar autenticado para visualizar la lista.

API18 – Proporcionar lista de competiciones.

Como: administrador del sistema

Quiero: brindar la lista de competiciones a los usuarios.

Para: que puedan seleccionar alguno y visualizar los partidos.

Condiciones:

- Este es un servicio que debe brindar la API RESTFul
- El usuario debe estar autenticado para visualizar la lista

API19 – Proporcionar lista de pagos.

Como: administrador del sistema

Quiero: brindar la lista de pagos según el partido a los usuarios.

Para: que puedan seleccionar alguno y realizar una apuesta multiplicando el monto por la cantidad que deseen apostar.

Condiciones:

- Este es un servicio que debe brindar la API RESTFul
- Los pagos deben pertenecer a un partido.
- El usuario debe estar autenticado para visualizar la lista

API20 – Proporcionar datos de usuario.

Como: administrador del sistema

Quiero: brindar los datos al usuario.

Para: que los pueda visualizar o manipular (cambiar contraseña)

Condiciones:

- Este es un servicio que debe brindar la API RESTFul.
- El usuario puede manipular alguno de sus datos, hay datos que deben permanecer intactos.
- El usuario debe estar autenticado para visualizar la lista.

Fuente: Elaboración propia.

Historias de usuarios para la aplicación que va a consumir los servicios de la API RESTFul Golwin.

HU02 - Registrarse en la aplicación

Como: usuario

Quiero: registrarme en la aplicación

Para: poder acceder a las opciones de la aplicación.

Condiciones:

- Debe agregarse un formulario para que el usuario ingrese sus datos personales.
- En el formulario debe mostrar lista de países.
- Debe permitir añadir un alias y password.
- Debe permitir añadir un correo electrónico.

HU01 - Login en la aplicación

Como: usuario

Quiero: acceder a la aplicación con una cuenta ya creada

Para: poder visualizar los partidos para realizar apuestas.

Condiciones:

- En la pantalla de login debe permitir al usuario ingresar con su alias y password
- Si el usuario no está registrado debe existir un botón que lleve a la pantalla de registro de usuario.

HU04 - Seleccionar un tipo de competición

Como: usuario

Quiero: visualizar la lista de tipos de competiciones

Para: poder acceder a ella y poder visualizar las competiciones que tiene.

Condiciones:

- Debe permitir listar los tipos de competiciones
- Debe permitir acceder a las competiciones al seleccionar una opción.

HU05 - Seleccionar una competición

Como: usuario

Quiero: visualizar la lista de competiciones, según el tipo de competición seleccionada.

Para: poder acceder a una ellas y poder visualizar los partidos disponibles que tiene.

Condiciones:

- Debe listar las competiciones que se encuentran disponibles en la base de datos.
- Debe permitir acceder a las opciones al seleccionar una competición.

HU03 - Seleccionar un partido

Como: usuario

Quiero: visualizar la lista de partidos, según la competición seleccionada.

Para: visualizar el detalle del partido seleccionado.

Condiciones:

- Debe listar los partidos que se encuentren en la base de datos según la competición seleccionada.
- Debe permitir acceder a un detalle del partido según se halla seleccionado.

HU06 - Pagos de cada partido

Como: usuario

Quiero: visualizar el detalle del partido como el monto a pagar según el resultado, entre otros, del partido seleccionado

Para: poder elegir qué tipo de apuesta deseo realizar.

poder multiplicarlo por el monto que deseo apostar.

Condiciones:

- Debe mostrar una lista con los tipos de apuesta y los pagos que se ofrecen para cada uno.
- Al seleccionar una opción debe permitir acceder a un formulario para poder realizar una apuesta.

HU08 - Agregar un monto para la apuesta

Como: usuario

Quiero: agregar el monto que deseo apostar.

Para: poder multiplicarlo por el pago que se ofrece para la opción seleccionada.

Condiciones:

- Debe permitir ingresar solo números positivos.
- Al ingresar un numero valido se debe realizar el cálculo entre el pago que se ofrece por la cantidad que se está apostando y mostrar el total.

HU07 - Realizar apuesta

Como: usuario

Quiero: poder realizar una apuesta de un determinado partido.

Para: poder acumular puntos en caso de acertar en la apuesta.

Condiciones:

- Debe existir un botón de cancelar o regresar atrás en caso no se quiera apostar y poder seleccionar otro tipo de apuesta o partido.
- Debe permitir registrar la apuesta en la base de datos.

HU09 – Menú principal

Como: usuario

Quiero: poder visualizar un menú principal con opciones importantes.

Para: poder permitirme saber qué es lo que se puede hacer en la aplicación.

Condiciones:

- Mostrar una lista de opciones, como, por ejemplo, visualizar partidos, realizar apuesta, Salir.

HU10 – Formulario de apuesta

Como: usuario

Quiero: poder visualizar un formulario.

Para: ingresar los diferentes valores que se necesiten al realizar una apuesta.

Condiciones:

- Mostrar que tipo de apuesta se realiza.
- Mostrar el partido al que se está haciendo la apuesta.
- Mostrar el monto de pago del partido

Fuente: Elaboración propia.

3.3.3.1.3. Priorizar y estimar el product Backlog

Se procederá a ordenar la lista de las historias de usuarios para priorizar y estimar que tareas se van a realizar primero, las cuales nos van a permitir avanzar con el desarrollo del resto de los requisitos y poder cumplir con el propósito.

Priorizar y estimar el product backlog para el sistema de gestión y de servicios de la API RESTFul Golwin.

Tabla 77.

Priorización y estimación del Product Backlog para el sistema de gestión y de servicios de la API.

Código	Descripción	Prioridad	Estimación – Planning Poker
API01	Crear la base de datos	1 Muy alta	13
API21	Preparar el ambiente para el desarrollo del sistema	2 Muy alta	13
API07	Registrar países	3 Muy alta	8
API02	Registro de usuarios	4 Muy alta	8
API06	Registrar los tipos de competiciones	5 Alta	5
API05	Registrar las competiciones	6 Alta	5
API10	Registrar Temporadas	7 Alta	3
API08	Registrar Equipos	8 Alta	13
API09	Registrar Lista de Equipos	9 Alta	13
API11	Registrar Jornadas	10 Alta	3

API04	Registrar los partidos	11	Alta	8
API13	Registrar Pagos	12	Alta	5
API12	Registrar Resultados	13	Alta	5
API03	Permitir realizar apuestas	14	Alta	13
API15	Permitir el registro de usuarios	15	Alta	8
API14	Permitir la Autenticación de usuarios	16	Alta	13
API17	Proporcionar lista de tipo de competiciones	17	Alta	5
API18	Proporcionar lista de competiciones	18	Alta	5
API16	Proporcionar lista de partidos	19	Alta	5
API19	Proporcionar lista de pagos	20	Alta	5
API20	Proporcionar datos de usuario	21	Alta	3

Nota: Se priorizan las tareas para realizarlas en un determinado orden. Fuente: Elaboración propia.

Tabla 78.

Priorización y estimación del Product Backlog para la aplicación del cliente.

Código	Descripción	Prioridad	Estimación – Planning Poker
HU02	Registrarse en la aplicación	1	Muy alta 5
HU01	Login en la aplicación	2	Muy alta 5
HU09	Menú principal	3	Muy alta 3
HU04	Seleccionar un tipo de competición	4	Alta 5
HU05	Seleccionar una competición	5	Alta 5
HU03	Seleccionar un partido	6	Alta 5
HU06	Pagos de cada partido	7	Alta 8
HU10	Formulario de apuesta	8	Alta 8

HU08	Agregar un monto para la apuesta	9	Alta	5
HU07	Realizar apuesta	10	Alta	13

Nota: Fuente: Se priorizan las tareas para realizarlas en un determinado orden. Elaboración propia.

3.3.3.1.4. División de tareas (Tasking) y Revisión de tareas (Spring Backlog)

Se procede a dividir en tareas más pequeñas para facilitar y agilizar el desarrollo y proceder a realizar entregables en corto plazo, la asignación del tiempo se da por horas, esto se debe a que el desarrollador (autor de la investigación) ya aprendió y conoce las herramientas que va a utilizar, siendo así se cree conveniente realizar la siguiente asignación del tiempo a cada tarea dividida.

Terminada la realización de la tarea se realiza la revisión de esta (Spring Backlog) lo que permite indicarle un estado a la tarea, ya sea no empezado, en proceso o hecho.

Tabla 79.

Tasking y Spring del product backlog del sistema de gestión y de servicios la API.

Req.	Código tarea	Descripción	Horas	Estado
API01	API01-T01	Crear la base de datos en postgresql	1	Hecho
	API01-T02	Definir las tablas y sus atributos	10	Hecho
	API01-T03	Definir las relaciones entre tablas	8	Hecho
API21	API21- T01	Crear el proyecto en el IDE Netbeans	1	Hecho
	API21- T02	Agregar las librerías necesarias	1	Hecho
	API21- T03	Crear e implementar las clases de configuración necesarias	10	Hecho

	API21- T04	Crear las clases Entidades, según las tablas de nuestra base de datos, y la respectiva configuración de Hibernate	8	Hecho
		Crear una clase Genérica con operaciones que se utilizaran con las entidades	4	Hecho
API07	API07-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API07-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API07-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API07-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API07-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API07-T06	Crear la interfaz de usuario.	4	Hecho
API02	API02-T01	Crear una clase Interface con operaciones necesarias para	2	Hecho

		realizar las tareas que se requiera. CRUD		
	API02-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API02-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API02-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API02-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API02-T06	Crear la interfaz de usuario.	4	Hecho
API06	API06-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API06-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API06-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho

	API06-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API06-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API06-T06	Crear la interfaz de usuario.	4	Hecho
API05	API05-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API05-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API05-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API05-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API05-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho

	API05-T06	Crear la interfaz de usuario.	4	Hecho
API10	API10-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API10-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API10-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API10-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API10-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API10-T06	Crear la interfaz de usuario.	4	Hecho
API08	API08-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API08-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho

	API08-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API08-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API08-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API08-T06	Crear la interfaz de usuario.	4	Hecho
API09	API09-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API09-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API09-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API09-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho

	API09-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API09-T06	Crear la interfaz de usuario.	4	Hecho
API11	API11-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API11-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API11-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API11-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API11-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API11-T06	Crear la interfaz de usuario.	4	Hecho
API04	API04-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho

	API04-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API04-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API04-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API04-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API04-T06	Crear la interfaz de usuario.	4	Hecho
API13	API13-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API13-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API13-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API13-T04	Crear una clase para realizar la implementación de la interface	2	Hecho

		con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)		
	API13-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API13-T06	Crear la interfaz de usuario.	4	Hecho
API12	API12-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API12-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API12-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API12-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API12-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller)	2	Hecho
	API12-T06	Crear la interfaz de usuario.	4	Hecho

API03	API03-T01	Crear una clase Interface con operaciones necesarias para realizar las tareas que se requiera. CRUD	2	Hecho
	API03-T02	Crear una clase para realizar la implementación de la Interface y extendiendo de la clase Genérica. Configuración de Spring MVC (@Repository)	4	Hecho
	API03-T03	Crear una clase interface con operaciones necesarias que servirá para crear servicios.	1	Hecho
	API03-T04	Crear una clase para realizar la implementación de la interface con la configuración de Spring MVC (@Service) esta es una clase transaccional (@Transactional)	2	Hecho
	API03-T05	Crear una clase que servirá de controlador con la configuración de Spring MVC (@Controller, @RestController)	4	Hecho
API15	API15-T01	Crear e implementar una clase que servirá de controlador con la configuración de Spring MVC (@RestController) y permita realizar la acción requerida.	4	Hecho
	API15-T02	Realizar la configuración para el cifrado de la contraseña del usuario	2	Hecho
API14	API14- T01	Crear e implementar las clases necesarias que permitan cumplir con la acción requerida.	8	Hecho

		Configuración de Spring Security Oauth 2		
API17	API17- T01	Crear e implementar una clase que servirá de controlador con la configuración de Spring MVC (@RestController) y permita realizar la acción requerida.	4	Hecho
API18	API18- T01	Crear e implementar una clase que servirá de controlador con la configuración de Spring MVC (@RestController) y permita realizar la acción requerida.	4	Hecho
API16	API16- T01	Crear e implementar una clase que servirá de controlador con la configuración de Spring MVC (@RestController) y permita realizar la acción requerida.	4	Hecho
API19	API19- T01	Crear e implementar una clase que servirá de controlador con la configuración de Spring MVC (@RestController) y permita realizar la acción requerida.	4	Hecho
API20	API20- T01	Crear e implementar una clase que servirá de controlador con la configuración de Spring MVC (@RestController) y permita realizar la acción requerida.	4	Hecho

Nota: Lista de tareas realizadas correctamente. Fuente: Elaboración propia.

Tabla 80.

Tasking y Spring del product backlog de la aplicación del cliente.

Requisito	Código Tarea	Descripción	Horas	Estado
-----------	-----------------	-------------	-------	--------

HU02	HU02-T01	Crear formulario de registro	2	Hecho
	HU02-T02	Programar el funcionamiento para poder mostrar la lista de países.	2	Hecho
	HU02-T03	Programar el funcionamiento para el registro de usuario.	4	Hecho
HU01	HU01-T01	Crear formulario de acceso a la aplicación.	2	Hecho
	HU01-T02	Programar el funcionamiento para realizar el acceso a la aplicación.	2	Hecho
HU09	HU09-T01	Crear pantalla con opciones del menú principal.	2	Hecho
	HU09-T02	Programar el funcionamiento de cada opción del menú principal.	2	Hecho
HU04	HU04-T01	Crear pantalla que muestre la lista de tipos de competiciones.	2	Hecho
	HU04-T02	Realizar la petición AJAX al servidor para solicitar los tipos de competiciones y mostrarlos en la pantalla.	2	Hecho
	HU04-T03	Programar la acción que tendrá cada tipo de competición al ser seleccionada.	2	Hecho
HU05	HU05-T01	Crear pantalla donde se mostrará la lista de las competiciones.	2	Hecho
	HU05-T02	Realizar la petición AJAX al servidor para solicitar la lista de competiciones según el tipo de competición seleccionada, y mostrarlas en la pantalla	2	Hecho
	HU05-T03	Realizar la acción que tendrá cada competición al ser seleccionada	2	Hecho

HU03	HU03-T01	Crear pantalla donde se mostrará la lista de partidos según la competición seleccionada.	2	Hecho
	HU03-T02	Realizar la petición AJAX al servidor para solicitar la lista de partidos según la competición seleccionada, y mostrarlos en pantalla	2	Hecho
	HU03-T03	Realizar la acción que tendrá cada partido al ser seleccionado.	2	Hecho
HU06	HU06-T01	Crear la pantalla donde se mostrarán los pagos que existen según el partido seleccionado, esto ayudará a saber qué tipo de apuesta se realizará.	2	Hecho
	HU06-T02	Realizar la petición AJAX al servidor para solicitar la lista de pagos según el partido seleccionado, y mostrarlos en pantalla	2	Hecho
	HU06-T03	Realizar la opción que tendrá cada monto de pago al ser seleccionado.	2	Hecho
HU10	HU10-T01	Crear la pantalla con el formulario para ingresar datos para realizar una apuesta, este aparece luego de seleccionar un monto de pago de partido	1	Hecho
	HU10-T02	Mostrar los datos que deben aparecer por defecto, como el tipo de apuesta, el monto del pago, el partido, entre otros que se crean convenientes	1	Hecho

HU08	HU08-T1	Al ingresar un monto de apuesta, se tiene que calcular el total de la apuesta, siendo el pago de apuesta por el monto apostado.	1	Hecho
HU07	HU07-T01	Validar que los campos ingresados en el formulario sean correctos.	2	Hecho
	HU07-T02	Realizar la acción del botón APOSTAR mediante AJAX al servidor para almacenarla en la base de datos.	2	Hecho
	HU07-T03	Después de realizar una apuesta exitosa, se debe de redirigir hacia la pantalla de tipos de competiciones para poder realizar el proceso en caso se requiera realizar otra apuesta.	1	Hecho

Nota: Lista de tareas realizadas correctamente. Fuente: Elaboración propia.

3.3.3.2. Arquitectura de la Api RestFul Golwin

En esta parte para poder implementar la API se utilizaron los siguientes frameworks, Spring MVC para la implementación, configuración de la arquitectura de la aplicación y la lógica del negocio, además permite la integración con otros frameworks, se utilizó también Spring Security Oauth2 para asegurar la API, el framework Hibernate para el mapeo de las tablas y para la interacción con la base de datos, se utilizó JSON (JavaScript Object Notation) como formato de texto ligero para el intercambio de datos. Estas tecnologías tenemos que agregarlas como dependencias para el proyecto en el archivo de configuración correspondiente para que se puedan agregar a nuestro proyecto y para poder utilizarlas.

Una vez implementado todo, el funcionamiento del API es de la siguiente manera.

En el lado del cliente, en este caso las aplicaciones desarrolladas con Ionic y JQuery Mobile está conectada a la web la cual hace una solicitud hacia la API que está en el lado del servidor, primeramente tiene pasar por la capa de seguridad la cual esta implementada con Spring Security Oauth2, la solicitud está dirigida hacia el servidor de recursos que es el encargado de recibir, procesar y devolver lo que el cliente está solicitando, pero spring security oauth2 hace una verificación de si el usuario que está haciendo la solicitud tiene autorización para poder acceder al recurso solicitado, en caso que no esté autorizado no podrá tener acceso al recurso solicitado, por lo tanto el usuario tendrá que autenticarse en el servidor de autorización, realizado con éxito la autenticación, el servidor de autorización retorna el token de acceso, este token es el que nos va a permitir tener acceso a los recursos solicitados, cabe resaltar que mediante este token de acceso el usuario solo tendrá que enviar sus credenciales una sola vez (en el login, usuario y contraseña), posteriormente con cada solicitud se envía el token de acceso junto con la URL de la solicitud.

Autenticado y obtenido el token de acceso tenemos que guardar este token de acceso para realizar las solicitudes hacia el servidor de recursos, para cada solicitud se tiene que agregar este token de acceso a la URL. El servidor de recursos se encarga de recibir, procesar y devolver el recurso solicitado por el cliente, para esto se tiene que comunicar con el controlador (@RestController) que es el encargado de procesar la petición HTTP hacia el método correspondiente y así obtener una respuesta apropiada, este pide los recursos hacia la capa de servicios (@Service) y este solicita a la capa de los DAOs (@Repository) donde se encuentra toda la lógica para poder obtener los recursos que se solicitan. (Desde el controlador hasta los DAOs esta implementado con el framework Spring MVC.) Para poder hacer su tarea tiene que comunicarse con las entidades las cuales están mapeadas con las tablas de la base de datos, este mapeo lo implementa el framework Hibernate el cual nos facilita las operaciones que se realizan hacia la base de datos, si todo el proceso de solicitud a tenido éxito y no ha ocurrido algún error, el servidor responde a la solicitud realizada por el cliente con los recursos que se solicitaron, estos

recursos (objetos) se devuelven en formato JSON, el cliente recibe este (os) objeto (s) en formato JSON, el cual tiene que procesarlos para poder mostrarlos y/o realizar alguna operación que crea conveniente, siempre y cuando la API lo permita.

En el siguiente grafico ejemplifica lo explicado anteriormente, se muestra una arquitectura de cómo es el funcionamiento de la API Rest Golwin.

ARQUITECTURA API REST GOLWIN

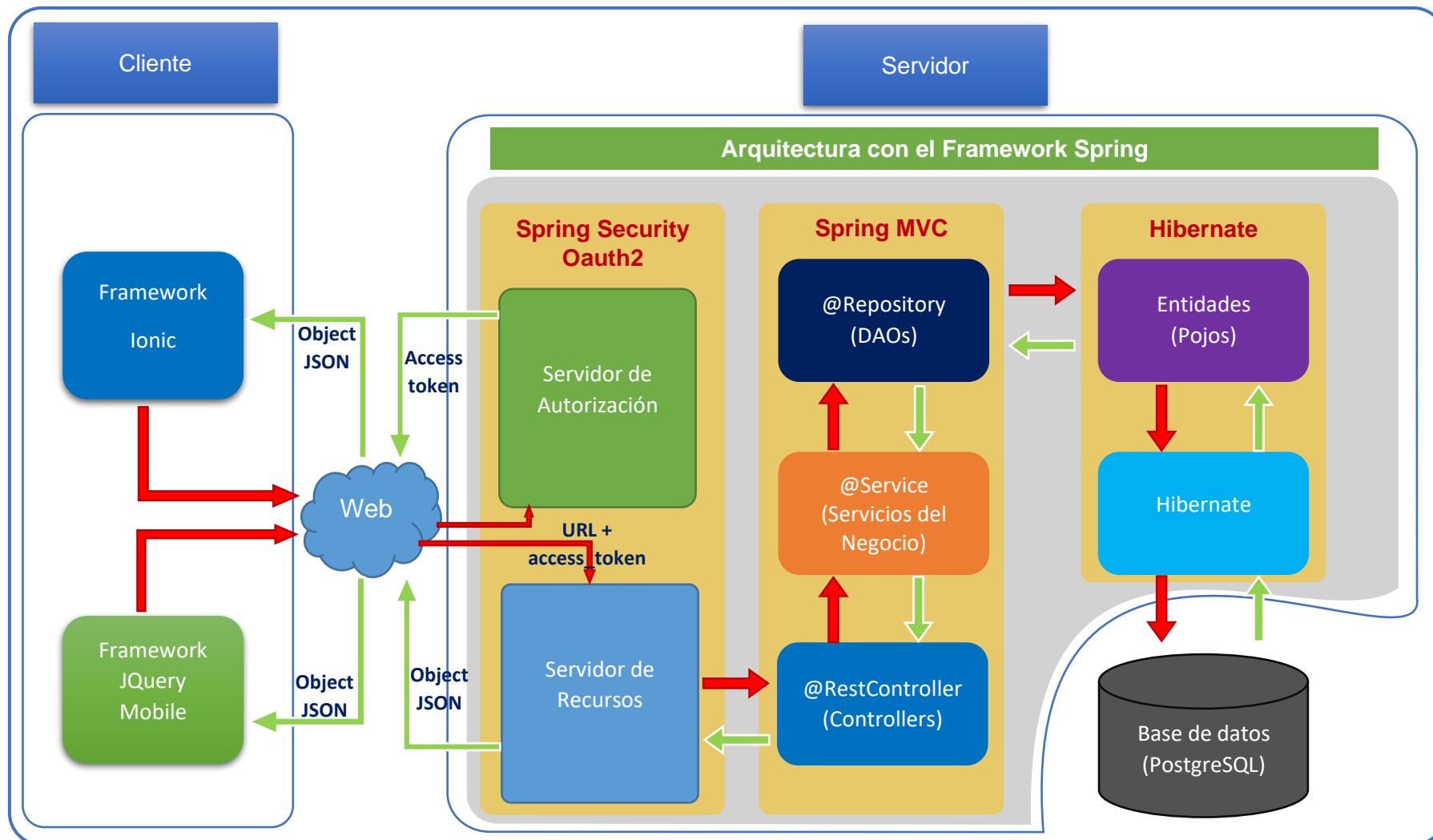


Figura 26. Arquitectura de cómo funciona la API Rest Golwin, proceso que se realiza desde la petición del cliente hacia el servidor y como este retorna los recursos en objetos JSON. Fuente: Elaboración propia.

Se explicará cada ítem del gráfico para entender como es el funcionamiento en cada uno. Empezaremos explicando cada ítem del lado del servidor, luego explicaremos los ítems del lado del cliente. En el lado del servidor se utilizaron los frameworks como son Spring MVC, Hibernate, Spring Security, Spring Security Oauth2.

3.3.3.3. Base de datos

La base de datos está implementada en PostgreSQL. A continuación, se muestra el diagrama relacional del Negocio.

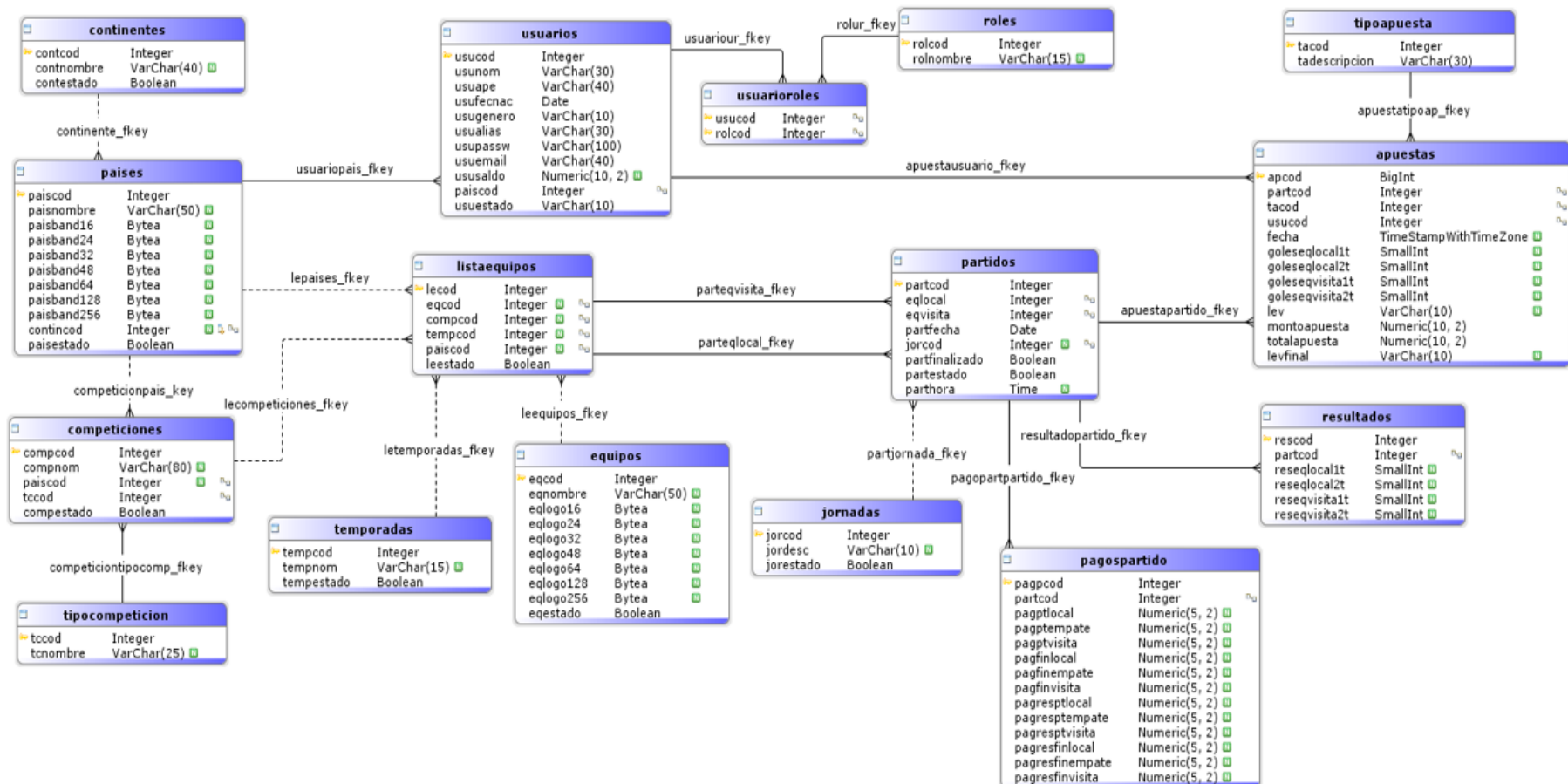


Figura 27. Diagrama relacional de la base de datos Golwin. Fuente: Elaboración propia

El siguiente diagrama muestra las tablas que se necesitan para implementar el protocolo de autorización OAuth 2. En la aplicación del servidor se implementará con Spring Security OAuth 2.

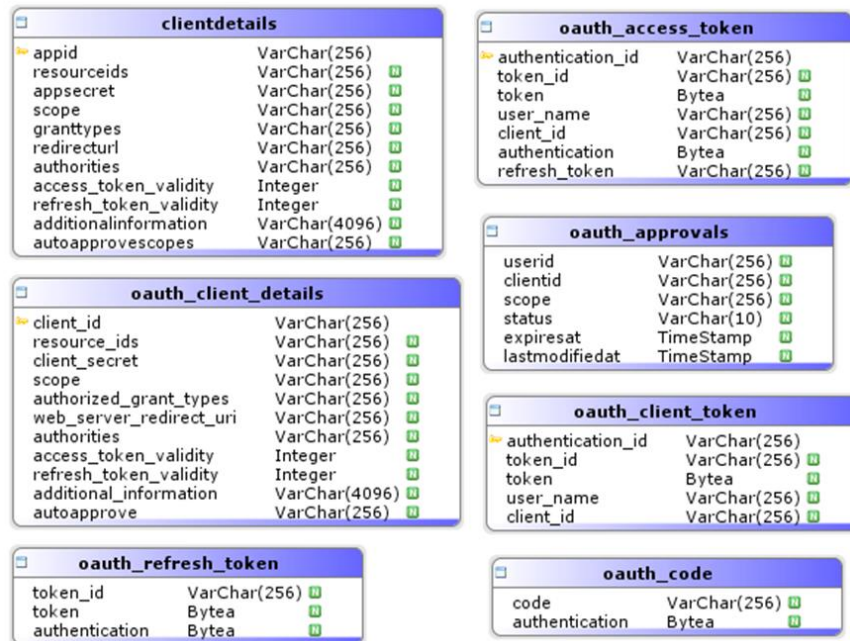


Figura 28. Tablas para implementar el protocolo de seguridad OAuth 2. Fuente: Elaboración propia.

3.3.3.4. Hibernate

Para poder trabajar con Hibernate debemos agregar las dependencias a nuestro proyecto en el archivo pom.xml tal como se muestra en la siguiente imagen.

```

12 <properties>
13   <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
14   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15   <hibernate.version>4.3.11.Final</hibernate.version>
16 </properties>
17
18 <dependencies>
19   <!-- Hibernate -->
20   <dependency>
21     <groupId>org.hibernate</groupId>
22     <artifactId>hibernate-core</artifactId>
23     <version>${hibernate.version}</version>
24   </dependency>

```

Figura 29. Dependencias necesarias para Hibernate. Fuente: Elaboración propia.

También debemos agregar las dependencias requeridas para conectarnos con nuestro motor de base de datos, en este caso utilizaremos PostgreSQL.

```

12 <properties>
13   <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
14   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15   <postgresql.version>9.2-1002-jdbc4</postgresql.version>
16 </properties>
17
18 <dependencies>
19   <!-- Postgresql -->
20   <dependency>
21     <groupId>org.postgresql</groupId>
22     <artifactId>postgresql</artifactId>
23     <version>${postgresql.version}</version>
24   </dependency>
25 </dependencies>

```

Figura 30. Dependencia para PostgreSQL. Fuente: Elaboración propia.

Ahora crearemos nuestra clase de configuración de hibernate para conectarnos a nuestra base de datos. La configuración la haremos utilizando programación java y no en archivos xml.

```

HibernateConfig.java
Source History
1 package com.golwin.application.config;
2
3 import java.util.Properties;
4 import javax.sql.DataSource;
5 import org.hibernate.SessionFactory;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.context.annotation.Bean;
8 import org.springframework.context.annotation.ComponentScan;
9 import org.springframework.context.annotation.Configuration;
10 import org.springframework.context.annotation.PropertySource;
11 import org.springframework.core.env.Environment;
12 import org.springframework.jdbc.datasource.DriverManagerDataSource;
13 import org.springframework.orm.hibernate4.HibernateTransactionManager;
14 import org.springframework.orm.hibernate4.LocalSessionFactoryBean;
15 import org.springframework.transaction.annotation.EnableTransactionManagement;
16
17 /**
18  *
19  * @author KRISTIAN IC
20  */
21 @Configuration → A
22 @EnableTransactionManagement → B
23 @ComponentScan({"com.golwin.application.config"}) → C
24 @PropertySource(value = {"classpath:jdbc.properties"}) → D
25 public class HibernateConfig {
26
27   @Autowired → E
28   private Environment environment;
29
30   @Bean → F
31   public LocalSessionFactoryBean sessionFactory() {
32     LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
33     sessionFactory.setDataSource(dataSource());
34     sessionFactory.setPackagesToScan(new String[] {
35       "com.golwin.application.model"});
36     sessionFactory.setHibernateProperties(getHibernateProperties());
37     return sessionFactory;
38   }
39 }

```

Annotations A through G are placed over the following code elements:

- A: @Configuration
- B: @EnableTransactionManagement
- C: @ComponentScan
- D: @PropertySource
- E: @Autowired
- F: @Bean
- G: The entire sessionFactory() method.

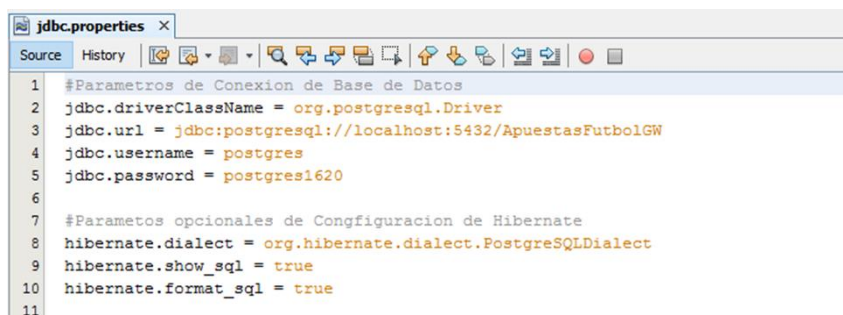
```

39
40
41 @Bean
42 public DataSource dataSource() {
43     DriverManagerDataSource dataSource = new DriverManagerDataSource();
44     dataSource.setDriverClassName(environment
45         .getRequiredProperty("jdbc.driverClassName"));
46     dataSource.setUrl(environment
47         .getRequiredProperty("jdbc.url"));
48     dataSource.setUsername(environment
49         .getRequiredProperty("jdbc.username"));
50     dataSource.setPassword(environment
51         .getRequiredProperty("jdbc.password"));
52     return dataSource;
53 }
54
55 private Properties getHibernateProperties() {
56     Properties properties = new Properties();
57     properties.put("hibernate.dialect",
58         environment.getRequiredProperty("hibernate.dialect"));
59     properties.put("hibernate.show_sql",
60         environment.getRequiredProperty("hibernate.show_sql"));
61     properties.put("hibernate.format_sql",
62         environment.getRequiredProperty("hibernate.format_sql"));
63     return properties;
64 }
65
66 @Bean
67 @Autowired
68 public HibernateTransactionManager transactionManager(
69     SessionFactory sessionFactory) {
70     HibernateTransactionManager txManager = new HibernateTransactionManager();
71     txManager.setSessionFactory(sessionFactory);
72     return txManager;
73 }

```

Figura 31. Clase de configuración de Hibernate. Fuente: Elaboración propia.

- A. Anotación @Configuration: indica que esta clase contiene uno o más métodos Bean anotados con @Bean, en nuestro caso esta clase representa la configuración de hibernate.
- B. Anotación @EnableTransactionManagement: permite la capacidad de gestión de transacciones de Spring basada en anotaciones.
- C. Anotación @ComponentScan: proporciona donde buscar los beans/clases administrados por spring.
- D. Anotación @PropertySource: se utiliza para declarar un conjunto de propiedades (definido en un archivo properties en el classpath de la aplicación) en el entorno de tiempo de ejecución de Spring, proporcionando flexibilidad para tener valores diferentes en diferentes entornos de aplicaciones. A continuación, se muestra el archivo de propiedades utilizado, se encuentra en la siguiente ruta del proyecto: */src/main/resources/jdbc.properties*



```
1 #Parametros de Conexion de Base de Datos
2 jdbc.driverClassName = org.postgresql.Driver
3 jdbc.url = jdbc:postgresql://localhost:5432/ApuestasFutbolGW
4 jdbc.username = postgres
5 jdbc.password = postgres1620
6
7 #Parametros opcionales de Configuracion de Hibernate
8 hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
9 hibernate.show_sql = true
10 hibernate.format_sql = true
11
```

Figura 32. Archivo de propiedades. Fuente: Elaboración propia.

- E. Anotación `@Autowired`: marca un constructor, un campo, un método setter o un método de configuración para ser auto-conectado por las instalaciones de inyección de dependencias de Spring.
- F. Anotación `@Bean`: se utiliza para explícitamente declarar un solo bean, en lugar que spring lo haga automáticamente. Devuelve un objeto que spring debe registrar como bean en el contexto de la aplicación.
- G. Métodos de configuración: el método `sessionFactory()` está creando un `LocalSessionFactoryBean`, que refleja exactamente la configuración basada en XML; necesitamos un `dataSource` (método `dataSource()`) y propiedades de hibernate (método `getHibernateProperties()`). Gracias a `@PropertySource`, podemos externalizar los valores reales en un archivo `".properties"` y usar el entorno de Spring para obtener el valor correspondiente a un elemento. Una vez que se crea `SessionFactory`, se inyectará en el Bean método `transactionManager()`, que eventualmente puede proporcionar soporte de transacción para las sesiones creadas por `sessionFactory`.

3.3.3.4.1. Entidades

Hibernate es un framework de mapeo objeto-relacional (ORM) que nos permite hacer el mapeo entre los atributos de una tabla de la base de datos y los atributos de una clase del modelo de objetos de la aplicación (Entidades/pojos), para la persistencia de los objetos utilizaremos anotaciones en los atributos de las entidades para establecer las relaciones.

Luego procederemos a crear nuestra(s) clase(s) donde haremos el mapeo entre los atributos la clase y los atributos de la tabla(s) de la base de datos. Se

mostrará la implementación de una clase de mapeo para caso de ejemplo, además, se agregarán algunos atributos de otras clases para explicar su funcionamiento.

A continuación, se muestran una serie de figuras, todas pertenecen a la clase Partidos en la cual se hizo el mapeo de la tabla Partido de nuestra base de datos. En cada imagen iremos explicando algunas cosas importantes, como son las anotaciones.

```
1 package com.golwin.application.model;
2
3 import java.util.Date;
4 import java.util.HashSet;
5 import java.util.Set;
6 import javax.persistence.Column;
7 import javax.persistence.Entity;
8 import javax.persistence.FetchType;
9 import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12 import javax.persistence.JoinColumn;
13 import javax.persistence.ManyToOne;
14 import javax.persistence.OneToMany;
15 import javax.persistence.Table;
16 import javax.persistence.Temporal;
17 import javax.persistence.TemporalType;
18 import org.springframework.format.annotation.DateTimeFormat;
19 import org.springframework.format.annotation.DateTimeFormat.ISO;
20
21 /**
22  * Partidos generated by hbm2java
23  */
24 @Entity
25 @Table(name = "partidos", schema = "public")
26 public class Partidos implements java.io.Serializable {
27
28     private int partcod;
29     private Jornadas jornadas;
30     private Listaequipos listaequiposByEqlocal;
31     private Listaequipos listaequiposByEqvisita;
32     private Date partfecha;
33     private Date parthora;
34     private boolean partfinalizado;
35     private boolean partestado;
36     private Set<Pagospartido> pagospartidos = new HashSet<Pagospartido>(0);
37     private Set<Apuestas> apuestas = new HashSet<Apuestas>(0);
38     private Set<Resultados> resultados = new HashSet<Resultados>(0);
39 }
```

The image shows a screenshot of an IDE with the code for the `Partidos.java` class. The code is annotated with red brackets and letters A, B, C, and D. Bracket A groups the import statements. Bracket B points to the `@Entity` annotation. Bracket C points to the `@Table` annotation. Bracket D groups the private attributes of the class.


```

39
40 public Partidos() {
41 }
42
43 public Partidos(int partcod, Listaequipos listaequiposByEqlocal,
44 Listaequipos listaequiposByEqvilita, Date partfecha,
45 Date parthora, boolean partfinalizado, boolean partestado) {
46 this.partcod = partcod;
47 this.listaequiposByEqlocal = listaequiposByEqlocal;
48 this.listaequiposByEqvilita = listaequiposByEqvilita;
49 this.partfecha = partfecha;
50 this.parthora = parthora;
51 this.partfinalizado = partfinalizado;
52 this.partestado = partestado;
53 }
54
55 public Partidos(int partcod, Jornadas jornadas,
56 Listaequipos listaequiposByEqlocal,
57 Listaequipos listaequiposByEqvilita, Date partfecha, Date parthora,
58 boolean partfinalizado, boolean partestado,
59 Set<Pagospartido> pagospartidos, Set<Apuestas> apuestas,
60 Set<Resultados> resultados) {
61 this.partcod = partcod;
62 this.jornadas = jornadas;
63 this.listaequiposByEqlocal = listaequiposByEqlocal;
64 this.listaequiposByEqvilita = listaequiposByEqvilita;
65 this.partfecha = partfecha;
66 this.parthora = parthora;
67 this.partfinalizado = partfinalizado;
68 this.partestado = partestado;
69 this.pagospartidos = pagospartidos;
70 this.apuestas = apuestas;
71 this.resultados = resultados;
72 }
73
74 @Id → F
75 @GeneratedValue(strategy = GenerationType.IDENTITY) → G
76 @Column(name = "partcod", unique = true, nullable = false) → H
77 public int getPartcod() {
78     return this.partcod;
79 }
80
81 public void setPartcod(int partcod) {
82     this.partcod = partcod;
83 }
84
85 @ManyToOne(fetch = FetchType.LAZY) → J
86 @JoinColumn(name = "jorccod") → K
87 public Jornadas getJornadas() {
88     return this.jornadas;
89 }
90
91 public void setJornadas(Jornadas jornadas) {
92     this.jornadas = jornadas;
93 }
94
95 @ManyToOne(fetch = FetchType.LAZY)
96 @JoinColumn(name = "eqlocal", nullable = false)
97 public Listaequipos getListaequiposByEqlocal() {
98     return this.listaequiposByEqlocal;
99 }
100
101 public void setListaequiposByEqlocal(Listaequipos listaequiposByEqlocal) {
102     this.listaequiposByEqlocal = listaequiposByEqlocal;
103 }
104
105 @ManyToOne(fetch = FetchType.LAZY)
106 @JoinColumn(name = "eqvilita", nullable = false)
107 public Listaequipos getListaequiposByEqvilita() {
108     return this.listaequiposByEqvilita;
109 }
110
111 public void setListaequiposByEqvilita(Listaequipos listaequiposByEqvilita) {
112     this.listaequiposByEqvilita = listaequiposByEqvilita;
113 }
114
115 @DateTimeFormat(iso = ISO.DATE) → L
116 @Temporal(TemporalType.DATE) → M
117 @Column(name = "partfecha", nullable = false, length = 13)
118 public Date getPartfecha() {
119     return this.partfecha;
120 }
121
122 public void setPartfecha(Date partfecha) {
123     this.partfecha = partfecha;
124 }
125

```

E

F

G

H

I

J

K

L

M

```

126 @DateTimeFormat(iso = ISO.TIME, pattern = "HH:mm") → N
127 @Temporal(TemporalType.TIME) → O
128 @Column(name = "parthora", length = 15)
129 public Date getParthora() {
130     return this.parthora;
131 }
132
133 public void setParthora(Date parthora) {
134     this.parthora = parthora;
135 }
136
137 @Column(name = "partfinalizado", nullable = false)
138 public boolean isPartfinalizado() {
139     return this.partfinalizado;
140 }
141
142 public void setPartfinalizado(boolean partfinalizado) {
143     this.partfinalizado = partfinalizado;
144 }
145
146 @Column(name = "partestado", nullable = false)
147 public boolean isPartestado() {
148     return this.partestado;
149 }
150
151 public void setPartestado(boolean partestado) {
152     this.partestado = partestado;
153 }
154
155 @OneToMany(fetch = FetchType.LAZY, mappedBy = "partidos") → P
156 public Set<Pagospartido> getPagospartidos() {
157     return this.pagospartidos;
158 }
159
160 public void setPagospartidos(Set<Pagospartido> pagospartidos) {
161     this.pagospartidos = pagospartidos;
162 }
163
164 @OneToMany(fetch = FetchType.LAZY, mappedBy = "partidos")
165 public Set<Apuestas> getApuestas() {
166     return this.apuestas;
167 }
168
169 public void setApuestas(Set<Apuestas> apuestas) {
170     this.apuestas = apuestas;
171 }
172
173 @OneToMany(fetch = FetchType.LAZY, mappedBy = "partidos")
174 public Set<Resultados> getResultadoses() {
175     return this.resultadoses;
176 }
177
178 public void setResultadoses(Set<Resultados> resultadoses) {
179     this.resultadoses = resultadoses;
180 }
181
182 }

```

Figura 33. Mapeo de la tabla Partidos en la clase Partidos. Fuente: Elaboración propia.

Explicación de los ítems que se encuentran en la clase:

A. import: se importan las clases que se requieran

B. Anotación @Entity: declara una clase como una entidad, es decir, una clase POJO persistente, por lo que debe tener un constructor sin argumentos que sea visible con al menos el ámbito protegido.

C. Anotación @Table: se establece en el nivel de la clase, permite especificar los detalles de la tabla que se utiliza para guardar la entidad en la base de datos. proporciona atributos como el nombre de la tabla, su catálogo y esquema,

además permite aplicar restricciones únicas en las columnas de la tabla. Por ejemplo, `uniqueConstraints={@UniqueConstraint(columnNames = "usualias"), @UniqueConstraint(columnNames = "usuemail")}`

- D. Atributos: atributos de la clase, las cuales también se encuentran en la tabla de la base de datos.
- E. Constructores: métodos de la clase para inicializar los objetos.
- F. Anotación `@Id`: permite definir que propiedad es el identificador de la entidad.
- G. Anotación `@GeneratedValue`: permite definir la estrategia de generación de identificadores.
- H. Anotación `@Column`: permite definir los detalles de la columna a la que se asignaran un campo o propiedad de una tabla de la base de datos.
- I. Getter y Setter: métodos `get` y `set` de cada propiedad de la clase.
- J. Anotación `@ManyToOne`: permite definir una relación de muchos a uno
- K. Anotación `@JoinColumn`: permite definir la columna de unión de la relación.
- L. Anotación `@DateTimeFormat`: permite definir un parámetro de campo debe tener el formato de fecha u hora, en este caso es de fecha (DATE).
- M. Anotación `@Temporal`: permite definir la precisión temporal del tiempo que se espera en la base de datos. Los datos temporales pueden tener precisión DATE, TIME o TIMESTAMP. En este caso es de fecha (DATE)
- N. Anotación `@DateTimeFormat`: permite definir un parámetro de campo debe tener el formato de fecha u hora, en este caso es de hora (TIME).
- O. Anotación `@Temporal`: permite definir la precisión temporal del tiempo que se espera en la base de datos. Los datos temporales pueden tener precisión DATE, TIME o TIMESTAMP. En este caso es de hora (TIME)
- P. Anotación `@OneToMany`: Permite definir una relación de uno a muchos.

3.3.3.5. Spring MVC

Para poder trabajar con el framework Spring MVC debemos agregar las dependencias de Spring MVC a nuestro proyecto en el archivo *pom.xml*, tal como se muestra en la siguiente imagen.

```

12 <properties>
13   <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
14   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15   <springframework.version>4.3.1.RELEASE</springframework.version>
16 </properties>
17
18 <dependencies>
19   <!--Spring-->
20   <dependency>
21     <groupId>org.springframework</groupId>
22     <artifactId>spring-core</artifactId>
23     <version>${springframework.version}</version>
24   </dependency>
25   <dependency>
26     <groupId>org.springframework</groupId>
27     <artifactId>spring-web</artifactId>
28     <version>${springframework.version}</version>
29   </dependency>
30   <dependency>
31     <groupId>org.springframework</groupId>
32     <artifactId>spring-webmvc</artifactId>
33     <version>${springframework.version}</version>
34   </dependency>
35   <dependency>
36     <groupId>org.springframework</groupId>
37     <artifactId>spring-tx</artifactId>
38     <version>${springframework.version}</version>
39   </dependency>
40   <dependency>
41     <groupId>org.springframework</groupId>
42     <artifactId>spring-orm</artifactId>
43     <version>${springframework.version}</version>
44   </dependency>
45 </dependencies>

```

Figura 34. Dependencias necesarias para Spring MVC. Fuente: Elaboración propia.

También es necesario agregar las dependencias de Java Servlet que nos van a servir para ampliar las capacidades del servidor, también agregaremos dependencias para Logger, tal como se muestra a continuación.

```

12 <properties>
13   <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
14   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15   <slf4j.version>1.7.6</slf4j.version>
16   <log4j.version>1.2.17</log4j.version>
17 </properties>
18
19 <dependencies>
20   <!-- Servlet -->
21   <dependency>
22     <groupId>javax.servlet</groupId>
23     <artifactId>javax.servlet-api</artifactId>
24     <version>3.1.0</version>
25   </dependency>
26   <!-- Logger -->
27   <dependency>
28     <groupId>log4j</groupId>
29     <artifactId>log4j</artifactId>
30     <version>${log4j.version}</version>
31   </dependency>
32   <dependency>
33     <groupId>org.slf4j</groupId>
34     <artifactId>slf4j-log4j12</artifactId>
35     <version>${slf4j.version}</version>
36   </dependency>
37   <dependency>
38     <groupId>org.slf4j</groupId>
39     <artifactId>slf4j-api</artifactId>
40     <version>${slf4j.version}</version>
41   </dependency>
42 </dependencies>

```

Figura 35. Dependencias necesarias para Servlet y Logger. Fuente: Elaboración propia.

Ahora crearemos nuestra clase de configuración de Spring. La configuración la haremos utilizando programación java y no en archivos xml.

```

1 package com.golwin.application.config;
2
3 import com.fasterxml.jackson.databind.SerializationFeature;
4 import com.fasterxml.jackson.datatype.hibernate4.Hibernate4Module;
5 import com.golwin.application.converter.RoleToUserProfileConverter;
6 import java.util.List;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.context.annotation.ComponentScan;
9 import org.springframework.context.annotation.Configuration;
10 import org.springframework.format.FormatterRegistry;
11 import org.springframework.http.converter.HttpMessageConverter;
12 import org.springframework.http.converter.json.Jackson2ObjectMapperBuilder;
13 import org.springframework.http.converter.json.MappingJackson2HttpMessageConverter;
14 import org.springframework.web.servlet.config.annotation.EnableWebMvc;
15 import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
16
17 /**
18  *
19  * @author KRISTIAN IC
20  */
21 @Configuration
22 @EnableWebMvc
23 @ComponentScan(basePackages = "com.golwin.application")
24 public class AppConfig extends WebMvcConfigurerAdapter {
25
26     @Autowired
27     RoleToUserProfileConverter roleToUserProfileConverter;
28
29     @Override
30     public void addFormatters(FormatterRegistry registry) {
31         registry.addConverter(roleToUserProfileConverter);
32     }
33     private Hibernate4Module hibernate4Module() {
34         return new Hibernate4Module().disable(Hibernate4Module.Feature.USE_TRANSIENT_ANNOTATION);
35     }
36     public MappingJackson2HttpMessageConverter jacksonMessageConverter() {
37         Jackson2ObjectMapperBuilder builder = new Jackson2ObjectMapperBuilder()
38             .featuresToDisable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS)
39             .modulesToInstall(hibernate4Module());
40         return new MappingJackson2HttpMessageConverter(builder.build());
41     }
42     @Override
43     public void configureMessageConverters(List<HttpMessageConverter<?>> converters) {
44         converters.add(jacksonMessageConverter());
45         super.configureMessageConverters(converters);
46     }
47 }

```

Figura 36. Clase de configuración de Spring MVC. Fuente: Elaboración propia.

- A. Anotación `@Configuration`: anotación que representa la configuración de Spring MVC.
- B. Anotación `@EnableWebMvc`: habilita las configuraciones por defecto de los componentes de Spring MVC. Permite soporte para las clases anotadas con `@Controller`/`@RestController` que utilizan `@RequestMapping` para mapear las solicitudes entrantes a un método específico.
- C. Anotación `@ComponentScan`: Habilita los elementos que tengan las anotaciones `@Component`, `@Repository`, `@Service` y `@Controller`/`@RestController`, mediante el atributo "basePackages" proporciona donde buscar los beans/clases (componentes) administrados por spring.
- D. Métodos para convertir la respuesta en el formato deseado; Spring utiliza `HttpMessageConverters` para convertir la respuesta en el formato deseado, en este caso en formato JSON. `MappingJackson2HttpMessageConverter` es

una implementación de `HttpMessageConverter` que puede leer y escribir JSON usando `ObjectMapper Jackson`. De forma predeterminada, este convertidor admite (`application/json`). Con el fin de servir JSON vamos a utilizar *Jackson library*, debemos agregar las dependencias a nuestro archivo `pom.xml` como se muestra a continuación.

```

12 <properties>
13 <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
14 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15 <jackson.library>2.8.3</jackson.library>
16 </properties>
17
18 <dependencies>
19 <!-- jackson -->
20 <dependency>
21 <groupId>com.fasterxml.jackson.core</groupId>
22 <artifactId>jackson-annotations</artifactId>
23 <version>${jackson.library}</version>
24 </dependency>
25 <dependency>
26 <groupId>com.fasterxml.jackson.core</groupId>
27 <artifactId>jackson-databind</artifactId>
28 <version>${jackson.library}</version>
29 </dependency>
30 <dependency>
31 <groupId>com.fasterxml.jackson.datatype</groupId>
32 <artifactId>jackson-datatype-hibernate4</artifactId>
33 <version>${jackson.library}</version>
34 </dependency>
35 </dependencies>

```

Figura 37. Dependencias para Jackson library. Fuente: Elaboración propia.

Para que Spring pueda ejecutarse correctamente debemos añadir una clase inicializadora extendida de `AbstractAnnotationConfigDispatcherServletInitializer`, esta clase se carga y crea una instancia de la clase de configuración `AppConfig.class` tal como se muestra a continuación.

```

1 package com.golwin.application.config;
2
3 import javax.servlet.Filter;
4 import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;
5
6 /**
7  *
8  * @author KRISTIAN IC
9  */
10 public class AppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
11
12     @Override
13     protected Class<?>[] getRootConfigClasses() {
14         return new Class[] {AppConfig.class};
15     }
16
17     @Override
18     protected Class<?>[] getServletConfigClasses() {
19         return null;
20     }
21
22     @Override
23     protected String[] getServletMappings() {
24         return new String[] {"/"};
25     }
26
27     @Override
28     protected Filter[] getServletFilters() {
29         Filter[] singleton = {new CORSFilter()};
30         return singleton;
31     }
32 }

```

Figura 38. Clase inicializadora de Spring MVC. Fuente: Elaboración propia.

1. Agregamos un filtro para el soporte CORS, añadimos nuestra clase de configuración de filtro de CORS a Spring.

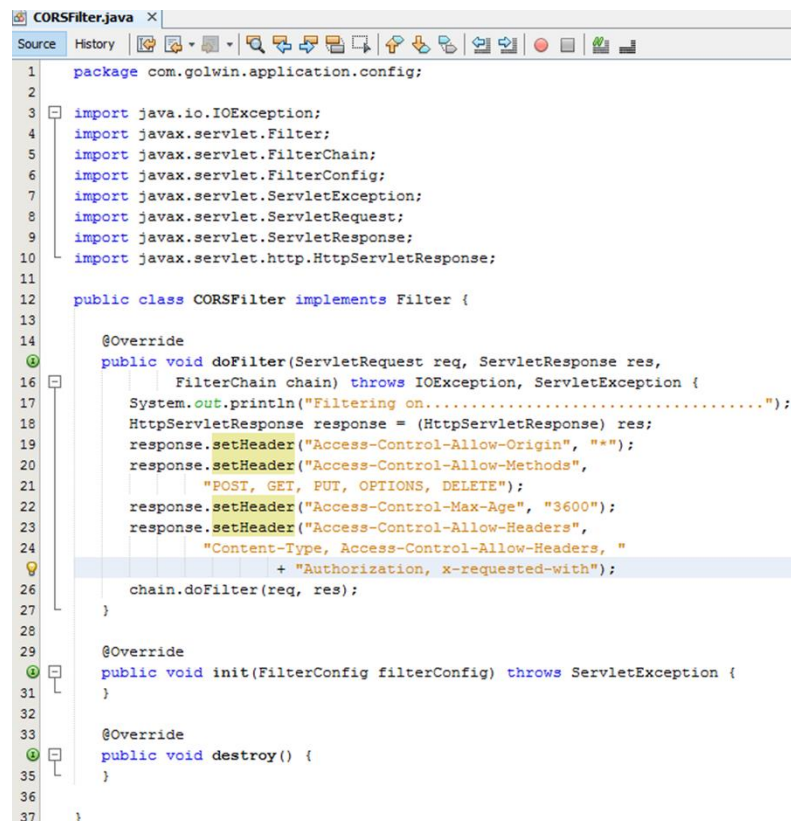
Al acceder a la API REST es posible enfrentarse con problemas relacionados con la misma política de origen.

Errores como:

“No ‘Access-Control-Allow-Origin’ header is present on the requested resource. Origin ‘http://127.0.0.1:8080’ is therefore not allowed access.” O “XMLHttpRequest cannot load http://abc.com/bla. Origin http://localhost:12345 is not allowed by Access-Control-Allow-Origin.” are common in such case.

La solución es el *intercambio de recursos de origen cruzado*. Básicamente, en el lado del servidor, podemos devolver encabezados (headers) de control de acceso CORS adicionales con la respuesta, lo que permitirá finalmente una mayor comunicación entre dominios.

Con Spring podemos escribir un filtro simple que agrega los encabezados específicos de CORS con cada respuesta, tal como se muestra a continuación.



```
1 package com.golwin.application.config;
2
3 import java.io.IOException;
4 import javax.servlet.Filter;
5 import javax.servlet.FilterChain;
6 import javax.servlet.FilterConfig;
7 import javax.servlet.ServletException;
8 import javax.servlet.ServletRequest;
9 import javax.servlet.ServletResponse;
10 import javax.servlet.http.HttpServletResponse;
11
12 public class CORSFilter implements Filter {
13
14     @Override
15     public void doFilter(ServletRequest req, ServletResponse res,
16         FilterChain chain) throws IOException, ServletException {
17         System.out.println("Filtering on.....");
18         HttpServletResponse response = (HttpServletResponse) res;
19         response.setHeader("Access-Control-Allow-Origin", "*");
20         response.setHeader("Access-Control-Allow-Methods",
21             "POST, GET, PUT, OPTIONS, DELETE");
22         response.setHeader("Access-Control-Max-Age", "3600");
23         response.setHeader("Access-Control-Allow-Headers",
24             "Content-Type, Access-Control-Allow-Headers, "
25             + "Authorization, x-requested-with");
26         chain.doFilter(req, res);
27     }
28
29     @Override
30     public void init(FilterConfig filterConfig) throws ServletException {
31     }
32
33     @Override
34     public void destroy() {
35     }
36
37 }
```

Figura 39. Implementación de la clase para CORS.Fuente: Elaboración propia.

3.3.3.5.1. DAOs (@Repository)

En esta capa se crearán las clases de persistencia que se encargarán de interactuar con la base de datos, se utiliza el framework Hibernate para las operaciones hacia la base de datos.

Crearemos una clase abstracta llamada “AbstractDao” con los métodos que más vamos a utilizar, para de esta manera poder utilizarla en las demás clases extendiéndola para utilizar sus métodos y así ahorrarnos código repetitivo.

```
1 package com.golwin.application.repository.dao;
2
3 import java.io.Serializable;
4 import java.lang.reflect.ParameterizedType;
5 import org.hibernate.Criteria;
6 import org.hibernate.Session;
7 import org.hibernate.SessionFactory;
8 import org.springframework.beans.factory.annotation.Autowired;
9 /**
10  *
11  * @author KRISTIAN IC
12  * @param <PK> Tipo Dato
13  * @param <T> Entidad
14  */
15 @
16 public abstract class AbstractDao<PK extends Serializable, T> {
17
18     private final Class<T> persistentClass;
19
20     @SuppressWarnings("unchecked")
21     public AbstractDao() {
22         this.persistentClass = (Class<T>) ((ParameterizedType) this.getClass()
23             .getGenericSuperclass()).getActualTypeArguments()[1];
24     }
25     @Autowired
26     private SessionFactory sessionFactory; } A
27
28     protected Session getSession() {
29         return sessionFactory.getCurrentSession(); } B
30
31     @SuppressWarnings("unchecked")
32     public T getByKey(PK key) {
33         return (T) getSession().get(persistentClass, key);
34     }
35     public void persist(T entity) {
36         getSession().persist(entity);
37     }
38     public void update(T entity) {
39         getSession().update(entity);
40     }
41     public void delete(T entity) {
42         getSession().delete(entity);
43     }
44     protected Criteria createEntityCriteria() {
45         return getSession().createCriteria(persistentClass);
46     }
47     protected Criteria createEntityCriteriaAlias(String alias) {
48         return getSession().createCriteria(persistentClass, alias);
49     }
50 }
```

Figura 40. Clase abstracta AbstractDao. Fuente: Elaboración propia.

A. La anotación @Autowired indica que este atributo (SessionFactory) deberá ser inyectado (obtener una instancia) automáticamente por Spring, de la clase “org.hibernate.SessionFactory”, que es el objeto que usamos para crear los

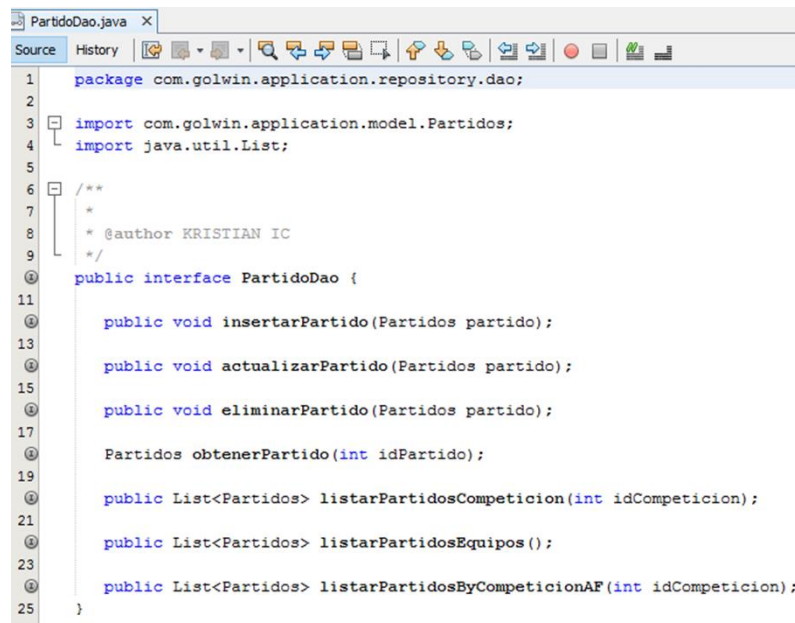
objetos “org.hibernate.Session” la cual nos va a permitir realizar operaciones sobre la base de datos.

B. Creamos una instancia de Session que nos va a proporcionar las operaciones (CRUD) hacia la base de datos.

C. Métodos implementados por “org.hibernate.Session” para realizar un CRUD (Create, Read, Update, Delete)

D. Métodos que utilizan la clase Criteria de Hibernate, la cual permite realizar consultas personalizadas a la base de datos.

Ahora procederemos a crear una interface que defina las operaciones que podremos realizar en la capa de persistencia sobre los registros de la tabla “Partidos”. Crearemos la interface llamada “PartidoDao” donde agregaremos los métodos que se utilizarán, tal como muestra la siguiente figura.



```
1 package com.golwin.application.repository.dao;
2
3 import com.golwin.application.model.Partidos;
4 import java.util.List;
5
6 /**
7  *
8  * @author KRISTIAN IC
9  */
10 public interface PartidoDao {
11
12     public void insertarPartido(Partidos partido);
13
14     public void actualizarPartido(Partidos partido);
15
16     public void eliminarPartido(Partidos partido);
17
18     Partidos obtenerPartido(int idPartido);
19
20     public List<Partidos> listarPartidosCompeticion(int idCompeticion);
21
22     public List<Partidos> listarPartidosEquipos();
23
24     public List<Partidos> listarPartidosByCompeticionAF(int idCompeticion);
25 }
```

Figura 41. Interface PartidoDao. Fuente: Elaboración propia.

Procederemos a crear la clase PartidoDaoImpl donde implementaremos la interface “PartidosDao” que hemos creado anteriormente y a la vez extendemos de la clase abstracta “AbstractDao” para poder utilizar sus métodos.

```
PartidoDaoImpl.java x
Source History
1 package com.golwin.application.repository.impl;
2
3 import com.golwin.application.model.Partidos;
4 import com.golwin.application.repository.dao.AbstractDao;
5 import com.golwin.application.repository.dao.PartidoDao;
6 import java.util.List;
7 import org.hibernate.Criteria;
8 import org.hibernate.criterion.Order;
9 import org.hibernate.criterion.Restrictions;
10 import org.springframework.stereotype.Repository;
11
12 /**
13  *
14  * @author KRISTIAN IC
15  */
16 @Repository("partidoDao") → A
17 public class PartidoDaoImpl extends AbstractDao<Integer, Partidos>
18     implements PartidoDao {
19
20     @Override
21     public void insertarPartido(Partidos partido) {
22         persist(partido);
23     }
24
25     @Override
26     public void actualizarPartido(Partidos partido) {
27         update(partido);
28     }
29
30     @Override
31     public void eliminarPartido(Partidos partido) {
32         delete(partido);
33     }
34
35     @Override
36     public Partidos obtenerPartido(int idPartido) {
37         Partidos partido = getByKey(idPartido);
38         return partido;
39     }
40
41     /**
42      * Metodo para Listar los partidos por competicion. Metodo utilizado para el
43      * Administrador
44      *
45      * @param idCompeticion recibe el id de la competicion
46      * @return listarPartidosCompeticion
47      */
48     @Override
49     @SuppressWarnings("unchecked")
50     public List<Partidos> listarPartidosCompeticion(int idCompeticion) {
51         Criteria criteria = createEntityCriteriaAlias("p")
52             .addOrder(Order.asc("p.partfecha"))
53             .addOrder(Order.asc("p.parthora"))
54             .createCriteria("p.listaequiposByEqlocal", "lelocal")
55             .createCriteria("lelocal.competiciones", "cl")
56             .add(Restrictions.eq("cl.compcod", idCompeticion))
57             .createCriteria("lelocal.paises")
58             .createCriteria("p.listaequiposByEquisita", "levisita")
59             .createCriteria("levisita.competiciones", "cv")
60             .add(Restrictions.eq("cv.compcod", idCompeticion))
61             .createCriteria("levisita.paises")
62             .createCriteria("p.jornadas", "j")
63             .addOrder(Order.asc("j.jordesc"));
64         //Para evitar datos duplicados
65         criteria.setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY);
66         List<Partidos> listarPartidosCompeticion = criteria.list();
67         return listarPartidosCompeticion;
68     }
69 }
```

```

69
70 /**
71  * Metodo para mostrar todos los partidos (equipos) cuyos partidos sigan
72  * activos y no hallan finalizado.
73  *
74  * @return listarPartidosEquipos
75  */
76 @Override
77 @SuppressWarnings("unchecked")
78 public List<Partidos> listarPartidosEquipos() {
79     Criteria criteria = createEntityCriteriaAlias("p")
80     .add(Restrictions.eq("p.partfinalizado", false))
81     .add(Restrictions.eq("p.partestado", true))
82     .createCriteria("p.listaequiposByEqlocal", "lelocal")
83     .createCriteria("lelocal.países")
84     .createCriteria("p.listaequiposByEqvisita", "levisita")
85     .createCriteria("levisita.países");
86     //Para evitar datos duplicados
87     criteria.setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY);
88     List<Partidos> listarPartidosEquipos = criteria.list();
89     return listarPartidosEquipos;
90 }
91
92 @Override
93 @SuppressWarnings("unchecked")
94 public List<Partidos> listarPartidosByCompeticionAF(int idCompeticion) {
95     Criteria criteria = createEntityCriteriaAlias("p")
96     .add(Restrictions.eq("p.partfinalizado", false))
97     .add(Restrictions.eq("p.partestado", true))
98     .addOrder(Order.asc("p.partfecha"))
99     .addOrder(Order.asc("p.parthora"))
100    .createCriteria("p.listaequiposByEqlocal", "lelocal")
101    .createCriteria("lelocal.competiciones", "cl")
102    .add(Restrictions.eq("cl.compcod", idCompeticion))
103    .createCriteria("lelocal.países")
104    .createCriteria("p.listaequiposByEqvisita", "levisita")
105    .createCriteria("levisita.competiciones", "cv")
106    .add(Restrictions.eq("cv.compcod", idCompeticion))
107    .createCriteria("levisita.países")
108    .createCriteria("p.jornadas", "j")
109    .addOrder(Order.asc("j.jordesc"));
110    //Para evitar datos duplicados
111    criteria.setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY);
112    List<Partidos> listarPartidosCompeticion = criteria.list();
113    return listarPartidosCompeticion;
114 }
115 }

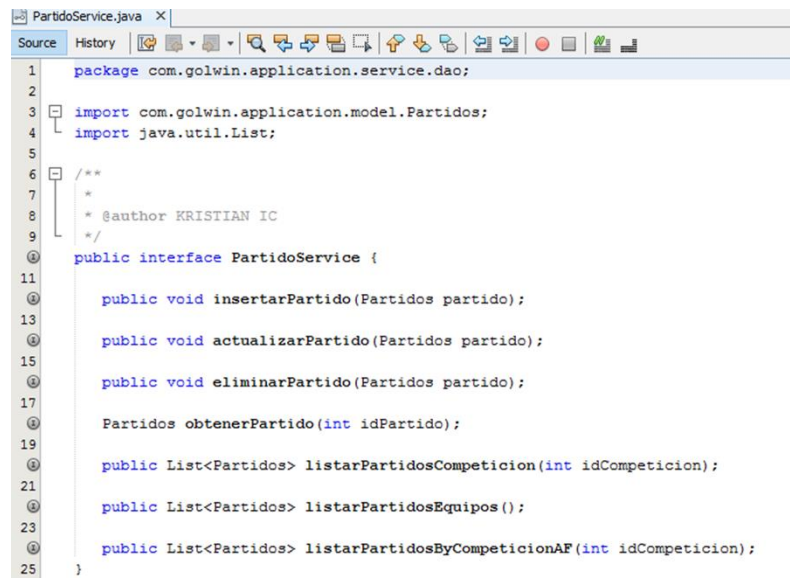
```

Figura 42. Clase PartidoDaoImpl. Fuente: Elaboración propia.

- A. La anotación `@Repository` le indica a Spring que esta será una clase que debe manejar como un bean y que el propósito del bean es realizar acceso a datos.
- B. Métodos implementados de la interface "PartidosDao", en los cuales se utilizan los métodos de la clase abstracta "AbstractDao" para poder realizar las operaciones (CRUD) correspondientes a la base de datos.
- C. Estos métodos tienen la misma estructura que los métodos anteriores con la diferencia que en estos se está usando la clase `Criteria` de Hibernate, la cual permite realizar consultas personalizadas hacia la base de datos, en la cual podemos agregar restricciones, ordenamiento de los datos, entre otras cosas más.

3.3.3.5.2. Servicios (@Service)

Aquí se crea la lógica de negocio, para eso creamos una interface que defina las operaciones que podremos realizar en la capa de servicios hacia la capa de persistencia. Crearemos la interface llamada “PartidoService” donde agregaremos los métodos que se utilizaran, tal como muestra la siguiente figura.



```
1 package com.golwin.application.service.dao;
2
3 import com.golwin.application.model.Partidos;
4 import java.util.List;
5
6 /**
7  *
8  * @author KRISTIAN IC
9  */
10 public interface PartidoService {
11
12     public void insertarPartido(Partidos partido);
13
14     public void actualizarPartido(Partidos partido);
15
16     public void eliminarPartido(Partidos partido);
17
18     Partidos obtenerPartido(int idPartido);
19
20     public List<Partidos> listarPartidosCompeticion(int idCompeticion);
21
22     public List<Partidos> listarPartidosEquipos ();
23
24     public List<Partidos> listarPartidosByCompeticionAF(int idCompeticion);
25 }
```

Figura 43. Interface PartidoService. Fuente: Elaboración propia.

Ahora procederemos a crear la clase PartidoServiceImpl donde implementaremos la interface “PartidosService” que hemos creado anteriormente.

```
PartidoServiceImpl.java
Source History
1 package com.golwin.application.service.impl;
2
3 import com.golwin.application.model.Partidos;
4 import com.golwin.application.repository.dao.PartidoDao;
5 import com.golwin.application.service.dao.PartidoService;
6 import java.util.List;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.stereotype.Service;
9 import org.springframework.transaction.annotation.Transactional;
10
11 /**
12  *
13  * @author KRISTIAN IC
14  */
15 @Service("partidoService") → A
16 public class PartidoServiceImpl implements PartidoService {
17
18     @Autowired → B
19     private PartidoDao partidoDao;
20
21     @Override
22     @Transactional → C
23     public void insertarPartido(Partidos partido) {
24         partidoDao.insertarPartido(partido);
25     }
26
27     @Override
28     @Transactional
29     public void actualizarPartido(Partidos partido) {
30         partidoDao.actualizarPartido(partido);
31     }
32
33     @Override
34     @Transactional
35     public void eliminarPartido(Partidos partido) {
36         partidoDao.eliminarPartido(partido);
37     }
38
39     @Override
40     @Transactional(readOnly = true)
41     public Partidos obtenerPartido(int idPartido) {
42         return partidoDao.obtenerPartido(idPartido);
43     }
44
45     @Override
46     @Transactional(readOnly = true)
47     public List<Partidos> listarPartidosCompeticion(int idCompeticion) {
48         return partidoDao.listarPartidosCompeticion(idCompeticion);
49     }
50
51     @Override
52     @Transactional(readOnly = true)
53     public List<Partidos> listarPartidosEquipos() {
54         return partidoDao.listarPartidosEquipos();
55     }
56
57     @Override
58     @Transactional(readOnly = true)
59     public List<Partidos> listarPartidosByCompeticionAF(int idCompeticion) {
60         return partidoDao.listarPartidosByCompeticionAF(idCompeticion);
61     }
62
63 }
64
```

Figura 44. Clase PartidoServiceImpl. Fuente: Elaboración propia.

- A. La anotación `@Service` mantiene la lógica de negocio, indica que la clase anotada está proporcionando un servicio de negocio a otras capas dentro de la aplicación (proporciona servicios a la capa de presentación que esta anotada con `@RestController`, la cual se verá más adelante), para eso hace una llamada a la capa de persistencia (DAOs) anotada con `@Repository`. Es

decir que el usuario no llama directamente a métodos de la capa de persistencia, sino que llamará a estos métodos con esta anotación.

- B. La anotación `@Autowired` indica que el atributo "PartidoDao", que hace referencia a la interface antes creada y que cuenta con su implementación, deberá ser inyectado (obtener una instancia) automáticamente por Spring
- C. La anotación `@Transactional` indica que el método debe manejar transacciones. Esta anotación se debe agregar solo en los métodos públicos, si lo agregamos en otro tipo de método no obtendremos ninguna excepción, pero tampoco funcionará.
- D. La anotación `@Transactional` con el atributo "readOnly = true" indica que solo obtienen información, son métodos de solo lectura.sds.

3.3.3.5.3. Controllers (@RestController)

En el enfoque de Spring para la creación de servicios web RESTful, las solicitudes HTTP son manejadas por un controlador. Estos componentes se identifican fácilmente mediante la anotación `@RestController`. Una diferencia clave entre un controlador MVC tradicional y el controlador de servicio web RESTful es la forma en que se crea el cuerpo de respuesta HTTP. En lugar de depender de una tecnología de vista para realizar la representación del lado del servidor de los datos a HTML, este controlador de servicio web RESTful simplemente llena y devuelve un objeto. Los datos del objeto se escribirán directamente en la respuesta HTTP como JSON.

Esta capa crearemos nuestros controladores, tal como se muestra en la siguiente figura.

```

1 package com.golwin.application.controller;
2
3 import com.golwin.application.model.Partidos;
4 import com.golwin.application.service.dao.PartidoService;
5 import java.util.List;
6 import org.slf4j.Logger;
7 import org.slf4j.LoggerFactory;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.http.HttpStatus;
10 import org.springframework.http.MediaType;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.web.bind.annotation.PathVariable;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RequestMethod;
15 import org.springframework.web.bind.annotation.RestController;
16
17 /**
18  *
19  * @author KRISTIAN IC
20  */
21
22 @RestController
23 @RequestMapping(value = "/api/v1")
24 public class PartidoRestController {
25
26     private static final Logger LOG = LoggerFactory
27         .getLogger(PartidoRestController.class);
28
29     @Autowired
30     private PartidoService partidoService;
31
32     //Lista de partidos por competicion
33     @RequestMapping(value = "/partidos/{idcompeticion}",
34         method = RequestMethod.GET,
35         produces = MediaType.APPLICATION_JSON_VALUE)
36     public ResponseEntity<List<Partidos>> listarPartidosByCompeticion(
37         @PathVariable(value = "idcompeticion")int idCompeticion){
38         LOG.info("Obteniendo la lista de partidos por competicion con id: {} ",
39             idCompeticion);
40         List<Partidos> partidos = partidoService
41             .listarPartidosByCompeticionAF(idCompeticion);
42         if (partidos.isEmpty()) {
43             return new ResponseEntity<List<Partidos>>(HttpStatus.NO_CONTENT);
44         }
45         return new ResponseEntity<List<Partidos>>(partidos, HttpStatus.OK);
46     }
47 }

```

Figura 45. Clase PartidoRestController. Fuente: Elaboración propia.

- A. Anotación `@RestController` marca la clase como un controlador donde cada metodo devuelve un objeto de dominio/pojo en lugar de una vista. Significa que ya no estamos usando view-resolvers, no estamos enviando directamente el html en respuesta, pero estamos enviando un objeto de dominio convertido en un formato entendido por los consumidores. En nuestro caso, debido a la biblioteca Jackson incluida en la ruta de clase, el objeto se convertirá en formato JSON. Esta anotación combina `@ResponseBody` y `@Controller`
- B. Anotación `@RequestMapping` agregado en la clase, se utiliza para asignar solicitudes web a clases de manejadores específicos y/o metodos de manejador. En nuestro caso, lo hemos aplicado tambien nivel de clase, que dice que esta clase es manejadora por defecto para todas las peticiones HTTP de tipo `"/api/v1"`. `@RequestMapping` tiene varios atributos [value, method,

params...] que pueden usarse para restringir su mapeo a una selección más específica.

- E. La anotación `@Autowired` indica que el atributo "PartidoService", que hace referencia a la interface antes creada en la capa de servicios y que cuenta con su implementación, deberá ser inyectado (obtener una instancia) automáticamente por Spring
- C. La anotación `@RequestMapping` agregado en el método, tiene una declaración de mapeo adicional con atributo *value*, servirá la solicitud de forma `/api/v1/partidos/1`. El atributo *method* se utiliza para especificar el tipo de solicitud HTTP que este método puede servir (GET, POST, PUT, DELETE), en este caso es GET. El atributo *produces* especifica el MediaType que se producirá o consumirá (utilizando el atributo *produce* o *consume*) por ese metodo de controlador en particular, para restringir aún más la asignación.
- D. El metodo `listarPartidosByCompeticion`, devuelve un `ResponseEntity` que contiene la lista de partidos (`ResponseEntity<List<Partidos>>`) con un código de estado 200 (`HttpStatus.OK`) en caso sea correcto, en caso contrario devolverá un 404 (`HttpStatus.NO_CONTENT`) se detalla lo siguiente
 1. `ResponseEntity`, representa toda la respuesta HTTP. Lo bueno de esto es que puede controlar cualquier cosa que entra en él. Puede especificar el código de estado, los encabezados y el cuerpo viene con varios constructores para llevar la información que desea enviar en respuesta HTTP.
 2. La anotación `@PathVariable` indica que un parámetro del método debe estar enlazado a una variable de plantilla de URI. En nuestro caso el parámetro es `"idCompeticion"`, se debe agregar en la URI.

3.3.3.6. Spring Security Oauth2

Vamos a asegurar nuestra API REST usando Oauth2 que permite a una aplicación de terceros obtener acceso limitado a un servicio HTTP, ya sea en nombre de un propietario de recursos.

El proyecto Spring Security OAuth proporciona toda la API necesaria que pueda necesitar para desarrollar una implementación compatible con OAuth2 con Spring.

Implementaremos un servidor de autorización y un servidor de recursos. Para tener acceso a los recursos primero tenemos que autenticarnos en el servidor de autorización, siempre y cuando que el recurso solicitado necesite autorización.

Para poder utilizar Spring Security OAuth2 necesitamos agregar las siguientes dependencias al proyecto en nuestro archivo pom.xml, tal como se muestra en la siguiente figura.

```
12 <properties>
13   <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
14   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15   <springsecurity.version>4.0.4.RELEASE</springsecurity.version>
16   <spring.oauth2.version>2.0.11.RELEASE</spring.oauth2.version>
17 </properties>
18
19 <dependencies>
20   <!-- Spring Security -->
21   <dependency>
22     <groupId>org.springframework.security</groupId>
23     <artifactId>spring-security-web</artifactId>
24     <version>${springsecurity.version}</version>
25   </dependency>
26   <dependency>
27     <groupId>org.springframework.security</groupId>
28     <artifactId>spring-security-config</artifactId>
29     <version>${springsecurity.version}</version>
30   </dependency>
31   <!-- Spring Security OAuth2-->
32   <dependency>
33     <groupId>org.springframework.security.oauth</groupId>
34     <artifactId>spring-security-oauth2</artifactId>
35     <version>${spring.oauth2.version}</version>
36   </dependency>
37 </dependencies>
```

Figura 46. Dependencias para Spring Security OAuth2. Fuente: Elaboración propia.

Crearemos la interface *OauthClientDetailsServiceDao* que contendrá métodos que nos van a permitir interactuar con objetos de tipo *OauthClientDetails*, más adelante realizaremos la implementación de estos métodos.

```

OauthClientDetailsServiceDao.java
Source History
1 package com.golwin.application.service.security.dao;
2
3 import com.golwin.application.model.OauthClientDetails;
4
5 /**
6  *
7  * @author KRISTIAN IC
8  */
9
10 public interface OauthClientDetailsServiceDao {
11
12     public void crearOauthClientDetails(OauthClientDetails oauthClientDetails);
13
14     public boolean isOauthClientDetailsDisponibile(String clientId);
15
16     public OauthClientDetails getOauthClientDetailsById(String clientId);
17
18     public OauthClientDetails actualizarOauthClientDetails(
19         OauthClientDetails oauthClientDetails);
20
21     public void eliminarOauthClientDetails(
22         OauthClientDetails oauthClientDetails);
23 }

```

Figura 47. Interface OauthClientDetailsServiceDao. Fuente: Elaboración propia.

Crearemos una clase CustomOauthClientDetailsService para implementar la interfaz creada anteriormente. Esta clase implementa de ClientDetailsService propio de Spring Security Oauth2, que es un servicio que nos va a proporcionar los detalles de un cliente Oauth2. Este tiene un método loadClientByClientId que carga un cliente por el ID del cliente, este método no debe de devolver null.

Esta implementación contiene operaciones básicas, como el CRUD.

```

CustomOauthClientDetailsService.java
Source History
1 package com.golwin.application.service.security.impl;
2
3 import com.golwin.application.model.OauthClientDetails;
4 import com.golwin.application.repository.dao.OauthClientDetailsDao;
5 import com.golwin.application.service.security.dao.OauthClientDetailsServiceDao;
6 import com.golwin.application.utils.Constantes;
7 import java.util.ArrayList;
8 import java.util.Arrays;
9 import java.util.HashSet;
10 import java.util.List;
11 import java.util.Set;
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.security.core.GrantedAuthority;
14 import org.springframework.security.core.authority.SimpleGrantedAuthority;
15 import org.springframework.security.oauth2.provider.ClientDetails;
16 import org.springframework.security.oauth2.provider.ClientDetailsService;
17 import org.springframework.security.oauth2.provider.ClientRegistrationException;
18 import org.springframework.security.oauth2.provider.client.BaseClientDetails;
19 import org.springframework.stereotype.Service;
20 import org.springframework.transaction.annotation.Transactional;
21 import org.springframework.util.StringUtils;
22
23 /**
24  *
25  * @author KRISTIAN IC
26  */
27 @Service("customOauthClientDetailsService")
28 @Transactional
29 public class CustomOauthClientDetailsService implements ClientDetailsService,
30     OauthClientDetailsServiceDao {
31
32     @Autowired
33     private OauthClientDetailsDao oauthClientDetailsDao;

```

```

34  @Override
35  public ClientDetails loadClientById(String clientId)
36      throws ClientRegistrationException {
37      if (oauthClientDetailsDao.isOauthClientDetailsDisponible(clientId)) {
38          throw new ClientRegistrationException(String.format(
39              Constantes.MSG_ERROR_CLIENTE_NO_REGISTRADO, clientId));
40      }
41      OauthClientDetails oauthClientDetails = oauthClientDetailsDao
42          .loadClientById(clientId);
43      BaseClientDetails baseClientDetails = new BaseClientDetails();
44      List<String> authorities = Arrays.asList(oauthClientDetails
45          .getAuthorizedGrantTypes().split(", "));
46      List<GrantedAuthority> authoritiesList=new ArrayList<GrantedAuthority>();
47      for (String s : authorities) {
48          authoritiesList.add(new SimpleGrantedAuthority(s));
49      }
50      Set<String> uris = new HashSet<String>(Arrays.asList(oauthClientDetails
51          .getWebServerRedirectUri().split(", ")));
52      baseClientDetails.setClientId(oauthClientDetails.getClientId());
53      baseClientDetails.setScope(Arrays.asList(oauthClientDetails.getScope()
54          .split(", ")));
55      baseClientDetails.setAuthorizedGrantTypes(Arrays.asList(
56          oauthClientDetails.getAuthorizedGrantTypes().split(", ")));
57      baseClientDetails.setAuthorities(authoritiesList);
58      baseClientDetails.setAccessTokenValiditySeconds(
59          oauthClientDetails.getAccessTokenValidity().intValue());
60      baseClientDetails.setClientSecret(oauthClientDetails.getClientSecret());
61      baseClientDetails.setRegisteredRedirectUri(uris);
62      baseClientDetails.setResourceIds(Arrays.asList(
63          oauthClientDetails.getResourceIds().split(", ")));
64
65      String approve = oauthClientDetails
66          .getAutoapprove() == null ? "false" : "true";
67
68      if (approve.equalsIgnoreCase("true")) {
69          baseClientDetails.setAutoApproveScopes(StringUtils
70              .commaDelimitedListToSet(oauthClientDetails.getAutoapprove()));
71      } else {
72          baseClientDetails.setAutoApproveScopes(new HashSet<String>());
73      }
74      return baseClientDetails;
75  }
}

76  @Override
77  public void crearOauthClientDetails(OauthClientDetails oauthClientDetails) {
78      if (oauthClientDetails != null) {
79          oauthClientDetailsDao.crearOauthClientDetails(oauthClientDetails);
80      }
81  }
82  }
83
84  @Override
85  public boolean isOauthClientDetailsDisponible(String clientId) {
86      if (clientId != null) {
87          oauthClientDetailsDao.getOauthClientDetailsById(clientId);
88      }
89      return false;
90  }
91  }
92
93  @Override
94  public OauthClientDetails getOauthClientDetailsById(String clientId) {
95      OauthClientDetails client = null;
96      if (!clientId.equalsIgnoreCase("")) {
97          client = oauthClientDetailsDao.getOauthClientDetailsById(clientId);
98      }
99      return client;
100  }
101
102  @Override
103  public OauthClientDetails actualizarOauthClientDetails(
104      OauthClientDetails oauthClientDetails) {
105      if (oauthClientDetails != null) {
106          oauthClientDetailsDao.actualizarOauthClientDetails(oauthClientDetails);
107      }
108      return oauthClientDetails;
109  }
110
111  @Override
112  public void eliminarOauthClientDetails(
113      OauthClientDetails oauthClientDetails) {
114      if (oauthClientDetails != null) {
115          oauthClientDetailsDao.eliminarOauthClientDetails(oauthClientDetails);
116      }
117  }
118  }
}

```

Figura 48. Clase CustomOauthClientDetailsService. Fuente: Elaboración propia.

Implementaremos la clase Constantes la cual nos permite tener mensajes y estos poder utilizarlos en otras clases como en la implementación anterior.

```
Constantes.java x
Source History
1 package com.golwin.application.utils;
2
3 /**
4  *
5  * @author KRISTIAN IC
6  */
7 public class Constantes {
8
9     public static final String RESOURCE_ID = "golwin";
10
11     public static final String MSG_ERROR_CLIENTE_NO_REGISTRADO =
12         "El cliente '%s' no está registrado";
13     public static final String MSG_ERROR_USUARIO_NO_REGISTRADO =
14         "El usuario '%s' no está registrado";
15 }
```

Figura 49. Clase Constantes. Fuente: Elaboración propia.

Implementaremos la clase CustomUserDetailsService la cual implementa de UserDetailsService que es propio de Spring Security, esto nos va a permitir recuperar la información del usuario que se quiere loguear.

```
CustomUserDetailsService.java x
Source History
1 package com.golwin.application.service.security.impl;
2
3 import com.golwin.application.model.Roles;
4 import com.golwin.application.model.Usuarios;
5 import com.golwin.application.service.dao.UsuarioService;
6 import java.util.ArrayList;
7 import java.util.List;
8 import org.slf4j.Logger;
9 import org.slf4j.LoggerFactory;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.security.core.GrantedAuthority;
12 import org.springframework.security.core.authority.SimpleGrantedAuthority;
13 import org.springframework.security.core.userdetails.UserDetails;
14 import org.springframework.security.core.userdetails.UserDetailsService;
15 import org.springframework.security.core.userdetails.UsernameNotFoundException;
16 import org.springframework.stereotype.Service;
17 import org.springframework.transaction.annotation.Transactional;
18
19 /**
20  *
21  * @author KRISTIAN IC
22  */
23 @Service("customUserDetailsService")
24 public class CustomUserDetailsService implements UserDetailsService{
25
26     private static final Logger LOG = LoggerFactory.getLogger(
27         CustomUserDetailsService.class);
28
29     @Autowired
30     private UsuarioService usuarioService;
31
32     @Override
33     @Transactional(readOnly = true)
34     public UserDetails loadUserByUsername(String usualias)
35     throws UsernameNotFoundException {
36         com.golwin.application.model.Usuarios usuario = usuarioService
37             .buscarUsuarioAlias(usualias);
38         if (usuario == null) {
39             throw new UsernameNotFoundException(
40                 "No se encontró información del usuario: " + usualias);
41         }
42         return new org.springframework.security.core.userdetails.User(
43             usuario.getUsualias(), usuario.getUsupassw(),
44             usuario.getUsuestado().equals("Active"), true, true, true,
45             getGrantedAuthorities(usuario));
46     }
```

```

47
48
49     private List<GrantedAuthority> getGrantedAuthorities(Usuarios usuario) {
50         List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();
51
52         for (Roles userRol : usuario.getRoleses()) {
53             LOG.info("UserRol : {}" + userRol);
54             authorities.add(new SimpleGrantedAuthority("ROLE_" +
55                 userRol.getRolnombre()));
56         }
57         LOG.info("authorities : {}" + authorities);
58         return authorities;
59     }

```

Figura 50. Clase CustomUserDetailsService. Fuente: Elaboración propia.

3.3.3.6.1. Servidor de Autorización

El servidor de autorización (authorization server) es el responsable de verificar las credenciales y si las credenciales están bien, proporcionar el token (Access_token así como refresh_token). También contiene información sobre los clientes registrados y los posibles ámbitos de acceso y tipos de concesión. El almacenamiento de token se utiliza para almacenar el token. Utilizaremos un almacén de token en la base de datos. @EnableAuthorizationServer habilita un servidor de Autorización (es decir, un AuthorizationEndPoint y un TokenEndpoint) en el contexto actual de la aplicación. La clase AuthorizationServerConfigurerAdapter implementa AuthorizationServerConfigurer que proporciona todos los métodos necesarios para configurar un servidor de autorización.

```

1  /**
2   * En esta clase especifica definimos tanto nuestro servidor de autenticación
3   * como nuestro servidor de recursos, especificamos que recursos están
4   * securizados, si queremos persistir el token en memoria o base de datos, etc..
5   */
6   package com.golwin.application.security.config;
7
8   import com.golwin.application.component.CustomAuthenticationEntryPoint;
9   import com.golwin.application.utils.Constantes;
10  import javax.sql.DataSource;
11  import org.springframework.beans.factory.annotation.Autowired;
12  import org.springframework.beans.factory.annotation.Qualifier;
13  import org.springframework.context.annotation.Bean;
14  import org.springframework.context.annotation.Configuration;
15  import org.springframework.context.annotation.Primary;
16  import org.springframework.security.authentication.AuthenticationManager;
17  import org.springframework.security.config.annotation.web.builders.HttpSecurity;
18  import org.springframework.security.config.http.SessionCreationPolicy;
19  import org.springframework.security.core.userdetails.UserDetailsService;
20  import org.springframework.security.oauth2.config.annotation.configurers
21     .ClientDetailsServiceConfigurer;
22  import org.springframework.security.oauth2.config.annotation.web.configuration
23     .AuthorizationServerConfigurerAdapter;
24  import org.springframework.security.oauth2.config.annotation.web.configuration
25     .EnableAuthorizationServer;
26  import org.springframework.security.oauth2.config.annotation.web.configuration
27     .EnableResourceServer;
28  import org.springframework.security.oauth2.config.annotation.web.configuration
29     .ResourceServerConfigurerAdapter;
30  import org.springframework.security.oauth2.config.annotation.web.configurers
31     .AuthorizationServerEndpointsConfigurer;
32  import org.springframework.security.oauth2.config.annotation.web.configurers
33     .AuthorizationServerSecurityConfigurer;
34  import org.springframework.security.oauth2.config.annotation.web.configurers
35     .ResourceServerSecurityConfigurer;
36  import org.springframework.security.oauth2.provider.ClientDetailsService;
37  import org.springframework.security.oauth2.provider.code
38     .AuthorizationCodeServices;
39  import org.springframework.security.oauth2.provider.code
40     .JdbcAuthorizationCodeServices;
41  import org.springframework.security.oauth2.provider.token.DefaultTokenServices;
42  import org.springframework.security.oauth2.provider.token.store.JdbcTokenStore;
43  import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
44
45  /**
46   *
47   * @author KRISTIAN IC
48   */
49  @Configuration
50  public class OAuth2Configuration {
51
52      @Configuration
53      @EnableAuthorizationServer
54      protected static class AuthorizationServerConfiguration
55          extends AuthorizationServerConfigurerAdapter {
56
57          @Autowired
58          @Qualifier("authenticationManagerBean")
59          private AuthenticationManager authenticationManager;
60
61          @Autowired
62          private DataSource dataSource;
63
64          @Autowired
65          @Qualifier("customOauthClientDetailsService")
66          private ClientDetailsService clientDetailsService;
67
68          @Autowired
69          @Qualifier("customUserDetailsService")
70          private UserDetailsService userDetailsService;
71
72          @Bean
73          public JdbcTokenStore tokenStore() {
74              return new JdbcTokenStore(dataSource);
75          }
76
77          @Bean
78          protected AuthorizationCodeServices authorizationCodeServices() {
79              return new JdbcAuthorizationCodeServices(dataSource);
80          }
81
82          @Override
83          public void configure(AuthorizationServerSecurityConfigurer oauthServer)
84              throws Exception {
85              oauthServer.allowFormAuthenticationForClients();
86          }
87

```

```

87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
}

@Override
public void configure(AuthorizationServerEndpointsConfigurer endpoints)
    throws Exception {
    endpoints
        .tokenStore(tokenStore())
        .authenticationManager(this.authenticationManager)
        .userDetailsService(this.userDetailsService);
}

@Override
public void configure(ClientDetailsServiceConfigurer clients)
    throws Exception {
    //con los datos del cliente en base de datos
    clients.withClientDetails(clientDetailsService);
}

@Bean
@Primary
public DefaultTokenServices tokenServices() {
    DefaultTokenServices defaultTokenServices = new DefaultTokenServices();
    defaultTokenServices.setSupportRefreshToken(true);
    defaultTokenServices.setTokenStore(tokenStore());
    return defaultTokenServices;
}

```

Figura 51. Clase Oauth2Configuration – Authorization Server. Fuente: Elaboración propia.

3.3.3.6.2. Servidor de Recursos

El Servidor de recursos (resource server) alberga los recursos (nuestra API REST) en los que el cliente está interesado. La anotación `@EnableResourceServer`, aplicada a Oauth2 Resource Servers, habilita un filtro Spring Security que autentica las peticiones usando un token Oauth2 entrante. La clase `ResourceServerConfigurerAdapter` implementa `ResourceServerConfigurer` que proporciona métodos para ajustar las reglas de acceso y las rutas protegidas por la seguridad Oauth2.

El método `configure` configura `HttpSecurity` lo que permite configurar la seguridad basada en la web para las solicitudes HTTP específicas. Por defecto se va a aplicar a todas las solicitudes, pero se puede restringir utilizando `antMatchers`.

En la configuración se dice que la URL `"/public/**"` no están aseguradas, cualquiera puede acceder a ellos. La URL `"/api/v1/secure"` solo puede ser accedida por alguien con rol ADMIN, y la URL `"/api/v1/**"` solo puede ser accedida por usuarios autenticados.

```

113
114 @Configuration
115 @EnableResourceServer
116 protected static class ResourceServerConfiguration
117     extends ResourceServerConfigurerAdapter {
118
119     @Autowired
120     private CustomAuthenticationEntryPoint customAuthenticationEntryPoint;
121
122     @Override
123     public void configure(ResourceServerSecurityConfigurer resources) {
124         resources
125             .resourceId(Constants.RESOURCE_ID);
126     }
127
128     @Override
129     public void configure(HttpSecurity http) throws Exception {
130         http
131             .exceptionHandling()
132             .authenticationEntryPoint(customAuthenticationEntryPoint)
133             .and()
134             .csrf()
135             .requireCsrfProtectionMatcher(new AntPathRequestMatcher(
136                 "/oauth/authorize"))
137             .disable()
138             .headers()
139             .frameOptions().disable()
140             .and()
141             .sessionManagement()
142             .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
143             .and()
144             .authorizeRequests()
145             .antMatchers("/public/**").permitAll()
146             .antMatchers("/api/v1/secure").hasRole("ADMIN")
147             .antMatchers("/api/v1/**").authenticated()
148             .and()
149             .logout()
150             .logoutUrl("/oauth/logout");
151     }
152 }
153
154
155 }

```

Figura 52. Clase Oauth2Configuration – Resource Server. Fuente: Elaboración propia.

Tenemos que crear una clase que extenderá de AuthenticationEntryPoint de Spring y reemplazará el método commence. Este método rechazará todas las solicitudes no autenticadas y enviará el código de error 401.


```

1 package com.golwin.application.component;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServletRequest;
6 import javax.servlet.http.HttpServletResponse;
7 import org.springframework.security.core.AuthenticationException;
8 import org.springframework.security.web.AuthenticationEntryPoint;
9 import org.springframework.stereotype.Component;
10
11 /**
12  *
13  * @author KRISTIAN IC
14  */
15 @Component
16 public class CustomAuthenticationEntryPoint implements AuthenticationEntryPoint{
17
18     @Override
19     public void commence(HttpServletRequest request,
20                         HttpServletResponse response, AuthenticationException authException)
21                         throws IOException, ServletException {
22         response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Access_Denied");
23     }
24
25 }

```

Figura 53. Clase CustomAuthenticationEntryPoint. Fuente: Elaboración propia.

3.3.3.6.3. Configurar Spring Security

Debemos de crear una configuración de Spring Security, esta configuración crea un filtro de servlet conocido como *springSecurityFilterChain* que es responsable de toda la seguridad (protección de las direcciones URL de la aplicación, la validación de usuario y contraseña, etc.) dentro de nuestra aplicación.

Tenemos un método `ConfigureGlobalSecurity` que configura `AuthenticationManagerBuilder` con credenciales de usuario y funciones permitidas. Este `AuthenticationManagerBuilder` crea `AuthenticationManager` que es responsable de procesar cualquier solicitud de autenticación. Notemos que estamos utilizando la autenticación en Base de Datos mediante `UserDetailsService` (lo inyectamos de la clase `CustomUserDetailsService` que antes hemos mencionado). Además, con el fin de cifrar la contraseña en la base de datos, estamos encriptando los password mediante `BCryptPasswordEncoder`.

Anotamos la clase con `@Configuration`, la cual permite indicar que esta clase es una clase de configuración y debe de tomarse como tal.

La anotación `@EnableWebSecurity` determina la adición de características e integración con Spring MVC en base a la ruta de clase. Además, proporciona configuración a través de `HttpSecurity` que permiten configurar el acceso basado

en patrones de URLs (autorizar o no una determinada URL), criterios de valoración de autenticación, manipuladores, etc.

Con `@Import` importamos nuestra clase de configuración de `oauth2` `Oauth2Configuration.class`.

La anotación `@EnableGlobalMethodSecurity` añade seguridad a los métodos de la capa de servicios. Proporciona soporte para la seguridad de anotaciones JSR-

250

```
SecurityConfig.java x
Source History
1 package com.golwin.application.security.config;
2
3 import java.security.NoSuchAlgorithmException;
4 import java.security.SecureRandom;
5 import javax.sql.DataSource;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.beans.factory.annotation.Qualifier;
8 import org.springframework.context.annotation.Bean;
9 import org.springframework.context.annotation.Configuration;
10 import org.springframework.context.annotation.Import;
11 import org.springframework.security.access.expression.method
12     .MethodSecurityExpressionHandler;
13 import org.springframework.security.authentication.AuthenticationManager;
14 import org.springframework.security.authentication.dao
15     .DaoAuthenticationProvider;
16 import org.springframework.security.config.annotation.authentication.builders
17     .AuthenticationManagerBuilder;
18 import org.springframework.security.config.annotation.method.configuration
19     .EnableGlobalMethodSecurity;
20 import org.springframework.security.config.annotation.method.configuration
21     .GlobalMethodSecurityConfiguration;
22 import org.springframework.security.config.annotation.web.configuration
23     .EnableWebSecurity;
24 import org.springframework.security.config.annotation.web.configuration
25     .WebSecurityConfigurerAdapter;
26 import org.springframework.security.core.userdetails.UserDetailsService;
27 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
28 import org.springframework.security.crypto.password.PasswordEncoder;
29 import org.springframework.security.oauth2.provider.expression
30     .OAuth2MethodSecurityExpressionHandler;
31
32 /**
33  *
34  * @author KRISTIAN IC
35  */
36 @Configuration
37 @EnableWebSecurity
38 @Import({Oauth2Configuration.class})
39 public class SecurityConfig extends WebSecurityConfigurerAdapter {
40
41     @Autowired
42     @Qualifier("customUserDetailsService")
43     private UserDetailsService userDetailsService;
44
45     @Autowired
46     private DataSource dataSource;
```

```

47
48
49 @Autowired
50 public void configureGlobalSecurity(AuthenticationManagerBuilder auth)
51     throws Exception {
52     auth.authenticationProvider(authProvider());
53 }
54
55 @Bean
56 public DaoAuthenticationProvider authProvider() {
57     DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
58     authProvider.setPasswordEncoder(passwordEncoder());
59     authProvider.setUserDetailsService(userDetailsService);
60     return authProvider;
61 }
62
63 @Bean
64 public PasswordEncoder passwordEncoder() {
65     SecureRandom random = null;
66     try {
67         random = SecureRandom.getInstance("SHA1PRNG");
68     } catch (NoSuchAlgorithmException e) {
69         e.printStackTrace();
70     }
71     PasswordEncoder encoder = new BCryptPasswordEncoder(16, random);
72     return encoder;
73 }
74
75 @Bean
76 @Override
77 public AuthenticationManager authenticationManagerBean() throws Exception {
78     return super.authenticationManagerBean();
79 }
80
81 @EnableGlobalMethodSecurity(prePostEnabled = true, jsr250Enabled = true)
82 private static class GlobalSecurityConfiguration
83     extends GlobalMethodSecurityConfiguration {
84
85     @SuppressWarnings("unused")
86     public GlobalSecurityConfiguration() {
87
88     }
89
90     @Override
91     protected MethodSecurityExpressionHandler createExpressionHandler() {
92         return new OAuth2MethodSecurityExpressionHandler();
93     }
94 }
95

```

Figura 54. Clase SecurityConfig. Fuente: Elaboración propia.

Creamos la clase RestSecurityInitializer que extiende de AbstractSecurityWebApplicationInitializer que solamente registra el filtro springSecurityFilterChain para cada URL de la aplicación.

```

RestSecurityInitializer.java
Source History
1 package com.golwin.application.security.config;
2
3 import org.springframework.security.web.context
4     .AbstractSecurityWebApplicationInitializer;
5
6 /**
7  *
8  * @author KRISTIAN IC
9  */
10 public class RestSecurityInitializer extends
11     AbstractSecurityWebApplicationInitializer{
12
13 }

```

Figura 55. Clase RestSecurityInitializer. Fuente: Elaboración propia.

Ahora explicaremos los frameworks que se utilizaron en el lado del cliente.

3.3.3.7. Framework Ionic

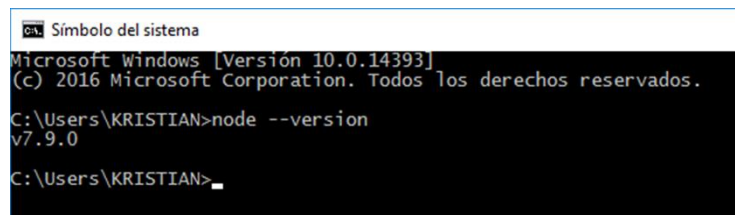
Para implementar la aplicación Golwin en el Framework Ionic se siguieron los siguientes procedimientos.

3.3.3.7.1. Instalar NodeJs

NodeJs es el encargado de realizar toda la automatización, es el que poner en marcha el CLI de Ionic.

Debemos descargar NodeJs de su página oficial dirigiéndonos a la siguiente url: <https://nodejs.org/> y descargar la versión para nuestro sistema operativo. Una vez descargado en nuestro ordenador procederemos a realizar la instalación.

Para verificar que se ha instalado correctamente abrimos una ventana de símbolo del sistema, y escribimos “node --version”, nos mostrará la versión que hemos instalado de nodejs.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\KRISTIAN>node --version
v7.9.0

C:\Users\KRISTIAN>
```

Figura 56. Comando para verificar la versión de NodeJS. Fuente: Elaboración propia.

3.3.3.7.2. Instalar CLI and Cordova

Para crear proyecto Ionic, necesitamos instalar la última versión de la CLI y de Cordova, para eso tenemos que crear una carpeta donde creamos conveniente en nuestro ordenador. Ya creada nuestra carpeta la abrimos y dentro de ella tenemos que abrir una ventana de símbolo del sistema. Para eso mantenemos presionado la tecla “SHIFT” en nuestro teclado y la vez hacemos click derecho, del menú que aparece tenemos que hacer click en “Abrir ventana de comando aquí”. Tal como muestra la siguiente figura.

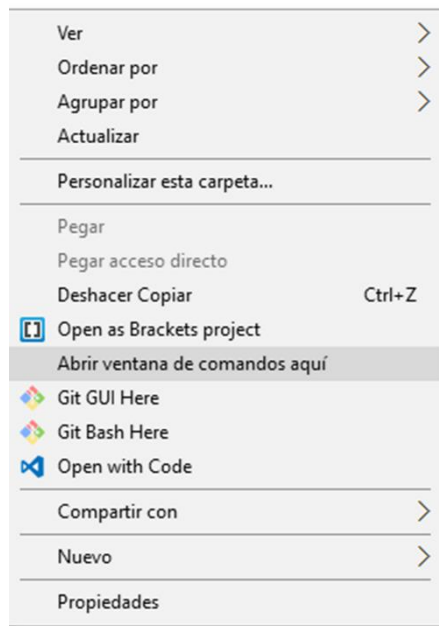


Figura 57. Menú para abrir la ventana de comandos aquí. Fuente: Elaboración propia.

Otra manera de realizar lo mismo es, dirigirnos a la pestaña Archivo del explorador de Windows y hacer lo que muestra la siguiente figura.

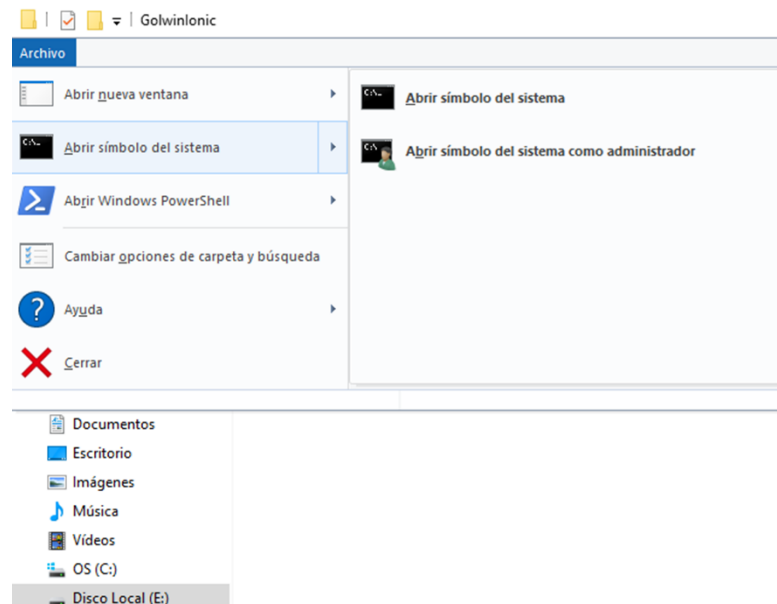


Figura 58. Abrir símbolo del sistema. Fuente: Elaboración propia

Ejecutando uno de los dos métodos anteriores nos aparece la siguiente ventana donde haremos la instalación introduciendo el siguiente comando, “npm install –

g ionic cordova”, y presionamos la tecla “ENTER” para que empiece la instalación (debemos contar con acceso a internet para que se puedan descargar los paquetes de instalación), tal como se muestra en la siguiente figura.

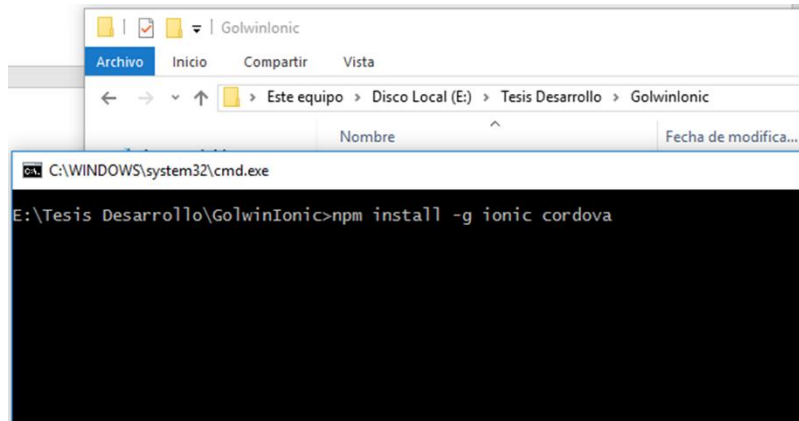


Figura 59. Ingresamos el comando para la instalación. Fuente: Elaboración propia.

3.3.3.7.3. Crear nuestra aplicación con Ionic - Golwin

Para crear nuestro proyecto con Ionic tenemos que ejecutar el siguiente comando en la ventana de símbolo del sistema “ionic start Golwin blank”. Tal como se muestra en la siguiente figura.

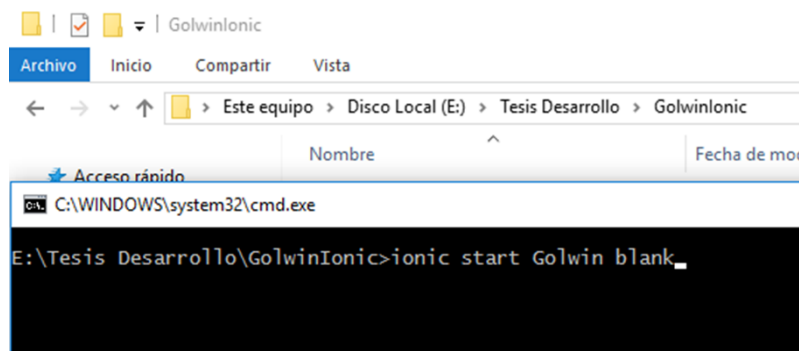


Figura 60. Comando para crear un nuevo proyecto. Fuente: Elaboración propia

Donde le indicamos a Ionic que nos cree un proyecto con el nombre “Golwin” y con una plantilla en blanco/vacía, Ionic empezará a crear el proyecto, primero nos creara una carpeta con el nombre del proyecto que hemos puesto (Golwin), después dentro de él me va a descargar y crear todos los elementos necesarios (Angular, Córdoba y toda la estructura necesaria para un proyecto de Ionic).

En la instalación nos va a preguntar si queremos utilizar notificaciones, o sea si vamos a utilizar una cuenta de Ionic para lanzar push notification en la aplicación, le dimos que no pulsando la tecla “N” y presionamos “ENTER”, haciendo esto la instalación estará terminada.

Nos dirigimos a la carpeta donde hemos creado nuestro proyecto y nos aparecerá la carpeta del proyecto y dentro de ella todos los archivos de un proyecto Ionic. Tal como se muestra en la siguiente figura.

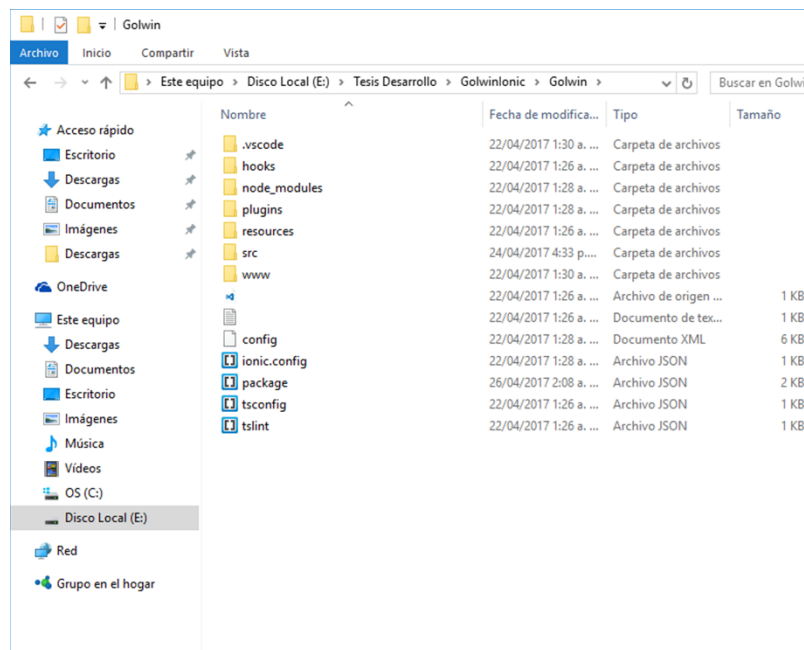


Figura 61. Carpeta con los archivos del proyecto. Fuente: Elaboración propia.

Para saber la versión de Ionic y otra información tanto de sistema como del proyecto. Para eso nos dirigimos dentro de la carpeta de nuestro proyecto para abrir una ventana de consola de comando, para eso mantenemos presionado la tecla “SHIFT” y a la vez hacemos click derecho, del menú que se despliega hacemos click sobre “Abrir ventana de comandos aquí” y escribimos el siguiente comando “ionic info”, tal como se muestra en la siguiente figura.

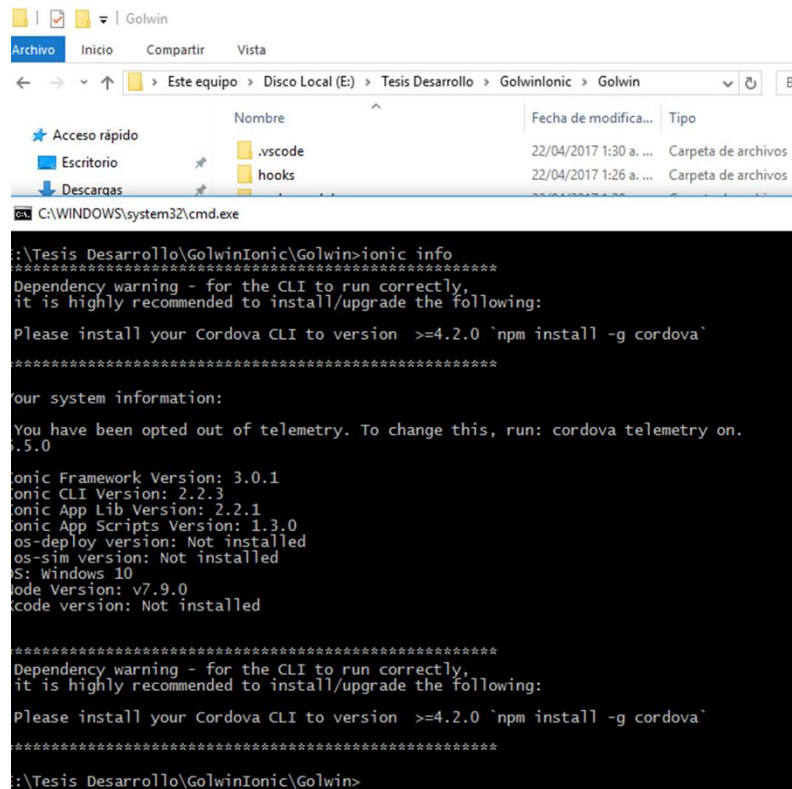


Figura 62. información del sistema. Fuente: Elaboración propia.

3.3.3.7.4. Levantar el servidor de ionic

Ionic levanta un servidor donde va a estar constantemente escuchando nuestra carpeta para cualquier cambio que hagamos y este haga automáticamente la actualización, este servidor de NodeJs en Express particularmente lo va a levantar en el explorador y vamos a estar viendo todos esos cambios. Para levantar el servidor ejecutamos el siguiente comando “ionic serve”, tal como se muestra en la siguiente figura.

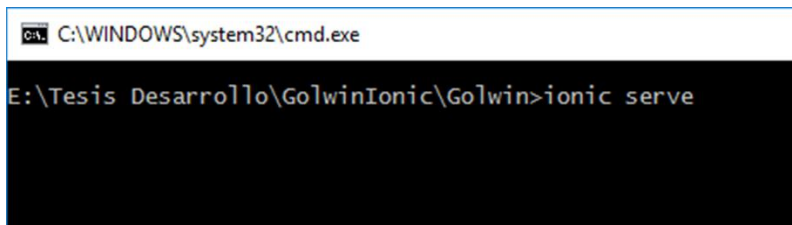


Figura 63. Comando para levantar el servidor. Fuente: Elaboración propia

Terminado de realizar el proceso para levantar el servidor, la aplicación se abrirá en el navegador (Google Chrome), Por defecto levanta un servidor “localhost” con el puerto “8100”. Tal como se muestra en la siguiente figura.

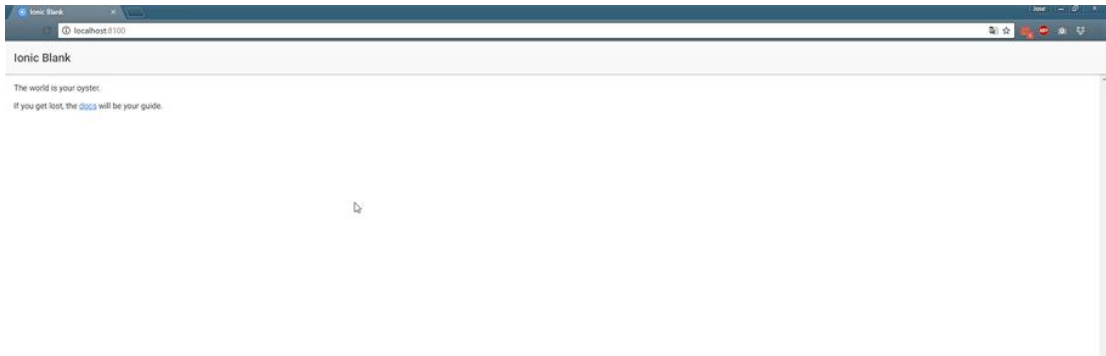


Figura 64. Aplicación desplegada en el navegador. Fuente: Elaboración propia.

Para una mejor visualización de nuestra aplicación procedemos a activar el modo de desarrollo en Google Chrome, para activarlo presionamos la tecla “F12”, donde podemos activar el modo dispositivo para simular el tamaño de un dispositivo móvil. Tal como se muestra en la siguiente figura.

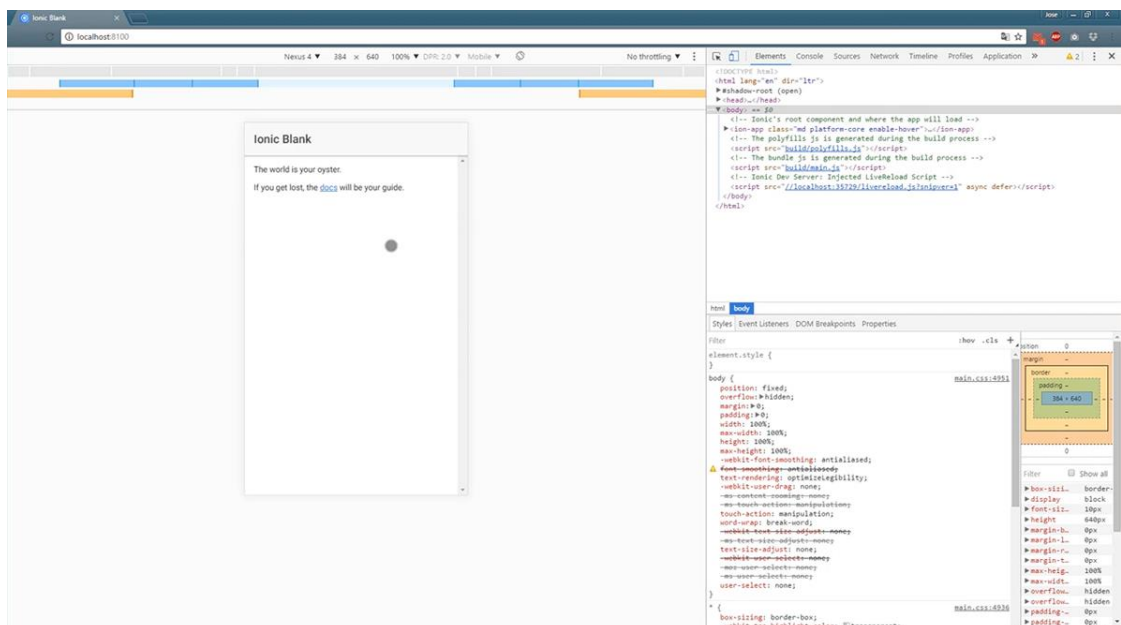


Figura 65. Visualización en modo dispositivo. Fuente: Elaboración propia.

3.3.3.7.5. Estructura de un proyecto con el framework Ionic

Para visualizar la estructura del proyecto vamos a abrirlo en un editor de código, en este caso utilizaremos Visual Studio Code como editor de código para nuestro proyecto. Tal como se muestra en la siguiente figura.

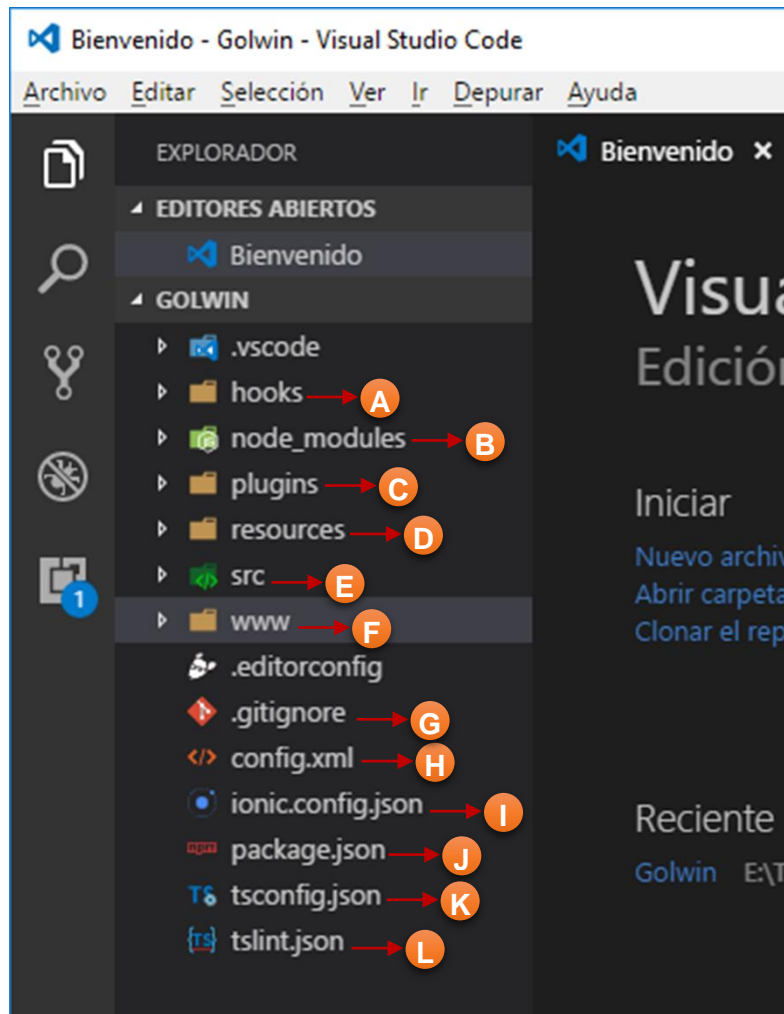


Figura 66. Proyecto abierto en Visual Studio Code. Fuente: Elaboración propia.

Se explicará las secciones importantes y de nuestro interés para el desarrollo de nuestra aplicación.

A. Carpeta hooks: En esta carpeta se añaden los *scripts* que se ejecutaran en el proceso de construcción de la aplicación.

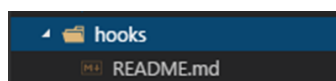


Figura 67. Contenido de la carpeta hooks. Fuente: Elaboración propia

B. Carpeta node_modules: esta carpeta contiene todas las dependencias/modules que se instalan para el proyecto.

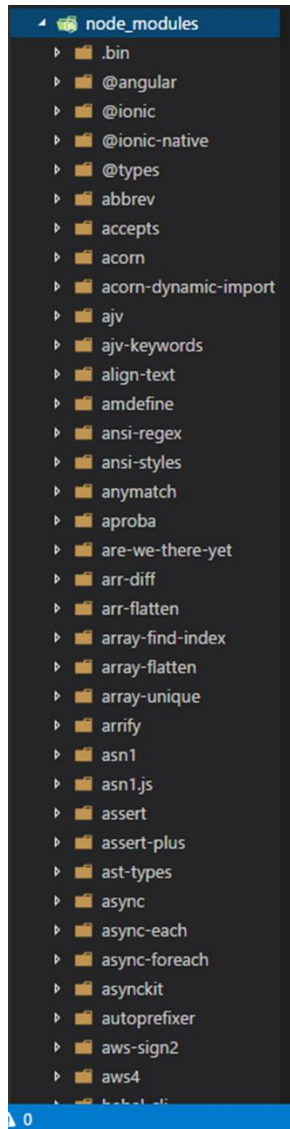


Figura 68. Contenido de la carpeta node_modules. Fuente: Elaboración propia

C. Carpeta plugins: Contiene todos los plugins nativos que vamos a utilizar, ya sean de Córdoba, del propio Ionic o de terceros, para el desarrollo de la aplicación. Aquí tampoco tendremos que modificar nada.

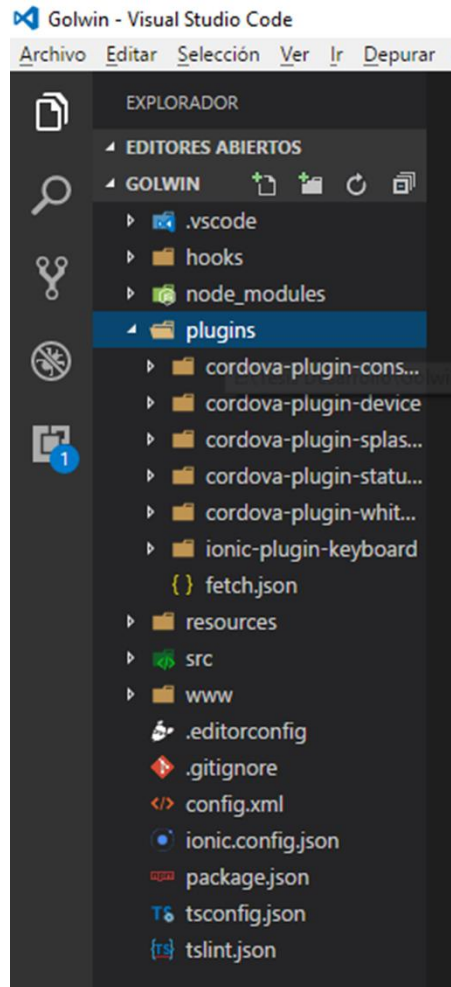


Figura 69. Contenido de la carpeta plugins. Fuente: Elaboración propia.

D. Carpeta resources: en esta carpeta están todos los archivos pertinentes a las imágenes, los iconos, los splash screen (pantalla de presentación) que va a tener nuestra aplicación cuando vallamos la compilemos dependiendo de la plataforma.

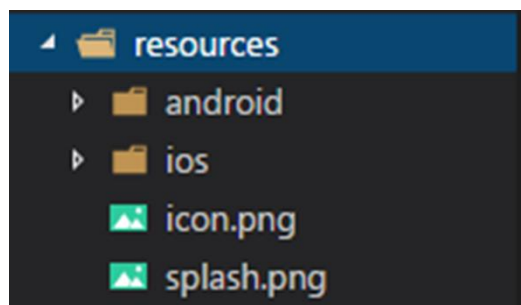


Figura 70. Contenido de la carpeta resources. Fuente: Elaboración propia.

E. Carpeta src: Esta es la carpeta más importante, en esta carpeta se encuentra toda nuestra aplicación, aquí es donde vamos a estar editando y configurando, vamos a agregar componentes, aquí realizaremos la implementación de nuestra aplicación.

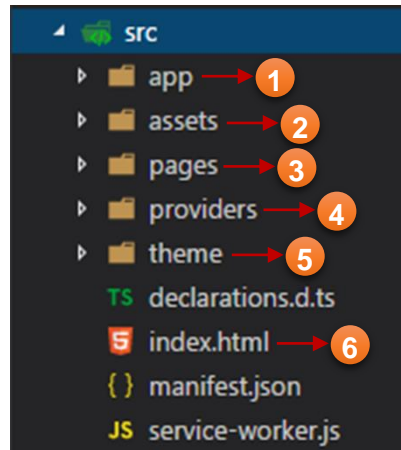


Figura 71. Contenido de la carpeta src. Fuente: Elaboración propia

Esta carpeta explicaremos las partes de su contenido, ya que es donde vamos a estar trabajando y necesitamos saber qué cosa hace o para que sirve cada una.

1. Carpeta app: en esta carpeta se encuentra los archivos raíz o principales de configuración de nuestra aplicación.

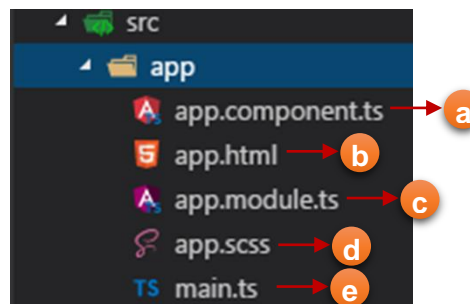


Figura 72. Contenido de la carpeta app. Fuente: Elaboración propia.

a. Archivo app.component.ts: Este archivo es el componente raíz de nuestra aplicación. Es cargado/declarado en el archivo "src/app/app.module.ts", que representa nuestra aplicación completa como un módulo, que se carga en "src/app/main.ts". Esta estructura soporta "Ahead of Time Compilation" (Antes del tiempo de compilación),

una característica de angular que descarga la compilación del paquete de aplicaciones.

```
app.component.ts
1 import { Component, ViewChild } from '@angular/core';
2 import { Platform, NavController, LoadingController } from 'ionic-angular';
3 import { StatusBar } from '@ionic-native/status-bar';
4 import { SplashScreen } from '@ionic-native/splash-screen';
5 import { Storage } from '@ionic/storage';
6 import { Menu } from '../pages/menu/menu';
7 import { Login } from '../pages/login/login';
8 import { Apuesta } from '../pages/apuesta/apuesta';
9 @Component({
10   templateUrl: 'app.html'
11 })
12 export class MyApp {
13   @ViewChild('NAV') nav: NavController
14   public rootPage: any;
15   public datosUsuario: any;
16   constructor(
17     platform: Platform,
18     statusBar: StatusBar,
19     splashScreen: SplashScreen,
20     public storage: Storage,
21     public loadingCtrl: LoadingController,
22   ) {
23     platform.ready().then(() => {
24       statusBar.styleDefault();
25       splashScreen.hide();
26     });
27     this.datosUsuario = [];
28     this.checkAutorizacion();
29   }
30   checkAutorizacion(){
31     this.storage.get('tokens').then( datosAcceso => {
32       if(datosAcceso){
33         let loader = this.loadingCtrl.create({content: 'Autenticando Usuario...'});
34         loader.present();
35         this.storage.get('usuario').then(datosUser => {
36           if(datosUser){
37             this.datosUsuario.push(datosUser);
38             this.nav.setRoot(Menu, {datosUsuario: this.datosUsuario});
39           }else{
40             this.nav.setRoot(Login);
41           }
42           loader.dismiss();
43         });
44       }else{
45         this.nav.setRoot(Login);
46       }
47     });
48   }
49 }
```

Figura 73. Archivo app.component.ts. Fuente: Elaboración propia.

- b. Archivo app.html: Este archivo es la vista raíz de nuestra aplicación. En este caso contiene a la pantalla de “Login” siempre y cuando un usuario tenga almacenada las credenciales, caso contrario muestra la pantalla “Menu”. Veremos esto más adelante.

```
app.html
1 <ion-nav #NAV [root]="rootPage"></ion-nav>
2
```

Figura 74. Archivo app.html. Fuente: Elaboración propia

- c. Archivo app.module.ts: este archivo es el modulo principal de la aplicación. Aquí se agregarán todas las dependencias, tanto de las paginas como de los componentes que estemos utilizando, también se deben agregar las librerías de terceros que vallamos a utilizar. Como son

declarations (las páginas de nuestra aplicación o las vistas), entryComponents (componentes que se utilizaran en toda la aplicación), providers (son los servicios que vamos a utilizar en la aplicación) son los que se van a mostrar en la sección root (<ion-app></ion-app>) de nuestro archivo “index.html”. Tal como muestra la siguiente figura.

```
app.module.ts
1 import { BrowserModule } from '@angular/platform-browser';
2 import { ErrorHandler, NgModule } from '@angular/core';
3 import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular';
4 import { SplashScreen } from '@ionic-native/splash-screen';
5 import { StatusBar } from '@ionic-native/status-bar';
6 import { HttpModule } from '@angular/http';
7 import { IonicStorageModule } from 'ionic/storage';
8 import { MyApp } from './app.component';
9 import { HomePage } from '../pages/home/home';
10 import { Registrarse } from '../pages/registrarse/registrarse';
11 import { Login } from '../pages/login/login';
12 import { Tipocompeticiones } from '../pages/tipocompeticiones/tipocompeticiones';
13 import { Competiciones } from '../pages/competiciones/competiciones';
14 import { Partidos } from '../pages/partidos/partidos';
15 import { Detallepartido } from '../pages/detallepartido/detallepartido';
16 import { Apuesta } from '../pages/apuesta/apuesta';
17 import { Perfil } from '../pages/perfil/perfil';
18 import { Acerca } from '../pages/acerca/acerca';
19 import { Menu } from '../pages/menu/menu';
20 @NgModule({
21   declarations: [
22     MyApp,
23     HomePage,
24     Registrarse,
25     Login,
26     Tipocompeticiones,
27     Competiciones,
28     Partidos,
29     Detallepartido,
30     Apuesta,
31     Perfil,
32     Acerca,
33     Menu
34   ],
35   imports: [
36     BrowserModule,
37     HttpModule,
38     IonicModule.forRoot(MyApp),
39     IonicStorageModule.forRoot()
40   ],
41   bootstrap: [IonicApp],
42   entryComponents: [
43     MyApp,
44     HomePage,
45     Registrarse,
46     Login,
47     Tipocompeticiones,
48     Competiciones,
49     Partidos,
50     Detallepartido,
51     Apuesta,
52     Perfil,
53     Acerca,
54     Menu
55   ],
56   providers: [
57     StatusBar,
58     SplashScreen,
59     {provide: ErrorHandler, useClass: IonicErrorHandler}
60   ]
61 })
62 export class AppModule {}
```

Figura 75. Archivo app.module.ts. Fuente: Elaboración propia

- d. Archivo app.scss: Este archivo es un archivo SASS. Aquí podemos añadir estilos CSS de manera avanzada, que se van a poder usar en cualquier parte (pantalla/vista) de la aplicación.

```
app.scss x
1 // http://ionicframework.com/docs/v2/theming/
2 |
3
4 // App Global Sass
5 // -----
6 // Put style rules here that you want to apply globally. These
7 // styles are for the entire app and not just one component.
8 // Additionally, this file can be also used as an entry point
9 // to import other Sass files to be included in the output CSS.
10 //
11 // Shared Sass variables, which can be used to adjust Ionic's
12 // default Sass variables, belong in "theme/variables.scss".
13 //
14 // To declare rules for a specific mode, create a child rule
15 // for the .md, .ios, or .wp mode classes. The mode class is
16 // automatically applied to the <body> element in the app.
17 .titulo{
18   color: map-get($colors, wet_asphalt);
19   text-align: center;
20 }
```

Figura 76. Archivo app.scss. Fuente: Elaboración propia.

- e. Archivo main.ts: Este archivo es que nos va a servir para cargar la aplicación. Carga el archivo “app.module.ts” que es donde tenemos todas las dependencias de nuestro proyecto.

```
main.ts x
1 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
2
3 import { AppModule } from './app.module';
4
5 platformBrowserDynamic().bootstrapModule(AppModule);
6
```

Figura 77. Archivo main.ts. Fuente: Elaboración propia.

- 2. Carpeta assests: En esta carpeta se encuentran los archivos de imágenes y otro tipo de materiales externos que vas a usar en las páginas.

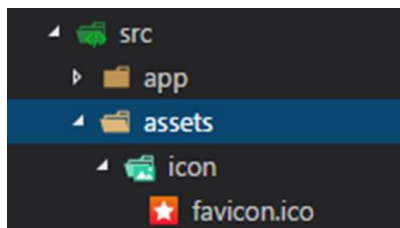


Figura 78. Contenido de la carpeta assests. Fuente: Elaboración propia.

- 3. Carpeta pages: En esta carpeta se irán colocando las paginas (pantallas) de la aplicación.

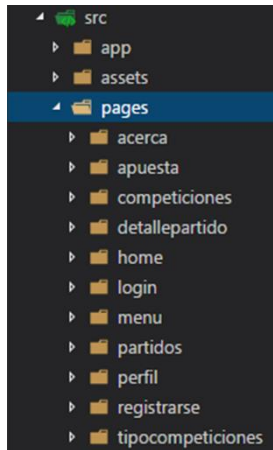


Figura 79. Contenido de la carpeta pages. Fuente: Elaboración propia.

4. Carpeta providers: En esta carpeta se crean los archivos de servicios que vamos a utilizar en nuestra aplicación. Por ejemplo, hacer peticiones HTTP.

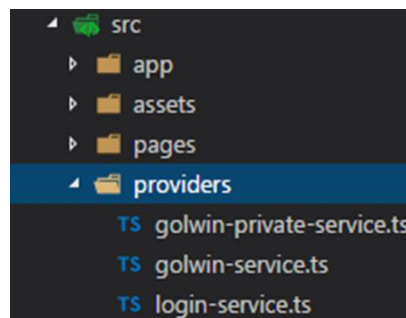


Figura 80. Contenido de la carpeta providers. Fuente: Elaboración propia.

5. Carpeta theme: Contiene archivos SASS que ayudan con la apariencia (Themes) de la aplicación.

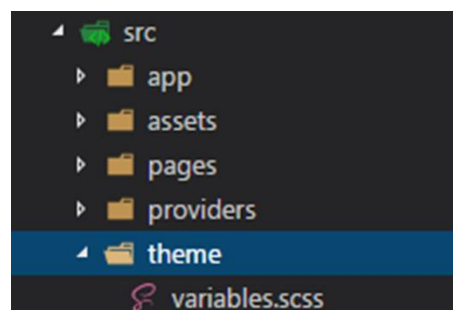


Figura 81. Contenido de la carpeta theme. Fuente: Elaboración propia.

La figura anterior muestra el contenido de la carpeta “theme”, este a su vez contiene un archivo “variable.css”, aquí, por ejemplo, podemos agregar los

colores que tendrá nuestra aplicación, para eso declaramos un nombre para el color y agregamos el valor del color, ahora podemos agregar este color en cualquier parte de nuestra aplicación solo con referenciar al nombre que le colocamos, tal como se muestra a continuación.

```
variables.scss ●
1 // Ionic Variables and Theming. For more info, please see:
2 // http://ionicframework.com/docs/v2/theming/
3 $font-path: "../assets/fonts";
4 @import "ionic.globals";
5
6 $colors: (
7   primary: #488aff,
8   secondary: #32db64,
9   danger: #f53d3d,
10  light: #f4f4f4,
11  dark: #212121,
12  wet_asphalt: #34495e,
13  belize_hole: #2980b9,
14  pomegranate: #c0392b,
15  midnight_blue: #2c3e50,
16  nephritis: #27ae60,
17  clouds: #ecf0f1
18 );
19
20 @import "ionic.theme.default";
21 // Ionicons
22 // -----
23 // The premium icon font for Ionic. For more info, please see:
24 // http://ionicframework.com/docs/v2/ionicons/
25 @import "ionic.ionicons";
26 // Fonts
27 // -----
28 @import "roboto";
29 @import "noto-sans";
```

Figura 82. Archivos variables.scss. Fuente: Elaboración propia.

6. Archivo index.html: a pesar que no se va a trabajar con este archivo, cabe decir que es importante ya que es aquí en donde se agregan los css, los javascripts, a pesar que este archivo a simple vista aparece vacío.

```
index.html
1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3 <head>
4   <meta charset="UTF-8">
5   <title>Ionic App</title>
6   <meta name="viewport" content="width=device-width, initial-scale=1.0,
7   minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
8   <meta name="format-detection" content="telephone=no">
9   <meta name="msapplication-tap-highlight" content="no">
10
11  <link rel="icon" type="image/x-icon" href="assets/icon/favicon.ico">
12  <link rel="manifest" href="manifest.json">
13  <meta name="theme-color" content="#4e8ef7">
14  <!-- cordova.js required for cordova apps -->
15  <script src="cordova.js"></script>
16  <!-- un-comment this code to enable service worker
17  <script>
18    if ('serviceWorker' in navigator) {
19      navigator.serviceWorker.register('service-worker.js')
20        .then(() => console.log('service worker installed'))
21        .catch(err => console.log('Error', err));
22    }
23  </script-->
24  <link href="build/main.css" rel="stylesheet">
25 </head>
26 <body>
27   <!-- Ionic's root component and where the app will load -->
28   <ion-app></ion-app>
29
30   <!-- The polyfills js is generated during the build process -->
31   <script src="build/polyfills.js"></script>
32
33   <!-- The bundle js is generated during the build process -->
34   <script src="build/main.js"></script>
35
36 </body>
37 </html>
```

Figura 83. Archivo index.html. Fuente: Elaboración propia.

F. Carpeta www: En esta carpeta se encuentra el proyecto ya compilado, el servidor levanta esta carpeta en el explorador. No tenemos que modificar nada aquí.

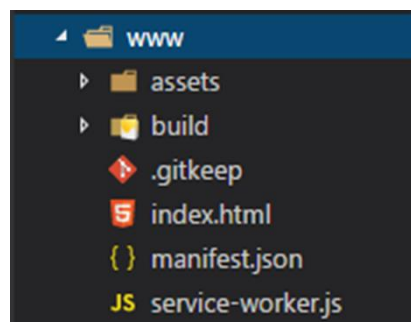


Figura 84. Contenido de la carpeta www. Fuente: Elaboración propia.

G. Archivo .gitignore: En este archivo se agregan todas las carpetas que vamos a ignorar cuando estemos usando git, es decir para subir el proyecto a GitHub por ejemplo.

```
.gitignore x
1  Specifies intentionally untracked files to ignore when using Git
2  # http://git-scm.com/docs/gitignore
3
4  *~
5  *.sw[mn]pcod]
6  *.log
7  *.tmp
8  *.tmp.*
9  log.txt
10 *.sublime-project
11 *.sublime-workspace
12 .vscode/
13 npm-debug.log*
14
15 .idea/
16 .sass-cache/
17 .tmp/
18 .versions/
19 coverage/
20 dist/
21 node_modules/
22 tmp/
23 temp/
24 hooks/
25 platforms/
26 plugins/
27 plugins/android.json
28 plugins/ios.json
29 www/
30 $RECYCLE.BIN/
31
32 .DS_Store
33 Thumbs.db
34 UserInterfaceState.xcuserstate
```

Figura 85. Archivo .gitignore. Fuente: Elaboración propia.

H. Archivo config.xml: Contiene parámetros que se utilizan cuando se construye un proyecto nativo a partir de un proyecto Ionic, aquí se escriben todas las configuraciones pertinentes de nuestra aplicación cuando la vallamos a compilar, ya sea para Android o IOS.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<widget id="com.ionicframework.golwin407402" version="0.0.1" xmlns="http://www.w3.org/ns/widgets"
  xmlns:cdv="http://cordova.apache.org/ns/1.0">
  <name>Golwin</name>
  <description>An awesome Ionic/Cordova app.</description>
  <author email="hi@ionicframework" href="http://ionicframework.com/">Ionic Framework Team</author>
  <content src="index.html"/>
  <access origin="*" />
  <allow-navigation href="http://ionic.local/*" />
  <allow-intent href="http://*/*" />
  <allow-intent href="https://*/*" />
  <allow-intent href="tel:*" />
  <allow-intent href="sms:*" />
  <allow-intent href="mailto:*" />
  <allow-intent href="geo:*" />
  <preference name="webViewBounce" value="false" />
  <preference name="UIWebViewBounce" value="false" />
  <preference name="DisallowOverscroll" value="true" />
  <preference name="android-minSdkVersion" value="16" />
  <preference name="BackupWebStorage" value="none" />
  <preference name="SplashMaintainAspectRatio" value="true" />
  <preference name="FadeSplashScreenDuration" value="300" />
  <preference name="SplashShowOnlyFirstTime" value="false" />
  <preference name="SplashScreen" value="screen" />
  <preference name="SplashScreenDelay" value="3000" />
  <platform name="android">
    <allow-intent href="market:*" />
    <icon src="resources/android/icon/drawable-ldpi-icon.png" density="ldpi" />
    <icon src="resources/android/icon/drawable-mdpi-icon.png" density="mdpi" />
    <icon src="resources/android/icon/drawable-hdpi-icon.png" density="hdpi" />
    <icon src="resources/android/icon/drawable-xhdpi-icon.png" density="xhdpi" />
    <icon src="resources/android/icon/drawable-xxhdpi-icon.png" density="xxhdpi" />
    <icon src="resources/android/icon/drawable-xxxhdpi-icon.png" density="xxxhdpi" />
    <splash src="resources/android/splash/drawable-land-ldpi-screen.png" density="land-ldpi" />
    <splash src="resources/android/splash/drawable-land-mdpi-screen.png" density="land-mdpi" />
    <splash src="resources/android/splash/drawable-land-hdpi-screen.png" density="land-hdpi" />
    <splash src="resources/android/splash/drawable-land-xhdpi-screen.png" density="land-xhdpi" />
    <splash src="resources/android/splash/drawable-land-xxhdpi-screen.png" density="land-xxhdpi" />
    <splash src="resources/android/splash/drawable-land-xxxhdpi-screen.png" density="land-xxxhdpi" />
    <splash src="resources/android/splash/drawable-port-ldpi-screen.png" density="port-ldpi" />
    <splash src="resources/android/splash/drawable-port-mdpi-screen.png" density="port-mdpi" />
    <splash src="resources/android/splash/drawable-port-hdpi-screen.png" density="port-hdpi" />
    <splash src="resources/android/splash/drawable-port-xhdpi-screen.png" density="port-xhdpi" />
    <splash src="resources/android/splash/drawable-port-xxhdpi-screen.png" density="port-xxhdpi" />
    <splash src="resources/android/splash/drawable-port-xxxhdpi-screen.png" density="port-xxxhdpi" />
  </platform>
  <platform name="ios">
    <allow-intent href="itms:*" />
    <allow-intent href="itms-apps:*" />
    <icon src="resources/ios/icon/icon.png" width="57" height="57" />
    <icon src="resources/ios/icon/icon@2x.png" width="114" height="114" />
    <icon src="resources/ios/icon/icon-40.png" width="40" height="40" />
    <icon src="resources/ios/icon/icon-40@2x.png" width="80" height="80" />
    <icon src="resources/ios/icon/icon-40@3x.png" width="120" height="120" />
    <icon src="resources/ios/icon/icon-50.png" width="50" height="50" />
    <icon src="resources/ios/icon/icon-50@2x.png" width="100" height="100" />
    <icon src="resources/ios/icon/icon-60.png" width="60" height="60" />
    <icon src="resources/ios/icon/icon-60@2x.png" width="120" height="120" />
    <icon src="resources/ios/icon/icon-60@3x.png" width="180" height="180" />
    <icon src="resources/ios/icon/icon-72.png" width="72" height="72" />
    <icon src="resources/ios/icon/icon-72@2x.png" width="144" height="144" />
    <icon src="resources/ios/icon/icon-76.png" width="76" height="76" />
    <icon src="resources/ios/icon/icon-76@2x.png" width="152" height="152" />
    <icon src="resources/ios/icon/icon-83.5@2x.png" width="167" height="167" />
    <icon src="resources/ios/icon/icon-small.png" width="29" height="29" />
    <icon src="resources/ios/icon/icon-small@2x.png" width="58" height="58" />
    <icon src="resources/ios/icon/icon-small@3x.png" width="87" height="87" />
    <splash src="resources/ios/splash/Default-568h@2x-iphone.png" width="640" height="1136" />
    <splash src="resources/ios/splash/Default-667h.png" width="750" height="1334" />
    <splash src="resources/ios/splash/Default-736h.png" width="1242" height="2208" />
    <splash src="resources/ios/splash/Default-Landscape-736h.png" width="2208" height="1242" />
    <splash src="resources/ios/splash/Default-Landscape@2x-iphad.png" width="2048" height="1536" />
    <splash src="resources/ios/splash/Default-Landscape@~iphadpro.png" width="2732" height="2048" />
    <splash src="resources/ios/splash/Default-Landscape~iphad.png" width="1024" height="768" />
    <splash src="resources/ios/splash/Default-Portrait@2x-iphad.png" width="1536" height="2048" />
    <splash src="resources/ios/splash/Default-Portrait~iphadpro.png" width="2048" height="2732" />
    <splash src="resources/ios/splash/Default-Portrait-iphad.png" width="768" height="1024" />
    <splash src="resources/ios/splash/Default@2x-iphone.png" width="640" height="960" />
    <splash src="resources/ios/splash/Default-iphone.png" width="320" height="480" />
  </platform>
  <plugin name="ionic-plugin-keyboard" spec="~2.2.1" />
  <plugin name="cordova-plugin-whitelist" spec="1.3.1" />
  <plugin name="cordova-plugin-console" spec="1.0.5" />
  <plugin name="cordova-plugin-statusbar" spec="2.2.1" />
  <plugin name="cordova-plugin-device" spec="1.1.4" />
  <plugin name="cordova-plugin-splashscreen" spec="~4.0.1" />
</widget>

```

Figura 86. Archivo de configuración config.xml. Fuente: Elaboración propia.

- I. Archivo `ionic.config.json`: contiene configuraciones para el propio ionic necesarias para compilar o ejecutar el comando “ionic serve”

```
ionic.config.json x
1 {
2   "name": "Golwin",
3   "app_id": "",
4   "v2": true,
5   "typescript": true
6 }
```

Figura 87. Archivo `ionic.config.json`. Fuente: Elaboración propia.

- J. Archivo `package.json`: archivo que contiene todas las dependencias/paquetes necesarios y que se quieran utilizar para el desarrollo de nuestra aplicación. También este archivo permite registrar los plugins de terceros que se vayan a utilizar.

```
package.json x
1 {
2   "name": "ionic-hello-world",
3   "version": "0.0.0",
4   "author": "Ionic Framework",
5   "homepage": "http://ionicframework.com/",
6   "private": true,
7   "scripts": {
8     "clean": "ionic-app-scripts clean",
9     "build": "ionic-app-scripts build",
10    "ionic:build": "ionic-app-scripts build",
11    "ionic:serve": "ionic-app-scripts serve"
12  },
13  "dependencies": {
14    "@angular/common": "4.0.0",
15    "@angular/compiler": "4.0.0",
16    "@angular/compiler-cli": "4.0.0",
17    "@angular/core": "4.0.0",
18    "@angular/forms": "4.0.0",
19    "@angular/http": "4.0.0",
20    "@angular/platform-browser": "4.0.0",
21    "@angular/platform-browser-dynamic": "4.0.0",
22    "@ionic-native/core": "3.4.2",
23    "@ionic-native/splash-screen": "3.4.2",
24    "@ionic-native/status-bar": "3.4.2",
25    "@ionic/storage": "2.0.1",
26    "ionic-angular": "3.0.1",
27    "ionicons": "3.0.0",
28    "rxjs": "5.1.1",
29    "sw-toolbox": "3.4.0",
30    "zone.js": "^0.8.4"
31  },
32  "devDependencies": {
33    "@ionic/app-scripts": "1.3.0",
34    "typescript": "~2.2.1"
35  },
36  "cordovaPlugins": [
37    "cordova-plugin-console",
38    "cordova-plugin-whitelist",
39    "cordova-plugin-statusbar",
40    "cordova-plugin-device",
41    "ionic-plugin-keyboard",
42    "cordova-plugin-splashscreen"
43  ],
44  "cordovaPlatforms": [],
45  "description": "Golwin: An Ionic project"
46 }
47 }
```

Figura 88. Archivo `package.json`. Fuente: Elaboración propia.

K. Archivo tsconfig.json: Este archivo contiene información necesaria a la hora de compilar TypeScript. Viene con la configuración por defecto, por lo que no necesitamos editar este archivo.

```
tsconfig.json x
1  {
2    "compilerOptions": {
3      "allowSyntheticDefaultImports": true,
4      "declaration": false,
5      "emitDecoratorMetadata": true,
6      "experimentalDecorators": true,
7      "lib": [
8        "dom",
9        "es2015"
10     ],
11     "module": "es2015",
12     "moduleResolution": "node",
13     "sourceMap": true,
14     "target": "es5"
15   },
16   "include": [
17     "src/**/*.ts"
18   ],
19   "exclude": [
20     "node_modules"
21   ],
22   "compileOnSave": false,
23   "atom": {
24     "rewriteTsconfig": false
25   }
26 }
```

Figura 89. Archivo tsconfig.json. Fuente: Elaboración propia.

L. Archivo tslint.json: Este archivo contiene información necesaria a la hora de compilar TypeScript. Viene con la configuración por defecto, por lo que no necesitamos editar este archivo.

```
tslint.json x
1  {
2    "rules": {
3      "no-duplicate-variable": true,
4      "no-unused-variable": [
5        true
6      ]
7    },
8    "rulesDirectory": [
9      "node_modules/tslint-eslint-rules/dist/rules"
10   ]
11 }
```

Figura 90. Archivo tslint.json. Fuente: Elaboración propia.

3.3.3.7.6. Implementando nuestra aplicación Golwin

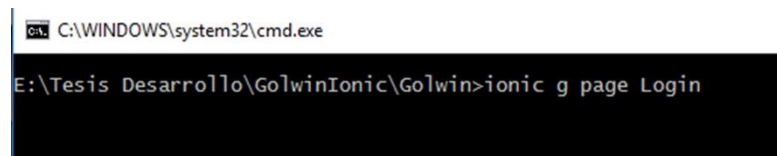
3.3.3.7.6.1. Levantar servidor

Como ya dijimos anteriormente primero tenemos que levantar un servidor para poder visualizar nuestra aplicación en el explorador. Para eso tenemos que ejecutar el siguiente comando “ionic serve”, tal como se explicó anteriormente.

3.3.3.7.6.2. Crear las páginas de nuestra aplicación.

Primero debemos de eliminar todo el contenido de la carpeta “src/pages”. A continuación, tenemos que agregar nuestras páginas/vistas/pantallas.

Para agregar paginas/vistas/pantallas a nuestra aplicación debemos ejecutar el siguiente comando dentro de la consola de comandos “ionic g page Login” (“g” es un alias para “generate”). Tal como se muestra en la siguiente figura.



```
C:\WINDOWS\system32\cmd.exe
E:\Tesis Desarrollo\GolwinIonic\Golwin>ionic g page Login
```

Figura 91. Comando para crear una nueva página. Fuente: Elaboración propia.

Con este comando le indicamos a ionic que nos genere una página con nombre Login. Ejecutado el comando se añadirá automáticamente una carpeta con el nombre que le pusimos “Login” dentro de la carpeta “src/pages”. De esta manera debemos crear las páginas que vallamos a necesitar, para nuestro caso crearemos las siguientes páginas: **Acerca** (contiene la página acerca de), **Apuesta** (página para realizar una apuesta), **Competiciones** (página que muestra las competiciones de un determinado tipo de competición), **Detallepartido** (página que muestra el detalle de pagos de un partido), **Home** (página principal cuando un usuario se ha autenticado), **Menu** (contendrá un menú lateral desplegable), **Partidos** (página que mostrará los partidos de una competición), **Perfil** (página que mostrará el perfil de un usuario), **Registrarse** (página para registrar un nuevo usuario), **Tipocompeticiones** (página que mostrará los tipos de competiciones), tal como se muestra en la siguiente figura.

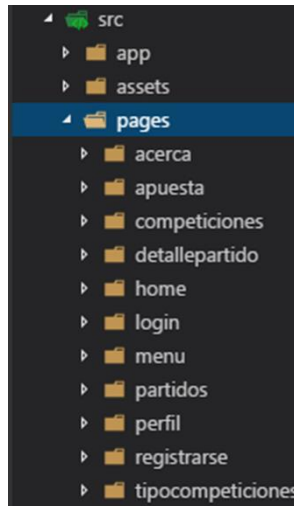


Figura 92. Estructura de páginas. Fuente: Elaboración propia.

3.3.3.7.6.3. Implementando la página de inicio de nuestra aplicación.

Vamos a implementar nuestra página de Login para nuestra aplicación, para eso nos dirigimos a la carpeta “**src/pages/login**”, la desglosaremos y veremos los siguientes archivos como se muestra en la siguiente figura.

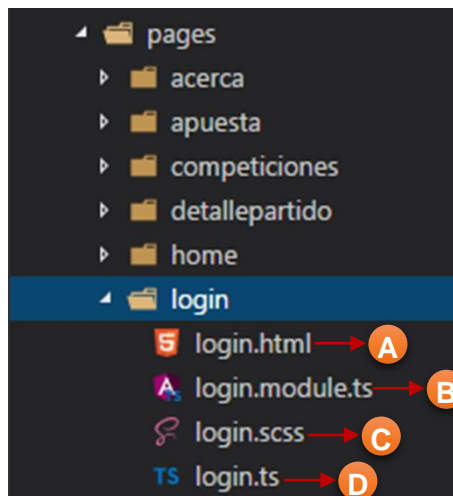


Figura 93. Estructura de archivos al crear page Login. Fuente: Elaboración propia.

A. Archivo Login.html: Este archivo contiene la estructura de la página (html) utilizando las directivas propias de Ionic para construir la página.

```

login.html x
1 <ion-header>
2 <ion-navbar color="belize_hole">
3 <ion-title center>Golwin</ion-title>
4 </ion-navbar>
5 </ion-header>
6
7
8 <ion-content padding>
9 <h1 class="titulo">Login</h1>
10 <form [formGroup]="loginForm" (ngSubmit)="onLogin(loginForm.value)">
11 <ion-item>
12 <ion-icon name="person" item-left</ion-icon>
13 <ion-label floating>Usuario</ion-label>
14 <ion-input formControlName="aliasuser" type="text" name="aliasuser" clearInput="true" required</ion-input>
15 </ion-item>
16 <ion-item *ngIf="loginForm.get('aliasuser').errors && loginForm.get('aliasuser').touched">
17 <p class="error-b" *ngIf="loginForm.get('aliasuser').hasError('required')">¡Por favor ingrese su usuario!</p>
18 <p class="error-b" *ngIf="loginForm.get('aliasuser').hasError('minlength')">¡Por favor ingrese un usuario válido!</p>
19 <p class="error-b" *ngIf="loginForm.get('aliasuser').hasError('maxlength')">¡Por favor ingrese un usuario válido!</p>
20 <p class="error-b" *ngIf="loginForm.get('aliasuser').hasError('pattern')">¡Por favor ingrese un usuario válido!</p>
21 </ion-item>
22 <ion-item>
23 <ion-icon name="key" item-left</ion-icon>
24 <ion-label floating>Contraseña</ion-label>
25 <ion-input formControlName="passworduser" type="password" name="passworduser" clearInput="true" required</ion-input>
26 </ion-item>
27 <ion-item *ngIf="loginForm.get('passworduser').errors && loginForm.get('passworduser').touched">
28 <p class="error-b" *ngIf="loginForm.get('passworduser').hasError('required')">¡Por favor ingrese su contraseña!</p>
29 <p class="error-b" *ngIf="loginForm.get('passworduser').hasError('minlength')">¡Por favor ingrese una contraseña válida!</p>
30 <p class="error-b" *ngIf="loginForm.get('passworduser').hasError('maxlength')">¡Por favor ingrese una contraseña válida!</p>
31 <p class="error-b" *ngIf="loginForm.get('passworduser').hasError('pattern')">¡Por favor ingrese una contraseña válida!</p>
32 </ion-item>
33 <div padding align="center">
34 <button ion-button round block large icon-right icon-only type="submit" color="peter_river" [disabled]="!loginForm.valid">
35 <ion-icon name="log-in" ></ion-icon>
36 Entrar
37 </button>
38 </div>
39 </form>
40 <div padding>
41 <button ion-button round block icon-right icon-only color="nephritis" (click)="goToPageRegistrarse()">
42 <ion-icon name="person-add" ></ion-icon>
43 Crear Cuenta
44 </button>
45 </div>
46
47 </ion-content>

```

Figura 94. Archivo Login.html. Fuente: Elaboración propia.

Explicaremos las directivas de Ionic del contenido de nuestro archivo login.html.

El `<ion-header>` representa el encabezado de la página, se puede agregar botones a la izquierda o la derecha, además de otros componentes.

El `<ion-navbar>` es responsable de la barra de navegación. Podríamos decir que es el componente de navegación en este caso.

El `<ion-content>` mantiene el contenido de la página. Tiene una directiva "padding" que ofrece un poco de relleno al contenido.

El `<ion-icon>` es un componente responsable del manejo de los iconos. Simplemente damos el nombre del icono sobre la base de esta lista iconos

Ionic (revisar la documentación oficial en <http://ionicframework.com/docs/ionicons/>).

La etiqueta `<form [formGroup]="loginForm" (ngSubmit)="onLogin(loginForm.value)">` representa un formulario. `(ngSubmit)="onLogin(loginForm.value)` representa la función que se ejecutará al realizar el submit del formulario. `[formGroup]="loginForm"` permite hacer validaciones para el formulario.

El `<ion-label>` representa a una etiqueta de texto.

El `<ion-input>` representa un elemento del DOM de tipo "text" en este caso. El atributo `[formControlName]="aliasuser"` representa a la variable ubicada en el controlador (archivo login.ts) la cual servirá para realizar la validación.

El `<ion-item>` representa una sección dentro del html. `*ngIf` permite realizar la validación del `input`, para eso obtiene el valor del input (aliasuser) del formulario (loginForm) que se encuentra en el controlador, y muestra las etiquetas "`<p>`" con el error correspondiente que se ha generado.

El `<button>` representa a un botón. Contiene algunas propiedades propias de Ionic. El atributo "color" contiene el nombre de un color que se encuentra definido en el archivo "src/theme/variables.scss. El atributo "disabled" permite realizar activar o desactivar el botón según se halla definido, en este caso le decimos que el botón debe estar desactiva si existen errores de validación.

El atributo "(click)" permite declarar el método a ejecutarse cuando se haga click en este botón, en nuestra archivo este atributo ejecutara el metodo "goToPageRegistrarse()" que nos redirigirá a la página para registrar un nuevo usuario.

Para mayor información visitar la documentación oficial de Ionic: <https://ionicframework.com/docs/>

B. Archivo login.module.ts: este archivo nos lo crea Ionic. Donde debemos modificar la parte del “imports”, dentro debemos cambiar a “IonicModule.forRoot(Login)” para eliminar el error que genera, o en caso contrario podemos eliminar este archivo.

```
login.module.ts x
1  import { NgModule } from '@angular/core';
2  import { IonicModule } from 'ionic-angular';
3  import { Login } from './login';
4
5  @NgModule({
6    declarations: [
7      Login,
8    ],
9    imports: [
10     IonicModule.forRoot(Login),
11   ],
12   exports: [
13     Login
14   ]
15 })
16 export class LoginModule {}
```

Figura 95. Archivo login.module.ts. Fuente: Elaboración propia.

C. Archivo login.scss: contiene el archivo sass para poder modificar el estilo de los componentes de nuestra página.

```
login.scss x
1  page-login {
2    .error-b {
3      position: relative;
4      z-index: 1;
5      padding: 10px;
6      border-radius: 10px;
7      color: map-get($colors, light);
8      width: 235px;
9      text-align: center;
10     font-size: 12px;
11     background: map-get($colors, pomegranate);
12   }
13 }
```

Figura 96. Archivo login.scss. Fuente: Elaboración propia.

En este archivo hemos agregado un estilo cuando se muestra el mensaje de error. En la propiedad “color” hemos agregado un color que tenemos en el archivo “src/theme/variables.scss” mediante “map-get(\$color, light)” y en la propiedad “background” hicimos los mismo.

D. Archivo login.ts: contiene el controlador de la página, donde se define el comportamiento de la misma.

```

15 login.ts
1 import { Component } from '@angular/core';
2 import { IonicPage, NavController, NavParams, AlertController, LoadingController } from 'ionic-angular';
3 import { FormBuilder, FormGroup, Validators } from '@angular/forms';
4 import { Registrarse } from '../registrarse/registrarse';
5 import { HomePage } from '../home/home';
6 import { Menu } from '../menu/menu';
7
8 @IonicPage()
9 @Component({
10   selector: 'page-login',
11   templateUrl: 'login.html'
12 })
13 export class Login {
14   loginForm: FormGroup;
15   public datosUsuario: any;
16
17   constructor(
18     public navCtrl: NavController,
19     public navParams: NavParams,
20     public alertCtrl: AlertController,
21     public loadingCtrl: LoadingController,
22     public FormBuilder: FormBuilder
23   ) {
24     this.loginForm = this.createLoginForm();
25     this.datosUsuario = [];
26
27     private createLoginForm(){
28       return this.formBuilder.group({
29         'aliasuser': ['', [Validators.required, Validators.minLength(3), Validators.maxLength(30), Validators.pattern(/^[a-z0-9_-]{3,30}$/)]],
30         'passworduser': ['', [Validators.required, Validators.minLength(6), Validators.maxLength(30), Validators.pattern(/^[a-z0-9_-]{6,30}$/)]],
31       });
32     }
33     gotoPageRegistrarse(){
34       this.navCtrl.push(Registrarse);
35     }
36     ionViewDidLoad() {
37     }
38     mostrarAlert(subTitulo){
39       let alert = this.alertCtrl.create({
40         title: 'Golwin',
41         subtitle: subTitulo,
42         buttons: ['OK']
43       });
44       alert.present();
45     }
46     onLogin(datosLoginForm: any){
47     }
48   }

```

Figura 97. Archivo login.ts. Fuente: Elaboración propia.

1. Import es utilizado para la importación de módulos que contienen librerías y clases para poder emplearlas en nuestro proyecto.

En este caso importamos el elemento Component de Angular; IonicPage, NavController, NavParams, AlertController, LoadingController de ionic-angular; FormBuilder, FormGroup, Validators de angular/forms; También importamos las páginas que vallamos a utilizar para la navegación desde esta página como son Registrarse, HomePage, Menu.

2. @Component es un decorador que registra un componente. Los decoradores definen como se va a comportar la clase y se colocan antes de la clase. Los decoradores que existen son: @Component, @Injectable, @Pipe, @Directive.

3. Se exporta la clase llamada "**Login**", para luego poderla importarla desde cualquier otro sitio de la aplicación.
4. Creamos una variable "**loginForm**" de tipo FormGroup, esta variable es la tenemos agregada en el html, la cual identifica a nuestro formulario, tal como se muestra en la siguiente figura.

```
10 <form [formGroup]="loginForm" (ngSubmit)="onLogin(loginForm.value)">
```

Figura 98. Variable del controlador como identificador de nuestro formulario en la vista html. Fuente: Elaboración propia.

También creamos una variable **datosUsuario** que nos va a servir para almacenar y enviar los datos de un usuario, al momento de autenticarse, hacia otra página mediante NavController, en este caso nos envía los datos del usuario hacia la página **Menu**.

5. En el constructor inicializamos las variables **loginForm** y **datosUsuario**.
6. Método **createLoginForm** nos permite realizar las validaciones de los campos (input) de nuestro formulario.
7. El metodo **goToPageRegistrarse** nos redirige hacia la página **Registrarse** usando NavController con su método **push**. Este método es llamado al momento de hacer click en el botón **CREAR CUENTA**, tal como se muestra en la siguiente figura.

```
<div padding>
  <button ion-button round block icon-right icon-only color="nephritis" (click)="goToPageRegistrarse()">
    <ion-icon name="person-add" ></ion-icon>
    Crear Cuenta
  </button>
</div>
```

Figura 99. Invocando al método desde el html. Fuente: Elaboración propia.

8. Esta función es la que se ejecuta justamente cuando el componente ya está listo, pero se ejecuta después del constructor.

9. El metodo **mostrarAlert** nos crea una alerta que es un componente de Ionic.

10. El metodo **onLogin**, mantiene toda la lógica de este proceso al presionar sobre el botón **Entrar** de tipo **submit**. Este método recibe como parámetro “**datosLoginForm: any**” este parámetro se envía desde el html, el cual contiene todos los campos de nuestro formulario, es decir, invocamos a nuestro formulario colocando su identificador y añadimos sus valores, tal como se muestra en la siguiente figura.

```
10 <form [formGroup]="loginForm" (ngSubmit)="onLogin(loginForm.value)">
```

Figura 100. Invocación del método onLogin enviando como parámetro los valores del formulario. Fuente: Elaboración propia.

Este método lo implementaremos más adelante cuando tengamos creado y funcionando nuestros servicios.

Implementando todo lo antes mencionado podremos visualizar nuestra página de **Login** en el navegador web, tal como se muestra en la siguiente figura.

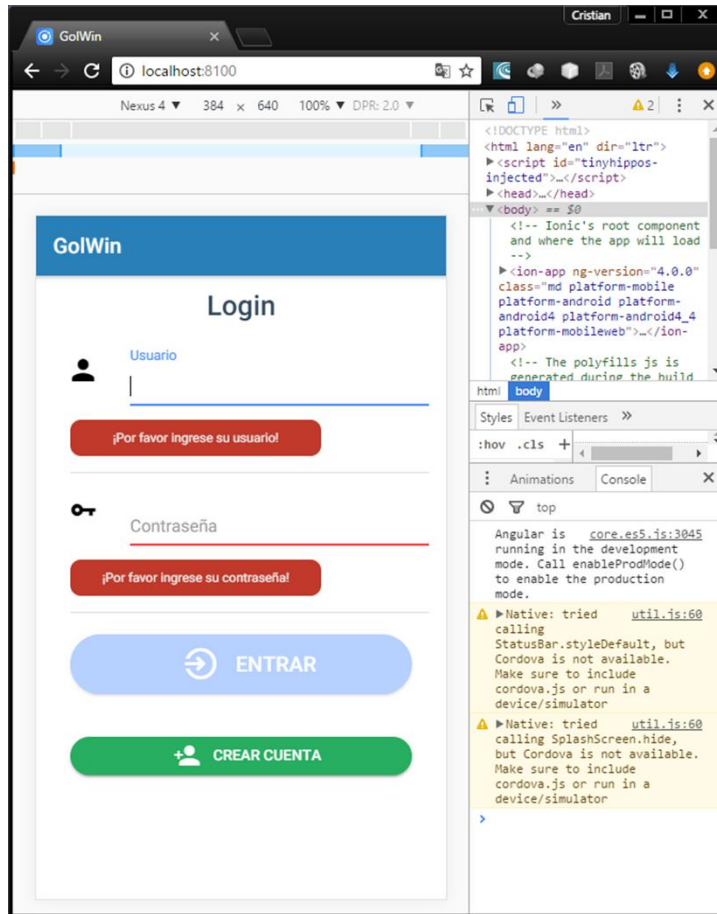


Figura 101. Pantalla de Login. Fuente: Elaboración propia.

3.3.3.7.6.4. Creando nuestros servicios (providers)

En esta parte crearemos nuestros servicios para poder usarlos en nuestra aplicación.

Para crear un *provider* lo hacemos mediante el siguiente comando desde la ventana de comandos “**ionic g provider LoginService**, tal como se muestra en la siguiente figura.

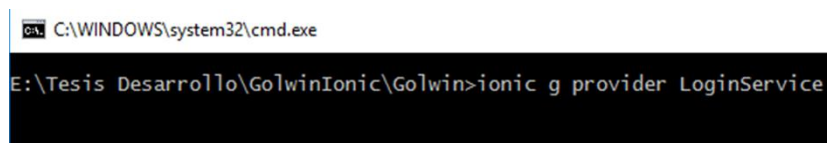


Figura 102. Comando para generar un provider. Fuente: Elaboración propia.

Al ejecutar el comando anterior le decimos a ionic que nos cree un provider con nombre *LoginService*. Ejecutado el comando se agregará una carpeta en **src/providers** con todos nuestros providers creados, para nuestro caso

crearemos dos providers mas, **GolwinService** y **GolwinPrivateService**, tal como se muestra en la siguiente figura.

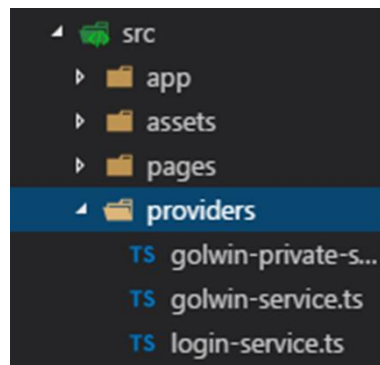


Figura 103. Contenido de la carpeta providers. Fuente: Elaboración propia.

Ahora abriremos nuestro archivo login-service y procederemos a implementarlo de la siguiente manera

```
login-service.ts
1  import { Injectable } from '@angular/core';
2  import { Http, Headers, RequestOptions } from '@angular/http';
3  import 'rxjs/add/operator/map';
4  import { Storage } from '@ionic/storage'
5
6  @Injectable()
7  export class LoginService {
8
9      public urlApi: string;
10     public urlOAuth: string;
11     public urlPetición: any;
12
13     constructor(public http: Http, public storage: Storage) {
14         this.urlApi = 'http://localhost:8084/GolwinAPI/';
15         this.urlOAuth = 'oauth/token?grant_type=password&client_id=GolwinIonic&client_secret=golwinionic&';
16         this.urlPetición = null;
17     }
18     getLogin(usuario){
19         this.urlPetición = this.urlApi + this.urlOAuth + 'username=' + usuario.aliasuser + '&password=' + usuario.passworduser;
20         let headers = new Headers({
21             "Content-Type": "application/json"
22         });
23         let options = new RequestOptions({ headers: headers });
24         return this.http.post(this.urlPetición, options)
25             .map(respuesta => respuesta.json());
26     }
27     getUsuario(alias, token){
28         let header = new Headers({
29             "Content-Type": "application/json"
30         });
31         let option = new RequestOptions({ headers: header });
32         this.urlPetición = this.urlApi + 'api/v1/usuarios/usualias/' + alias + '?access_token=' + token;
33         return this.http.get(this.urlPetición, option)
34             .map( respuesta => respuesta.json() );
35     }
36     limpiarStorage(){
37         this.storage.clear();
38     }
39     setStorageTokenAcceso(tokensAccess){
40         this.storage.set('tokens', tokensAccess);
41     }
42     setStorageUsuario(userDate){
43         this.storage.set('usuario', userDate);
44     }
45 }
```

Figura 104. Archivo login-service. Fuente: Elaboración propia.

1. Importamos las dependencias que vallamos a utilizar en nuestro servicio.
2. Creamos variables necesarias para facilitar el funcionamiento de nuestros servicios.
3. Inicializamos nuestras variables en el constructor.
4. Método que recibe como parámetro un array con el alias y password del usuario. Hace uso de **Http, Headers, RequestOptions** necesarios para trabajar con las peticiones hacia nuestro servidor. El método **getLogin** realiza la petición de tokens de acceso hacia el servidor, los datos que recibimos con el token de acceso es el que nos va a permitir realizar las futuras peticiones hacia el servidor agregando el token de acceso al final de la URL.
5. Método que recibe como parámetro el alias del usuario y el token de acceso solicitado en el método **getLogin**, este método solicita al servidor los datos del usuario, enviando como parámetro el alias recibido, además debemos agregar el token de acceso al final de la solicitud.
6. Métodos que utilizan el **Storage** que hemos importado, el cual nos va a permitir almacenar los objetos de las peticiones que hagamos hacia el servidor como es el caso del token de acceso que debemos guardarlo para poder utilizarlo en futuras peticiones, además debemos de guardar los datos del usuario para futuras operaciones. Los métodos reciben estos datos que los enviaremos desde el controlador.

3.3.3.7.6.5. Utilizando nuestro servicio LoginService en la implementación del método onLogin

Procederemos a darle la lógica de funcionamiento del método **onLogin**, este método se encargará de realizar un correcto acceso de usuario utilizando el método **getLogin** y **getUsuario** de nuestro servicio, además desde el controlador enviaremos los datos/objetos que deseamos que se guarden en nuestra aplicación, en este caso será el **token de acceso** y **los datos del usuario**, hacia los métodos de nuestro servicio **LoginService**.

Abrimos nuestro archivo **login.ts** ubicado en **src/pages/login** y procederemos a implementar del método, tal como se muestra en la siguiente figura.

```

1 import { Component } from '@angular/core';
2 import { IonicPage, NavController, NavParams, AlertController, LoadingController } from 'ionic-angular';
3 import { FormBuilder, FormGroup, Validators } from '@angular/forms'
4
5 import { Registrarse } from '../registrarse/registrarse';
6 import { HomePage } from '../home/home';
7 import { Menu } from '../menu/menu';
8
9 import { LoginService } from '../../providers/login-service';
10 @IonicPage()
11 @Component({
12   selector: 'page-login',
13   templateUrl: 'login.html',
14   providers: [ LoginService ]
15 })
16 export class Login {
17   loginForm: FormGroup;
18   public datosUsuario: any;
19
20   constructor(
21     public navCtrl: NavController,
22     public navParams: NavParams,
23     public loginServ: LoginService,
24     public alertCtrl: AlertController,
25     public loadingCtrl: LoadingController,
26     public formBuilder: FormBuilder
27   ) {
28     this.loginForm = this.createLoginForm();
29     this.datosUsuario = [];
30   }
31
32   private createLoginForm(){
33     return this.formBuilder.group({
34       'aliasuser': ['', [Validators.required, Validators.minLength(3), Validators.maxLength(30),
35         Validators.pattern(/^[a-z0-9_-]{3,30}$/)]],
36       'passworduser': ['', [Validators.required, Validators.minLength(6), Validators.maxLength(30),
37         Validators.pattern(/^[a-z0-9_-]{6,30}$/)]],
38     });
39   }
40
41   goToPageRegistrarse(){
42     this.navCtrl.push(Registrarse);
43   }
44
45   //funcion que se ejecuta justamente al cargar la pagina
46   ionViewDidLoad() {
47   }
48
49   onLogin(datosLoginForm: any){
50     let loader = this.loadingCtrl.create({content: "Conectando..."});
51     loader.present();
52     this.loginServ.getLogin(datosLoginForm)
53     .subscribe(
54       datosToken => {
55         if(datosToken){
56           this.loginServ.setStorageTokenAcceso(datosToken);
57           this.loginServ.getUsuario(datosLoginForm.aliasuser, datosToken.access_token).subscribe(
58             datosUser => {
59               if(datosUser){
60                 this.loginServ.setStorageUsuario(datosUser);
61                 this.datosUsuario.push(datosUser);
62               }
63             },
64             error => {
65               this.mostrarAlert('Error obteniendo datos de usuario. Inténtelo nuevamente.');

```

Figura 105. Implementación del método onLogin en el archivo login.ts. Fuente: Elaboración propia.

1. Importamos nuestro servicio LoginService.
2. Agregamos nuestro servicio LoginService como un providers
3. Añadimos nuestro LoginService al constructor de la clase.
4. El metodo **onLogin**, mantiene toda la lógica del proceso al presionar sobre el botón **Entrar** de tipo **submit**. Este método recibe como parámetro “**datosLoginForm: any**” este parámetro se envía desde el html, el cual contiene todos los campos de nuestro formulario, es decir, invocamos a nuestro formulario colocando su identificador y añadimos sus valores, tal como se muestra en la siguiente figura.

```
10 <form [formGroup]="loginForm" (ngSubmit)="onLogin(loginForm.value)">
```

Figura 106. Invocación del método onLogin enviando como parámetro los valores del formulario. Fuente: Elaboración propia.

Lo primero que hacemos es agregar el componente **loader** y mostrarlo, muestra una animación de cargando mientras se realiza el proceso.

Ahora vamos a hacer uso de nuestro servicio que hemos importado como **LoginService**, en cual hemos definido en el constructor como **loginServ**. En nuestro servicio tenemos el metodo **getLogin** el cual recibe como parámetro **datosLoginForm** que son los campos de nuestro formulario, si obtenemos una respuesta correcta guardaremos los datos/objetos mediante el metodo **setStorageTokenAcceso** el cual recibe como parámetro **datosToken** que son los datos devueltos por la petición de nuestro método anterior, como paso siguiente realizamos otra solicitud mediante el metodo **getUsuario** que recibe como parámetros el alias del usuario que se ingresó en el formulario y el token de acceso que se recibió de la solicitud del método anterior; este metodo solicita los datos del usuario y los almacena en el metodo **setStorageUsuario** de nuestro servicio, además debemos almacenarlo en la variable **datosUsuario** que hemos creado, esto nos servirá para enviar los datos del usuario a la siguiente página. En cualquiera de las dos peticiones anteriores sucediera algún error procederemos a mostrar una alerta con un mensaje adecuado.

Si todo ha sido correcto procederemos a **detener el loader** y nos rediregiremos a la página **Menu** y a la vez enviaremos los datos del usuario almacenado en la variable **datosUsuario**, utilizando el **NavController** mediante su metodo **setRoot**. La página menú contiene un menú lateral donde la página de inicio es Home.

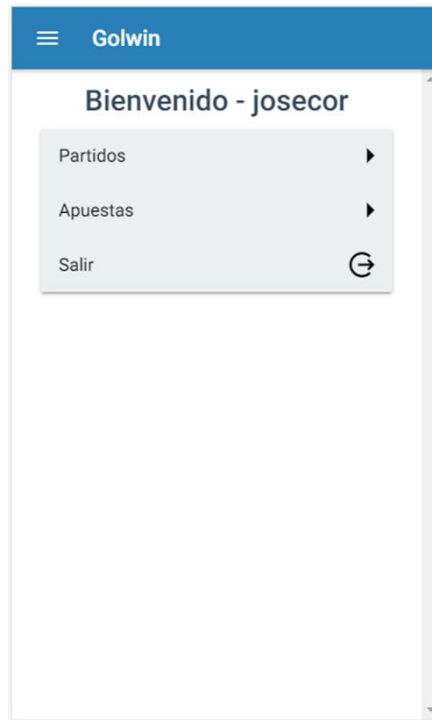


Figura 107. Página Home. Fuente: Elaboración propia.

A continuación, se muestra el menú lateral

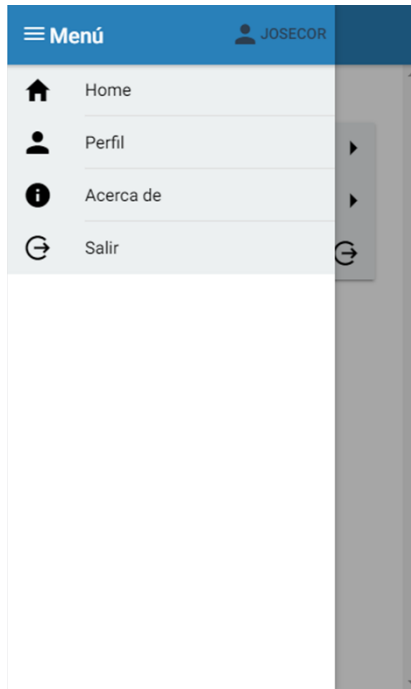


Figura 108. Página Menu (Menú Lateral). Fuente: Elaboración propia.

3.3.3.7.6.6. Visualizar lista de partidos

Mostraremos una lista con los partidos que están disponibles para realizar una apuesta.

Primero crearemos nuestro servicio, para eso abrimos nuestro archivo **src/providers/golwin-private-service.ts** y lo implementaremos tal como muestra la siguiente figura.

```

5 golwin-private-service.ts
1 import { Injectable } from '@angular/core';
2 import { Http, Headers, RequestOptions, Response } from '@angular/http';
3 import 'rxjs/add/operator/map';
4 import { Storage } from '@ionic/storage';
5 import { Observable } from 'rxjs/Rx';
6 import 'rxjs/add/operator/catch';
7 import 'rxjs/add/operator/mergeMap';
8 import 'rxjs/add/observable/fromPromise';
9
10 @Injectable()
11 export class GolwinPrivateService {
12
13   public apiUrl: string;
14   public tokenAces:any;
15
16   constructor(public http: Http, public storage: Storage) {
17     this.apiUrl = 'http://localhost:8084/GolwinAPI/api/v1/';
18     this.tokenAces = null;
19   }
20   // Obtenemos el token de acceso guardado en el Storage
21   getAccessToken(){
22     return Observable.fromPromise(this.storage.get('tokens'));
23   }
24   //realizamos la peticion para obtener los partidos
25   getPartidos(compCod){
26     return this.getAccessToken().flatMap( token => {
27       if( token ) {
28         this.tokenAces = token.access_token;
29       }
30       return this.http.get(this.apiUrl + 'partidos/' + compCod +'?access_token=' + this.tokenAces)
31         .timeout(60000)
32         .map( respuesta => respuesta.json() )
33     });
34   }

```

Figura 109. Archivo golwin-private-service.ts. Fuente: Elaboración propia.

Ahora implementaremos el controlador abriendo el archivo `src/pages/partidos/partidos.ts` de la siguiente manera.

```

15 partidos.ts
1 import { Component } from '@angular/core';
2 import { IonicPage, NavController, NavParams, LoadingController, AlertController } from 'ionic-angular';
3 import { Detallepartido } from '../detallepartido/detallepartido';
4 import { Login } from '../login/login';
5 import { GolwinPrivateService } from '../../providers/golwin-private-service';
6
7 @IonicPage()
8 @Component({
9   selector: 'page-partidos',
10  templateUrl: 'partidos.html',
11  providers: [GolwinPrivateService]
12 })
13 export class Partidos {
14
15   public competicion: any;
16   public partidos: any;
17   public jornadas: any;
18   public loading: any;
19
20   constructor(
21     public navCtrl: NavController,
22     public navParams: NavParams,
23     public golwinPrivServ: GolwinPrivateService,
24     public loadingCtrl: LoadingController,
25     public alertCtrl: AlertController
26   ) {
27     this.competicion = this.navParams.get('competicion');
28     this.partidos = [];
29     this.jornadas = [];
30   }

```

```

31
32 ionViewDidLoad() {
33   this.presentLoading();
34   this.golwinPrivServ.getPartidos(this.competicion.compcod).subscribe(
35     datosPartido => {
36       if(datosPartido){
37         this.partidos = datosPartido;
38       }else {
39         this.showAlert('No se encontraron partidos registrados para la ' +
40           'competición ' + this.competicion.compnom);
41       }
42     },
43     error => {
44       this.loading.dismiss();
45       if(error.name == 'TimeoutError'){
46         this.showAlert('Tiempo de espera agotado para realizar esta solicitud.'+
47           'No se pudo establecer la conexión con el servidor. Compruebe su conexión a internet');
48       }
49       if(error.status == 0){
50         this.showAlert('Datos se sesión no válidos. Vuelva a iniciar sesión');
51         this.navCtrl.setRoot(Login);
52       }
53     },
54     () => {
55       this.loading.dismiss();
56     }
57   );
58 }
59 presentLoading() {
60   this.loading = this.loadingCtrl.create({
61     content: "Cargando datos..."
62   });
63   this.loading.present();
64 }
65 showAlert(subTitulo) {
66   let alert = this.alertCtrl.create({
67     title: 'Golwin',
68     subTitle: subTitulo,
69     buttons: ['OK']
70   });
71   alert.present();
72 }
73 goToDetallePartido(part){
74   this.navCtrl.push(Detallepartido, {partido: part} );
75 }
76 }

```

Figura 110. Implementación del archivo partidos.ts. Fuente: Elaboración propia.

Ahora implementaremos el archivo **src/pages/partidos/partidos.html** de la siguiente manera.


```

partidos.html
1 <ion-header>
2 <ion-navbar color="belize_hole">
3 <ion-title>Partidos - {{competicion.compnom}}</ion-title>
4 </ion-navbar>
5 </ion-header>
6
7 <ion-content padding>
8 <div *ngFor="let part of partidos; let i=index">
9 <h4 *ngIf="i==0" class="titulo" >{{part.jornadas.jordesc}}</h4>
10 </div>
11 <ion-grid>
12 <ion-row>
13 <ion-col class="table table-header">Equipos</ion-col>
14 <ion-col class="table table-header">Fecha-Hora</ion-col>
15 <ion-col class="table table-header">Detalle</ion-col>
16 </ion-row>
17 <ion-row *ngFor="let part of partidos">
18 <ion-col class="table table-celdaleft">
19 
20 {{part.listaequiposByEqlocal.paises.paisnombre}} - {{part.listaequiposByEqvisita.paises.paisnombre}}
21 
22 </ion-col>
23 <ion-col class="table table-celdacenter">
24 {{part.partfecha}} {{part.parthora}}
25 </ion-col>
26 <ion-col class="table table-celdacenter">
27 <button ion-button round color="wet_asphalt" (click)="goToDetallePartido(part)">
28 <ion-icon name="list-box"></ion-icon>
29 </button>
30 </ion-col>
31 </ion-row>
32 </ion-grid>
33
34 </ion-content>

```

Figura 111. Implementación del archivo partidos.html. Fuente: Elaboración propia.

Agregaremos algunos estilos a nuestra página de la siguiente manera en el archivo `src/pages/partidos/partidos.scss`

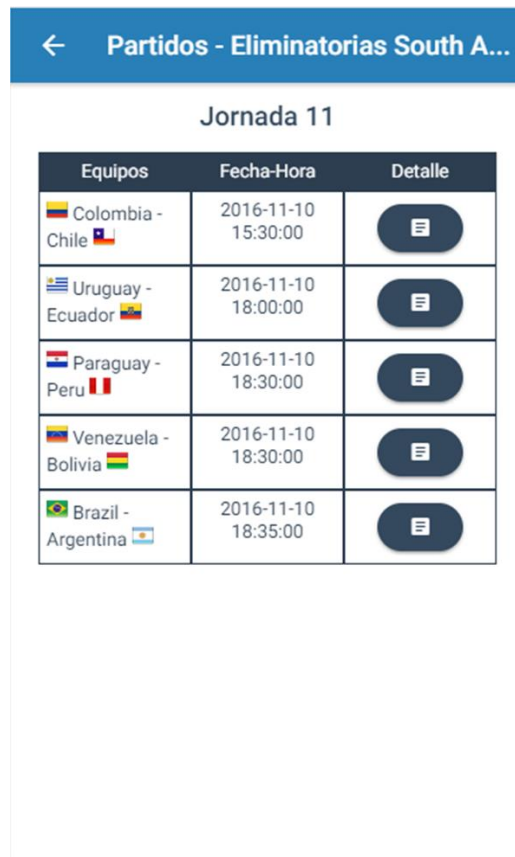
```

partidos.scss
1 page-partidos {
2   .table {
3     color: map-get($colors, midnight_blue);
4     border: 1px solid #2c3e50;
5   }
6   .table-header {
7     text-align: center;
8     color: #fff;
9     background: #2c3e50;
10  }
11  .table-celdacenter {
12    text-align: center;
13  }
14  .table-celdaleft {
15    text-align: left;
16  }
17
18 }
19

```

Figura 112. Archivo partidos.scss. Fuente: Elaboración propia.

Terminado toda la implementación de nuestra página podemos visualizarla, tal como se muestra en la siguiente figura.



The screenshot shows a mobile application interface with a blue header bar containing a back arrow and the text 'Partidos - Eliminatorias South A...'. Below the header, the text 'Jornada 11' is centered. A table with three columns: 'Equipos', 'Fecha-Hora', and 'Detalle' lists five matches. Each match row includes the names of the teams with their respective flags, the date and time of the match, and a circular button with a list icon for more details.

Equipos	Fecha-Hora	Detalle
Colombia - Chile	2016-11-10 15:30:00	[Detalle]
Uruguay - Ecuador	2016-11-10 18:00:00	[Detalle]
Paraguay - Peru	2016-11-10 18:30:00	[Detalle]
Venezuela - Bolivia	2016-11-10 18:30:00	[Detalle]
Brazil - Argentina	2016-11-10 18:35:00	[Detalle]

Figura 113. Vista de la página Partidos. Fuente: Elaboración propia.

3.3.3.8. Framework jQuery Mobile

Para la implementación de la aplicación Golwin con jQuery Mobile se realizó el siguiente procedimiento.

3.3.3.8.1. Descargar JQuery Mobile

Para descargar jQuery Mobile debemos dirigirnos a su página oficial ingresando en la siguiente URL: <http://jquerymobile.com/download-builder/> seleccionamos todo y luego haremos click en el botón **Build My Download**.

3.3.3.8.2. Descargar JQuery

jQuery mobile depende de jQuery para su correcto funcionamiento, así que tenemos que descargarlo para poder utilizarlo, para descargar jQuery nos

dirigimos a su página oficial en la siguiente URL: <https://code.jquery.com/> y descargamos la versión 2.x jQuery.

3.3.3.8.3. Creando nuestro proyecto en el IDE NetBeans

Abrimos nuestro IDE NetBeans y crearemos un nuevo proyecto HTML5. Nos dirigimos al menú **File** luego hacemos click sobre **New Project...**

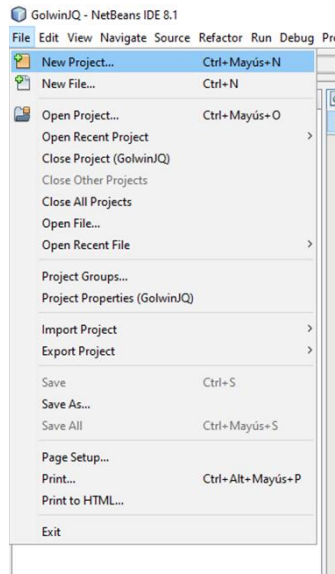


Figura 114. Click sobre New Project... Fuente: Elaboración propia.

De la ventana que nos aparece debemos elegir la categoría **HTML5/JavaScript** y dentro de ello seleccionamos un proyecto **HTML5/JS Application** y damos click en el botón **Next**

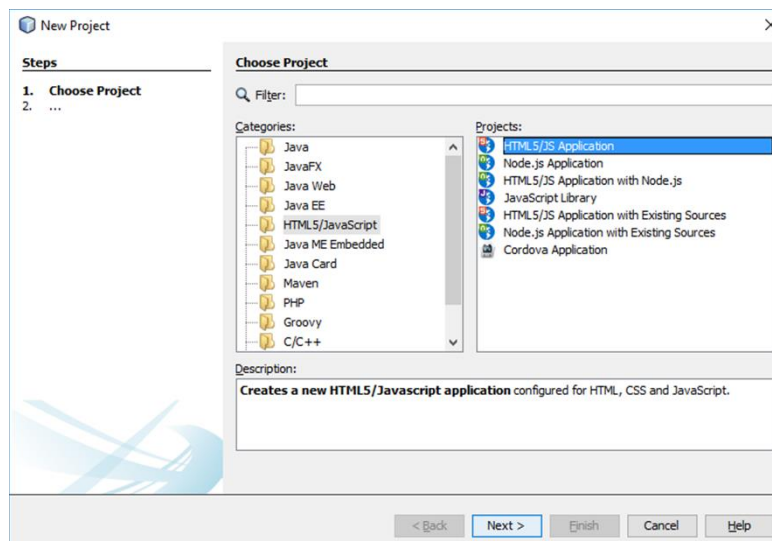


Figura 115. Seleccionamos el tipo de proyecto. Fuente: Elaboración propia.

De la ventana que aparece colocaremos el nombre de nuestro proyecto y elegimos un directorio donde guardarlo y damos click al botón **Next**.

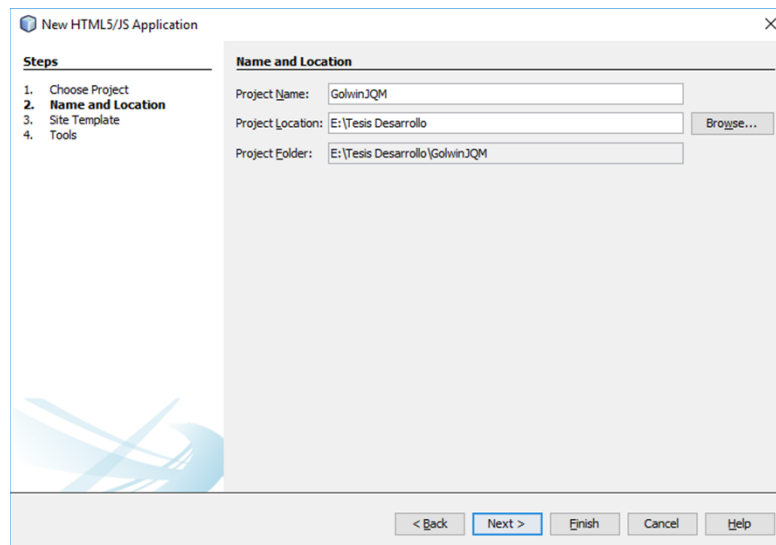


Figura 116. Nombre y directorio del proyecto. Fuente: Elaboración propia.

De la siguiente ventana elegimos la opción **No Site Template** y damos click en **Next**.

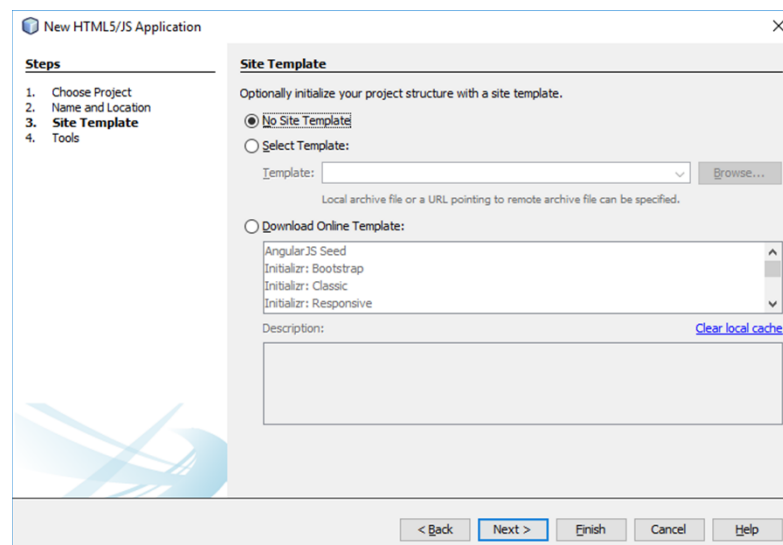


Figura 117. No elegimos un Template. Fuente: Elaboración propia.

En la ventana que nos aparece desmarcamos todos los checks y presionamos click sobre el botón **Finish**.

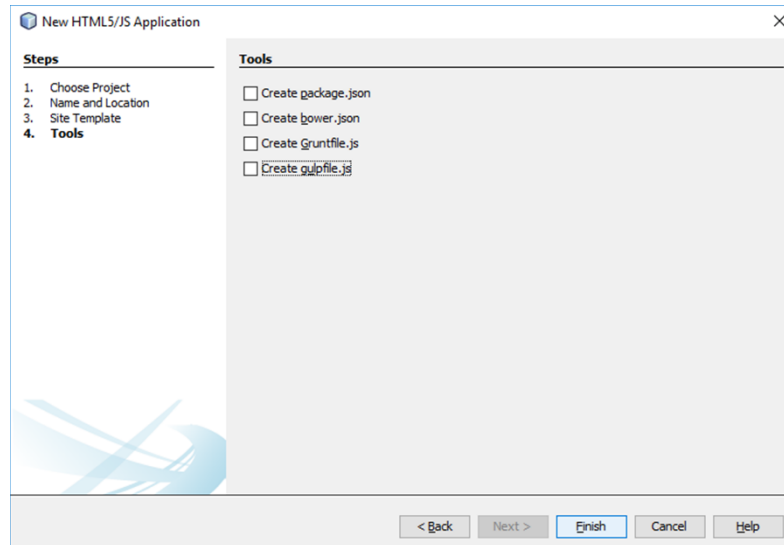


Figura 118. Desmarcamos los checks. Fuente: Elaboración propia.

Ahora ya tenemos creado nuestro proyecto creado, tal como lo podemos observar en la siguiente figura.

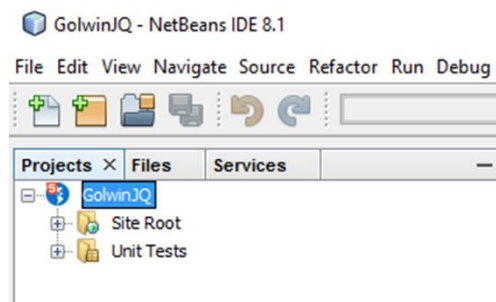


Figura 119. Proyecto creado. Fuente: Elaboración propia.

3.3.3.8.4. Agregando jQuery Mobile y jQuery a nuestro proyecto

Vamos a crear unas carpetas donde colocaremos los archivos que hemos descargado anteriormente, jQuery Mobile y jQuery.

Además, debemos descargar dos archivos más para poder almacenar nuestras variables y poder pasarlas de una página a otra.

jsStorage.js: <https://github.com/andris9/jStorage/blob/master/jstorage.js>

json2.js: <http://cdnjs.cloudflare.com/ajax/libs/json2/20110223/json2.js>

Crearemos una carpeta que la llamaremos **css** donde colocaremos todos nuestros archivos css, también aquí colocaremos la carpeta **images** que viene con jQuery Mobile (debemos descomprimir el zip que descargamos de jQuery Mobile), además debemos crear una carpeta que la llamaremos **js** donde colocaremos nuestros archivos JavaScript que hemos descargado.

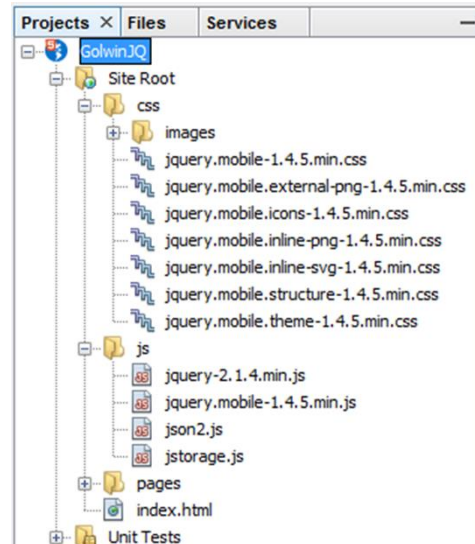


Figura 120. Estructura de carpetas y archivos del proyecto. Fuente: Elaboración propia.

3.3.3.8.5. Incluir los archivos en nuestro html.

Para construir una aplicación con jQuery Mobile debemos incluir los archivos respectivos dentro de la etiqueta **head** de nuestro archivo **html**. Debemos incluirlos de la siguiente manera, tal como se muestra en la siguiente figura.

```
index.html x
Source History
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Golwin - Login</title>
5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <link rel="icon" href="data:;base64,iVBORw0KGgo=">
8 <link href="css/jquery.mobile-1.4.5.min.css" rel="stylesheet" type="text/css"/>
9 <script src="js/json2.js" type="text/javascript"></script>
10 <script src="js/jquery-2.1.4.min.js" type="text/javascript"></script>
11 <script src="js/jstorage.js" type="text/javascript"></script>
12 <script src="js/jquery.mobile-1.4.5.min.js" type="text/javascript"></script>
13 <script ...84 lines />
97 </head>
98 <body ...26 lines />
124 </html>
```

Figura 121. Archivos incluidos en el head del html. Fuente: Elaboración propia.


3.3.3.8.6. Creación de la interfaz Login

Lo primero que tenemos que hacer es recurrir a la documentación y los demos de ejemplo que nos ofrece jQuery Mobile en su página oficial, para eso accedemos a las siguientes url:

Api Documentacion: <http://api.jquerymobile.com/>

Demos: <http://demos.jquerymobile.com/1.4.5/>

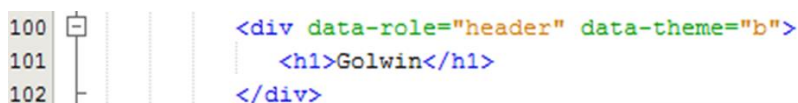
Para crear una página dentro de la etiqueta **body** se utiliza el atributo **data-role = "page"**, jQuery Mobile agrega código y eventos necesarios para que se comporte y vea como tal.



```
98 <body>
99   <div data-role="page" id="pageLogin">
101 </div>
102 </body>
```

Figura 122. Creando una pagina. Fuente: Elaboración propia.

Para agregar un encabezado a la pagina que vamos a crear usamos el mismo principio utilizando el atributo **data-role = "header"**.



```
100 <div data-role="header" data-theme="b">
101   <h1>Golwin</h1>
102 </div>
```

Figura 123. Header de la página. Fuente: Elaboración propia.

Ahora crearemos el cuerpo de nuestra página utilizando **data-role = "main"** y la clase **class = "ui-content"** que nos va a agregar el código y los eventos necesarios para que se comporte como el cuerpo del contenido de nuestra página.

```

<div data-role="main" class="ui-content">
  <center>
    <h2>Login</h2>
  </center>
  <form id="loginusuario" method="post">
    <div class="ui-field-contain">
      <label for="usualias">Usuario</label>
      <input type="text" name="usualias" id="usualias" placeholder="Ingrese Usuario"
        data-clear-btn="true" required="required" autofocus="autofocus"/>
    </div>
    <div class="ui-field-contain">
      <label for="usupassw">Password</label>
      <input type="password" name="usupassw" id="usupassw" placeholder="Ingrese Password"
        required="required" data-clear-btn="true"/>
    </div>
    <input type="submit" value="Entrar" data-icon="check" data-iconpos="right"
      data-transition="pop"/>
  </form>
  <div class="ui-field-contain">
    <a href="pages/registro.html" data-role="button" data-icon="user" data-iconpos="right"
      data-transition="slide">Crear Cuenta</a>
  </div>
</div>

```

Figura 124. Contenido del cuerpo de nuestra página. Fuente: Elaboración propia.

Por último, crearemos el footer de la página con el atributo **data-role = "footer"**

```

<div data-role="footer" data-theme="b">
  <h1>&copy; 2017</h1>
</div>

```

Figura 125. Footer de la página. Fuente: Elaboración propia

Si abrimos nuestra página que hemos creado en el navegador de Google Chrome y presionamos la tecla F12, además activamos a vista de dispositivo lo visualizaremos de la siguiente manera:

The image shows a mobile application login screen. At the top, a black bar contains the text 'Golwin'. Below this, the word 'Login' is centered in a bold, black font. The form consists of two input fields: the first is labeled 'Usuario' and contains the placeholder text 'Ingrese Usuario'; the second is labeled 'Password' and contains the placeholder text 'Ingrese Password'. Below the input fields are two buttons: 'Entrar' with a checkmark icon to its right, and 'Crear Cuenta' with a person icon to its right. At the bottom of the screen, a black bar contains the text '© 2017'.

Figura 126. Pantalla de Login. Fuente: Elaboración propia.

3.3.3.8.7. Dando funcionalidad a la página pageLogin

En esta parte utilizaremos jQuery para la funcionalidad del Login, para hacer las peticiones via AJAX a nuestro servidor.

Primero haremos una petición para poder obtener el token de acceso que lo utilizaremos para poder realizar futuras peticiones que requieran autenticación. Si esta petición tiene éxito procederemos a realizar internamente otra petición Ajax al servidor solicitando los datos del usuario que se acaba de logear, en esta URL tenemos que agregar el token de acceso al final para poder realizar la petición y obtener los datos de forma exitosa.

Para poder hacer esto agregamos el siguiente código a nuestro archivo.

```

<script>
$(document).ready(function () {
    $("#loginusuario").submit(function (event) {
        var usuario; //variable para almacenar el usuario que va a iniciar sesion
        usuario = $("#usualias").val();
        event.preventDefault();
        //creamos y mostramos el loading
        $.mobile.loading("show", {
            text: 'Conectando...',
            textVisible: true,
            textonly: false,
            html: ""
        });
        //realizamos la peticion ajax para obtener el token de acceso
        $.ajax({
            url: "http://localhost:8084/GolwinAPI/oauth/token?grant_type=password" +
                "&client_id=clientjqm&client_secret=123456789&username=" +
                $("#usualias").val() + "&password=" + $("#usupassw").val(),
            timeout: 2000,
            dataType: 'json',
            type: 'POST',
            success: function (datostoken, textStatus, jqXHR) {
                //almacenamos el token con jStorage
                $.jStorage.set("access_token", datostoken.access_token);
                /* Realizamos la peticion via ajax para obtener los datos del usuario
                 * como ya tenemos el token de acceso lo podemos utilizar en esta
                 * petición */
                $.ajax({
                    headers: {
                        "Accept": "application/json",
                        "Content-Type": "application/json"
                    },
                    url: "http://localhost:8084/GolwinAPI/api/v1/usuarios/usualias/" +
                        usuario + "?access_token=" + datostoken.access_token,
                    dataType: 'json',
                    type: 'GET',
                    success: function (datosusuario, textStatus, jqXHR) {
                        //guardamos el codigo del usuario actual
                        $.jStorage.set("usucod", datosusuario.usucod);
                        //guardamos el nombre del usuario actual
                        $.jStorage.set("usunom", datosusuario.usunom);
                        //guardamos el alias del usuario actual
                        $.jStorage.set("usualias", datosusuario.usualias);
                    }
                });
            },
            //ocultamos el loading
            $.mobile.loading("hide");
            // hacemos el redireccionamiento a la siguiente pagina
            $.mobile.changePage("#pageHome", {transition: "pop"});
            //window.location = "pages/home.html";
        },
        error: function (jqXHR, textStatus, errorThrown) {
            $.mobile.loading("hide");
            alert("Error: " + jqXHR.status + "\n" +
                "Message: " + jqXHR.statusText + "\n" +
                "Response: " + jqXHR.responseText + "\n" +
                errorThrown);
        }
    });
});
$(document).delegate("#pageHome", "pageshow", function () {
    //monstramos el alias del usuario en la otra pagina
    $("#useralias").html($.jStorage.get("usualias"));
});
});
</script>

```

Figura 127. Script para el Login de un usuario. Fuente: Elaboración propia.

En el metodo **delegate** le decimos a jQuery Mobile que cuando la página con **id = "pageHome"** se muestre debe ejecutar la siguiente función, debe de mostrar el **usualias** guardado con **jStorage** en el elemento que contenga **id = "useralias"** ubicado en la página **pageHome**.

Si el login ha sido exitoso se mostrará la pantalla **pageHome** con un menú principal.

3.3.3.8.8. Creando nuestro menú principal "pageHome"

En una nueva etiqueta **div** crearemos una nueva página que contendrá nuestro menú principal, tal como se muestra en la siguiente figura.

```
<div id="pageHome" data-role="page">
  <div data-role="header" data-theme="b">
    <h1>Golwin</h1>
    <a id="useralias" class="ui-btn ui-btn-corner-all ui-shadow ui-icon-user ui-btn-icon-left ">
      usualias
    </a>
  </div>
  <div data-role="main" class="ui-content">
    <center>
      <h3>Menú Principal</h3>
    </center>
    <ul data-role="listview" data-inset="true">
      <li><a href="competiciones.html">Partidos</a></li>
      <li><a href="">Apostar</a></li>
      <li><a href="#pageLogin">Salir</a></li>
    </ul>
  </div>
  <div data-role="footer" data-theme="b">
    <h1>&copy; 2017</h1>
  </div>
</div>
```

Figura 128. Página pageHome. Fuente: Elaboración propia.

Al acceder a esta pantalla se mostrará de la siguiente manera, debemos de fijarnos que aún no hemos dado funcionalidad a esta página para que muestre el **usualias** que hemos enviado de la página anterior.

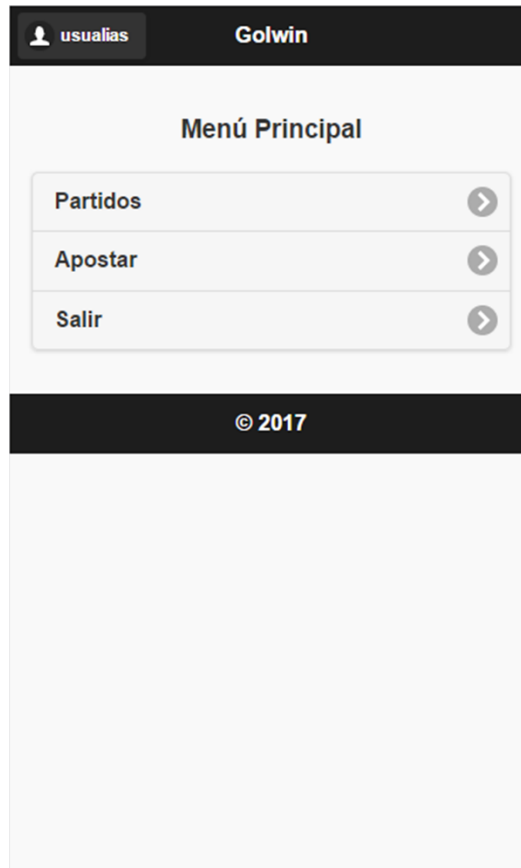


Figura 129. Pantalla Menú Principal. Fuente: Elaboración propia.

3.3.3.8.9. Dando funcionalidad a la página pageHome

Debemos de agregar el siguiente código para poder dar la funcionalidad que queremos obtener en nuestra página **pageHome**.

Verificamos si existe el token de acceso, si no existe mostramos la pantalla de Login, caso contrario nos muestre el **usualias** del usuario actual.

```

<script type="text/javascript">
  $(function () {
    //consultamos si el access_token existe
    var token = $.jStorage.get("access_token");
    //alert(token);
    //si no existe mandamos al usuario a la pagina de login(index.html)
    if (token === null) {
      $.mobile.changePage("#pageLogin", {transition: "pop"});
    } else {
      //si el usuario ha hecho login, asignamos la variable usualias
      //y lo asignamos al navbar
      $("#useralias").html($.jStorage.get("usualias"));
    }
  });
</script>

```

Figura 130. Código para la página pageHome. Fuente: Elaboración propia.

Si todo esta correcto podemos visualizar la pantalla Menú Principal con el **usualias** del usuario actual, tal como se muestra en la siguiente figura.

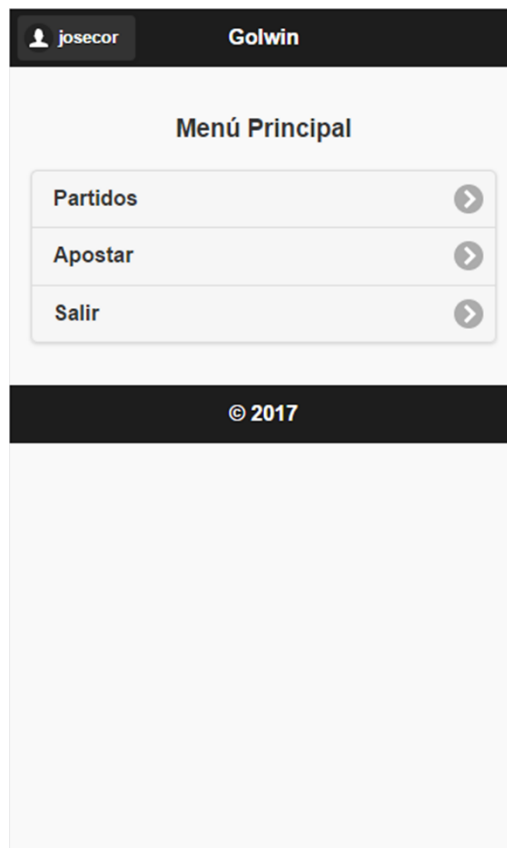


Figura 131. Pantalla Menu Principal que muestra el usualias. Fuente: Elaboración propia.

3.3.3.8.10. Obteniendo y mostrando la lista de partidos

Lo que haremos será hacer una petición vía Ajax a nuestro servidor solicitando la lista de partidos que se encuentren activos según la competición y así poder realizar una apuesta a ese partido. Solo mostraremos la lista de partidos en esta parte.

Creamos nuestro HTML de la siguiente manera.

```

<!--Pagina que muestra la lista de partidos de cada competicion -->
<div id="pagelistapartidos" data-role="page">
  <div data-role="header" data-theme="b" data-position="fixed">
    <h1>Partidos</h1>
    <a href="#pagelistacompeticiones"
      class="ui-btn ui-corner-all ui-icon-arrow-l ui-btn-icon-notext "
      data-transition="slide" data-direction="reverse">
    </a>
  </div>
  <div data-role="main" class="ui-content">
    <center>
      <h4 id="nombrecompeticion">Competicion</h4>
      <h5 id="nombrejornada">Jornada</h5>
    </center>
    <div id="listapartidosbycompeticion">
      <table id="tblistaPartido" data-role="table" data-mode="columntoggle"
        class="ui-responsive ui-shadow" border="1">
        <thead>
          <tr >
            <th>Equipos</th>
            <th data-priority="1">Fecha - Hora</th>
            <th data-priority="1">Detalle</th>
          </tr>
        </thead>
        <tbody>
          </tbody>
        </table>
    </div>
  </div>
  <div data-role="footer"></div>
</div>
<!--Fin de la pagina -->

```

Figura 132. Estructura de la página pageListaPartidos. Fuente: Elaboración propia.

Ahora procederemos a darle funcionalidad con el siguiente código JavaScript, utilizando jQuery.

```

//funcion para mostrar los partidos al hacer clic en una determinada competicion
$("#competiciones").on("click", "a", function () {
    var enlacepartido;
    competicion = $(this).text();
    //obtenemos los enlaces
    enlacepartido = $(this).attr("rel");
    //limpiamos el contenido de la tabla para luego volver a llenarla
    $("#tbody tr").remove();
    //Mostramos el loading
    $.mobile.loading("show", {
        text: "Cargando datos...",
        textVisible: true,
        textonly: false,
        html: ""
    });
    //Realizamos la peticion ajax
    $.getJSON(enlacepartido + "?access_token=" + $.jStorage.get("access_token"), null, function (datos) {
        var html = "";
        if (datos.length > 0) {
            for (var i = 0; i < datos.length; i++) {
                //creamos el contenido dinamicamente
                html += "<tr>";
                html += "<td><img src=\"data:image/png;base64,\" +
                    datos[i].listaequiposByEqlocal.paises.paisband16 +
                    "\" /> + datos[i].listaequiposByEqlocal.paises.paisnombre +
                    \" - \" + <img src=\"data:image/png;base64,\" +
                    datos[i].listaequiposByEqvisita.paises.paisband16 +
                    "\" /> + datos[i].listaequiposByEqvisita.paises.paisnombre +
                    \"</td>";
                html += "<td> + datos[i].partfecha + \" - \" + datos[i].parthora + \"</td>";
                html += "<td>";
                html += "<div data-role=\"controlgroup\" data-type=\"horizontal\" data-theme=\"b\" >";
                html += "<a href=\"#detallepartido\" \" +
                    \"rel=\"http://localhost:8084/GolwinAPI/api/v1//detallespartido/\" +
                    datos[i].partcod + \"\" +
                    \"class=\"ui-btn ui-corner-all ui-icon-action ui-btn-icon-notext ui-btn-inline\" \" +
                    \"data-transition=\"slide\" id=\"\" + datos[i].partcod + \"\"></a>";
                html += "</div>";
                html += "</td>";
                html += "</tr>";
                //guardamos la jornada actual
                jornada = datos[i].jornadas.jordesc;
            }
        }
        //añadimos a nuestra tabla todos los datos encontrados mediante la funcion html
        $("#tblistaPartido tbody").html(html);
        $.mobile.loading("hide");
    });
    //asignamos el nombre de la competicion a mostrar
    $("#nombrecompeticion").text(competicion);
    //asignamos la jornada a mostrar
    $("#nombrejornada").text(jornada);
});

```

Figura 133. Código para mostrar la lista de partidos. Fuente: Elaboración propia.

Si todo esta correcto y abrimos nuestra página en el explorador se nos mostrara la siguiente pantalla con la lista de partidos de su respectiva competición.

Equipos	Fecha - Hora	Detalle
Colombia - Chile	2016-11-10 - 15:30:00	
Uruguay - Ecuador	2016-11-10 - 18:00:00	
Paraguay - Peru	2016-11-10 - 18:30:00	
Venezuela - Bolivia	2016-11-10 - 18:30:00	
Brazil - Argentina	2016-11-10 - 18:35:00	

© 2017

Figura 134. Pantalla con la lista de partidos. Fuente: Elaboración propia.

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

a. Se realizó la búsqueda de los framework basados en tecnología web para dispositivos móviles existentes en la actualidad, siendo un total de 49 frameworks los cuales son los que se mencionan por la comunidad, para esto se realizó la búsqueda en Google y se ordenó por cantidad de resultados encontrados. Esta búsqueda y ordenamiento se realizó con el fin de obtener que frameworks son los más buscados por la comunidad para poder enfocarnos en ellos y posteriormente realizar el análisis comparativo. Además, se realizó una selección de 9 frameworks los cuales son los que más se mencionan en las diferentes paginas consultadas, para de esta manera saber cuál de todos ellos son los que más se mencionan y/o utilizan por la comunidad de desarrolladores y/o interesados en estos frameworks,

obteniendo como resultados a los frameworks Ionic, jQuery Mobile y PhoneGap los más sobresalientes, pero para este estudio se eligieron los dos primeros, ya que jQuery Mobile se puede integrar con PhoneGap para permitir crear aplicaciones tal como lo hace el framework Ionic.

- b. Se aplicó una evaluación teórica mediante el modelo QSOS, para determinar que framework tiene más características en la matriz QSOS, siendo de la siguiente manera:

Durabilidad intrínseca: el framework que sobresale es Ionic.

Industrialización de desarrollo: ambos frameworks tienen el mismo puntaje.

Estrategia: existe un empate entre ambos frameworks ya que pertenecen al mismo ámbito.

Solución industrializada: existe un empate entre ambos frameworks, ya que ambos cuentan con muchos recursos para su aprendizaje.

Disponibilidad de plataformas: ambos frameworks tienen el mismo puntaje, ya que ambos son frameworks implementados con tecnología web y pueden funcionar en muchas plataformas.

Adaptabilidad técnica: existe un empate entre ambos frameworks ya que están compuestos de módulos muy bien definidos.

- c. Se desarrolló una aplicación de apuestas de fútbol denominada Golwin, esta aplicación se implementó en ambos frameworks para realizar la evaluación con cada uno de ellos. Golwin se conecta a una API RESTful para consumir los recursos que esta brinda sobre los partidos y así realizar las apuestas, utilizando JSON para el intercambio de datos.

- d. Se desarrolló una API RESTful para brindar los servicios hacia las aplicaciones del lado del cliente desarrollados con los frameworks seleccionados, esta se implementó utilizando los frameworks como son Spring MVC, Hibernate y Spring Security OAuth 2.

- e. Se realizó la evaluación de los frameworks mediante el modelo de calidad ISO 25010, el cual se adecua de forma correcta para realizar la evaluación de los

frameworks, también se utilizó el método IQMC en conjunto para obtener mayores resultados.

- f. Según el análisis de los resultados obtenidos se pudo determinar que el framework Ionic cumple de la mejor manera con las características de adecuación funcional, eficiencia de desempeño, usabilidad, fiabilidad y portabilidad, superando a jQuery Mobile en estas características del modelo de calidad ISO 25010. En las características de compatibilidad, seguridad y mantenibilidad ambos frameworks están empatados, lo cual indica que ambos cuentan y/o cumplen con estas características; cabe indicar que para estas características solo se aplicaron algunos atributos para la evaluación ya que por ser framework para el lado del cliente, solo dichos atributos se pueden aplicar a la evaluación.

4.2.Recomendaciones

Para construir un modelo de evaluación flexible, robusto y escalable se recomienda utilizar la norma ISO 25010 conjuntamente con el modelo IQMC para realizar la evaluación de frameworks u otro software, ya que estos en conjunto representan una amplia lista de factores de calidad esenciales en un producto software.

Se recomienda utilizar el modelo de evaluación que se ha desarrollado en este proyecto cuando se necesite seleccionar la mejor opción entre varios frameworks para el lado del cliente que se desarrollan utilizando tecnología web, ya que este modelo posee características concretas y fáciles de medir.

Según los resultados obtenidos en la evaluación se recomienda utilizar el framework Ionic sin importar el nivel de experiencia en el uso de HTML, CSS, JavaScript, Angular, TypeScript para el desarrollo de aplicaciones orientadas a dispositivos móviles, ya que posee una API extensa, fácil de aprender y utilizar, contando así con el apoyo de una gran comunidad, teniendo a disposición gran cantidad de tutoriales, libros y foros.

REFERENCIAS.


- Albors Aznar, L. (2014). Diseño E Implementación De Una Base De Datos Relacional Para La Gestión De Apuestas De Fútbol, 82.
- Almeida Acosta, J. M., & Romero Rivero, J. A. (2016). Desarrollo de una aplicación móvil sobre la plataforma Android para la liga de fútbol sala de la facultad de ciencias de la universidad central de Venezuela, 122.
- análisis - Wikcionario. (n.d.). Retrieved June 15, 2017, from <https://es.wiktionary.org/wiki/análisis>
- Bermeo Rodríguez, L. I. (2014). Análisis comparativo de frameworks javascript jquery y mootools, para la implementación de aplicaciones web en la empresa sofya. Aplicacion a un caso de estudio, 169.
- Cuenca Aznar, D. (2014). Diseño e implementación de una base de datos relacional para la gestión de apuesta de fútbol, 84. Retrieved from <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf>
- Estrada Salazar, K. L. (2012). DISEÑO Y DESARROLLO DE UN SISTEMA DENOMINADO OCAÑA GOL PARA REALIZAR LAS APUESTAS DEL CAMPEONATO LIGA POSTOBON DEL FUTBOL COLOMBIANO. 200.93.148.28, 67. Retrieved from <http://200.93.148.28/drupal/files/D1dCBLxWr0Gnvid.pdf>
- Hale, M. L., & Hanson, S. (2015). A Testbed and Process for Analyzing Attack Vectors and Vulnerabilities in Hybrid Mobile Apps Connected to Restful Web Services. *Proceedings - 2015 IEEE World Congress on Services, SERVICES 2015*, 181–188. <https://doi.org/10.1109/SERVICES.2015.35>
- Hibernate ORM. (n.d.). Retrieved June 15, 2017, from <http://hibernate.org/orm/>
- Ionic Framework - Concepts. (2017). Retrieved June 15, 2017, from <http://ionicframework.com/docs/intro/concepts/>
- Iskander Morine, R. J. (2013). *Estudio comparativo de alternativas y frameworks de programación , para el desarrollo de aplicaciones móviles en entorno Android* . Universidad Politécnica de Cataluña. Retrieved from http://upcommons.upc.edu/bitstream/handle/2099.1/18249/Proyecto_Final_de_Carrera_UPC_-_Ricardo_Iskandar.pdf?sequence=5
- ISO 25010. (n.d.). Retrieved June 15, 2017, from <http://iso25000.com/index.php/normas-iso-25000/iso-25010>

- jQuery Mobile - About. (2017). Retrieved June 15, 2017, from <http://jquerymobile.com/about/>
- Katzmaier, A., & Hanneghan, M. (2013). Design Pattern Evaluation of Mobile and Web Based Application Frameworks. *2013 Sixth International Conference on Developments in eSystems Engineering*, 157–162. <https://doi.org/10.1109/DeSE.2013.36>
- Lisandro Nahuel, D. (2017). Desarrollo de aplicaciones móviles multiplataforma. *Universidad Nacional de La Plata*. Retrieved from <http://repositorio.unapiquitos.edu.pe/handle/UNAP/4515?show=full>
- Marenkov, J., Robal, T., & Kalja, A. (2015). A Study on Effective Knowledge Reuse in Multi-Platform Web Applications User Interfaces, 1351–1361.
- NORMAS ISO 25000. (2017). Retrieved June 15, 2017, from <http://iso25000.com/index.php/normas-iso-25000>
- Open Source | Open Source Initiative. (n.d.). Retrieved June 15, 2017, from <https://opensource.org/docs/osd>
- QSOS - Wikipedia. (n.d.). Retrieved June 15, 2017, from <https://en.wikipedia.org/wiki/QSOS>
- Ramos, G., & Páez, J. (2011). Analisis del metodo para calificacion de software qsos para la seleccion de software aplicable a procesos educativos, 23.
- Roque Hernández, R. V., Negrete Hoz, E., & Salinas Escandón, J. M. (2013). Aprendiendo a Desarrollar Aplicaciones Para Android Con La Metodología Ágil Scrum: Un Caso De Estudio. *XVIII Congreso Internacional de Contaduría,administracion E Informatica- ANFECA*, 12. Retrieved from <http://congreso.investiga.fca.unam.mx/docs/xviii/docs/8.01.pdf>
- Salazar Cardona, J. A., Angarrita, D. A., & Montoya, J. D. (2016). Tendencias en desarrollo móvil bajo las tecnologías Android e IOS. *Revista de Investigacion de La Faculta de Ingenieria EAM*, 3, 28–35.
- Sanchez Acosta, C. (2015). Análisis Comparativo de Frameworks para el Desarrollo de Aplicaciones Web en Java, *1*(1), 13.
- Santos Hernández, W. D., & Serrano Parreño, J. A. (2017). Desarrollo de una API REST con sus aplicaciones web y movil para la venta de ropa online de la empresa Rossman, 119.
- Schwaber, K., & Sutherland, J. (2013). La Guía de Scrum. *Scrumguides.Org*, 1,

21. Retrieved from <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf>
- Spring Framework Reference Documentation. (n.d.). Retrieved June 15, 2017, from <http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/>
- Spring Security. (n.d.). Retrieved June 15, 2017, from <http://projects.spring.io/spring-security/>
- Spring Security OAuth. (n.d.). Retrieved June 15, 2017, from <http://projects.spring.io/spring-security-oauth/docs/Home.html>
- Toapanta Chancusi, K., Vergara Ordoñez, M., & Campaña Ortega, M. (2014). Método ágil Scrum, aplicado a la implantación de un sistema informático para el proceso de recolección masica de información con tecnología móvil, 1–12.
- Uva, M., Daniele, M., Zorzán, F., Frutos, M., & Arsaute, A. (2014). Propuesta para documentar trabajos finales utilizando metodologías ágiles. *IX Congreso Sobre ...*, 8. Retrieved from <http://sedici.unlp.edu.ar/handle/10915/38424>
- Moisés Rodríguez, Óscar Pedreira y Carlos M. Fernández. 2015. Certificación de la Mantenibilidad del Producto Software: Un Caso Práctico. *Revista Latinoamericana de Ingeniería de Software*, 3(3): 127-134, ISSN 2314-2642. Recuperado de <http://sistemas.unla.edu.ar/sistemas/redisla/ReLAIS/relais-v3-n3-127-134.pdf>
- Coral Calero, A. M. 2010. *Calidad del producto y proceso software*. Madrid: Ra-Ma

ANEXOS

Anexo 1: Resolución de aprobación del trabajo de investigación.



FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N° 1547-2015/FIAU-USS

Chiclayo, 03 de setiembre de 2015

VISTO:

El Dictamen 27 de agosto de 2015, presentado por el Jurado Evaluador del Tesis designado por Resolución N° 1532-2015/FIAU-USS de fecha 03 de setiembre de 2015; en el cual se establece la procedencia para la ejecución de la Tesis titulada: **"ANÁLISIS COMPARATIVO DE FRAMEWORKS OPEN SOURCE PARA EL DESARROLLO DE APLICACIONES MÓVILES ANDROID BASADAS EN TECNOLOGÍA WEB"** presentada por el(los) estudiante(s) **INOÑAN CORONADO, CRISTIAN WILFREDO** de la Escuela Académico Profesional de **INGENIERÍA DE SISTEMAS** y;

CONSIDERANDO:

Que, de conformidad con el Reglamento de Grados y Títulos de la USS en su artículo 5° que a la letra dice: *"Las comisiones permanentes de Grados y Títulos, de cada Escuela Académico Profesional, estarán conformadas por dos miembros (Director de Escuela y un docente de la especialidad según su modalidad de estudios) designados por el Decano de la Facultad. Se encargarán de la revisión y calificación de los expedientes de los egresados para recibir los grados académicos y títulos profesionales correspondiente"*;

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;


SE RESUELVE:

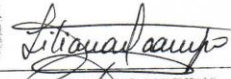
ARTÍCULO 1°: APROBAR, el Proyecto de Tesis denominado: **"ANÁLISIS COMPARATIVO DE FRAMEWORKS OPEN SOURCE PARA EL DESARROLLO DE APLICACIONES MÓVILES ANDROID BASADAS EN TECNOLOGÍA WEB"**, a cargo de(l) (los) estudiante(s) **INOÑAN CORONADO, CRISTIAN WILFREDO** de la Escuela Académico Profesional de **INGENIERÍA DE SISTEMAS**.

ARTÍCULO 2°: ESTABLECER, que la inscripción de la Tesis sea en la fecha de emitida la presente resolución.

ARTÍCULO 3°: DISPONER que el tema de investigación aprobado tendrá una vigencia máxima de 02 años, a partir de emitida la presente resolución.


REGÍSTRESE, COMUNIQUESE Y ARCHÍVESE.


Mg. AGUILAR CASTILLO SUSY DEL PILAR
DECANO(e) DE LA FACULTAD DE INGENIERÍA,
ARQUITECTURA Y URBANISMO


Ing. ROSA LILIANA OCAMPO MORENO
SEC. ACADEMICA DE LA FACULTAD DE
INGENIERÍA, ARQUITECTURA Y URBANISMO

cc.: Vicerrectorado Académico, CPGYT, Interesado(s), Archivo

CAMPUS UNIVERSITARIO
KM. 5 CARRETERA A PIMENTEL
TELEFONO (+51) (74) 481610 / FAX 203861
WWW.USS.EDU.PE
CHICLAYO - PERU

USS  UNIVERSIDAD
SEÑOR DE SIPÁN

Anexo 2: Hoja de evaluación del software. Método de evaluación QSOS

Durabilidad intrínseca		Puntuación		
		0	1	2
Madurez	Edad	Menos de tres meses	Entres tres meses y un año	Más de tres años
	Estabilidad	Software inestable con muchas versiones	Software estable con versiones pero viejas	Software estable. Las nuevas versiones tienen errores corregidos, pero en especial nuevas funcionalidades
	Historial de problemas	Muchos problemas que pueden ser prohibitivos	No se le conoce grandes problemas o crisis	Historial de buena administración de situaciones críticas
	Probabilidad de bifurcación	El software muy probablemente se expandirá en otro	El software proviene de otro, pero hay pocas posibilidades de que se expanda en otro	El software tiene muy pocas posibilidades que de bifurque en otro

Adopción	Popularidad	Muy pocos usuarios identificados	Uso detectable en internet	Numerosos usuarios y numerosas referencias
	Referencias	Ninguna	Pocas referencias, no es usos importantes	Implementado frecuentemente en aplicaciones críticas
	Calidad de la comunidad	No existe comunidad con notable actividad	Existe comunidad con notable actividad	Gran comunidad, mucha actividad en fotos, muchos contribuyentes
	Libros	No existen libre del software	Menos de 5 libros disponibles	Más de 5 libros y en muchos lenguajes
Estilo de liderazgo	Tamaño del equipo de desarrollo	1 o 2 individuos	Entre 2 y 5 personas independientes	Más de 5 personas
	Estilo de la administración del equipo	Completa dictadura	Despotismo culto	Consejo de desarrollo con un líder identificado
Actividad	Numero de desarrolladores	Menos de 3 desarrolladores no identificados claramente	Entre 4 y 7 desarrolladores, o mas no identificados	Más de 7 desarrolladores claramente identificados, equipo estable

	Actividad de errores	Poca actividad en foros o ninguna nota que arregle el error	Actividad detectada pero sin un proceso claramente expuesto	Alta actividad, basada en roles y asignación de tareas
	Actividad en funcionalidad	Ninguna o pocas funcionalidad	Evolución del producto introducido por el equipo de desarrollo pero sin un proceso formal.	Herramientas para administrar peticiones de características, alta interacción con el roadmap
	Actividad en liberaciones de nuevas versiones	Poca actividad en equipos de producción y desarrollo	Actividad en producción y desarrollo, frecuentemente lanzamientos menores (parches de errores)	Actividad importante el lanzamientos menores y lanzamientos mayores planeados en relación con el roadmap.

Fuente: (Ramos & Páez, 2011)

Solución industrializada		Puntuación		
		0	1	2
Servicios	Formación	No existen ofertas de formación	Existe oferta pero está restringida geográficamente y solo en un lenguaje	Existe y es provista por muchas empresas, en varios lenguajes y dividen el

				aprendizaje en módulos
	Soporte	No ofrecen soporte, excepto vía foros	Existe soporte pero es provisto únicamente por una empresa, sin el compromiso necesaria del servicio	Múltiples proveedores del servicio con gran compromiso
	Consultoría	No ofrece servicio de consultoría	Existe oferta pero está restringida geográficamente y solo en un lenguaje	Proveedores de consultoría de diversas empresas y con varios lenguajes
Documentación	Documentación	No existe documentación	Existe documentación pero esta desactualizada y en un solo lenguaje	La documentación en esta siempre a la fecha y posiblemente adaptada para distintos usuarios
Método de calidad	Aseguramiento de calidad	No existe aseguramiento de calidad	Existe un proceso de calidad pero	Proceso automatizado de calidad

			informal y sin herramienta	con publicación de resultados
	Herramientas	No existe herramienta para administración	Herramientas estándar	Herramientas especializadas

Fuente: (Ramos & Páez, 2011)

Disponibilidad de plataformas		Puntuación		
		0	1	2
Paquetes	Windows	No puede ser instalado en Windows	Existe un puerto para la instalación con problemas	Windows es soportado y existe paquete para la instalación
	Debían/Ubuntu	No esta empaquetado para Debian/Ubuntu	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución
	Red Hat/Fedora	No esta empaquetado para Red Hat/Fedora	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución
	Suse	No esta empaquetado para Suse	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución

	Mandriva	No esta empaquetado para Mandriva	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución
	Mac OS X	No esta empaquetado para Mac OX	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución
	Solaris	No esta empaquetado para Solaris	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución

Fuente: (Ramos & Páez, 2011)

Adaptabilidad técnica		Puntuación		
		0	1	2
Modularidad	Modularidad	Software estructurado sin módulos	Presencia de módulos de alto nivel permitiendo adaptación en el primer nivel	Concepción modular, permite una fácil adaptación del software seleccionado modulo o desarrollando nuevos
	Facilidad de extensión de código	No provee facilidad de extensión	Se puede extender el código con dificultad	Arquitectura plugin, diseñado para

				extensiones dinámicas
--	--	--	--	-----------------------

Fuente: (Ramos & Páez, 2011)

Estrategia		Puntuación		
		0	1	2
Licencia	Permisividad	Licencia muy estricta como I aGPL	Licencia moderadamente permisiva	Licencia muy permisiva como la Apache
	Protección contra extensiones del framework	Licencia muy permisiva como la Apache	Licencia moderadamente permisiva	Licencia muy estricta como I aGPL
Copyright	Dueños de Copyright	Derechos mantenidos por pocos individuos, los que fácilmente podrían cambiar la licencia	Derechos mantenidos por numerosos individuos, dueños del código en una forma homogénea	Derechos mantenidos por una entidad legal, de la cual la comunidad confía
Modificación del código fuente	Modificación del código fuente	No existe forma práctica de modificar el código	Provee herramientas para modificar el código pero no es la misma usada para el desarrollo	El proceso de modificación del código es bien definido, expuesto y respetado

RoadMap	RoadMap (mapa de versiones y liberaciones)	Sin publicaciones de roadmap	Existe un roadmap pero sin la debida planificación	Roadmap con las versiones, planificado y con detalles de medición
Patrocinador	Patrocinador	El software no tiene patrocinador, el equipo desarrollador no es pagado	El software solo tiene un patrocinador que puede determinar su estrategia	El software es patrocinado por la industria
Independencia a estrategia	Independencia a estrategia	No se encuentra una estrategia para el desarrollo	Visión estratégica compartida con otros productos open-source	Alta independencia del equipo de desarrollo una entidad legal sostiene los derechos

Fuente: (Ramos & Páez, 2011)

Industrialización del desarrollo		Puntuación		
		0	1	2
Herramientas	IDE	No tiene IDE para el desarrollo	Tiene un IDE definido para el desarrollo	Existe una gran cantidad de IDEs para el desarrollo

	Herramientas de construcción	No cuenta con herramientas que simplifiquen la construcción del aplicativo	Cuenta con pocas herramientas para la construcción del aplicativo	Cuenta con herramientas de construcción e integración con otros marcos de trabajo
	Herramientas de pruebas	No soporta la integración de herramientas de pruebas	Soporta integración de herramientas de pruebas	Soporta integración de herramientas de pruebas e incluye su propia herramienta de pruebas
	Herramienta grafica para el desarrollo de interfaces	Interfaces desarrolladas sin herramientas	Posee una herramienta para interfaces con pocos componentes	Herramienta de interfaces con gran número de componentes disponibles

Fuente: (Ramos & Páez, 2011)

Anexo 3. Detalle de las tablas de la base de datos

Tabla: Continentes


PK	Name	Data Type	NULL	Auto	Comment	FK
	contcod	Integer		✓	Código	
	contnombre	VarChar(40)	✓		Nombre	
	contestado	Boolean			Estado: Activo/Inactivo	

Tabla: Países



PK	Name	Data Type	NULL	Auto	Comment	FK
	paiscod	Integer		✓	Código	
	paisnombre	VarChar (50)	✓		Nombre	
	paisband16	Bytea	✓		Tamaño bandera 16px	
	paisband24	Bytea	✓		Tamaño bandera 24px	
	paisband32	Bytea	✓		Tamaño bandera 32px	
	paisband48	Bytea	✓		Tamaño bandera 48px	
	paisband64	Bytea	✓		Tamaño bandera 64px	
	paisband128	Bytea	✓		Tamaño bandera 128px	
	paisband256	Bytea	✓		Tamaño bandera 256px	
	contincod	Integer	✓		Código del continente al que pertenece	✓
	paisestado	Boolean			Estado: Activo/Inactivo	

Tabla: Usuarios

PK	Name	Data Type	NULL	Auto	Comment	FK
	usucod	Integer		✓	Código del usuario	
	usunom	VarChar(30)			Nombres del usuario	
	usuape	VarChar(40)			Apellidos del usuario	
	usufecnac	Date			Fecha de nacimiento del usuario	
	usugenero	VarChar(10)			Genero del usuario (Masculino/Femenino)	
	usualias	VarChar(30)			Alias del usuario	
	usupassw	VarChar(100)			Contraseña del usuario	
	usuemail	VarChar(40)			correo electrónico del usuario	
	ususaldo	Numeric(10, 2)	✓		Crédito actual del usuario	

paiscod	Integer	Código de 241aís del usuario	✓
usuestado	VarChar(10)	Estado actual del usuario (Active/Inactive/Deleted /Locked)	

Tabla: Roles

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	rolcod	Integer		✓	Codigo	
	rolnombre	VarChar(15)	✓		Nombre	

Tabla: usuarioroles

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑 1	usucod	Integer			Código de usuario	✓
🔑 2	rolcod	Integer			Código de Roles	✓

Tabla: TipoCompeticiones

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	compcod	Integer		✓	Código	
	compnom	VarChar(80)	✓		Nombre	
	paiscod	Integer	✓		Código de país	✓
	tccod	Integer			Código Tipo de Competición	✓
	compestado	Boolean			Estado: Activo/Inactivo	

Tabla: Competiciones

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	compcod	Integer		✓	Código	
	compnom	VarChar(80)	✓		Nombre	

paiscod	Integer	✓	Código de país	✓
tccod	Integer		Código Tipo de Competición	✓
compestado	Boolean		Estado: Activo/Inactivo	

Tabla: Temporadas

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	tempcod	Integer		✓	Código	
	tempnom	VarChar(15)	✓		Nombre	
	tempestado	Boolean			Estado (Activo/Inactivo)	

Tabla: Equipos

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	eqcod	Integer		✓	Código	
	eqnombre	VarChar(50)	✓		Nombre	
	eqlogo16	Bytea	✓		Logo 16px	
	eqlogo24	Bytea	✓		Logo 24px	
	eqlogo32	Bytea	✓		Logo 32px	
	eqlogo48	Bytea	✓		Logo 48px	
	eqlogo64	Bytea	✓		Logo 64px	
	eqlogo128	Bytea	✓		Logo 128px	
	eqlogo256	Bytea	✓		Logo 256px	
	equestado	Boolean			Estado (Activo/Inactivo)	

Tabla: ListaEquipos

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	lecod	Integer		✓	Código	
	eqcod	Integer	✓		Código de Equipo	✓
	compcod	Integer	✓		Código de Competición	✓
	tempcod	Integer	✓		Código de Temporada	✓

paiscod	Integer	✓	Código de País	✓
leestado	Boolean		Estado (Activo/Inactivo)	

Tabla: Jornadas

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	jorcod	Integer		✓	Códigodigo	
	jordesc	VarChar(10)	✓		Descripción	
	jorestado	Boolean			Estado (Activo/Inactivo)	

Tabla: PagosPartido

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	pagpcod	Integer		✓	Código	
	partcod	Integer			Código de Partido	✓
	pagptlocal	Numeric (5, 2)	✓		Pago primer tiempo local	
	pagptempate	Numeric (5, 2)	✓		pago primer tiempo empate	
	pagptvisita	Numeric (5, 2)	✓		Pago primer tiempo visita	
	pagfinlocal	Numeric (5, 2)	✓		Pago final local	
	pagfinempate	Numeric (5, 2)	✓		Pago final empate	
	pagfinvisita	Numeric (5, 2)	✓		Pago final visita	
	pagresptlocal	Numeric (5, 2)	✓		Pago resultado primer tiempo local	
	pagresptempate	Numeric (5, 2)	✓		Pago resultado primer tiempo empate	
	pagresptvisita	Numeric (5, 2)	✓		Pago resultado primer tiempo visita	

pagresfinlocal	Numeric (5, 2)	✓	Pago resultado final tiempo local
pagresfinempate	Numeric (5, 2)	✓	Pago resultado final tiempo empate
pagresfinvisita	Numeric (5, 2)	✓	Pago resultado final tiempo visita

Tabla: Resultados:

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	rescod	Integer		✓	Código	
	partcod	Integer			Código de Partido	✓
	reseqlocal1t	SmallInt	✓		Resultado equipo local 1 tiempo	
	reseqlocal2t	SmallInt	✓		Resultado equipo local 2 tiempo	
	reseqvisita1t	SmallInt	✓		Resultado equipo visita 1 tiempo	
	reseqvisita2t	SmallInt	✓		Resultado equipo visita 2 tiempo	

Tabla: Partidos

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	partcod	Integer		✓	Código	
	eqlocal	Integer			Codigo equipo local	✓
	eqvisita	Integer			Codigo equipo visita	✓
	partfecha	Date			Fecha del partido	
	jorcod	Integer	✓		Código de la jornada	✓
	partfinalizado	Boolean			Partido finalizado (Si/No)	

partestado	Boolean		Estado (Activo/Inactivo)
parthora	Time	✓	Hora del partido

Tabla: TipoApuesta

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	tacod	Integer		✓	Código	
	tadescripcion	VarChar(30)			Descripción	

Tabla: Apuestas

PK	Name	Data Type	NULL	Auto	Comment	FK
🔑	apcod	BigInt		✓	Código	
	partcod	Integer			Código del partido	✓
	tacod	Integer			Código tipo de apuesta	✓
	usucod	Integer			Código de usuario	✓
	fecha	TimeStampWith TimeZone	✓		Fecha de la apuesta	
	goleseqlocal1t	SmallInt	✓		Goles equipo local 1 tiempo	
	goleseqlocal2t	SmallInt	✓		Goles equipo local 2 tiempo	
	goleseqvisita1t	SmallInt	✓		Goles equipo visita 1 tiempo	
	goleseqvisita2t	SmallInt	✓		Goles equipo visita 2 tiempo	
	lev	VarChar(10)	✓		Local-Empate-Visita 1 tiempo	
	montoapuesta	Numeric(10, 2)			Monto apostado	
	totalapuesta	Numeric(10, 2)			Total de apuesta	
	levfinal	VarChar(10)	✓		Local-Empate-Visita 2 tiempo	

Tabla: ClientDetails:


PK	Name	Data Type	NULL	Auto	FK
	<u>appid</u>	VarChar(256)			
	<u>resourceids</u>	VarChar(256)	✓		
	<u>appsecret</u>	VarChar(256)	✓		
	<u>scope</u>	VarChar(256)	✓		
	<u>granttypes</u>	VarChar(256)	✓		
	<u>redirecturl</u>	VarChar(256)	✓		
	<u>authorities</u>	VarChar(256)	✓		
	<u>access_token_validity</u>	Integer	✓		
	<u>refresh_token_validity</u>	Integer	✓		
	<u>additionalinformation</u>	VarChar(4096)	✓		
	<u>autoapprovescopes</u>	VarChar(256)	✓		

Tabla: Oauth Client Details


PK	Name	Data Type	NULL	Auto	FK
	client_id	VarChar(256)			
	resource_ids	VarChar(256)	✓		
	client_secret	VarChar(256)	✓		
	scope	VarChar(256)	✓		
	authorized_grant_types	VarChar(256)	✓		
	web_server_redirect_uri	VarChar(256)	✓		
	authorities	VarChar(256)	✓		
	access_token_validity	Integer	✓		
	refresh_token_validity	Integer	✓		
	additional_information	VarChar(4096)	✓		
	autoapprove	VarChar(256)	✓		

Tabla: Oauth Refresh Token

PK	Name	Data Type	NULL	Auto	FK
	token_id	VarChar(256)	✓		
	token	Bytea	✓		
	authentication	Bytea	✓		

Tabla: Oauth Access Token


PK	Name	Data Type	NULL	Auto	FK
	authentication_id	VarChar(256)			
	token_id	VarChar(256)	✓		
	token	Bytea	✓		
	user_name	VarChar(256)	✓		
	client_id	VarChar(256)	✓		
	authentication	Bytea	✓		
	refresh_token	VarChar(256)	✓		

Tabla: Oauth Approvals

PK	Name	Data Type	NULL	Auto	FK
	userid	VarChar(256)	✓		
	clientid	VarChar(256)	✓		
	scope	VarChar(256)	✓		
	status	VarChar(10)	✓		
	expiresat	TimeStamp	✓		
	lastmodifiedat	TimeStamp	✓		

Tabla Oauth Client Token


PK	Name	Data Type	NULL	Auto	FK
	authentication_id	VarChar(256)			
	token_id	VarChar(256)	✓		
	token	Bytea	✓		
	user_name	VarChar(256)	✓		
	client_id	VarChar(256)	✓		

Tabla: Oauth Code

PK	Name	Data Type	NULL	Auto	FK
	code	VarChar(256)	✓		
	authentication	Bytea	✓		

Estructura de Foreign Keys

Name	Parent Table	Child Table	Parent Columns	Child Columns
apuestapartido_fkey	partidos	apuestas	partcod	partcod
apuestatipoap_fkey	tipoapuesta	apuestas	tacod	tacod
apuestausuario_fkey	usuarios	apuestas	usucod	usucod
competicionpais_key	países	competiciones	paiscod	paiscod
competiciontipocomp_fkey	tipocompeticion	competiciones	tccod	tccod
lcompeticiones_fkey	competiciones	listaequipos	compcod	compcod
leequipos_fkey	equipos	listaequipos	eqcod	eqcod
lepaises_fkey	países	listaequipos	paiscod	paiscod
letemporadas_fkey	temporadas	listaequipos	tempcod	tempcod
pagopartpartido_fkey	partidos	pagospartido	partcod	partcod
continente_fkey	continentes	países	contcod	contincod
parteqlocal_fkey	listaequipos	partidos	lecod	eqlocal
parteqvisita_fkey	listaequipos	partidos	lecod	eqvisita
partjornada_fkey	jornadas	partidos	jorcod	jorcod
resultadopartido_fkey	partidos	resultados	partcod	partcod
rolur_fkey	roles	usuarioroles	rolcod	rolcod
usuariour_fkey	usuarios	usuarioroles	usucod	usucod

Fuente: Elaboración propia

Anexo 4. Wireframes de la aplicación para el cliente

Pantalla de Registro

GoWin
Crear Cuenta

Pais: Seleccione Pais

Nombre: Nombres

Apellidos: Apellidos

Fecha Nac.: 4/22/2012

Genero: Masculino

Alias: Alias

Password: Password

E-mail: Email Address

Monto:

Registrar

Cancelar

Pantalla de Login

GoWin
Login

Usuario: Ingrese Usuario

Password: Ingrese Password

Entrar

Crear Cuenta

Pantalla Menú Principal

GoWin
Menú Principal

Partidos

Apostar

Salir

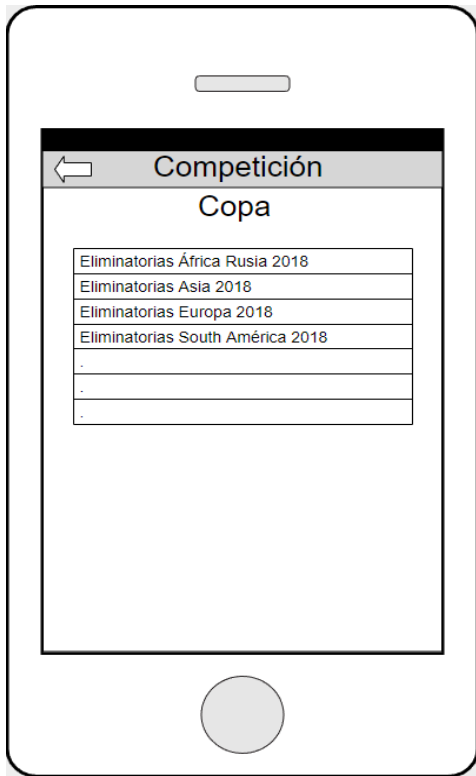
Pantalla Lista tipo de competición

GoWin
Tipo Competición

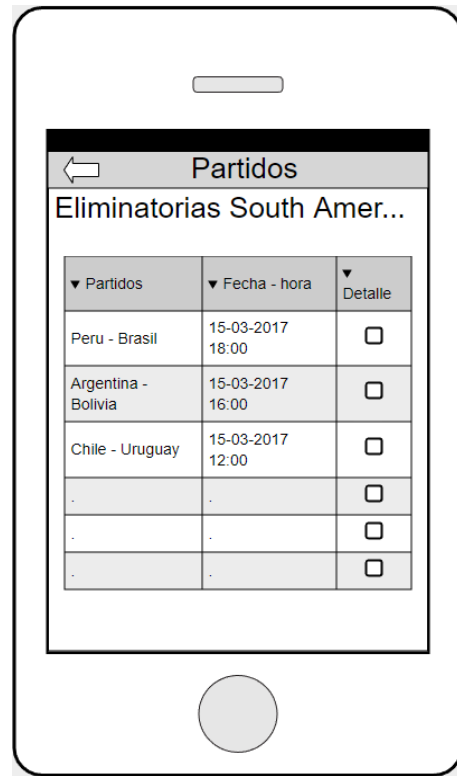
Copa

Liga

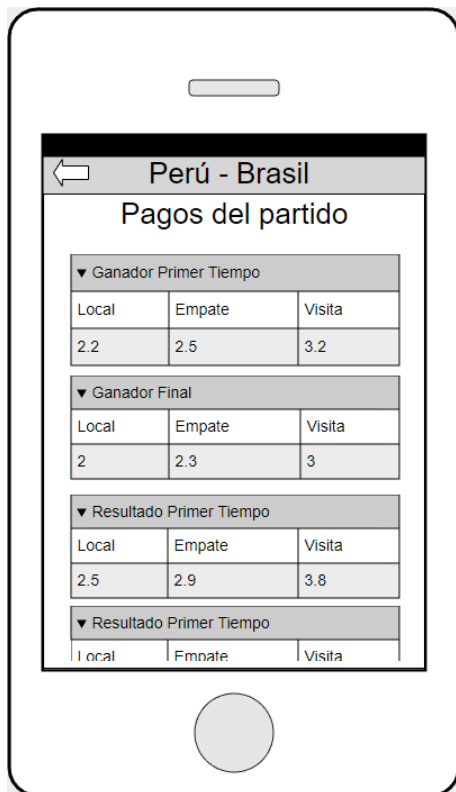
Pantalla lista de competiciones



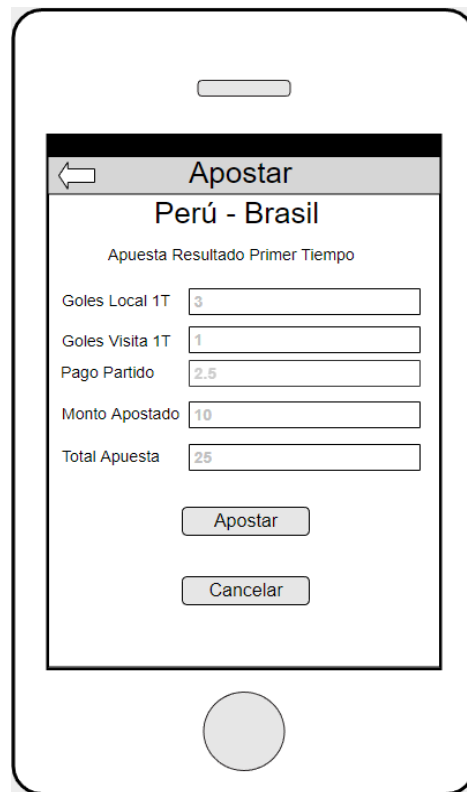
Pantalla lista de partidos



Pantalla de Pagos Partido



Pantalla Realizar Apuesta



Fuente: Elaboración propia

Anexo 5. Pruebas de Tiempo de carga Cliente - Servidor en entorno local

Pruebas de Tiempo de carga de JQuery Mobile

The screenshot displays a mobile application interface on the left and a network performance monitoring tool on the right. The application shows a list of football matches for the 'Eliminatorias South America Rusia 2018' tournament, specifically 'Jornada 11'. The matches listed are:

Equipos	Fecha - Hora	Detalle
Colombia - Chile	2016-11-10 - 15:30:00	[Detail Icon]
Uruguay - Ecuador	2016-11-10 - 18:00:00	[Detail Icon]
Paraguay - Peru	2016-11-10 - 18:30:00	[Detail Icon]
Venezuela - Bolivia	2016-11-10 - 18:30:00	[Detail Icon]
Brazil - Argentina	2016-11-10 - 18:35:00	[Detail Icon]

The network tool on the right shows a list of network requests. The top requests are XHR requests for 'jquery-2.1.4.min.js', with various sizes and times. Below these are requests for 'ajax-loader.gif' and multiple 'chromecastcheck.js' requests.

Name	Status	Type	Initiator	Size	Time	Waterfall
1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	115 ms	
1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	116 ms	
1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	115 ms	
1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	113 ms	
12?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	113 ms	
1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	13.0 KB	167 ms	
1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	13.0 KB	135 ms	
1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	13.0 KB	129 ms	
2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	134 ms	
2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	120 ms	
2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	119 ms	
2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	139 ms	
2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	148 ms	
2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	113 ms	
2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	113 ms	
51?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	121 ms	
51?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	129 ms	
52?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	126 ms	
52?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	170 ms	
53?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	121 ms	
53?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	136 ms	
54?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	128 ms	
54?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	119 ms	
54?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	131 ms	
55?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	123 ms	
55?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	123 ms	
ajax-loader.gif	200	gif	jquery-2.1.4.min.js:4	6.2 KB	27 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	2 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	2 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	2 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	4 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	2 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	2 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	2 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	4 ms	
chromecastcheck.js	200	script	content.js:136	(from ...)	2 ms	

Resultados de tiempo de carga de JQuery Mobile, y tiempo promedio de carga total de la aplicación.

N°	name	Status	Type	Initiator	Size	time
1	1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	115 ms
2	1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	116 ms
3	1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	115 ms
4	1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	113 ms
5	1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	686 B	113 ms
6	1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	13.0 KB	167 ms
7	1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	13.0 KB	135 ms
8	1?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	13.0 KB	129 ms
9	2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	134 ms
10	2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	120 ms
11	2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	119 ms
12	2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	139 ms
13	2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	148 ms
14	2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	113 ms
15	2?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.4 KB	113 ms
16	51?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	121 ms
17	51?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	129 ms
18	52?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	126 ms
19	52?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	170 ms

20	53?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	121 ms
21	53?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	136 ms
22	54?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	128 ms
23	54?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	119 ms
24	54?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	131 ms
25	55?access_token=fd7ab240-c42d-4832-9c57-ab657f329141	200	xhr	jquery-2.1.4.min.js:4	1.0 KB	123 ms
						Promedio 128 ms

Fuente: Elaboración propia

Pruebas de Tiempo de carga de Ionic

The screenshot shows a mobile application interface on the left and a network performance tool on the right.

Partidos - Eliminotorias South ...

Jornada 11

Equipos	Fecha-Hora	Detalle
Colombia - Chile	2016-11-10 15:30:00	[Info Icon]
Uruguay - Ecuador	2016-11-10 18:00:00	[Info Icon]
Paraguay - Peru	2016-11-10 18:30:00	[Info Icon]
Venezuela - Bolivia	2016-11-10 18:30:00	[Info Icon]
Brazil - Argentina	2016-11-10 18:35:00	[Info Icon]

The Network tab shows the following requests:

Name	Status	Type	Initiator	Size	Time	Waterfall
token?grant_type=password&client_id=Golwinionic&client_secret=golwinioni...	200	xhr	polyfills.js:3	576 B	174 ms	
token?grant_type=password&client_id=Golwinionic&client_secret=golwinioni...	200	xhr	Other	755 B	370 ms	
josecor?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	553 B	105 ms	
josecor?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	Other	916 B	112 ms	
tipocompeticiones?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	666 B	163 ms	
2?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.4 KB	124 ms	
1?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	13.0 KB	128 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
data:image/png;base...	200	png	Other	(from ...)	0 ms	
52?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	116 ms	
51?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	127 ms	
52?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	136 ms	
53?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	114 ms	
54?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	121 ms	
55?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	115 ms	
1?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	686 B	135 ms	
10?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	204	xhr	polyfills.js:3	485 B	126 ms	
2?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.4 KB	121 ms	
1?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	13.0 KB	130 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	
data:image/png;base...	200	png	platform-browser.es5.js:2914	(from ...)	0 ms	

Resultados de tiempo de carga de Ionic, y tiempo promedio de carga total de la aplicación.

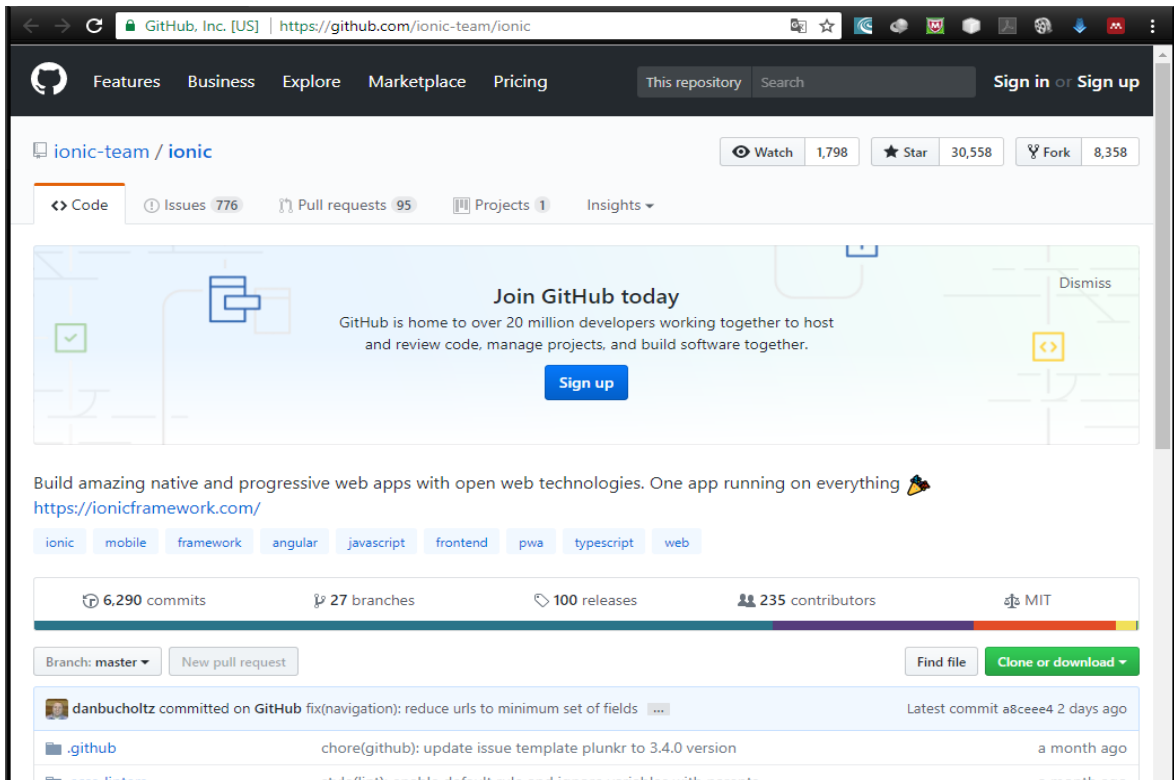
N°	name	Status	Type	Initiator	Size	time
1	2?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.4 KB	124 ms
2	1?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	13.0 KB	128 ms
3	52?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	116 ms
4	51?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	127 ms
5	52?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	136 ms
6	53?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	114 ms
7	54?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	121 ms
8	55?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	115 ms
9	1?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	686 B	135 ms
10	10?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	204	xhr	polyfills.js:3	485 B	126 ms
11	2?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.4 KB	121 ms
12	1?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	13.0 KB	130 ms
13	51?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	131 ms
14	52?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	124 ms
15	53?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	129 ms
16	55?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	124 ms
17	1?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	13.0 KB	132 ms
18	54?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	115 ms
19	53?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	134 ms

20	55?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	123 ms
21	51?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	125 ms
22	52?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.0 KB	130 ms
23	2?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.4 KB	119 ms
24	2?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.4 KB	124 ms
25	2?access_token=2a53733f-ab9e-40e0-9f8f-6219fdb7a239	200	xhr	polyfills.js:3	1.4 KB	121 ms
Promedio						125 ms

Fuente: Elaboración propia

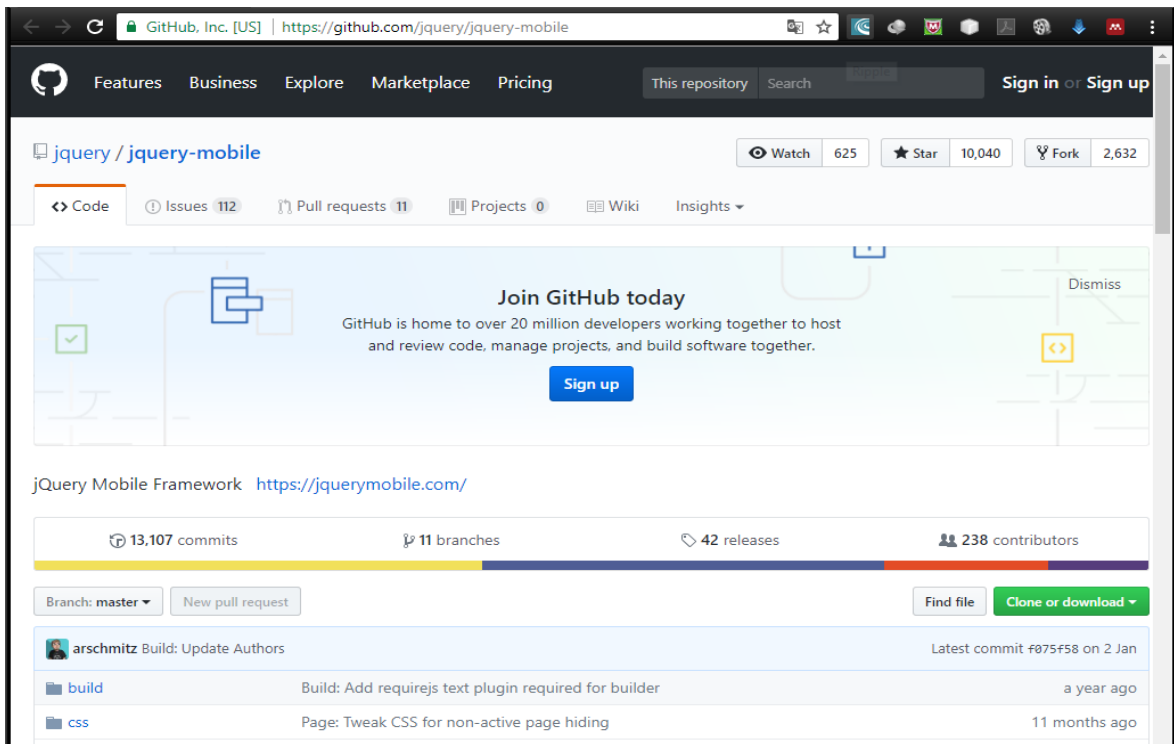
Anexo 6. Popularidad de los frameworks en GitHub

Cantidad de estrellas del framework Ionic y repositorio oficial en GitHub



The screenshot shows the GitHub repository page for the Ionic framework. The repository is owned by the 'ionic-team' and is named 'ionic'. It has 1,798 watchers, 30,558 stars, and 8,358 forks. The repository is currently on the 'master' branch. The page features a 'Join GitHub today' banner with a 'Sign up' button. Below the banner, there is a description of the framework: 'Build amazing native and progressive web apps with open web technologies. One app running on everything'. The repository statistics show 6,290 commits, 27 branches, 100 releases, and 235 contributors. The license is MIT. The commit history shows the latest commit by 'danbucholtz' on 2 days ago, with a commit message 'fix(navigation): reduce urls to minimum set of fields'.

Cantidad de estrellas del framework jQuery Mobile y repositorio oficial en GitHub



The screenshot shows the GitHub repository page for the jQuery Mobile framework. The repository is owned by 'jquery' and is named 'jquery-mobile'. It has 625 watchers, 10,040 stars, and 2,632 forks. The repository is currently on the 'master' branch. The page features a 'Join GitHub today' banner with a 'Sign up' button. Below the banner, there is a description of the framework: 'jQuery Mobile Framework'. The repository statistics show 13,107 commits, 11 branches, 42 releases, and 238 contributors. The license is MIT. The commit history shows the latest commit by 'arschmitz' on 2 Jan, with a commit message 'Build: Update Authors'.

Fuente: Elaboración propia