UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TRABAJO DE INVESTIGACIÓN

REVISIÓN DE CRITERIOS DE CALIDAD PARA MODELAR UNA ARQUITECTURA TECNOLÓGICA DE LA PLATAFORMA DE INTEROPERABILIDAD DEL ESTADO PERUANO BASADO EN SERVICIOS WEB REST

Para Optar el Grado de Bachiller en Ingeniería de Sistemas

Autor:

Maza Llacsahuanga Dilberto

Asesor:

Mg. Mejia Cabrera Heber Ivan

Línea de Investigación:

Tecnologías de la Información – Arquitectura Tecnológica

Paita – Perú

2020

TITULO

REVISIÓN DE CRITERIOS DE CALIDAD PARA MODELAR UNA ARQUITECTURA TECNOLÓGICA DE LA PLATAFORMA DE INTEROPERABILIDAD DEL ESTADO PERUANO BASADO EN SERVICIOS WEB REST

AUTOR

Maza Llacsahuanga Dilberto

Resumen

El presente trabajo de investigación propone una optimización de la Plataforma de Interoperabilidad del Estado (PIDE) mediante la propuesta de una arquitectura tecnológica basada en el intercambio de información a través de servicios web REST. Para cumplir con el objetivo de este trabajo, se analiza los servicios web de Sistema de seguimiento de inversiones (SSI) y Consulta pública de inversiones, ambos administrados por el Ministerio de Economía y Finanzas (MEF). Con base en la revisión de criterios de calidad de estos servicios, se propone una arquitectura tecnológica basada en servicios web REST. La revisión de la arquitectura propuesta evidencia el margen de mejora para los servicios web de la PIDE, especialmente en términos de desempeño.

Palabras Clave: REST, PIDE, MEAN Stack, Calidad de software, Gobierno electrónico.

Abstract

This research paper proposes an optimization of the State Interoperability Platform (PIDE) through the proposal of a technological architecture based on the exchange of information through REST web services. To fulfill the objective of this work, we analyze the web services of Investment Monitoring System (SSI) and Public Investment Consultation, both managed by the Ministry of Economy and Finance (MEF). Based on the review of quality criteria for these services, a technological architecture based on REST web services is proposed. The review of the proposed architecture shows the room for improvement for the PIDE web services, especially in terms of performance.

Keywords: REST, PIDE, MEAN Stack, Software quality, E-government.

ÍNDICE

CAPÍTULO I
INTRODUCCIÓN

1.1 Realidad problemática

La modernización de los servicios web del Estado es una agenda pendiente de suma relevancia. La implementación poco efectiva e inarticulada de políticas orientadas a este fin ha ocasionado un descenso importante en la posición del país frente a sus pares latinoamericanos en materia de Gobierno Electrónico. Por ejemplo, en el ranking del Índice de Servicio Online de Gobierno, el cual es diseñado por las Naciones Unidas e indica si los servicios web del Estado permiten que el ciudadano promedio pueda obtener información o encontrar soluciones de manera rápida o intuitiva, el Perú se encuentra solo por encima de Bolivia y Venezuela a nivel sudamericano.

Uno de los problemas en los servicios web del Estado se constituye en no contar con una arquitectura ágil y segura para garantizar el intercambio de información a través de una amplia variedad de recursos y sistemas, esto es, garantizar la interoperabilidad del mismo. Además, los servicios web del Estado presentan limitaciones en términos de desempeño, accesibilidad, diseño de arquitectura y seguridad.

Por tanto, resulta necesario implementar arquitecturas para el desarrollo de aplicaciones que permitan hacer más eficientes los procesos de las instituciones públicas y con ello mejorar los servicios para los ciudadanos. Las arquitecturas Representational State Transfer (REST) son una de las más usadas en la actualidad para conseguir dicho objetivo. Ejemplos de ello son las Application Programming Interface (API) de Twitter, Google, Facebook entre otras.

Por tal motivo, se propone el desarrollo de una arquitectura REST para mejorar la interoperabilidad de los servicios web del Estado. Los criterios de calidad utilizados para este desarrollo se basan en una selección de los criterios de la norma ISO 25010. Sobre las tecnologías de la arquitectura, se plantea el uso del modelo Vista Controlador (MVC) empleando tecnologías de JavaScript. En específico, se propone el modelo MEAN Stack, que se compone de las tecnologías MongoDB, ExpressJS, AngularJS y NodeJS.

1.2 Antecedentes de Estudio

La literatura científica internacional sobre el gobierno electrónico señala una serie de determinantes del desempeño de los servicios web estatales. Uno de los determinantes

más importantes es la interoperabilidad. Por tal motivo, Almutairi y Khan (2016) proponen una arquitectura tecnológica que optimice la interoperabilidad de un servicio web de datos públicos en Arabia Saudita. Ellos observaron que la redundancia de información se estaba convirtiendo en un tema crítico para todas las transacciones gubernamentales, pues aquélla sobrecargaba las bases de datos. De este modo, uno de los desafíos clave era la recopilar los datos RAW de todos los almacenes de datos, clasificarlos y luego aplicar la optimización en estas fuentes de datos. En la misma línea, Castrillón, González y López (2012) evalúan un modelo arquitectónico para garantizar la interoperabilidad entre las instituciones prestadoras de salud en Colombia. Los autores proponen el uso de SOA como referencia para definir la arquitectura de software y el estándar HL7 para el intercambio de mensajes y documentos clínicos. Los autores encuentran que el modelo propuesto permite diseñar arquitecturas interoperables y que es suficientemente abierto, flexible y escalable para construir arquitecturas para el sector salud.

Por otro lado, Sun, Ku y Shih (2015) proponen que la interoperabilidad se logra no solamente con el diseño de una arquitectura tecnológica, sino que se necesita la coordinación institucional de los agentes involucrados. En específico, ellos analizan la implementación del E-Government 2.0. Este tipo de gobierno eletrónico comprende una serie de servicios web orientados al usuario que interoperan y se brindan a través de tecnologías Web 2.0, como RSS, blogs, redes sociales, etc., y que son accesibles desde varios canales. Sin embargo, los autores indican que construir E-Government 2.0 es difícil porque la transición de E-Government a E-Government 2.0 debería ser una agenda de integración organizacional, no solo tecnológica. Ellos evalúan el grado de avance respecto al marco de implementación propuesto de tres países: Corea del Sur, Antigua y Barbuda y Ecuador. Para ello, verifican el nivel de desarrollo de cuatro indicadores: Integración de procesos, integración de recursos, integración back-office e integración front-office Una de las observaciones clave sobre el progreso en la innovación de servicios es que las agencias necesitan un marco institucional para ayudarlas a avanzar.

Además, el seguimiento de estándares en la implementación de una arquitectura tecnológica de interoperabilidad debe tomar en cuenta problemas potenciales tales como la escasez de personal, alta demanda, plazos ajustados y la falta de métodos de desarrollo específicos (Oliveira & Eler, 2017).

Por otra parte, la cuestión más importante en la implementación de un gobierno electrónico exitoso es la aceptación y el uso de los ciudadanos. Los ciudadanos deben recibir capacitación y educación para utilizar los servicios de portal electrónico disponibles en la estructura correspondiente (Sarrayrih & Sriram, 2015).

Asimismo, considerar el enfoque de todo el gobierno nos permite tener una comprensión más profunda de cómo los servicios electrónicos del gobierno pueden ser

interoperables entre sí y lograr las tres dimensiones de interoperabilidad: 1) proceso de negocios o la interoperabilidad organizacional, 2) información o interoperabilidad semántica, 3) interoperabilidad técnica (Reyes & Tangkeko, 2017).

Por otro lado, la literatura internacional también comprende estudios empíricos donde se implementa una arquitectura tecnológica del tipo REST para mejorar los servicios web. Asimismo, existe un creciente número de estudios que implementan la herramienta llamada MEAN STACK para diseñar dicha arquitectura. Esta herramienta emplea las tecnologías de MongoDB, Express, Angular y NodeJS.

Poulter, Johnston y Cox (2015), por ejemplo, demuestran los beneficios de la herramienta MEAN STACK y su idoneidad para ser utilizada en la implementación de servicios web REST para dispositivos de Internet de las cosas (IoT). Khue, Binh, Chang, Kim y Chung (2017) encuentran que la herramienta MEAN STACK aplicada a un sistema de señalización digital muestra un mejor rendimiento que un conocido sistema del mismo tipo, pero basado en la herramienta LAMP STACK. Sin embargo, los autores también observan que se necesita mejoras más extensas para garantizar la confiabilidad de la arquitectura diseñada por la herramienta MEAN SATCK.

En la literatura nacional, se puede mencionar el estudio de Orozco (2019), el cual propone una arquitectura basada en REST implementada a la gestión de procesos en entidades públicas. Como caso de estudio particular, el autor evalúa la División de Programación Operativa de la Superintendencia Nacional de Aduanas y de Administración Tributaria (SUNAT). También, se encuentra el estudio de Atencio y Mamani (2017), quienes desarrollan un servicio web basado en API REST para mejorar la interoperabilidad de las aplicaciones web de la Municipalidad Provincial de Lampa en el departamento de Puno. Ellos emplearon el lenguaje de Modelamiento Unificado UML y la metodología SCRUM para documentar los procesos de desarrollo. Por otro lado, el estudio de Burgos (2017) desarrolla un análisis comparativo entre la arquitectura REST y la arquitectura SOAP. Para ello, diseña arquitecturas de ambos tipos para un sistema de planificación de recursos empresariales llamado AdrisERP. Asimismo, Huanca (2017) propone una arquitectura de microservicios para el despliegue de servicios web. Con este fin, el autor analiza las necesidades en el desarrollo de software y en la implementación de servicios web, así como las nuevas tendencias en estilos arquitectónicos para servicios web.

En la literatura local, no se encontró estudios relacionados al diseño y/o implementación de arquitecturas tecnológicas para servicios web del Estado.

1.3 Teorías relacionadas al tema

1.3.1 Calidad

Según Deming (1989), la calidad es “un grado predecible de uniformidad y fiabilidad a bajo coste, adecuado a las necesidades del mercado”.

Para Juran y Gryna (1993), la calidad se define como: “La adecuación al uso, esta definición implica una adecuación del diseño del producto o servicio (calidad de diseño) y la medición del grado en que el producto es conforme con dicho diseño (calidad de fabricación o conformidad”.

Según Crosby (1987), define calidad como “La conformidad con las especificaciones o cumplimiento de los requisitos y entiende que la principal motivación de la empresa es el alcanzar la cifra de cero defectos. Su lema es "Hacerlo bien a la primera vez y conseguir cero defectos".

1.3.1.1 ISO/IEC 25010

El modelo de calidad determina qué criterios de calidad se consideran al revisar un producto de software. El modelo de calidad definido en ISO / IEC 25010 comprende ocho características que se explican a continuación.

Adecuación funcional

Este criterio considera la medida en que un producto de software proporciona funciones que satisfacen las necesidades establecidas e implícitas cuando se usa en condiciones específicas. Este criterio se manifiesta en los siguientes parámetros:

Completitud funcional. El conjunto de funciones cubre todas las tareas especificadas y los objetivos del usuario.
Corrección funcional. El producto de software proporciona los resultados correctos con el nivel de precisión necesario.
Pertinencia funcional Las funciones facilitan el cumplimiento de tareas y objetivos determinados.

Eficiencia de desempeño

Este criterio considera el rendimiento en relación con la cantidad de recursos utilizados en las condiciones establecidas. Se manifiesta en los siguientes parámetros:

Comportamiento temporal. Los tiempos de respuesta y procesamiento y las tasas de desempeño de un producto de software, al realizar sus funciones, se adaptan a los requerimientos.
Utilización de recursos. Las cantidades y tipos de recursos utilizados por un producto de software, al realizar sus funciones, se adaptan a los requerimientos.
Capacidad. Los límites máximos de un producto o parámetro del sistema se adaptan a los requerimientos.

Compatibilidad

Nivel en que un producto de software puede intercambiar información con otros, y/o realizar sus funciones, mientras comparten el mismo entorno. Este criterio se manifiesta en los siguientes parámetros:

Coexistencia. El producto puede realizar sus funciones de manera eficiente mientras comparte un entorno y recursos comunes con otros productos, sin un impacto perjudicial en ningún otro producto.
Interoperabilidad. Dos o más sistemas, productos o componentes intercambian información y la utilizan.

Usabilidad

Cuando un producto de software puede ser utilizado por usuarios determinados para lograr objetivos determinados con efectividad, eficiencia y satisfacción en un contexto de uso determinado. Este criterio se manifiesta en los siguientes parámetros:

Inteligibilidad. Los usuarios pueden reconocer si un producto de software es apropiado para sus necesidades.
Aprendizaje. Cuando los usuarios pueden aprender a usar el producto de software con efectividad, eficiencia, libertad de riesgos y satisfacción en un contexto de uso determinado.
Operabilidad. El producto de software tiene atributos que facilitan su operación y control.
Protección frente a errores del usuario. El sistema protege a los usuarios contra errores.
Estética. La interfaz de usuario permite una interacción agradable y satisfactoria.
Accesibilidad. El producto de software puede ser utilizado por personas con la más amplia gama de características y capacidades.

Fiabilidad

Nivel en que un sistema, producto o componente realiza funciones específicas en condiciones particulares durante un período de tiempo determinado. Este criterio se manifiesta en los siguientes parámetros:

Madurez. El producto de software cumple con las necesidades de fiabilidad en una situación corriente.
Disponibilidad. El producto de software está operativo y accesible cuando se requiere para su uso.
Tolerancia a fallos. El producto de software funciona según lo previsto a pesar de la presencia de fallos.
Capacidad de recuperación. En caso de interrupción o falla, un producto de software puede recuperar los datos directamente afectados y restablecer el estado deseado del sistema.

Seguridad

El producto de software protege la información y los datos para que las personas u otros productos o sistemas tengan el nivel de acceso a los datos apropiados a sus tipos y niveles de autorización. Este criterio se manifiesta en los siguientes parámetros:

Confidencialidad. El producto de software garantiza que los datos estén accesibles solo para aquellos autorizados a tener acceso.
Integridad. El producto de software impide el acceso no autorizado.
No repudio. Se puede demostrar las acciones que han tenido lugar, de modo que no puedan ser repudiados más tarde.
Responsabilidad. Las acciones de una entidad se pueden rastrear de forma exclusiva a la misma.
Autenticidad. Se puede demostrar que la identidad de un sujeto o recurso es la reclamada.

Mantenibilidad

Este criterio considera el nivel de efectividad y eficiencia con que un producto de software puede modificarse para mejorarlo, corregirlo o adaptarlo a los cambios en el entorno y en los requisitos. Este criterio se manifiesta en los siguientes parámetros:

Modularidad. El sistema o programa de computadora se compone de componentes discretos de manera que un cambio en un componente tiene un impacto mínimo en otros componentes.
Reusabilidad Se puede usar un activo en más de un sistema, o en la construcción de otros activos.
Analizabilidad. Nivel de efectividad y eficiencia con el que es posible evaluar el impacto en un producto de software de un cambio previsto en una o más de sus partes, o diagnosticar un producto por deficiencias o causas de fallas, o identificar partes para modificar.
Capacidad de ser modificado El producto de software puede modificarse de manera efectiva y eficiente sin introducir defectos o degradar la calidad del producto existente.
Capacidad de ser probado Nivel de efectividad y eficiencia con el que se pueden establecer los criterios de prueba para un sistema, producto o componente y se pueden realizar pruebas para determinar si se han cumplido esos criterios.

Portabilidad

Nivel de efectividad y eficiencia con el cual un producto de software puede transferirse de un entorno operativo o de uso a otro. Este criterio se manifiesta en los siguientes parámetros:

Adaptabilidad. El producto de software se puede adaptar de manera efectiva y eficiente para entornos operativos o de uso diferentes o en desarrollo.
Facilidad de instalación Nivel de efectividad y eficiencia con el que un producto de software se puede instalar y / o desinstalar con éxito en un entorno determinado.
Capacidad de ser reemplazado. El producto puede reemplazar a otro producto de software especificado para el mismo propósito en el mismo entorno.

1.3.2 Arquitectura Tecnológica de Plataforma de Interoperabilidad

1.3.2.1 Arquitectura Tecnológica

Ruiz (2014) señala que la arquitectura tecnológica es: "Una estructura de software y hardware, incluyendo área de comunicaciones y soporte".

Por otro lado, Thompson (2016), indica que una arquitectura tecnológica es:
"Un modelo conceptual que define la estructura, comportamiento, gobernabilidad y relaciones entre el hardware, software, redes, datos, interacción humana y el ecosistema que rodea nuestros procesos de negocios. Se puede visualizarla como una representación de nuestro entorno tecnológico actual y futuro donde describimos de manera general cada aspecto y lo vamos detallando por capas hasta llegar al nivel más bajo, manteniendo nuestra vista enfocada en las nuevas tecnologías que se desarrollarán."

1.3.2.2 Arquitectura de Aplicación

Ruiz (2014), indica que: "La arquitectura de aplicación identifica cada uno de los sistemas y su relación con el negocio. La arquitectura de aplicación analiza si cada uno de los sistemas satisface ciertos criterios de calidad respecto a los procesos de negocio. Concluyendo de esta manera la importancia de la aplicación para la organización".

1.3.2.3 Interoperabilidad

Según la Secretaría del Gobierno Digital, la interoperabilidad es la: "Habilidad de los sistemas TIC, y de los procesos de negocios que ellas soportan, de intercambiar datos y posibilitar compartir información y el conocimiento".

De la misma forma, Castañeda (2004) señala que la interoperabilidad es “la capacidad de dos o más sistemas para intercambiar la información y utilizarla”.

Por otro lado, el Decreto Supremo N° 083-2011-PCM, lo define como:
“Una infraestructura tecnológica que permite la implementación de servicios públicos en línea, por medios electrónicos y el intercambio de los datos entre entidades del Estado a través de internet, móviles y otros medios tecnológicos”.

1.3.3 Servicios Web Rest

1.3.3.1 Servicios Web

“Es un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios”. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web (Rodríguez & Besteiro, 2014).

Por lo tanto, los servicios web “proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar” (Rodríguez & Besteiro, 2014).

Por lo tanto, es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre las aplicaciones.

1.3.3.2 REST

Según Paredes (2015), REST es “una arquitectura que describe el acto de transferir un estado por sus representaciones”.

1.3.3.2.1 Modelo de madurez de Richardson

Este modelo clasifica las API web en cuatro niveles de madurez según su diseño. El nivel más bajo consiste en usar HTTP solo como un protocolo de transferencia de datos, sin seguir la semántica de los métodos HTTP y los principios arquitectónicos REST. El formato más común para la representación de datos es XML, con un estilo XML simple

(POX). Las interacciones se realizan en estilo RPC, proporcionando acceso a todos los servicios a través de un único punto de acceso.

El segundo nivel comprende las API web organizadas siguiendo el concepto de recursos web. La información accesible a través de la API se modela como múltiples recursos. Cada recurso individual se identifica y aborda de forma única, lo que permite al cliente manejar los recursos individualmente. Se puede acceder a cada recurso a través de su propio conjunto de operaciones.

El tercer nivel de madurez exige la conformidad con la semántica del protocolo HTTP. Requiere el uso correcto de los mensajes HTTP para manejar los recursos en consecuencia, es decir, POST crea un recurso, PUT lo modifica, GET lo recupera y DELETE elimina el recurso correspondiente. Los códigos de estado HTTP también deben usarse en los mensajes de respuesta para describir con coherencia el resultado de las operaciones realizadas en los recursos.

El nivel de madurez más alto requiere la adhesión al principio HATEOAS a través de la provisión de controles hipermedia por la API web. La provisión de controles hipermedia permite a los clientes de la API web explorar y navegar a través de los recursos sin estar estrechamente acoplados a las características internas de la implementación.

1.4 Formulación del problema

¿Cómo la revisión de criterios de calidad basados en servicios web REST permitirá modelar una arquitectura tecnológica de la Plataforma de Interoperabilidad del Estado Peruano?

1.4.1 Delimitaciones de la investigación

La población está constituida por todos los servicios web que actualmente soporta la PIDE. Aproximadamente 210 son los servicios web que se exponen a través de la PIDE. La muestra para la revisión se conforma de los servicios web denominados “Sistema de seguimiento y consulta de inversiones públicas” y “Consulta pública de inversiones”, sobre los cuales se hace una revisión y propuesta de mejora.

1.5 Justificación e importancia del estudio

El proyecto surge debido a la pobre informatización e interoperabilidad del Estado, esto ha generado en los últimos años un descenso significativo de nuestra posición en comparación a los países de la región en términos de gobierno electrónico.

El retraso en el desarrollo del Gobierno Electrónico se convierte en el obstáculo más relevante en el proceso de modernización del Estado. El limitado desarrollo institucional y la lenta implementación de arquitecturas tecnológicas modernas por parte del Estado provocan además serias ineficiencias.

El desarrollo de la presente tesis busca una manera ágil de desarrollar aplicaciones web que permitan hacer más eficientes los procesos de las instituciones de Estado. Las reformas propuestas buscan mejorar el desempeño e interoperabilidad en los organismos públicos y contribuir en una mejor organización de la información.

1.6 Hipótesis

La revisión de criterios de calidad permitirá mejorar un modelo de arquitectura tecnológica de plataformas de interoperabilidad del Estado Peruano basado en Servicios web REST.

1.7 Objetivos

1.7.1 Objetivo general

Revisar los criterios de calidad para modelar una arquitectura tecnológica de la plataforma de interoperabilidad del Estado Peruano basado en Servicios Web Rest.

1.7.2 Objetivos específicos

- Definir los criterios de calidad para el modelado de la arquitectura REST
- Revisar los servicios web del Estado de “Sistema de seguimiento y consulta de inversiones públicas” y “Consulta pública de inversiones”,
- Proponer una arquitectura REST para los servicios web revisados
- Revisar el resultado de la arquitectura modelada

CAPÍTULO II
MATERIAL Y MÉTODOS

2.1 Tipo y diseño de investigación

El tipo de investigación empleada para el estudio se clasifica en Cuantitativa. La razón principal se debe al empleo del análisis de la realidad a través de diferentes procedimientos basados en la medición.

El diseño de investigación es descriptivo. Se evalúan las características de los servicios web sin alterar el código fuente ni experimentar con los resultados de posibles modificaciones.

2.2 Población y muestra

La población está constituida por todos los servicios web que actualmente soporta la PIDE. Aproximadamente 210 son los servicios web que se exponen a través de la PIDE.

La muestra se conforma de los servicios web de "Sistema de seguimiento y consulta de inversiones públicas" y "Consulta pública de inversiones".

2.3 Variables y operacionalización

La variable independiente de la investigación es la Revisión de Criterios de Calidad. La variable dependiente el Modelo de Arquitectura Tecnológica Web Rest.

Para medir la variable independiente se emplea una serie de indicadores, que podemos agrupar en tres criterios de calidad:

1. Desempeño:
   Se relaciona con los criterios de "Eficiencia de desempeño" y "Fiabilidad" de la norma ISO 25010 y se mide a través de los siguientes indicadores:

   a. Tiempo de respuesta: Tiempo que tarda en aparecer algún elemento en la pantalla del usuario, al momento de hacer la petición en su navegador.
   b. Presencia de fragmentos de código no empleados
   c. Presencia de múltiples redirecciones
   d. Presencia de archivos multimedia de gran tamaño: Archivos (Imágenes, documentos, etc.) no optimizados que pueden generar un retraso en el tiempo de respuesta.
   e. Número de peticiones GET fallidas

2. Accesibilidad:

Se relaciona con el criterio de “Usabilidad” y “Compatibilidad” de la norma ISO 25010 y se mide a través de los siguientes indicadores:

a. Etiquetado de los elementos de la web
b. Asignación de atributos al contenido multimedia
c. Facilidad de navegación

3. Buenas prácticas:
   Se relaciona con el criterio de “Seguridad” de la norma ISO 25010 y se mide a través de los siguientes indicadores:

   a. Imágenes con ratio de aspecto incorrecto
   b. Librería de JavaScript poco seguras
   c. Enlaces a destinos poco seguros: Estos enlaces pueden derivar a páginas con posibles amenazas a la seguridad.
   d. Uso de protocolos HTTP no seguros

La elección de estos criterios tiene limitaciones. En específico, no se están considerando los criterios de la norma ISO 25010 de Adecuación Funcional, Mantenibilidad y Portabilidad. Ello se debe a dos razones: i) dichos criterios son más difíciles de medir solo con una navegación por la plataforma web, requiriendo la instalación del servicio web y la revisión por parte de usuarios, y ii) el software empleado para el análisis, Lighthouse®, al igual que otros softwares que evalúan arquitecturas tecnológicas, no comprende indicadores para dichos criterios.

Criterios de calidad de la norma ISO 25010	Criterios de calidad revisados por Lighthouse®		
	Desempeño	Accesibilidad	Buenas prácticas
Eficiencia de desempeño	X		
Compatibilidad		X	
Usabilidad		X	
Fiabilidad	X		
Seguridad			X

Tabla 1. Correspondencia entre los criterios de calidad de la norma ISO 25010 y los criterios de calidad revisados por Lighthouse®.

2.4 Técnicas e instrumentos de recolección de datos

La recolección de datos se hace mediante la observación. La observación es el registro visual de lo que ocurre en una situación real. Viene dado por el comportamiento de los modelos, Es decir, la observación de las características y especificaciones reales, las cuales se pueden observar y documentar.

En el caso específico de esta investigación, se observa el desempeño de los servicios web de “Sistema de seguimiento y consulta de inversiones públicas” y “Consulta

pública de inversiones" tomando como referencia los criterios de calidad antes mencionados.

2.5 Procedimientos de análisis de datos

El análisis de datos consiste en inspeccionar, limpiar y transformar datos con el objetivo de resaltar información útil, lo que sugiere conclusiones, y apoyo a la toma de decisiones.

En el caso concreto del presente estudio, el análisis se realiza mediante una auditoría de los servicios web mediante el software Lighthouse®. Esta auditoría evalúa los indicadores de interés. Además, se repiten las auditorías para comprobar la robustez de los resultados hallados en distintos escenarios.

Lighthouse® es una herramienta de código abierto, propiedad de Google y que sirve para revisar el estado de las páginas web, a través de cinco categorías:

A. Rendimiento: Se define como el rendimiento del sitio web, y se calcula mediante los resultados de la prueba de velocidad, comparando la velocidad del sitio web analizado con otros.
B. Progressive Web App: Es definida como un conjunto de directrices de diseño según Google, esta medida debe proporcionar una experiencia fluida similar a la de un aplicativo móvil, enfocándose en la velocidad de carga.
C. Accessibility: Este criterio hace referencia a la capacidad del sitio web para ser utilizada por tantos usuarios como sea posible.
D. Best Practices: Las mejores prácticas tienen que ver con el análisis del uso de HTTP/2 o HTTP/1.1, de bibliotecas JavaScript front-end con vulnerabilidades de seguridad conocidas, o las dimensiones de visualización de las imágenes.
E. SEO: En esta categoría se busca que la web contenga suficientes metadatos para los motores de búsqueda, como las etiquetas meta del encabezado HTML, el title, etc.

Esta revisión permite saber que mejoras técnicas requiere el sitio web revisado. Al finalizar la revisión de la página se otorga una puntación sobre 100 de acuerdo con los aspectos analizados y aquellos que estén bien implementados.

Luego de la auditoría de los servicios web, se propone una arquitectura REST basada en el modelo MEAN STACK. Este modelo se basa en los criterios de calidad definidos y en las limitaciones encontradas en las auditorías de los servicios web.

2.6 Criterios éticos

Presento esta investigación siguiendo los principios éticos para garantizar la veracidad del presente documento, apoyándome en las citas de diversos autores y las normas APA para las referencias bibliográficas.

Tomo en cuenta el "Código de ética de investigación de la Universidad Señor de Sipán" (Código de ética de la universidad señor de Sipán versión 2.0 elaborado por la dirección de investigación, revisado por el Vicerrectorado de Investigación. Aprobado con Resolución de Directorio N° 024-2017/PD-USS)

También tomo criterios del Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica:

Objetividad: pues la presente investigación no se ve afectada bajo ningún grupo de interés, y tiene como único propósito, explicar y desarrollar una investigación científica para demostrar cómo mejorar un sistema operativo.

Originalidad: citando alguna información que sirva de apoyo para plantear las bases de mis fundamentos que serán muestra de mi propio desarrollo intelectual.

Veracidad: finalmente para demostrar que todo lo desarrollado aquí tiene un fundamento real y no es un invento de la nada

Concluyendo que durante toda mi investigación se puede seguir un proceso de revisión para comprobar el cambio entre el antes y el después de la arquitectura del software.

2.7 Criterios de rigor científico

Mi investigación cuenta con los siguientes criterios de rigor científico:

Credibilidad: Que en la arquitectura diseñada siguió los pasos estándar para su elaboración evitando ser alterada por causas externas que modifiquen la realidad de la investigación.

Fiabilidad: Las técnicas utilizadas para la medición del framework, arrojan resultados similares a los reales y pueden ser replicadas para confianza de otros investigadores.

Validación: Toda la información obtenida ha sido previamente analizada de distintas fuentes para asegurarse que no haya información falsa que pueda perjudicar los resultados.

Replicabilidad: Anotando cada paso que se tomó durante la elaboración de la arquitectura, se deja a disponibilidad su revisión para que pueda obtenerse los mismos resultados.

CAPÍTULO III
RESULTADOS

3.1 Resultados en Tablas y Figuras: Auditoría de servicios web

3.1.1 Servicio web: Sistema de seguimiento de inversiones (SSI)

El SSI es una plataforma del Ministerio de Economía y Finanzas que permite la consulta en línea del funcionamiento de los diversos proyectos de inversión pública existentes, desde su formulación hasta su posterior realización. Para ello es necesario contar con un código SNIP (Sistema Nacional de Inversión Pública) que identifica a cada proyecto. Este código es parte del sistema administrativo del Estado que a través de un conjunto de principios, métodos, procedimientos y normas técnicas certifica la calidad de los Proyectos de Inversión Pública (PIP).

RESULTADOS DE LA AUDITORÍA:

RENDIMIENTO:

El rendimiento de la página web del SSI se sitúa en apenas 2 puntos, encontrándose en una zona bastante deficiente, y considerando que el puntaje más alto es de 100 (atribuido a aquella web que posee la mayor velocidad respecto de todas las webs existentes), se asume que la web revisada presenta una velocidad bastante menor que la mayoría de las webs. Asimismo, en la parte de oportunidades, Lighthouse muestra las oportunidades que ha detectado en forma de recursos que pueden optimizarse para acelerar el sitio web.

ACCESSIBILIDAD:

En esta categoría se observa una asignación de 88 puntos, encontrándose en la zona naranja que indica una puntuación regular. Este puntaje se debería al fácil acceso que tienen los usuarios para poder realizar sus consultas y es muy probable que los usuarios con algún tipo de discapacidad dispongan de una mejor experiencia. Además, se espera que la web tenga una estructura sólida y presentable, y que los atributos estén regularmente definidos, proporcionando metadatos suficientes para los lectores, como los aria-label.

MEJORES PRÁCTICAS:

En esta categoría se ha asignado un puntaje de 71, lo cual indica que no se están utilizando la extensión HTTP2. Esto se ve claramente en las 51 solicitudes no

atendidas a través de HTTP/2. Asimismo, la página carece de un documente HTTP, lo que provoca que la identificación pública esperada en modo peculiar sea una cadena vacía. Por otro lado, la web Incluye bibliotecas front-end de JavaScript con vulnerabilidades de seguridad.

SEO:
En esta categoría se ha realizado una asignación de 75 puntos, lo que indica que la web no contiene suficientes metadatos para los motores de búsqueda. Asimismo, se recomienda formatear la extensión HTML de manera que permita a los rastreadores comprender mejor el contenido de la aplicación.

3.1.2 Servicio web: Consulta a la base de datos de los proyectos de inversión pública:

Es una herramienta WEB que permite consultar toda la Base de Datos de proyectos SNIP a través de dos criterios de búsqueda (código SNIP, localidad), Puede usarse al menos un criterio y a diferencia de la anterior web, solo es necesario tener la localidad. La consulta, te lleva a una pestaña donde los resultados son plasmados una hoja de cálculo para un trabajo mucho más a detalle por parte del usuario.

RESULTADOS DE LA AUDITORÍA:
RENDIMIENTO:
El rendimiento de la página web del SSI se sitúa en 18 puntos, encontrándose en una zona bastante deficiente, al igual que el anterior sitio web. Asimismo, la web revisada presenta una velocidad bastante menor que la mayoría de las webs, y el tiempo de ejecución de ficheros como Java es bastante desproporcionado. Por otro lado, uno existe la utilización de imágenes optimizadas ni el uso de formato web en las mismas.

ACCESSIBILIDAD:
En esta categoría se observa una asignación de 53 puntos, este puntaje se debe a que la parte estética de la web ha sido descuidada, la auditoría señala que los colores de fondo y primer plano no tienen ninguna relación de contraste suficiente. Por otro lado, el documento no tiene un elemento <title>, los elementos de dibujo no tienen atributos [alt] y los elementos de formulario no tienen etiquetas asociadas a la internalización y localización.

MEJORES PRÁCTICAS:
En cuento a las mejores prácticas, se ha asignado un puntaje de 79, situado en la zona regular. Al igual que la web anteriormente revisada, esta no utiliza HTTP/2 y posee un total de 18 solicitudes no atendidas a través de HTTP/2. Un punto a favor

para la web es la seguridad conocida en las bibliotecas front-end, a través de Java Script.

SEO:
En esta categoría se ha realizado una asignación de 67 puntos, lo que indica que la web no contiene suficientes metadatos para los motores de búsqueda. Asimismo, se recomienda formatear la extensión HTML de manera que permita a los rastreadores comprender mejor el contenido de la aplicación. Además, según la auditoría el documento no contiene un elemento <title> ni una meta descripción.

Figura 1. Plataforma web del sistema de seguimiento de inversiones (SSI). Recuperado de: https://ofi5.mef.gob.pe/ssi/

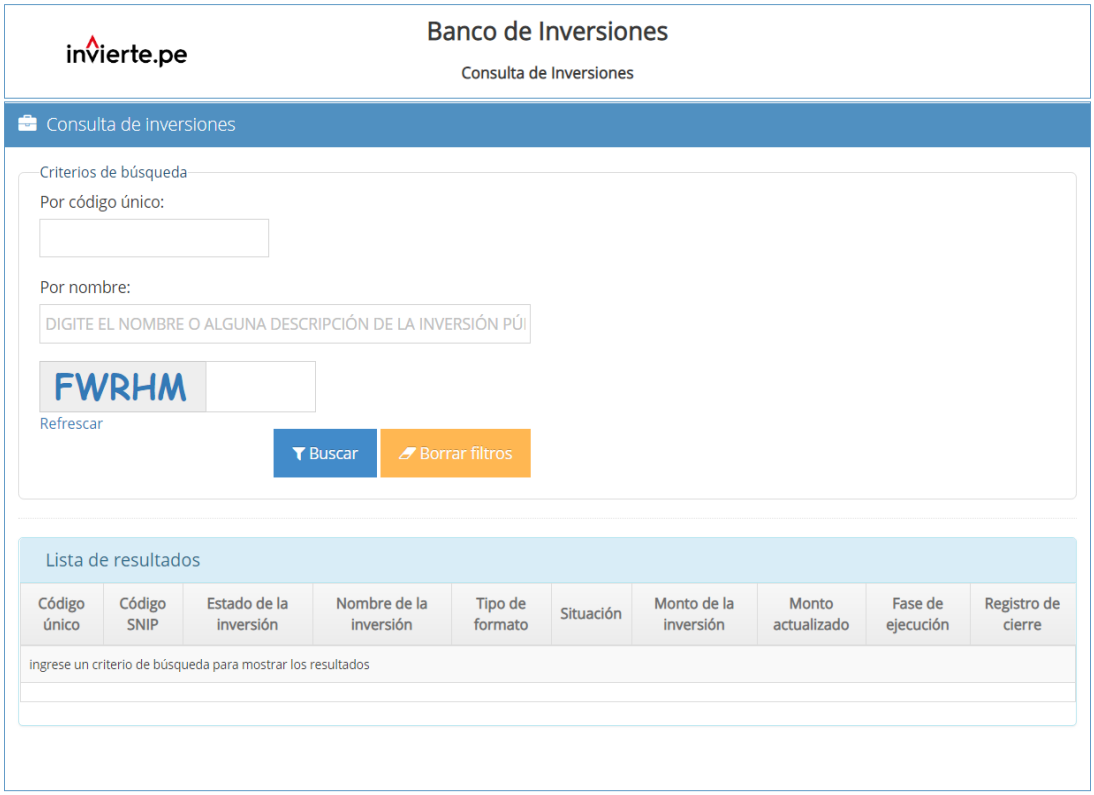

Figura 2. Plataforma web de la consulta pública de inversiones. Recuperado de: *https://ofi5.mef.gob.pe/invierte/consultapublica/consultainversiones*

Medidas

Primera página completa	13.7 s	Primera imagen real	13.7 s
Índice de velocidad	22.6 s	Tiempo de inactividad del CPU	13.7 s
Tiempo de interacción	17.1 s	Tiempo de 1° retardo de entrada	580 ms

Los valores son estimados y pueden variar. La puntuación de rendimiento se basa solo en estas métricas.

Oportunidades

Estas sugerencias pueden ayudar a que la página cargue más rápido. No afectan directamente al puntaje de rendimiento.

Oportunidad	Ahorro estimado
Eliminar recursos que bloquean el renderizado	14.95 s
Minimizar Java Script	0.73 s
Pre-conectar a los orígenes requeridos	0.3 s

Figura 3. Resultados de la auditoría para el SSI: Rendimiento.

Accesibilidad

Estas comprobaciones resaltan oportunidades para mejorar la accesibilidad de la aplicación web. Solo se puede detectar automáticamente un subconjunto de problemas de accesibilidad, por lo que también se recomienda realizar pruebas manuales

Oportunidades

Estos elementos destacan las mejores prácticas de accesibilidad

Los atributos [id] en la página deben ser únicos

Internacionalización y localización

Estas son oportunidades para mejorar la interpretación del contenido por parte de los usuarios en diferentes lugares

El elemento <html> debe tener un atributo [lang]

Nombres y etiquetas

Estas son oportunidades para mejorar la semántica de los controles en la aplicación. Esto puede mejorar la experiencia para los usuarios de tecnología de asistencia, como un lector de pantalla.

Los elementos de formulario deben tener etiquetas asociadas

Mejores Prácticas

No utiliza HTTP/2 para todos sus recursos. – 51 solicitudes no atendidas a través de HTTP/2

Utiliza document.write()

La página carece del tipo de documento HTML, lo que desencadena el modo *quirks*

Incluye bibliotecas javascript de *front-end* con vulnerabilidades de seguridad conocidas – 4 vulnerabilidades detectadas

Figura 4. Resultados de la auditoría para el SSI: Accesibilidad y Mejores Prácticas

SEO

Estas comprobaciones aseguran que la página esté optimizada para el *ranking* de resultados del motor de búsqueda. Hay factores adicionales que Lighthouse no verifica que pueden afectar la posición en el *ranking*.

Amigable para Móviles

Asegurar de que las páginas son compatibles con dispositivos móviles para que los usuarios no tengan que pellizcar o hacer zoom para leer las páginas de contenido

No tener una etiqueta < meta name = viewport > con *width* o *initial-scale* – No se encontró metaetiqueta de ventana gráfica

Mejores Prácticas de Contenido

Formatear el código HTML de manera que permita a los rastreadores comprender mejor el contenido de las aplicaciones

El documento no tiene una descrpción meta

Figura 5. Resultados de la auditoría para el SSI: SEO.

Medidas

Primera página completa	8.4 s	Primera imagen real	9.3 s
Índice de velocidad	9.1 s	Tiempo de inactividad del CPU	9.3 s
Tiempo de interacción	9.5 s	Tiempo de 1° retardo de entrada	290 ms

Los valores son estimados y pueden variar. La puntuación de rendimiento se basa solo en estas métricas.

Oportunidades

Estas sugerencias pueden ayudar a que la página se cargue más rápido. No afectan directamente al puntaje de rendimiento.

Oportunidad	Ahorro estimado
Eliminar recursos que bloquean el renderizado	7.44 s
Habilitar la compresión de texto	4.83 s
Eliminar CSS no utilizado	4.8 s
Minimizar CSS	2.1 s

Figura 6. Resultados de la auditoría para la Plataforma web de la consulta pública de inversiones: Rendimiento.

Accesibilidad

Estas comprobaciones resaltan oportunidades para mejorar la accesibilidad de la aplicación web. Solo se puede detectar automáticamente un subconjunto de problemas de accesibilidad, por lo que también se recomienda realizar pruebas manuales

Contraste

Estas son oportunidades para mejorar la legibilidad del contraste

Los colores de fondo y primer plano deben tener una relación de contraste suficiente

Nombres y etiquetas

Estas son oportunidades para mejorar la semántica de los controles en la aplicación. Esto puede mejorar la experiencia para los usuarios de tecnología de asistencia, como un lector de pantalla.

El documento no tiene un elemento de <título>

Los elementos de imagen no tienen atributos [alt]

Los elementos de formulario no tienen etiquetas asociadas

Internacionalización y localización

Estas son oportunidades para mejorar la interpretación del contenido por parte de los usuarios en diferentes lugares

El elemento<html> no tiene un atributo [lang]

Mejores Prácticas

No utiliza HTTP/2 para todos sus recursos – 18 solicitudes no atendidas a través de HTTP/2

Incluye bibliotecas javascript de front-end con vulnerabilidades de seguridad conocidas – 6 bvulnerabilidades detectadas

Errores del navegador registrados en la consola

Figura 7. Resultados de la auditoría para la Plataforma web de la consulta pública de inversiones: Accesibilidad y Mejores Prácticas.

SEO

Estas comprobaciones aseguran que la página esté optimizada para el *ranking* de resultados del motor de búsqueda. Hay factores adicionales que Lighthouse no verifica que pueden afectar la posición en el *ranking*.

Mejores prácticas de contenido

Formatear el código HTML de manera que permita a los rastreadores comprender mejor el contenido de las aplicaciones

El documento no tiene un elemento de <título>

El documento no tiene una descripción meta

Los elementos de imagen no tienen atributos [alt]

Figura 8. Resultados de la auditoría para la Plataforma web de la consulta pública de inversiones: SEO.

3.2 Aporte práctico: Arquitectura tecnológica propuesta

3.2.1 MEAN Stack

La mayoría de servicios web del Estado emplea la pila (*stack* en inglés) de tecnologías LAMP, que cuenta con un sistema operativo Linux, un servidor Web: APACHE, una base de datos en MySQL y un lenguaje de programación basado en PHP. No obstante, LAMP presentaba ciertas deficiencias en el rendimiento de MySQL para sitios web con gran volumen de inserción de datos (Poulter, Johnston, & Cox, 2015). Además, las aplicaciones tradicionales desarrolladas en la pila LAMP tienen desventajas como la dependencia del sistema operativo y menos dinamismo en las aplicaciones del lado del cliente (Bharath, Reddy, & Dey, 2018).

A raíz de esta problemática, ha surgido una nueva herramienta para el desarrollo web llamada MEAN STACK, esta herramienta está basada únicamente en Java Script y es de código abierto, lo que permite hacer un desarrollo completo de aplicaciones web. MEAN STACK ahora está ganando más impulso como arquitectura para desarrollar aplicaciones basadas en la web, ya que es más flexible, rápido y conveniente en comparación con las arquitecturas convencionales basadas en LAMP (Khue, Binh, Chang, Kim, & Chung, 2017).

MEAN toma su nombre de las cuatro herramientas a las que agrupa:

M: Mongo DB, es una base de datos multiplataforma de código abierto, escrita en C++, almacena datos binarios en formato JSON (Java Script Object Notation). Además, cuenta con la capacidad de escalar una aplicación de forma rápida y rentable al poder agregar más servidores y a diferencia de las bases de datos relacionales, los datos se almacenan con colecciones en pares clave-valor que recuerden a las matrices asociativas. La ventaja de usar MongoDB es la alta capacidad de almacenamiento y velocidad de en las consultas, lo que permite un gran rendimiento.

E: Express JS, es un framework de servidor maduro, flexible y liviano; diseñado para crear aplicaciones web hibridas de una sola página o múltiples páginas. Este framework ligero utiliza el motor Pug para proporcionar soporte a las plantillas. Una de las ventajas de Express es su diseño rápido y fácil de usar, lo que permite al servidor trabajar de manera eficiente con NODE JS dado que se adapta a su filosofía. Asimismo, permite organizar la asignación de rutas hasta el manejo de solicitudes y vistas. En una arquitectura VMC (Model View Controller), Express permite conectar todos los componentes, gestionar y procesar los datos y enviarlos a las capas

A: ANGULAR JS, es un framework de Java Script con el que se puede desarrollar el fronted de una aplicación, es decir la parte en la que los usuarios interactúan directamente. Angula JS es de código abierto mantenido principalmente por Google. Tiene como objetivo introducir la arquitectura MVC (Model View Controller) en la aplicación basada en un navegador que facilita el proceso de desarrollo y prueba. Asimismo, permite usar HTML como lenguaje de plantilla, por lo tanto, puede extender la sintaxis de HTML para expresar los componentes de la aplicación.

N: NODE JS, es el entorno de desarrollo de la capa del servidor, es de código abierto y trabaja con lenguaje Java Script. Además, permite desarrollar rápidamente aplicaciones escalables a nivel servidor. La naturaleza asincrónica de Node.js permite al servidor interactuar con múltiples clientes sin interrumpir las conexiones individuales, garantizando así la escalabilidad del sistema (De Silva, Soysa, Bandara, & Rathnathilake, 2017).

Estas cuatro herramientas permiten el uso de Java Script en cada capa de aplicación, lo que reduce el uso de varios lenguajes de programación y marcos para desarrollar las soluciones propuestas.

MEAN STACK funciona como una un conjunto de herramientas integradas para el desarrollo de aplicaciones web. El proceso para llevar a cabo el desarrollo de aplicaciones web se extiende de la siguiente manera:

1. El cliente o usuario interactúa con ANGULAR JS que es la parte visible para los los usuarios.
2. ANGULAR JS a su vez realiza una solicitud al servidor de NODE JS.
3. El servidor analiza la solicitud y la envía hacia EXPRESS JS.
4. En este punto EXPRESS JS gestiona la obtención de datos, para ello necesitara de la base de datos de MONGO DB mediante una solicitud. MONGO DB recupera los datos solicitados y devuelve la solicitud a EXPRESS JS.
5. Express JS envía los datos a NODE JS, quien se encarga de devolver la solicitud al cliente.
6. Por el lado del cliente o usuario se necesitará la colaboración de ANGULAR JS para mostrar el resultado obtenido en MONGO DB

La principal ventaja de usar estas tecnologías juntas es que todas dependen de JavaScript que se puede usar tanto en el cliente como en el lado del servidor de la aplicación (Ramappa & Bein, 2018).

3..2.2 Ventajas de JavaScript

Con la maduración de JavaScript en los últimos años, la mayoría de sus debilidades se han corregido o mitigado, lo que ha provocado el aumento del valor de sus ventajas (Štajcer & Oreščanin, 2016). De estas ventajas, podemos mencionar las siguientes:

a. El navegador web es la aplicación más utilizada en el mundo: la mayoría de las personas dejan la ventana del navegador abierta durante todo el día, para asegurarse de que pueden usarla cuando lo deseen. El acceso a una aplicación de JavaScript está a solo un clic de favoritos.

b. JavaScript en el navegador es uno de los entornos de ejecución más ampliamente distribuidos del mundo. En los últimos 3 años, el aumento en la activación de teléfonos móviles Android e iOS causó un aumento significativo en el envío de implementaciones de JavaScript robustas en teléfonos, tabletas, computadoras portátiles y computadoras de escritorio en todo el mundo.

c. La implementación de JavaScript es trivial: al alojarla en un servidor HTTP, podríamos hacer que la aplicación de JavaScript esté disponible para más de mil millones de usuarios web.

d. JavaScript es útil para el desarrollo multiplataforma: ahora los SPA se pueden crear con Windows, Mac OS X o Linux, y se puede implementar una sola aplicación no solo en todas las máquinas de escritorio, sino también en tabletas y teléfonos inteligentes. Las bibliotecas maduras como jQuery y PhoneGap que suavizan las inconsistencias y la convergencia de las implementaciones de estándares en los navegadores han contribuido a lo mencionado.

e. JavaScript se ha vuelto sorprendentemente rápido y puede, a veces, competir con lenguajes compilados: por su velocidad, podemos agradecer a una competencia constante y acalorada entre Mozilla Firefox, Google Chrome, Opera y Microsoft. Las implementaciones modernas de JavaScript han experimentado optimizaciones avanzadas, como la compilación JIT para el código de máquina nativo, la predicción de ramificación, la inferencia de tipos y el subprocesamiento múltiple.

f. La evolución de JavaScript le ha permitido incluir características avanzadas: estas características incluyen el objeto nativo JSON, los selectores nativos de estilo jQuery y capacidades AJAX más

consistentes. Las bibliotecas maduras como Strophe y Socket.IO han hecho que la mensajería push sea mucho más fácil.

g. Ha habido un avance significativo en los estándares y el soporte de HTML5, SVG y CSS3; gracias a estos avances, se permite la representación de gráficos perfectos en píxeles que pueden competir con la velocidad y la calidad producidas por Java o Flash.

h. JavaScript se puede utilizar en todo un proyecto web: ahora se puede utilizar el excelente servidor web Node.js y almacenes de datos como MongoDB, que se comunican en JSON. Incluso las bibliotecas se pueden compartir entre el servidor y el navegador.

3.2.3 Diseño propuesto

La característica más importante de MEAN STACK es su interoperabilidad entre plataformas y sistemas operativos, a diferencia de LAMP (Bharath, Reddy, & Dey, 2018). MEAN admite plataformas cruzadas o múltiples y es más flexible. Se utiliza un solo lenguaje, JavaScript, en todas las pilas, tanto en el lado del cliente como en el lado del servidor. Mientras que, en la pila LAMP, se requieren al menos dos idiomas para implementar el lado del cliente y el lado del servidor. El uso de JSON en MEAN hace que los datos fluyan directamente sin formatear. Debido al formato JSON, las operaciones CRUD (CREATE, READ, UPDATE, DELETE) están en un formato similar que a su vez brinda flexibilidad. Aunque LAMP tiene bibliotecas integradas, falla en el lado del cliente, ya que es difícil y confuso. Por las ventajas mencionadas anteriormente, MEAN stack se emplea en este documento.

En el caso particular de la arquitectura MEAN Stack diseñada para esta investigación, se crean dos directorios correspondientes al *back-end* y al *front-end* de la arquitectura.

La Figura 10 muestra la estructura del *back-end* de la arquitectura. Como se puede apreciar, dicha estructura se encuentra en el directorio denominado “ProjectManagerServer”. Allí se desarrolla el servidor a través de las tecnologías NodeJS y ExpressJS, las cuales interactúan con MongoDB para gestionar la información. Se observa las subcarpetas de “controllers” y “data_models”, que corresponden a los controladores y modelos de datos con los que trabaja el servidor, respectivamente.

Se identifica cada controlador con su respectivo modelo de datos. Los controladores y modelos de datos llamados “project" son los que se encargan de gestionar la información sobre cada proyecto. En específico, se encargan de enlistar, añadir, actualizar, eliminar y requerir información sobre cada proyecto, haciendo uso de los

métodos HTTP propios de una arquitectura Web REST. La Figura 12 muestra ejemplos de cómo se emplean estos métodos.

Los controladores y modelos de datos cuyos archivos se llaman "parenttask" se encargan de gestionar la información sobre cada actividad principal dentro de cada proyecto. En concreto, enlistan añaden y requieren información sobre cada actividad principal. Aquellos con archivos denominados "task" gestionan la información sobre cada actividad específica dentro de cada actividad principal. Enlistan, añaden, requieren, actualizan y finalizan cada actividad. Finalmente, aquellos controladores y de modelos de datos denominados "user" gestionan la información sobre cada usuario. Enlistan, añaden, actualizan, eliminan, requieren, asignan funciones y asignan tareas para los usuarios.

La Figura 11 muestra la estructura del *front-end* de la arquitectura. Como se puede apreciar, dicha estructura se encuentra en el directorio denominado "ProjectManagerClient". En la carpeta "app" de esta estructura se encuentran los principales componentes del cliente, el cual es diseñado mediante la tecnología AngularJS.

3.2.4 Autorización y autentificación de usuarios

Autentificación significa la capacidad del servidor para determinar la identidad de los clientes en una transacción. Autorización significa permitir o prevenir el uso de un recurso o servicio particular por parte del servidor. Para el sistema propuesto, JWT (JSON Web Token) se utiliza para autenticación y autorización, cuando se requieren firmas de seguridad. JWT es un protocolo de autenticación, que ha ganado mucho apoyo en los últimos tiempos por su simplicidad y eficiencia. A diferencia de oAuth, que tiene estado, JWT es un mecanismo de autenticación / autorización sin estado, ya que el estado del usuario nunca se guarda en la memoria del servidor. JWT es un medio de representar reclamos que se transferirán entre el cliente y el servidor en la nube. Los reclamos en un JWT están codificados como un objeto JSON que se firma digitalmente utilizando una firma web JSON y opcionalmente se cifra mediante el cifrado web JSON. En la autenticación, cuando un cliente inicia sesión con éxito, el sistema creará un token y lo devolverá. Una vez que el cliente tenga ese token, lo almacenará en almacenamiento local o cookies. Este token es necesario para pasar con cada solicitud en el encabezado de autorización para obtener información después de eso, y el sistema validará el token en cada solicitud utilizando el middleware de ruta.

Se propone tres niveles de usuarios: Grupo de administración (Superadministrador, Administrador, Moderador), Grupo de usuarios (Agente, Usuario final, Anónimo) e Invitado. Super-Admin y Admin pueden administrar todo, pero Admin puede ser agregado o eliminado. Por otro lado, Super-Admin tiene privilegios de *root* y puede administrar todo, y nunca ser eliminado. El moderador puede administrar la información, excepto la información del sistema.

Cada usuario final asocia al menos un SCP con él[1]. Cada grupo de usuarios finales puede ser representado por un agente. Un usuario final puede pertenecer a un grupo administrado por un agente declarando un agente asociado. Toda la gestión de usuarios está diseñada para procesarse en la arquitectura. El usuario está relacionado con la autenticación, autorización y administración de dispositivos (SCP). Según el rol del usuario, el sistema decidirá otorgar el permiso para ejecutar una función o no.

3.2.5 Diseño de la interfaz de usuario

La Figura 13 muestra la interfaz que permite añadir un proyecto. Como se puede apreciar, el usuario debe introducir los datos del proyecto: Nombre, fecha de inicio y fin, y prioridad. Luego de completada la información, el usuario debe dar click en "Añadir". Para el caso de los usuarios que quieren consultar información de forma rápida sobre algún proyecto, se puede observar que existe una opción de búsqueda. Además, se muestra una lista de los proyectos.

La Figura 14 presenta la interfaz para añadir una actividad dentro de un proyecto. Se observa que primero debe agregarse el nombre del proyecto y la actividad principal a los que pertenecen la actividad a ingresar, con la opción de buscar el proyecto y la actividad principal entre los ya existentes. No es obligatorio especificar una actividad principal a la que pertenece porque la actividad ingresada puede ser una actividad principal en sí misma. Luego se ingresa el nombre de la actividad, se especifica si se trata de una actividad principal, se establece la prioridad, las fechas de inicio y término y el usuario encargado de la misma. Luego de completada la información, el usuario debe dar click en "Agregar".

La Figura 15 muestra la interfaz para añadir usuario. Como se observa, la información que debe agregarse es el nombre, apellido e identificador. Una vez subida esta información, el usuario registrados puede ser asignado a administrador de una actividad o director de un proyecto. Además, se muestra un buscador de usuarios y una lista de los usuarios que puede ordenarse según el nombre, apellido e identificador.

La Figura 16 muestra la interfaz para seguir un proyecto. Aquí se enlistan las actividades y además el usuario puede ordenar esta lista según la fecha de inicio y fin, prioridad y estado de completado. Además, hay una opción de búsqueda de actividades.

[1] Signage Content Player (SCP) en este documento significa un sistema integrado responsable de mostrar contenido en un dispositivo de visualización asociado.

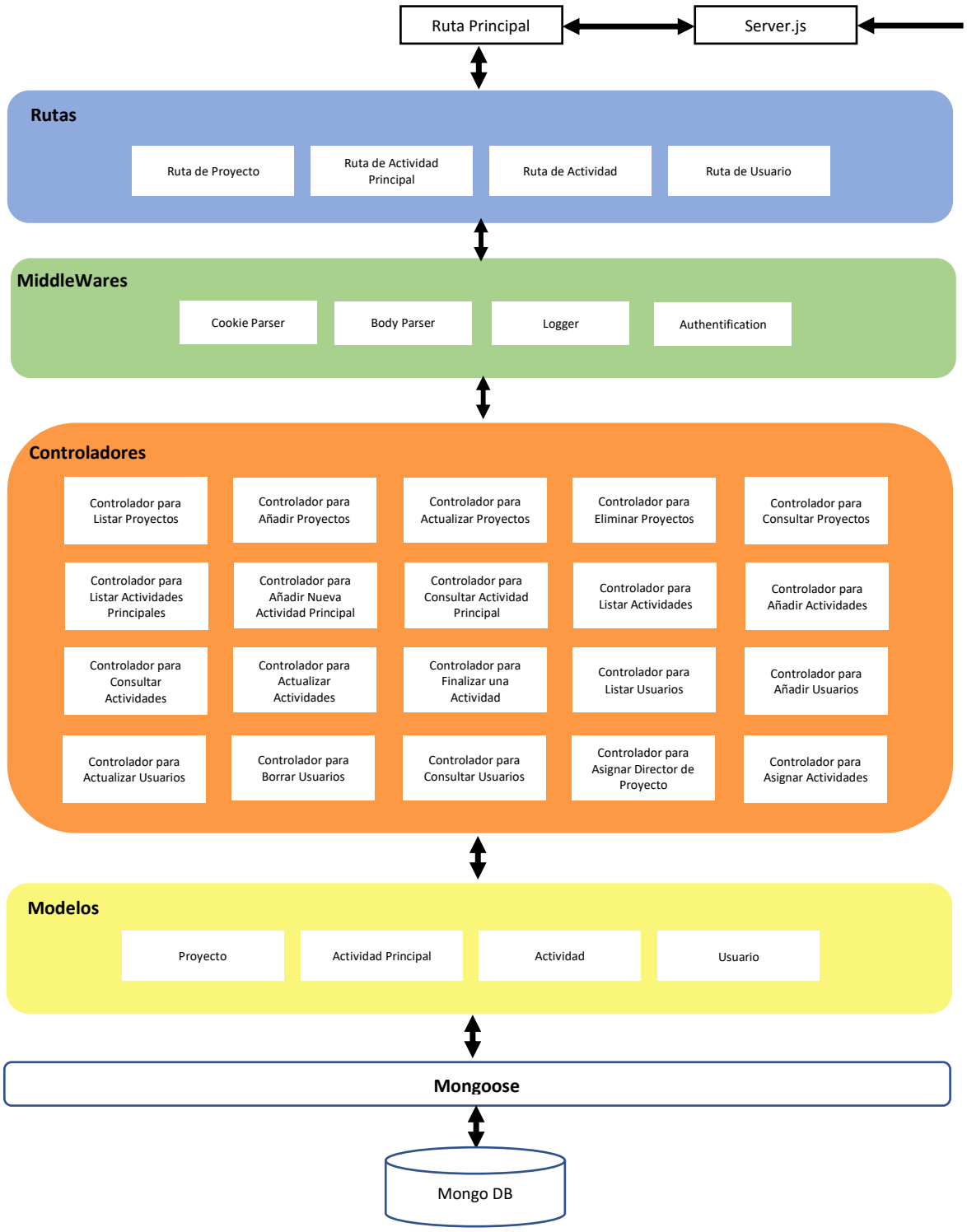

Figura 9. Estructura del back-end de la arquitectura.

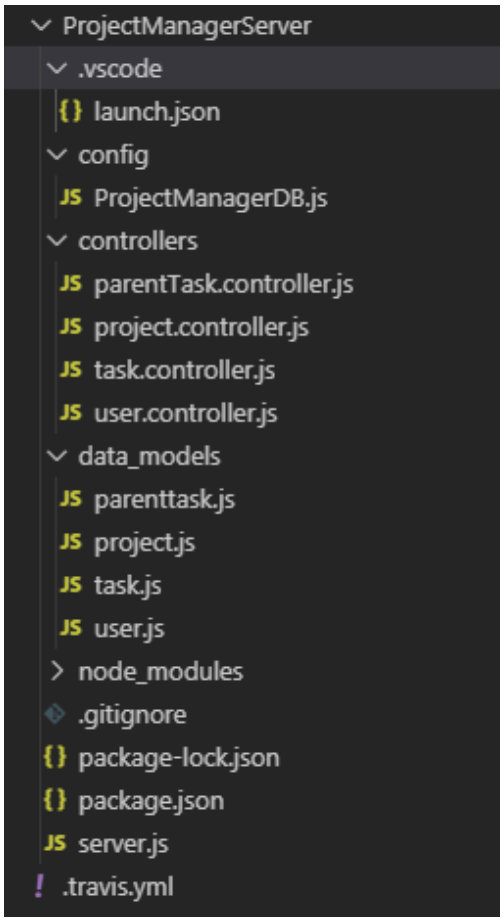

Figura 10. Estructura de carpetas del back-end de la arquitectura.

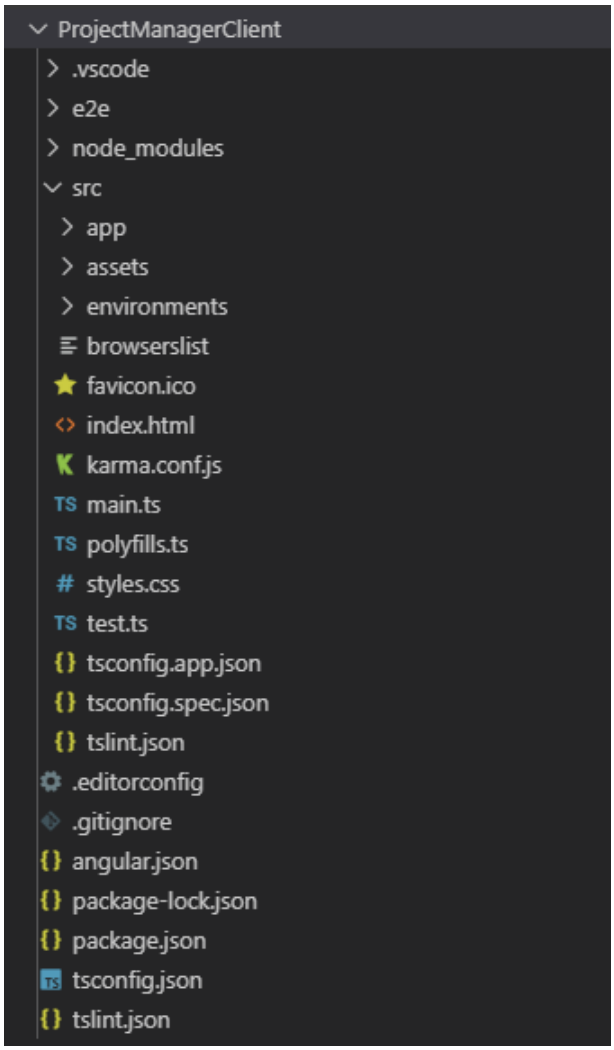

Figura 11. Estructura de carpetas del front-end de la arquitectura.

```
app.get('/api/v1/proyectos/:_id', function(req, res){
    proyecto.getproyectoById(req.params._id, function(err, proyectos){
        if(err){
            throw err;
        }
        res.json(proyectos);
    });
});

app.post('/api/v1/proyectos', function(req, res){
    var proyectos=req.body;
    proyecto.addproyectos(proyectos,function(err, proyectos){
        if(err){
            throw err;
        }
        res.json(proyectos);
    });
});

app.put('/api/v1/proyectos/:_id', function(req, res){
    var id=req.params._id;
    var proyectos=req.body;
    proyecto.updateproyectos(id,proyectos,{},function(err, proyectos){
        if(err){
            throw err;
        }
        res.json(proyectos);
    });
});
```

Figura 12. API REST de la arquitectura.

Figura 13. Interfaz de usuario para añadir proyecto.

Figura 14. Interfaz de usuario para añadir una actividad de un proyecto.

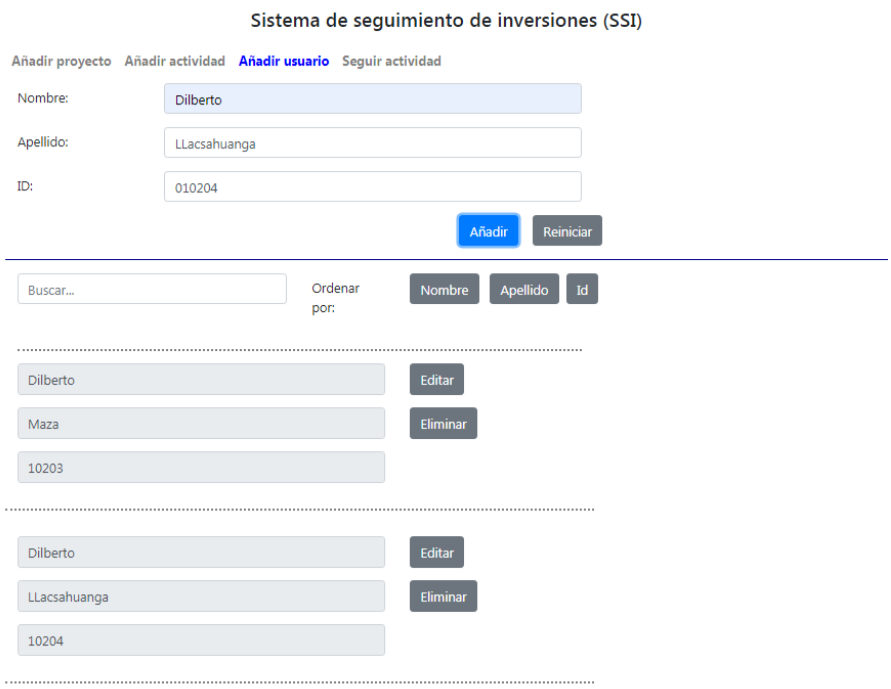

Figura 15. Interfaz de usuario para añadir usuario.

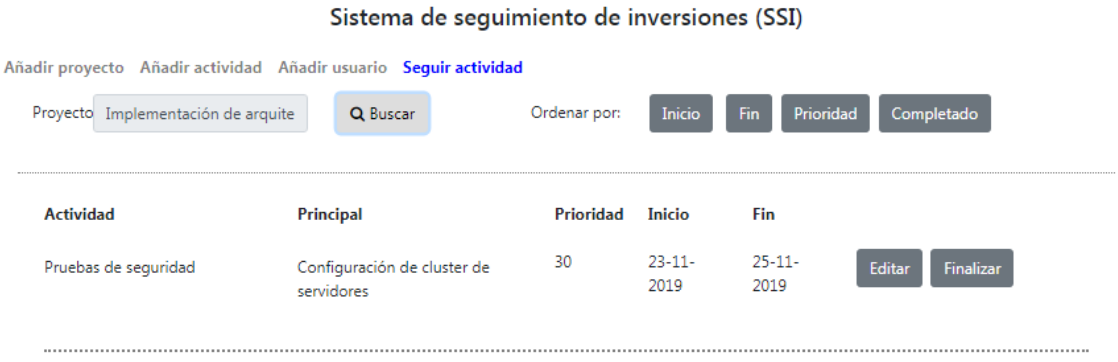

Figura 16. Interfaz de usuario para seguir un proyecto.

CAPÍTULO IV
DISCUSIÓN

Los servicios web auditados reflejan una seria deficiencia en aspectos esenciales como la presentación, la velocidad de carga, la accesibilidad y el uso de los recursos. Ello va de la mano con el uso de tecnologías no acordes con el desarrollo actual de los servicios web. En conjunto, esta situación hace que las plataformas web del Estado no interoperen de forma óptima. Además, una consecuencia principal de ello es que los ciudadanos no puedan acceder de forma eficiente a los servicios e información pública que el Estado debe brindar. Por tanto, se propone una arquitectura REST que emplea la pila de tecnologías MEAN.

Como se aprecia en la Figura 17, la arquitectura propuesta representa una mejora significativa en el desempeño del servicio web. En concreto, se observan mejores tiempos de carga. Por otra parte, la accesibilidad también presenta mejoras. Por tanto, la aplicación de una arquitectura MEAN STACK representaría una mejora notable en los servicios web del Estado y representaría un aporte significativo a la Plataforma de Interoperabilidad del Estado Peruano. En específico, la arquitectura propuesta mejora los criterios de calidad de software que la norma ISO 25010 denomina usabilidad y eficiencia en el desempeño.

Las buenas prácticas y el SEO, por el contrario, no presentan mejoras significativas, como se aprecia en la Figura 18. Con respecto a las buenas prácticas, se señala que no todos los recursos de la página web usan protocolos de seguridad adecuados. Por tanto, una mejora inmediata para la arquitectura propuesta es el mejorar los criterios de seguridad del servicio web. En relación al SEO, también sería una oportunidad de mejora el definir y describir los recursos del servicio web de forma que sea más fácilmente ubicable en Internet.

Asimismo, conviene notar que las pruebas realizadas tendrían mayor validez si se efectúan en una implementación de la arquitectura en un *cluster* de servidores. De este modo, se podría revisar criterios como el de mantenibilidad, escalabilidad y portabilidad, que indican si un diseño de software se puede actualizar de forma ágil, si permite la entrada de múltiples usuarios y si puede ser implementado en diversas plataformas o sistemas operativos.

Medidas

Primera página completa	1.5 s	Primera imagen real	1.5 s
Índice de velocidad	1.8 s	Tiempo de inactividad del CPU	2.5 s
Tiempo de interacción	2.5 s	Tiempo de 1° retardo de entrada	20 ms

Los valores son estimados y pueden variar. La puntuación de rendimiento se basa solo en estas métricas.

Oportunidades

Estas sugerencias pueden ayudar a que la página cargue más rápido. No afectan directamente al puntaje de rendimiento.

Oportunidad	Ahorro estimado
Eliminar recursos que bloquean el renderizado	2.36 s
Minimizar Java Script	0.95 s
Eliminar CSS no utilizado	0.95 s

Estas comprobaciones resaltan oportunidades para mejorar la accesibilidad de la aplicación web. Solo se puede detectar automáticamente un subconjunto de problemas de accesibilidad, por lo que también se recomienda realizar pruebas manuales

Mejores prácticas

Estos elementos destacan las mejores prácticas de accesibilidad comunes

Los atributos [id] en la página deben ser únicos

Figura 17. Resultados de la revisión mediante Lighthouse®: Rendimiento y Accesibilidad.

Mejores Prácticas

No utiliza HTTP/2 para todos sus recursos. – 12 solicitudes no atendidas a través de HTTP/2

Utiliza document.write()

Incluye bibliotecas javascript de *front-end* con vulnerabilidades de seguridad conocidas
2 vulnerabilidades detectadas

SEO

Estas comprobaciones aseguran que la página esté optimizada para el *ranking* de resultados del motor de búsqueda. Hay factores adicionales que Lighthouse no verifica que pueden afectar la posición en el *ranking*.

Mejores Prácticas de Contenido

Formatear el código HTML de manera que permita a los rastreadores comprender mejor el contenido de las aplicaciones

El documento debe tener una descrpción meta

Figura 18. Resultados de la revisión mediante Lighthouse®: Mejores Prácticas y SEO.

CAPÍTULO V
CONCLUSIONES

En la presente investigación se propone una arquitectura que permite mejorar los criterios de calidad de los servicios web analizados. Se trata de la arquitectura REST y se propone como tecnologías para su implementación al MEAN STACK. Este diseño permite optimizar los recursos a través del diseño de páginas web ágiles. En particular, la arquitectura demostró presentar mejores indicadores de desempeño, expresado en tiempos de carga menores. Además, el desarrollo de servicios web se torna más sencillo con estas tecnologías debido a que todas ellas emplean el JavaScript como lenguaje de programación.

En específico, el diseño de arquitectura que se recomienda para los servicios web de "Sistema de seguimiento y consulta de inversiones públicas" y "Consulta pública de inversiones" se basa en un desarrollo web que permita una interacción fluida entre el cliente y el servicio.

Además, se propone extender la arquitectura propuesta con los siguientes puntos a realizar:

1. Un diseño amigable para el cliente: Presentación
2. Estética coherente (colores de fondo, botones, etc.).
3. Establecer de forma más segura el registro y creación de un usuario para realizar el seguimiento de los proyectos más importantes para el cliente.
4. Reforzar los protocolos de seguridad del servicio web.
5. Describir y etiquetar cada recurso de forma que sea rápidamente identificable en un navegador web.
6. Establecer un catálogo de proyectos clasificándolos por localidad, fecha de antigüedad, sector, etc.
7. Permitir la visualización de todos los aspectos del proyecto, dado que existen aún aspectos importantes de cada proyecto que no se pueden visualizar.

Además, conviene notar que una de las limitaciones del estudio es que la arquitectura no se ha implementado en un cluster de servidores. Por tanto, queda como agenda pendiente para investigaciones posteriores el realizar tests de la arquitectura a mayor escala.

REFERENCIAS

Almutairi, B., & Khan, A. (2016). Persistent architecture for optimizing web service for e-government implementation. *2016 IEEE International Symposium on Systems Engineering (ISSE)*, 1-4.

Atencio, D., & Mamani, D. (2017). Interconectividad basado en API REST en aplicaciones web de la municipalidad provincial de Lampa.

BBVA (2017). *Perú Avances en digitalización*. Recuperado de: Naciones Unidas, E-Government survey (2016); World Economic Forum (2016). Recuperado de: https://www.bbvaresearch.com/wp-content/uploads/2017/11/Peru_Avances-en-digitalizacion_nov-17I.pdf

Bharath, M., Reddy, K., & Dey, R. (2018). Implementation of IoT Architecture for Intruder Alert System using MQTT Protocol and MEAN Stack. *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, 1-5.

Burgos, L. (2017). Análisis y evaluación de las arquitecturas REST y SOAP para el desarrollo de servicios web aplicados al ERP "AdrisERP" y su versión móvil en Android.

Castañeda, L. (2004). Interoperabilidad. Estándares. *Revista Digital Universitaria, 5*(10).

Castrillón, H., González, C., & López, D. (2012). Modelo Arquitectónico para Interoperabilidad entre Instituciones Prestadoras de Salud en Colombia. *Revista Ingeniería Biomédica, 6*(12).

Crosby, P. (1987). *La calidad no cuesta. El arte de cerciorarse de la calidad.* México D.F.: CECSA.

De Silva, S., Soysa, W., Bandara, I., & Rathnathilake, R. (2017). A MEAN stack based remote monitoring system for electrical distribution feeders in Sri Lanka. *TENCON 2017-2017 IEEE Region 10 Conference*, 3033-3038.

Deming, E. (1989). Foundation for management of quality in the western world.

Fortunato, F. (2019). Lighthouse architecture demystified. Recuperado de: https://izifortune.com/lighthouse-architecture-demystified/

Huanca, F. (2017). Arquitectura para el desarrollo e implementación de servicios web.

Juran, J., & Gryna, F. (1993). *Manual de control de calidad* (Cuarta ed., Vol. I). Madrid, España: Mc Graw-Hill.

Khue, T., Binh, N., Chang, W., Kim, C., & Chung, S.-T. (2017). Design and implementation of MEAN stack-based scalable real-time Digital Signage System. *2017 8th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, 1-6.

Ministerio De Economía Y Finanzas (2020). Plataforma web de la consulta pública de inversiones. Recuperado de: https://ofi5.mef.gob.pe/invierte/consultapublica/consultainversiones

Ministerio De Economía Y Finanzas (2020). Plataforma web del sistema de seguimiento de inversiones (SSI). Recuperado de: https://ofi5.mef.gob.pe/ssi/

Oliveira, A., & Eler, M. (2017). Strategies and challenges on the accessibility and interoperability of e-government web portals: A case study on Brazilian federal universities. *2017 IEEE 41st Annual Computer Software and Applications Conference, 1*, 737-742.

Orozco, E. (2019). Modelo de una arquitectura basada en REST aplicada a la gestión de procesos en entidades públicas, Caso: División de Programación Operativa de la Administración Tributaria - SUNAT.

Paredes, M. (2015). A systematic review of tools, languages, and methodologies for mashup development. *Software: Practice and Experience, 45*(3), 365-397.

Pham, P. (2016). Hybrid mobile application with Ionic and MEAN stack.

Poulter, A., Johnston, S., & Cox, S. (2015). Using the MEAN stack to implement a RESTful service for an Internet of Things application. *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 280-285.

Ramappa, V., & Bein, D. (2018). MusiqGlobe. fm using MEAN stack. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 661-664.+

Reyes, R., & Tangkeko, M. (2017). Interoperability Framework for Government E-Services of the Philippines. *Pacific Asia Conference on Information Systems 2017 Proceedings, 33*.

Rodríguez, M., & Besteiro, M. (2014). *Servicios web.* Madrid, España.

Ruiz, D. (2014). Diseño de arquitectura empresarial en el sector educativo colombiano: caso colegio privado en Bogotá. *Repositorio Institucional Universidad Católica de Colombia* .

Sarrayrih, M., & Sriram, B. (2015). Major challenges in developing a successful e-government: A review on the Sultanate of Oman. *Journal of King Saud University-Computer and Information Sciences, 27*(2), 230-235.

Štajcer, M., & Oreščanin, D. (2016). Using MEAN stack for development of GUI in real-time big data architecture. *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 524-529.

Sun, P.-L., Ku, C.-Y., & Shih, D.-H. (2015). An implementation framework for E-Government 2.0. *Telematics and Informatics, 32*(3), 504-520.

Thompson, P. (2016). Web-based enterprise management architecture. *IEEE Communications Magazine, 36*(3), 80-86.