Comparativa de Frameworks para el Desarrollo Web en el lado del cliente basado en métricas de desempeño Web Vitals

García Yánez, Renán Andrés y Zurita Hidalgo, Kevin Mateo

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Trabajo de integración curricular, previo a la obtención del título de Ingeniera de Software

Ing. Lascano, Jorge Edison, PhD.

4 de febrero del 2023

ANALYSIS SUMMARY
magister

20230206Garcia-Zurita Proyecto de Titulación

7% Similarities

2% Text between quotes
< 1% similarities between quotation marks
< 1% Language not recognised

Document name: 20230206Garcia-Zurita Proyecto de Titulación.pdf
Document ID: 42f5d050b79efb74f7c1ae45617d49e98aeb68b2
Original document size: 1.19 Mo

Submitter: JORGE EDISON LASCANO
Submission date: 2/7/2023
Upload type: interface
analysis end date: 2/7/2023

Number of words: 21,684
Number of characters: 157,907

Location of similarities in the document:

Main sources detected

No.	Description	Similarities	Locations	Additional information
1	**repositorio.espe.edu.ec** | Estudio comparativo entre paradigmas de programación y ... http://repositorio.espe.edu.ec:8080//bitstream/21000/22123/5/T-ESPE-043669.pdf.txt	6%		Identical words : 6% (1,434 words)
2	**dspace.uazuay.edu.ec** | Revisión sistemática de literatura para la identificación de fa... http://dspace.uazuay.edu.ec/bitstream/datos/7910/3/13650.pdf.txt	< 1%		Identical words : < 1% (25 words)
3	**es.vuejs.org** | La instancia Vue — Vue.js https://es.vuejs.org/v2/guide/instance.html	< 1%		Identical words : < 1% (21 words)

Sources with incidental similarities

No.	Description	Similarities	Locations	Additional information
1	**web.dev** | Web Vitals https://web.dev/i18n/es/vitals/?autm_content=blog_emails	< 1%		Identical words : < 1% (38 words)
2	**openaccess.uoc.edu** | Portal web para la gestión de comunidades de vecinos: Vecino... https://openaccess.uoc.edu/bitstream/10609/126487/9/ccorderorTFG0121memoria.pdf	< 1%		Identical words : < 1% (28 words)
3	**www.academia.edu** | (PDF) Towards a Framework for Shared Understanding and Sh... https://www.academia.edu/39211633/Towards_a_Framework_for_Shared_Understanding_and_Shared_...	< 1%		Identical words : < 1% (20 words)
4	**doi.org** | COMPARISON OF FRONT-END FRAMEWORKS FOR WEB APPLICATIONS DEVE... https://doi.org/10.31784/zvr.6.1.19	< 1%		Identical words : < 1% (15 words)
5	**runebook.dev** | Vue - Template Syntax Vue utiliza una sintaxis de plantilla basada e... https://runebook.dev/es/docs/vue/guide/essentials/template-syntax	< 1%		Identical words : < 1% (15 words)

Referenced sources (without similarities detected)

These sources were cited in the paper without finding any similarities.

1 https://insights.stackoverflow.com/trends?tags=jquery,angularjs,angular,reactjs,next.js
2 https://web.dev/vitals/
3 https://doi.org/10.1109/ACSAT.2014.22
4 https://doi.org/10.1016/j.jss.2018.01.010
5 https://angular.io/guide/what-is-angular

Departamento de Ciencias de la Computación

Carrera de Software

Certificación

Certifico que el trabajo de integración curricular, "**Comparativa de Frameworks para el Desarrollo Web en el lado del cliente basado en métricas de desempeño Web Vitals**" fue realizado por los señores **García Yánez, Renán Andrés** y **Zurita Hidalgo, Kevin Mateo,** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 4 de febrero de 2023

Firma:

Firmado electrónicamente por:
JORGE EDISON
LASCANO .

.....

Ing. Lascano, Jorge Edison, PhD.

C. C.: 1710893114

Departamento de Ciencias de la Computación

Carrera de Software

Responsabilidad de Autoría

Nosotros, **García Yánez, Renán Andrés** con cédula de ciudadanía N° **1719026187** y **Zurita Hidalgo, Kevin Mateo** con cédula de ciudadanía N° **1804603296** declaramos que el contenido, ideas y criterios del trabajo de integración curricular**: Comparativa de Frameworks para el Desarrollo Web en el lado del cliente basado en métricas de desempeño Web Vitals** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 4 de febrero de 2023

Firma

Firma

Firmado electrónicamente por:
RENAN ANDRES GARCIA YANEZ

Firmado electrónicamente por:
KEVIN MATEO ZURITA HIDALGO

García Yánez, Renán Andrés

C.C.: 1719026187

Zurita Hidalgo, Kevin Mateo

C.C.: 1804603296

Departamento de Ciencias de la Computación

Carrera de Software

Autorización de Publicación

Nosotros, **García Yánez, Renán Andrés** con cédula de ciudadanía N° **1719026187** y **Zurita Hidalgo, Kevin Mateo** con cédula de ciudadanía N° **1804603296**, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Comparativa de Frameworks para el Desarrollo Web en el lado del cliente basado en métricas de desempeño Web Vitals** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 4 de febrero de 2023

Firma

Firma

Firmado electrónicamente por: RENAN ANDRES GARCIA YANEZ

.....

García Yánez, Renán Andrés

C.C.: 1719026187

Firmado electrónicamente por: KEVIN MATEO ZURITA HIDALGO

.....

Zurita Hidalgo, Kevin Mateo

C.C.: 1804603296

Dedicatoria

Dedico esta tesis a mis padres, quienes siempre estuvieron conmigo durante toda esta etapa, y siempre supieron apoyarme en todo aspecto.

A mi tío Víctor, América quienes les debo mucho por todo su apoyo brindado durante toda etapa.

Y en general a todos los que pudieron formar parte de esta etapa que llega a culminar, les agradezco por todo.

García Renán

Dedico esta tesis a mis padres, quieres supieron darme todo su apoyo en todo lo que he hecho, siempre han sido y serán una inspiración para mí.

A mis hermanos, quienes siempre me han acompañado y aconsejado, sus palabras y consejos me ayudaron muchos momentos cuando me sentía perdido.

A mis tíos que siempre estaban pendientes de mí, en especial a mi tía Anita y tía Rosa, quienes me ayudaron mientras estudiaba en Quito, sin ellas, esto no hubiera sido posible.

A mis mejores amigos, Sebastián y Daniel, quienes estuvieron presentes en todo momento para hacerme reír y olvidar las preocupaciones.

A mis compañeros, en especial a aquellos que fueron y son miembros del club de Software, fue una de las mejores experiencias que tuve dentro de la universidad. Y a todos, quienes, con su granito de arena, aportaron a mi vida universitaria, siempre se los agradeceré.

Zurita Kevin

Agradecimiento

Agradezco a la Universidad de las Fuerzas Armadas ESPE, por haber sido el establecimiento en el que pude invertir todo este tiempo y que pudo permitirme culminar mi carrera.

A toda la planta docente que compartieron sus conocimientos conmigo y con todos quienes compartimos las aulas. Un agradecimiento especial a Edison que nos guio durante toda la realización de este trabajo.

Finalmente, gracias al Ing. Mauricio Campaña que siempre nos brindó una mano cada que pudo. Gracias a ti Kevin por permitirme realizar este trabajo contigo, siempre paciente y predispuesto, gracias.

García Renán

Agradezco a la Universidad de las Fuerzas Armadas ESPE el haberme permitido estudiar en su establecimiento, siempre recordaré el tiempo invertido en sus aulas y como me ayudo a formarme no solo como profesional sino como persona.

A todos los profesores que con sus enseñanzas me permitieron llegar a donde estoy. En especial a Edison por haber tenido la paciencia de guiarnos durante el proceso del proyecto de titulación, no lo habríamos logrado sin usted.

A mis amigos, quienes me acompañaron en las aulas, no hubiera sido lo mismo sin ustedes. Un agradecimiento especial a Renán con quien realice este trabajo, siempre dispuesto a ayudar, lo logramos.

Zurita Kevin

Índice de Contenido

Índice de Tablas

Índice de figuras

Resumen

El desarrollo web es una de las tendencias más importantes en la ingeniería de software debido, en parte, al aumento de usuarios de Internet en los últimos años y al hecho de que casi todas las aplicaciones interactúan con la Web. Ha aparecido un vasto conjunto de nuevas tecnologías para cubrir la necesidad de desarrollar aplicaciones web modernas. Sin embargo, hay muchos aspectos a considerar al seleccionar una tecnología de desarrollo web sobre otras. Nuestro propósito es proponer un framework para hacer comparaciones entre tecnologías específicas de desarrollo web Front-End. Este framework se basa tanto en las aplicaciones desarrolladas como en los desarrolladores, nuestro objetivo es determinar el mejor framework basado en el desarrollo de aplicaciones pequeñas, el rendimiento de los desarrolladores y la curva de aprendizaje. Planificamos y ejecutamos un experimento con la participación de los estudiantes del Departamento de Informática de la Universidad de las Fuerzas Armadas-ESPE. Comparamos el rendimiento de los desarrolladores, la percepción del aprendizaje y el rendimiento de las aplicaciones al usar dos frameworks front-end: React y Vue. Los resultados muestran que nuestro framework es útil para determinar diferencias estadísticamente significativas a favor de una de las tecnologías. Según los resultados de la encuesta, React tuvo una curva de aprendizaje más fácil que Vue. Además, el tiempo promedio dedicado a resolver los laboratorios propuestos fue más corto en React que en Vue. Pudimos establecer un framework de comparación útil para contrastar el desempeño de diferentes tecnologías web, el desempeño de los desarrolladores y la facilidad para aprender las tecnologías. Sin embargo, pensamos que no es posible generalizar los resultados sin replicar el experimento en contextos más amplios. En conclusión, la selección de la tecnología de desarrollo front-end puede influir en el rendimiento del desarrollador y del software.

Palabras Clave: Vue y React, rendimiento del desarrollador y percepción de aprendizaje, rendimiento de las aplicaciones web, framework de comparación.

Abstract

Web development is one of the most important trends in software engineering due, in part, to the increase of Internet users in the past years, and to the fact that almost every application interacts with the Web. A vast set of new technologies has appeared to cover the need of developing modern web applications. Nevertheless, there are many aspects to consider when selecting one web development technology over others. Our purpose is to propose a framework to make comparisons among specific Front-End web development technologies. This framework is based on both the developed applications and the developers, we aim to determine the best framework based on the development of small applications, the performance of the developers, and the learning curve. We planned and executed an experiment with the participation of the Computer Science Department students of Universidad de las Fuerzas Armadas-ESPE. We compared developers' performance, learning perception, and applications performance when using two Front-End Frameworks: React and Vue. The results show that our framework is useful to determine statistically significant differences in favor of one of the technologies. Based on the survey's results, React had an easier learning curve than Vue. Also, the average time spent on resolving the proposed laboratories was shorter in React than in Vue. We were able to establish a comparison framework useful to contrast the performance of different web technologies, the performance of the developers and the easiness to learn the technologies. Nevertheless, we think that it is not possible to generalize the results without replicating the experiment in broader contexts. In conclusion, the front-end development technology selection may influence the developer's and the software's performance.

Key Words: Vue and React, developer performance and learning perception, web applications performance, comparison framework.

Capítulo I: Introducción

Antecedentes

La Ingeniería de Software aparece por primera vez en 1968, como una solución a los crecientes problemas existentes en el desarrollo de proyectos de software a los cuales se enfrentaban las comunidades de programadores y diseñadores. Normalmente estos proyectos tenían varias funcionalidades implementadas incorrectamente, resultando en un mayor gasto tanto de tiempo como de dinero para el desarrollador y su cliente (Sommerville, 2015).

Una parte creciente de estos proyectos son las aplicaciones web, las cuáles desde la aparición de la primera versión de Internet, la ARPANET (Deitel, 2007), hasta la actualidad han adquirido mayor relevancia, puesto que el crecimiento de la cantidad de información y servicios disponibles y el número de usuarios en búsqueda de acceder dichos servicios ha causado un incremento en la latencia de la Web, resultando en un declive del desempeño de estas aplicaciones su corrección (Makker & Rathy, 2011), debido a la rapidez con la que estás aplicaciones deben ser desarrolladas.

De esta manera, en la actualidad, el desarrollo de aplicaciones web puede ser comúnmente dividido en 2 secciones: el front end y el back end. “El front end es la parte que el usuario observa y con la cual interactúa, como son las pantallas de menú y formularios, mientras que el back end consta del servidor, la aplicación en sí y una base de datos” (Abdullah & Zeki, 2014). Esto ha provocado que los componentes de las aplicaciones web no se encuentren compartiendo un mismo espacio, sino que se encuentran divididos entre varios servicios dependientes uno del otro.

Esta es la razón por lo cual hoy en día, una petición realizada a una aplicación web debe ser autenticada en una base de datos, y los contenidos cargados para la aplicación son leídos desde varios servidores, tanto locales, como remotos y de terceros, esto provoca tiempos de peticiones y carga más

lentos (Shaw, 2000). Esto puede llegar a ser perjudicial ya que los usuarios son sensibles a los retrasos en la respuesta de la computadora, se considera que una demora de más de 10 segundos es un distractor e interrumpe el flujo de trabajo del usuario (Shneiderman et al., 2016).

Como una de las respuestas para resolver este problema por parte del desarrollo front-end, surgen los frameworks. Un framework es un conjunto integrado de artefactos de software (clases, objetos y componentes), que facilitan la realización de una arquitectura para favorecer la reutilización en una familia de aplicaciones relacionadas (Rebitzer et al., 2004). Esto permite agilizar el proceso de desarrollo, reduciendo el tiempo de esfuerzo por parte del desarrollador y por ende los costos del desarrollo de la aplicación (Ramírez et al., 2019). Pero esto, no significa que un framework sea siempre la solución a los problemas de rendimiento de las aplicaciones web o de los desarrolladores.

El uso adecuado de un framework permite la reutilización de código, utilización de patrones de diseño, implementación de interfaces y muchos beneficios más (Ramírez et al., 2019). Dada la gran variedad de frameworks que existen, es necesario analizar los requerimientos de la aplicación y del cliente al momento de decidir con cual se trabajara, ya que todos tienen tanto ventajas como desventajas.

La elección de un framework debe ser realizada tomando en cuenta varios factores como son: costo, calidad, escalabilidad, desempeño, seguridad, ambiente y las necesidades tanto del cliente como del usuario (McCarthy et al., 2019).

Planteamiento del Problema

En un inicio el desarrollo web estaba basado prácticamente en el lenguaje de marcado HTML, juntamente con hojas de estilo (CSS) y para agregarle dinamismo a las páginas web se añadió JavaScript.

Conforme fue avanzando el desarrollo web, nuevas tecnologías emergieron a manera de bibliotecas y frameworks. Estas agilizaron el desarrollo y también mejoraron el rendimiento de las aplicaciones web.

Se crearon bibliotecas como React (sus autores fueron ingenieros de Facebook) para resolver los desafíos que implican el desarrollo de interfaces de usuario complejas con conjuntos de datos que cambian con el tiempo. Esta no es una tarea trivial y la aplicación no solo debe ser mantenible sino también escalable (Xu, 2021).

En relación con frameworks, se tiene por ejemplo Vue, que puede considerarse uno de los frameworks más nuevos, aunque su lanzamiento inicial ocurrió en 2012, dos años antes del lanzamiento de Angular 2. Pero como este último se basó en varios conceptos de la primera iteración (es decir, AngularJS), se podría argumentar que Vue es el más reciente en términos de actualidad.

Otro framework, entre los más usados en la actualidad es Angular, que fue creado originalmente por los empleados de Google, Misko Hevery y Adam Abrons, en 2008. Inicialmente, se lo conocía como AngularJS y fue desarrollado en JavaScript simple. Esto fue en un momento en que la mayoría de los sitios web se basaban en el enfoque de aplicación de varias páginas: cuando un usuario hacía clic en un enlace, el navegador tenía que recuperar el documento HTML solicitado del servidor (Xu, 2021).

En la actualidad, se tiene varias opciones de frameworks y bibliotecas front-end, la tarea de selección de la herramienta apropiada para el desarrollo web es compleja. Por esto se debe establecer criterios para elegir el framework correcto en determinadas situaciones y una metodología o framework que permita compararlos representando sus fortalezas y debilidades.

Justificación

De acuerdo con varios trabajos previos, existen diversos factores a considerar al momento de decidir qué framework usar para el desarrollo del Front End de una aplicación web, aquí es donde se

tiene dos puntos de vista de desempeño: desde el lado del desarrollador donde este se puede evaluar con pruebas, puntos de función; y del lado del usuario, donde se puede evaluar el rendimiento de manera subjetiva, pero que principalmente obedece a los tiempos de carga.

Uno de los principales argumentos en las comparaciones de frameworks es la falta de conocimiento y de comprensión sobre los frameworks y las herramientas utilizadas para producir contenido web. Esto provoca que el desarrollador haga mal uso de las características propias de cada framework, lo que afectarían al desempeño en general de la aplicación (Ollila et al., 2022), por lo tanto, el desempeño de la aplicación podría ser influenciado por la dificultad del aprendizaje y el mal uso del framework seleccionado.

Por lo tanto, es necesario determinar ventajas desde el punto de vista de la facilidad de aprendizaje y del desempeño de un desarrollador al utilizar distintos frameworks. Con esto el desarrollador podría tener un conocimiento previo acerca de cuál sería la mejor biblioteca o framework para obtener el mayor rendimiento posible en un desarrollo web, considerando la complejidad y la capacidad de aprendizaje.

Objetivos

Objetivo General

Comparar la funcionalidad de frameworks de desarrollo Web para el lado del cliente, mediante un experimento, con el propósito de verificar la facilidad de aprendizaje y de desarrollo, y el desempeño de cada aplicación web, mediante métricas de percepción del aprendizaje y métricas Web Vitals para medir el rendimiento y buenas prácticas, en el contexto de estudiantes de pregrado de Ingeniería de Software y Tecnologías de la Información de la Universidad de las Fuerzas Armadas ESPE.

Objetivo Especifico

Elaborar una búsqueda al estado del arte actual en relación con el funcionamiento y desempeño de los frameworks de lado del cliente, desempeño de los desarrolladores y comparaciones entre tecnologías web, para formar una base sólida para la investigación.

Diseñar un experimento con el desarrollo de una aplicación web, para comparar el rendimiento de la aplicación desarrollada, la facilidad de aprendizaje de las herramientas y el desempeño de los desarrolladores al utilizar diferentes frameworks del lado del cliente.

Determinar la validez del experimento propuesto para que los resultados sean representativos dentro del contexto de la investigación.

Implementar el experimento diseñado, dentro de los parámetros establecidos para obtener datos sobre la ejecución del proyecto y experiencia de los sujetos de estudio (estudiantes de pregrado ESPE).

Evaluar e interpretar los resultados del experimento, comparando el desempeño de los desarrolladores durante la implementación de una aplicación web siguiendo instrucciones básicas, mediante el uso de los frameworks/bibliotecas seleccionadas, las métricas de percepción de aprendizaje y las métricas Web Vitals que medir el desempeño de la aplicación desarrollada.

Alcance

Con la finalidad de establecer los límites de la investigación actual, es necesario plantear preguntas de investigación que faciliten el alcance de los objetivos. Estas preguntas se encuentran en la Tabla 1.

Tabla 1

Preguntas de investigación

	Objetivo Especifico	Preguntas de Investigación
I.	Realizar una revisión preliminar de lectura sobre el desempeño en las aplicaciones web y sobre los frameworks a analizar, incluyendo trabajos de comparación previos, para definir las bases del estudio.	a) ¿Qué comparativas se han realizado entre los distintos frameworks? b) ¿Qué es lo que más influye dentro del desempeño dentro de las aplicaciones web c) ¿Un framework es mejor que otro en términos de desempeño?
II.	Determinar la validez del experimento propuesto para que los resultados sean representativos dentro del contexto de la investigación	a) ¿Un experimento sobre estudiantes de pregrado es la mejor opción para este tipo de investigación? b) ¿El experimento cumple con las características de validez?

	Objetivo Especifico	Preguntas de Investigación
III.	Diseñar un experimento sobre el desarrollo de una aplicación web para comparar el desempeño entre distintos frameworks: React y Vue.	a) ¿Es posible aislar el experimento de manera que los resultados sean influenciados únicamente por el framework elegido? a) ¿Las métricas establecidas dentro del experimento se pueden medir con exactitud?
IV.	Realizar el experimento diseñado dentro de los parámetros establecidos para obtener datos sobre la ejecución del proyecto y experiencia de sujetos de estudio (estudiantes de pregrado ESPE)	b) ¿Cuáles son las principales amenazas a la validez que se pueden dar al realizar el experimento? c) ¿es posible generar contingencias adecuadas para estas amenazas a la validez?
V.	Evaluar e interpretar los resultados del experimento, comparando desempeño entre los distintos frameworks y de la misma manera la facilidad con la que el desarrollador adoptó estos.	a) ¿Existe una relación entre los frameworks y el desempeño obtenido? b) ¿Existe una relación entre los frameworks y la facilidad de desarrollo?

Nota. Esta tabla muestra las preguntas de investigación definidas para el proyecto.

Definición de la Investigación

De acuerdo con (Genero Bocco et al., 2015), un experimento en el área de la Ingeniera de Software es un estudio práctico, por lo que es necesario seguir un procedimiento. Este será incorporado en base a la siguiente metodología de trabajo:

Definición del Alcance

El alcance de la investigación ha sido limitado de acuerdo con las metas de esta. Teniendo esto consideración se define la funcionalidad del objeto experimental:

- Una aplicación web que consuma un api para consultar lugares de interés en una ciudad definida.
- Una aplicación web que consuma un api para mostrar información de personajes de una serie animada de televisión.
- Una aplicación web que consuma un api para mostrar información de personajes de un videojuego.

Planificación

a. Contexto

   i. Objetos experimentales

      1. Front end usando React
      2. Front end usando Vue
      3. Back end
      4. Encuestas

         a. Facilidad de aprendizaje

b. Tiempo de desarrollo

ii. Sujetos

Alumnos de la carrera Ingeniería de Software cursando la asignatura Desarrollo Web Avanzado y la asignatura Desarrollo de Aplicaciones Web de la carrera de Ingeniería en Tecnologías de la Información, de la Universidad de las Fuerzas Armadas ESPE.

b. Selección de sujetos

La selección de los sujetos para las pruebas del experimento se hará de manera aleatoria, adicionalmente todos los sujetos responderán la encuesta sobre la facilidad de aprendizaje de acuerdo con el framework asignados a ellos aleatoriamente.

c. Variables

i. Variables independientes:

1. Elección de framework de desarrollo

ii. Variables dependientes

1. Facilidad de aprendizaje

2. Desempeño de la aplicación desarrollada

d. Hipótesis

Las hipótesis nulas y alternativas han sido escritas en base a las variables dependientes desarrolladas en la investigación, detalladas en la Tabla 2.

Tabla 2

Hipótesis nulas y alternativas del experimento

Hipótesis Nula	Hipótesis alternativas
$H_{0,1}$: $LCP_{React} = LCP_{Vue}$	$H_{1,1,1}$: $LCP_{React} \neq LCP_{Vue}$ $H_{1,1,2}$: $LCP_{React} > LCP_{Vue}$ $H_{1,1,3}$: $LCP_{React} < LCP_{Vue}$
$H_{0,2}$: $FID_{React} = FID_{Vue}$	$H_{1,2,1}$: $FID_{React} \neq FID_{Vue}$ $H_{1,2,2}$: $FID_{React} > FID_{Vue}$ $H_{1,2,3}$: $FID_{React} < FID_{Vue}$
$H_{0,3}$: $Performance_{React} = Performance_{Vue}$	$H_{1,3,1}$: $Performance_{React} \neq Performance_{Vue}$ $H_{1,3,1}$: $Performance_{React} > Performance_{Vue}$ $H_{1,3,1}$: $Performance_{React} < Performance_{Vue}$
$H_{0,4}$: $Dev\ Performance_{React} = Dev\ Performance_{Vue}$	$H_{1,4,1}$: $Dev\ Performance_{React} \neq Dev\ Performance_{Vue}$ $H_{1,4,1}$: $Dev\ Performance_{React} > Dev\ Performance_{Vue}$ $H_{1,4,1}$: $Dev\ Performance_{React} < Dev\ Performance_{Vue}$
$H_{0,5}$: $Learning_{React} = Learning_{Vue}$	$H_{1,5,1}$: $Learning_{React} \neq Learning_{Vue}$ $H_{1,5,1}$: $Learning_{React} > Learning_{Vue}$ $H_{1,5,1}$: $Learning_{React} < Learning_{Vue}$

Nota. Esta tabla describe las distintas hipótesis nulas y alternativas.

Operación

La operación de la investigación consiste en la ejecución del experimento propuesto, haciendo uso de un primer laboratorio, en el cual se busca igualar el nivel de conocimiento de los participantes, para posteriormente ejecutar dos laboratorios para la recolección de datos, los cuales serán limpiados y procesados.

Análisis e Interpretación

El análisis se realizará posterior a la ejecución de pruebas estadísticas que permitan determinar la distribución de los datos, con lo cual se obtendrá suficiente evidencia para rechazar o aceptar las hipótesis y concluir con la investigación.

Hipótesis de Investigación

Se ha planteado la siguiente hipótesis de trabajo:

- **Hipótesis de Investigación:** La elección del framework de desarrollo y su facilidad de aprendizaje influye en el desempeño del desarrollador al momento de usarlo e implementarlo dentro de un proyecto de software.

  De acuerdo con la elección del framework de desarrollo y la facilidad de aprendizaje que tiene cada uno, aunque este llegue a ser más difícil que otro, es directamente proporcional al desempeño del desarrollo y del producto final desarrollado.

Capítulo II: Marco Teórico y Estado del Arte

Marco Teórico

Antes de poder desarrollar la investigación planteada, hace falta introducir varios conceptos previos que funcionarán como fundamentos teóricos que ayudarán a entender el contexto del presente trabajo. Se tratan los temas referentes a las variables dependientes e independientes de la hipótesis de investigación: desempeño del desarrollador y la percepción de la facilidad con la que el desarrollador percibe a los a frameworks de desarrollo para aplicaciones web.

Aplicaciones Web

Una aplicación web es un programa que utiliza una arquitectura cliente / servidor, e interactúa con los usuarios a través de un cliente web y con otros sistemas mediante el protocolo HTTP. Para un usuario, lo más probable es que el cliente web sea un navegador web como Google Chrome o Safari, pero puede hacerlo desde otros programas, o desde un agente de usuario HTTP que actúa como un navegador automatizado.

El usuario final visualiza páginas web y puede interactuar enviando información hacia el sistema. Las funciones realizadas pueden variar desde tareas relativamente simples, hasta aplicaciones que realizan actividades totalmente sofisticadas como gestión de inventarios, o manejo de procesos bancarios. Se debe tener en cuenta que las aplicaciones web se encuentran formadas por dos partes componentes: Front End y Back End (Kaluža & Vukelić, 2018).

Front End

El front end se define como la parte que el usuario visualiza y con la cual interactúa, por ejemplo, menús y formularios de contacto. Para diseñar y desarrollar una interfaz web front-end, se debe utilizar ciertas herramientas y tecnologías, que generalmente son una combinación de HTML, CSS y

JavaScript. Actualmente, las aplicaciones web sofisticadas suelen incluir los servicios de front end y back end. Es indispensable considerar que para un mejor desempeño de las aplicaciones web se utilizan frameworks y bibliotecas para desarrollo de front-end web (Kaluža & Vukelić, 2018).

Frameworks de Front End

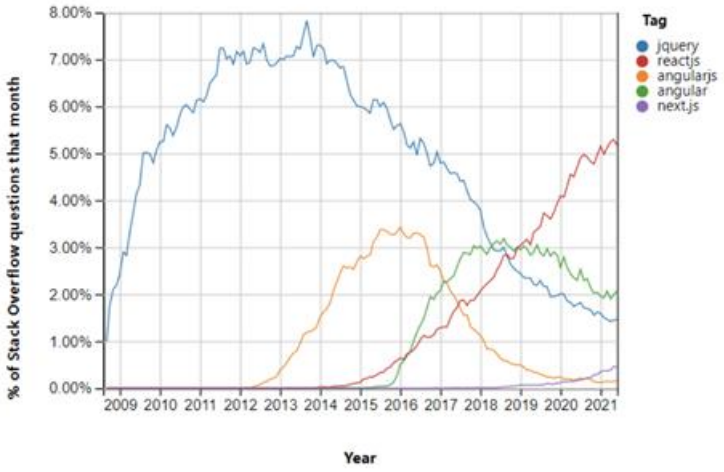
Los frameworks son considerados tecnologías básicas para el desarrollo de aplicaciones, tienen varias soluciones informáticas que ofrecen las siguientes ventajas:

- Incremento en la velocidad de desarrollo.
- Reducción del número de errores de programación.
- Facilidad para colaboración.
- Programación compleja.

(Kaluža & Vukelić, 2018) mencionan como los frameworks son utilizados dependiendo de su propósito, por ejemplo, si un framework se utiliza para implementar SPAs (Single page applications) o MPAs (Multiple Page Applications). Concluyen, que en su mayoría los frameworks pueden ser utilizados para ambos tipos de aplicaciones. Pero existen excepciones, como Angular, que es inadecuado para MPAs. Esta podría ser una de las razones que han provocado la baja de su popularidad, como se puede ver en la Figura 1. Ahí se muestran las tendencias de bibliotecas y frameworks de JavaScript (JQuery, React.js, Angular.js, Angular, Next.js) en el sitio StackOverflow en años recientes.

Figura 1

Tendencias JavaScript Frameworks.



Nota. El gráfico representa las tendencias de los frameworks dentro de JavaScript.

https://insights.stackoverflow.com/trends?tags=jquery%2Cangularjs%2Cangular%2Creactjs%2Cnext.js

React

React es una biblioteca de JavaScript desarrollada por Facebook (Meta Platforms, Inc.) que se puede utilizar para crear interfaces de usuario para la web. Se publicó como software de código abierto en 2013 y desde entonces ha ganado mucho la atención en el mundo de los desarrolladores. Algunos de los casos de uso más populares de React incluyen Instagram y WhatsApp. Si bien React a menudo está conectado al desarrollo web, su núcleo es una biblioteca independiente y se puede combinar con react-dom, para desarrollar interfaces de usuario para la web. JavaScript es una parte inherente de React, porque es su principal lenguaje de desarrollo (Wohlgethan, 2018). Debido a que React solo es

responsable de la vista de una aplicación, el proceso de desarrollo requiere una lista de varias tecnologías orientadas al desarrollo de front end para ser efectivo:

- Compilador (para JSX)
- Módulos (y un cargador apropiado) para la estructura de la aplicación
- Proceso de construcción
- Enrutamiento
- Manejo de estados

El núcleo de React está hecho de componentes. El objetivo general es transformar un determinado estado de la aplicación en una vista que se pueda mostrar en el navegador. Es posible escribir componentes con dos enfoques diferentes: componentes como funciones y componentes como clases ES6.

Componentes como Funciones. Hay funciones puras que devuelven exactamente un ReactElement. El nombre del componente es también el nombre de la función. Este enfoque, sin embargo, tiene sus limitaciones: no se puede alterar el estado ni se pueden usar métodos de ciclo de vida (p. ej., componentDidMount).

Props. Los componentes de React se pueden configurar con propiedades y tener un estado mutable internamente. Sin embargo, las propiedades dentro de un componente son inmutables, pero se pueden configurar desde un punto externo de la aplicación en todo momento.

Estado. El estado es responsable de la gestión de datos dentro de un componente.

JSX. React no utiliza un lenguaje de plantilla específico, pero el código JavaScript se puede enriquecer con fragmentos de código HTML. Esta extensión se llama JSX, que se traduce aproximadamente como 'JavaScript extenso'. Al usar este enfoque, se requiere un compilador para transformar el código JSX en JavaScript simple (Wohlgethan, 2018).

En React normalmente se estructuran los proyectos de dos formas, pero la más común es por tipo de archivos, es decir, si son archivos de API o de Componentes ahí se van estructurando.

Vue.js

Vue.js (también conocido como Vue), puede considerarse el framework más nuevo en esta comparación, aunque su lanzamiento inicial ocurrió dos años antes de Angular 2. Pero como Angular 2 se basó en conceptos de la primera versión (AngularJS), es decir que Vue, de hecho, es el más reciente ya que fue lanzado en 2014 por EvanYou, ex empleado de Google, donde trabajó con AngularJS. Sin embargo, la versión 1.0.0 de Vue.js recién llegó en octubre de 2015.

Vue.js utiliza una sintaxis de marcado basada en HTML, que se vincula a la instancia de JavaScript subyacente de Vue. Todas las plantillas de Vue son HTML válidos. Los datos provenientes de la aplicación Vue se pueden vincular a HTML usando la sintaxis "Bigote" (llaves dobles), por ejemplo:

* Mensaje: {{ msj }} .*

Todo lo escrito dentro de llaves dobles se evaluará como JavaScript en el ámbito de datos de una instancia de Vue.

Las aplicaciones Vue están conformadas por componentes reutilizables llamados instancias de Vue, que aceptan opciones como datos, métodos y enlaces de ciclo de vida. Es común que una aplicación se organice en un árbol de componentes anidados. Para usar componentes en plantillas, deben estar registrados para que Vue los conozca. Los componentes se pueden registrar global y localmente. Un componente registrado globalmente está disponible para todos los subcomponentes de la instancia de Vue.

Los datos de la aplicación se almacenan en la función de datos de los componentes, que actúa como un estado de los componentes. Los datos se pueden pasar como apoyo de los componentes principales a los secundarios. Cuando se pasa un valor como un atributo prop, se convierte en propiedad

de esa instancia de componente. Se puede acceder al valor de propiedad en el componente de la misma manera que un atributo de datos. Un componente puede tener una cantidad ilimitada de árboles de accesorios.

Un componente de Vue es una instancia reutilizable con un nombre. Una *plantilla* de Vue utiliza una sintaxis de template basada en HTML, que le permite vincular de forma declarativa el DOM renderizado a los datos de la instancia de Vue subyacente. Las *props* de Vue son atributos personalizados que se puede registrar en un componente, cuando se pasa un valor a un atributo prop, se convierte en una propiedad en esa instancia de componente.

La estructura de carpetas se compone básicamente de una carpeta *public* donde se encuentran el *favicon* y el *html* público, aparte de esta, se crea la carpeta *src* donde se incluyen los componentes, los assets y las vistas.

Similar a React, Vue usa Virtual DOM para actualizar la página. Cada instancia de Vue pasa por una serie de pasos de inicialización cuando se crea. Necesita configurar la observación de datos, compilar la plantilla, montar la instancia de Vue en el DOM y actualizar el DOM cuando cambien los datos. Durante este proceso, también ejecuta los enlaces del ciclo de vida, que permiten a los usuarios agregar su código en las etapas específicas.

Los paquetes de terceros en Vue se pueden instalar a través de NPM o importándolos como dependencias a través de una etiqueta de secuencia de comandos. Las bibliotecas de terceros proporcionan a Vue una funcionalidad adicional y mejoran su facilidad de uso. Uno de los paquetes de terceros más utilizados para Vue JS es VueX.

VueX es un patrón de administración de estado y una biblioteca para aplicaciones Vue. Sirve como un almacén de datos centralizado para todos los componentes de la aplicación, lo que garantiza

que el estado se pueda cambiar solo de manera predecible. VueX está inspirado en Flux, Redux y The Elm Architecture, con la única excepción de que está diseñado específicamente para Vue (Saks, 2019).

A pesar de que React es considerada una biblioteca desde el principio de su lanzamiento, en la actualidad, gracias a la gran cantidad de funcionalidades que ha acumulado a lo largo de los años, muchos lo consideran un framework de desarrollo front end. De acuerdo con (Gackenheimer, 2015), React es un framework de escala completa como Angular por ejemplo, puesto que maneja una arquitectura MVC siguiendo estándares comunes entre varios frameworks.

Por otra parte, (Boduch & Derks, 2020) en su libro React and React Native: A complete hands-on guide to modern web and mobile development with React.js, 3ra edición, dicen que mucha gente piensa que React solo permite manejar la vista de las interfaces de una aplicación, pero gracias a la existencia de componentes, estados y contextos permite hacer la función de framework, un especializado en soluciones simples y de alto rendimiento.

Adicionalmente (Saks, 2019), en su trabajo de graduación ”JavaScript Frameworks: Angular vs React vs Vue”, habla como React es el framework más popular de JavaScript, debido a su facilidad de aprendizaje y rápido escalado, puesto que muchas veces no es necesaria una gran complejidad cuando solo se requiere crear una SPA (Single Page Application).

Angular

Angular es un framework de diseño de aplicación y plataforma de desarrollo para crear sofisticadas y eficientes aplicaciones de una sola página SPA (*Angular - What Is Angular?*, n.d.). Este framework es uno de los más populares para el desarrollo web (Baida et al., 2020). No se puede negar su importancia y uso en el desarrollo web en la actualidad.

Angular se basa en el principio de Pareto o también conocida como la regla 80-20, esto significa que permite hacer el 80% de las tareas con un 20% del esfuerzo. La filosofía detrás de Angular es errar

del lado de la configuración sobre la convención, al aumentar la simplicidad. Los frameworks basados en convenciones, aunque pueden parecer elegantes desde el exterior, dificultan que los desarrolladores novatos aprendan Angular. Los frameworks basados en la configuración tienen como objetivo exponer su funcionamiento interno a través de una configuración y hooks explícitos, donde se adjuntan los comportamientos personalizados al framework (Uluca, 2018).

Para entender a Angular, es necesario comprender las funcionalidades que ofrece de manera básica al momento de programar. La primera funcionalidad son los llamados “Components” o componentes, estos son bloques de construcción que componen a una aplicación. Este hace uso de un decorador en el cual se especifica un selector de CSS, o estilo de la página, una plantilla de HTML que indica como renderizar el componente a Angular y un conjunto opcional de CSS para definir la apariencia de los elementos de la plantilla de HTML (*Angular - What Is Angular?*, n.d.).

La segunda funcionalidad básico son los “Templates” o plantillas, esto como se mencionó anteriormente se encuentra dentro de cada componente de Angular, ya que define como se renderiza. Angular brinda un soporte adicional a HTML que permite insertar valores dinámicamente desde cualquier componente, siendo estos actualizados automáticamente en el DOM una vez se ha realizado un cambio (*Angular - What Is Angular?*, n.d.).

Finalmente, una de las grandes funcionalidades y ultima a destacar de Angular son las “Dependency injections” o inyecciones de dependencia, esto facilita la declaración de las dependencias de las clases de TypeScript sin la necesidad de instanciar las mismas. Este patrón de diseño permite escribir código más flexible y testeable. Este es uno de los servicios más beneficiosos de Angular pero que a su vez no es de los más dominados por los desarrolladores (*Angular - What Is Angular?*, n.d.).

Estructura de un Proyecto en Angular. Todo proyecto creado en Angular parte de un mismo formato, el cual es generado automáticamente al crearlo. Esto se logra mediante la ejecución del comando:

$ng new nombre-del-proyecto

El resultado del uso de este comando son una serie de carpetas, cada una con una o más funciones específicas, las cuales se detallan a continuación:

- ***e2e*:** Más conocida como "end to end". Esta carpeta contiene una serie de archivos, cuya función es realizar pruebas automáticas, como si un usuario interactuara con la aplicación. Se ejecuta con el comando *e2e*.
- ***node_modules*:** Es la carpeta que contiene todas dependencias del proyecto, en otras palabras, bibliotecas y funciones.
- ***src***: Es el directorio del código del proyecto, donde se almacenan todos los ficheros que contienen el código desarrollado (HTML, CSS, TypeScript, JavaScript). Este a su vez cuenta con más carpetas internas y varios archivos:
    - **Carpeta app**: Aquí se encuentra toda la implementación de los componentes, templates y archivos de estilo.
    - **Carpeta assets**: Contiene como su nombre indicia los assets y archivos adicionales del proyecto. Los assets son cualquier archivo o ítem necesario para el proyecto, pero no necesariamente vinculado a la programación, por ejemplo: imágenes, videos.
    - **Carpeta enviroments**: Almacena los archivos de configuración y las variables de entorno del proyecto.
    - **Favicon.ico:** Es el archivo del icono del proyecto.

- **Index.html**: Es el archivo de la página principal del proyecto.
- **Main.ts**: Es un archivo del lenguaje TypeScript donde se almacena las configuraciones globales del proyecto.

Comparación entre React, Vue y Angular

A pesar de las diferencias que puedan tener estos frameworks como se puede observar en la Tabla 3, también tienen varias características y ventajas comunes. Una de las más importantes es la mejora de la calidad del proyecto, como proveer una estructura definida y evitar código repetitivo (Ramírez et al., 2019). Esto adicionalmente aumenta el rendimiento de los desarrolladores, ya que acorta los tiempos de desarrollo y promueve las buenas prácticas (Fagerholm et al., 2015).

Tabla 3

Comparativa React, Angular y Vue

Atributos	Angular	React	Vue.js
Tipo	Javascript Framework	Librería JS Open Source	Framework Progresivo JS
Descargas Semanales NPM	444,594	5,036,078	996,293
Velocidad de codificación	Lenta	Normal	Rápida

Atributos	Angular	React	Vue.js
Documentación	Sí	Sí	Sí
Renderizado	Lado del Cliente	Lado del Servidor	Lado del servidor
Modelo	MVC	Virtual	Virtual

Nota. Esta tabla describe las principales similitudes y diferencias entre los 3 frameworks de JavaScript

Calidad de un Proyecto de Software

La calidad de software es una compleja combinación de varios factores y perspectivas, que cambian dependiendo del tipo de software a desarrollar y las necesidades del cliente. (Pressman, 2005) presenta tres puntos importantes al momento de evaluar la calidad de un proyecto de software: un proceso eficaz de desarrollo de software, un producto software útil y un valor agregado para el desarrollador y los stakeholders.

Un proceso eficaz de software permite evitar errores y confusiones durante el proyecto pues funciona de base para la planificación. Para esto se puede usar varias metodologías o estrategias creadas por desarrolladores e ingenieros, estas metodologías facilitan el análisis de los requerimientos y problemas planteados por el cliente, permiten encontrar una solución correcta y el método óptimo para lograr un producto útil y con la menor cantidad de fallas.

Un producto útil se entiende como aquel que cumple con su propósito y las expectativas del usuario con la mínima necesidad de conocimiento por su parte. Un software útil debe cumplir con estas características, además de con otras propias de un software como son requerimientos funcionales,

como son el tipo de plataforma en la que se puede usar, la cantidad de recursos que debe usar, y requerimientos no funcionales, como son la usabilidad, portabilidad, accesibilidad.

Agregar valor es no limitarse a los requerimientos establecidos por el cliente, debido a que varias veces las funcionalidades solicitadas por este no dan como resultado un software útil para el usuario. Es por esta razón que, como desarrolladores, se tiene que buscar mejoras y soluciones a los problemas del usuario, ya que esto no sólo añade valor al desarrollo, sino también permite tener un mejor conocimiento sobre distintos aspectos relacionados con el negocio.

Es por esto por lo que se define el concepto de calidad dentro del desarrollo de software, como multidimensional, dependiendo del punto de vista: la calidad del producto y la calidad del proceso. Pero la mejora de los procesos no asegura la calidad del producto. Para garantizar un resultado uniforme en un procedimiento es necesario estandarizar un modelo procedimental, pero al no ser el apropiado, estos productos no cumplirían con la calidad esperada (B. Kitchenham & Pfleeger, 1996).

Según (Valenciano López, 2015) otras perspectivas son:

- **Perspectiva Transcendental:** Se basa en la subjetividad del investigador, donde reconoce la existencia de calidad mas no se tiene la capacidad de definirla.
- **Perspectiva del Usuario:** Se fundamenta en la experiencia del usuario con el objeto en cuestión, enfocándose en el cumplimiento de necesidades antes que la calidad objetiva.
- **Perspectiva basada en Aspectos Económicos:** La calidad está fuertemente ligada con cualquier aspecto monetario, incluido costos y ganancias.

Pero lo mencionado anteriormente, no significa que no existan estándares formales al momento de evaluar la calidad de un proyecto de software, por ejemplo, las normas ISO/IEC 25000 establecen un

framework común para evaluar la calidad del producto de software, convirtiéndose así en la piedra angular de esta área de la Ingeniería de Software (Rodríguez et al., 2015).

El objetivo de estas normas ISO es proveer una revisión general de los estándares SQuaRE (Standards named Systems and software Quality Requirements and Evaluation), modelos de referencia y definiciones comunes (*NORMAS ISO 25000*, n.d.). En la Tabla 4 se muestra la composición de SQuaRE:

Tabla 4

Modelo de calidad ISO/IEC 25000

Numeración de la ISO	Nombre de la ISO	Descripción de la ISO
ISO/IEC 2500n	División de gestión de calidad	Se definen todos los modelos comunes, términos y referencias
ISO/IEC 2501n	División del modelo de calidad	Modelo de calidad detallado, características para la calidad interna, externa y en uso
ISO/IEC 2502n	División de mediciones de calidad	Modelo de referencia de calidad del producto de software. Aplicaciones de métricas para la calidad interna, externa y en uso
ISO/IEC 2503n	División de requisitos de calidad	Especificación de requisitos de calidad. Se guía en la norma ISO/IEC 15288
ISO/IEC 2504n	División de evaluación de la calidad	Requisitos, recomendaciones y guías para la evaluación de un producto

Numeración de la ISO	Nombre de la ISO	Descripción de la ISO
ISO/IEC 25050 – 25099	Estándares de extensión de SQuaRE	Requisitos para la calidad de productos de software “Off-The-Self”

Nota. Esta tabla describe las distintas normas ISO que componen a la ISO/IEC 25000 y los estándares SQuaRE

Es en la norma ISO/IEC 25010, donde se definen algunos modelos de calidad y se han elegido varias características y factores para ser evaluados refiriéndose a la calidad de un proyecto de software, como se detalla en la Figura 2.

Figura 2

Factores de Calidad de Software en ISO/IEC 25010

Nota. El gráfico representa las distintas características y factores de la norma ISO/IEC 25010. Tomado de Certificación de la Mantenibilidad de Producto de Software: Un Caso Práctico (p.3), por Rodriguez, Pedreira, & Fernández, 2015, Revista Latinoamericana de Ingeniería de Software.

Para este trabajo, nos centraremos en el aspecto de rendimiento o performance. El rendimiento es la característica que representa al desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones, como el tiempo de respuesta, la cantidad de contenido cargado en cierto tiempo o el retraso del primer input. También es importante el rendimiento del desarrollador.

Rendimiento de un Desarrollador en un Proyecto de Software

El desempeño de un desarrollador de software representa la vida o muerte de un proyecto o innovación, siendo necesario introducir evaluaciones durante el desarrollo. Esto representa un problema puesto la complejidad de los procesos desarrollo de software no es baja, siendo mutable y extensible (Chen, 2008). Por esta razón las métricas de evaluación de desempeño no pueden ser reutilizadas o adaptadas de otros lugares, necesitando nuevos métodos específicos para cada caso.

Una de las teorías que se mantiene dice que para mejorar el rendimiento de los desarrolladores, es necesario enfocarse en la persona antes que en el programa, debido a que la persona es el elemento más volátil dentro de cualquier proyecto (Boehm & Papaccio, 1988). Para atacar este elemento varias empresas y compañías ofrecen incentivos a manera de bienes materiales, como dinero, reconocimientos, vacaciones, pero este solo es un aspecto entre los factores que afectan el rendimiento de un desarrollador.

Existe un nivel de desconocimiento sobre los métodos de evaluación para el desempeño de los desarrolladores y proyectos de software, incluso al interior de la misma comunidad, ni los desarrolladores saben cómo medir su rendimiento. Muchas veces esto es aplacado mediante el uso del control de calidad, pero normalmente es limitado al rendimiento del aplicativo, mas no al de la persona (Chen, 2008).

(Graziotin et al., 2014) sostiene que la emoción y el ánimo están fuertemente relacionados con el desempeño de un desarrollador. Estos estímulos se presentan en la vida personal, profesional,

además de factores psicológicos y físicos que pueden parecer insignificantes, pero impactan de manera palpable en el rendimiento del desarrollador.

La libertad de trabajo y opinión ha mostrado un aumento en el desempeño de los desarrolladores, puesto que permite al desarrollador sentirse integrado con la empresa y su trabajo. Al ocurrir este fenómeno las obligaciones del desarrollador llegan a tener un vínculo personal, siendo asociado como un proyecto propio, motivando al cumplimiento y perfeccionamiento del mismo (Graziotin et al., 2014).

Rendimiento de una Aplicación de Software

(Smith & Williams, 1993) define al rendimiento como un aspecto importante, pero muchas veces ignorado en las metodologías de desarrollo de software. El rendimiento se refiere a la sensibilidad de un sistema cada vez que se requiere una respuesta a un evento específico, o al número de eventos procesadores en un intervalo de tiempo dado. Es por esto, que el desempeño de software es una cualidad persistente difícil de entender, debido a que es afectada en cada aspecto del diseño, el código y el ambiente de ejecución (Woodside et al., 2007).

Los enfoques de ingeniería de rendimiento para predicción, creación de perfiles y las pruebas de carga proporcionan herramientas poderosas para desarrolladores y especialistas de operaciones, para investigar las propiedades de tiempo de ejecución relacionadas con el tiempo y los recursos de los sistemas de software. Los datos registrados y los hallazgos derivados suelen ser complejos y extensos. Si bien las herramientas existentes brindan apoyo para el análisis de datos e informan sobre problemas potenciales, aún se requiere que el desarrollador realice revisiones manuales para asegurar una mayor calidad del producto software (Srivastava et al., 2021).

La estimación del rendimiento en las primeras etapas del desarrollo de software es difícil, ya que hay muchos aspectos que pueden ser desconocidos o inciertos, como las decisiones de diseño, el código

y el entorno de ejecución. En muchos dominios de aplicaciones, como sistemas distribuidos heterogéneos, aplicaciones empresariales y computación en la nube, la evaluación del rendimiento también se ve afectada por factores externos, como cargas de trabajo cada vez más fluctuantes y escenarios cambiantes. Como resultado, el ingeniero de software a menudo se ve obligado a tomar decisiones bajo incertidumbre (es decir, relacionadas con el diseño o la refactorización del sistema), sin conocer el impacto exacto de esas decisiones en el rendimiento (Aleti et al., 2018).

Desde una perspectiva histórica, el rendimiento del software se abordó inicialmente tratando de exportar modelos y medidas de rendimiento del dominio del hardware al dominio del software. Esto fue bastante sencillo al considerar la capa del sistema operativo, pero ha asumido una nueva dimensión cuando el enfoque está orientado a las aplicaciones de software.

Además, con el aumento de la complejidad del software, se reconoció que el rendimiento del software no podía enfrentarse localmente a nivel de código mediante el uso de técnicas de optimización, ya que los problemas de rendimiento a menudo resultan de las primeras elecciones de diseño. Esta conciencia impulsó la necesidad de anticipar el análisis de rendimiento en las primeras etapas del desarrollo de software (Cortellessa et al., 2011).

Hoy en día, se prefiere en gran medida un enfoque de rendimiento de software a uno de rendimiento de sistema, debido a la complejidad de los sistemas de software. Las soluciones de software, de hecho, contribuyen a administrar (y posiblemente disminuir) de manera efectiva tal complejidad al buscar alternativas de software que exploten mejor la plataforma subyacente. Por lo tanto, cuando se evidencia un problema de rendimiento, siempre se deben buscar primero soluciones factibles de software, ya que suelen ser menos costosas y contribuyen a actuar sobre la complejidad del software.

Adicionalmente, tales soluciones sobreviven en caso de portabilidad de software, a diferencia de soluciones de sistema vinculadas a una plataforma específica. Obviamente, hay situaciones en las que ninguna solución de software elimina eficazmente un problema de rendimiento. En estos casos, como última posibilidad, se debe adoptar una solución de sistema, si es factible (Cortellessa et al., 2011).

Métricas Web Vitals

Web Vitals es una iniciativa de Google para proporcionar una guía unificada de indicadores que son esenciales para brindar una excelente experiencia de usuario en la web, mediante una serie de herramientas se puede medir y generar informes de rendimiento de aplicaciones web, mediante las métricas conocidas como Core Web Vitals (*Web Vitals*, 2023).

Core Web Vitals son un subconjunto de Web Vitals que se aplica a todas las páginas web, cada uno de estos representa una faceta distinta de la experiencia del usuario que se puede medir y refleja la experiencia en el mundo real de un usuario. Estas métricas evolucionarán con el tiempo, actualmente se centran en tres aspectos: carga, interactividad y estabilidad visual (*Web Vitals*, 2023), siendo representadas por estas métricas en la Figura 3:

Figura 3

Métricas Core Web Vitals

Nota. Las tres métricas Core Web Vitals representadas con sus escalas respectivas. Tomado de Web Vitals, 2023 https://web.dev/vitals/

Donde:

- Largest Contentful Paint (LCP): Es el tiempo usado para desplegar el componente o contenido más grande dentro de una aplicación web. Esto sirve para medir el rendimiento de carga, siendo un tiempo de 2.5 segundos el tiempo recomendado (*Web Vitals*, 2023).
- First Input Delay (FID): Es la demora existente hasta el primer ingreso de un input. Mide la interactividad y se recomienda un FID de 100 milisegundos como tiempo recomendado (*Web Vitals*, 2023).
- Cumulative Layout Shift (CLS): Es tiempo usado para hacer un cambio en el diseño de una aplicación web. Mide la estabilidad visual y se recomienda mantener un CLS de menos de 0,1 (*Web Vitals*, 2023).

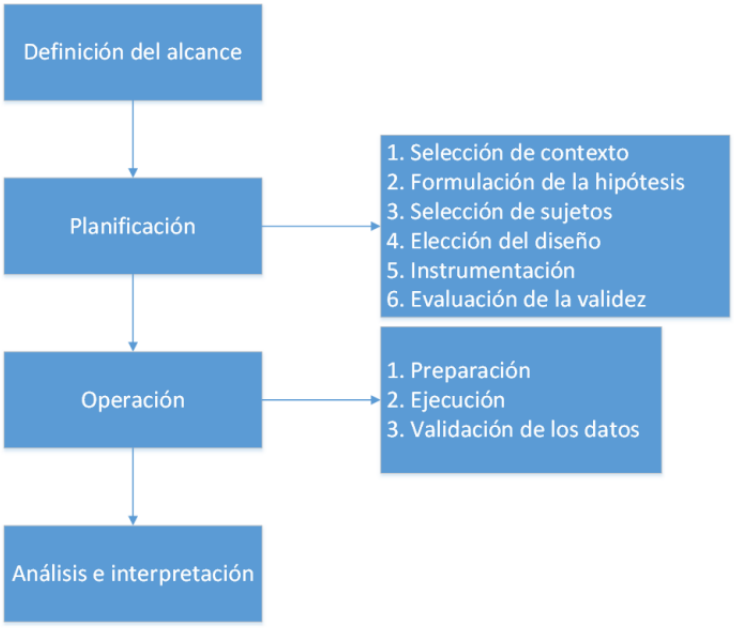
Otra métrica para considerar son las denominadas best practices, en el sentido de programación web, se entiende best web practices como buenas prácticas de programación que aseguran un buen producto de software. Las buenas prácticas han sido definidas por programadores experimentados desde el punto de vista de la calidad de código y del rendimiento del producto (*Web Vitals*, 2023).

Proceso Experimental

El proceso de un experimento requiere un nivel de consistencia alto de acuerdo con los objetivos planteados y su teoría. Para garantizar este aspecto se ha hecho uso de una guía metodología como es “Métodos de investigación en ingeniería del software” (Genero Bocco et al., 2015). Este proceso puede ser visualizado en la Figura 4.

Figura 4

Proceso Experimental



Nota. Representación gráfica del proceso experimental a seguir.

Definición del Alcance

Esta definición se refiere a el enfoque principal del experimento, mediante el cual se alcanzará los objetivos de la investigación, y por lo tanto las hipótesis plantadas.

Para este tipo de investigación existen varias pautas metodológicas ya establecidas, para el presente trabajo se hará usó de la guía Goal-Question-Metric (Basili et al., 1994), debido a la necesidad de comparar resultados basados en métricas.

Planificación

En la planificación se contemplan varias tareas que serán un coadyuvante para que se mantenga clara la idea del desarrollo del experimento, estas son:

Selección del Contexto

Se establece el entorno sobre el que el experimento será realizado, y con esto se busca decidir sobre qué condiciones los resultados del experimento serán válidos. Por lo general el entorno del experimento está compuesto por cuatro aristas diferentes, según (Genero Bocco et al., 2015) estas son:

- Online vs Offline.
- Profesionales vs Estudiantes.
- Problemas de "juguete" o proyectos reales.
- Especifico o general.

Formulación de la hipótesis

Dentro de un trabajo experimental se tiene que establecer lo que se intentara probar, esto se realiza formulando una hipótesis. Normalmente se inicia a partir de las llamadas hipótesis nulas, estas pueden ser una única o varias, mediante esto lo que se busca es rechazar estas, para así dar validez al experimento.

Selección de variables

Existen varios tipos de variables que se deben tomar en cuenta durante un experimento, las cuales son:

- **Variables Independientes:** Pueden ser alteradas estas sirven para analizar el efecto que producen las alteraciones

- **Variables Dependientes:** son observables y por lo general se utilizan para visualizar que efectos produjo el cambio en las variables independientes.
- **Variables Controladas:** estas son similares a las variables independientes solo que estas pueden ser manipuladas de manera específica.
- **Variables Enmascaradas:** variables que no pueden ser controladas y también pueden producir cambios en las variables dependientes.
- **Variables Aleatorias:** variables que pueden ser consideradas como errores aleatorios, vale decir que estas no son controlables

Según (Genero Bocco, 2015) establecer las variables también implica que también se debe establecer las distintas métricas y los valores que se emplearán. Siempre considerando que se debe evitar que las variables no controladas no generen un efecto dentro del experimento

Elección de Diseño

La elección del diseño experimental es un paso crucial durante la elaboración de un experimento ya que este es la base para la replicación y confirmación del experimento. En esta fase se dispone de diferentes métodos de control que ayudan a minimizar el error experimental (Genero Bocco et al., 2015).

- **Aleatorización:** Se refiere a la distribución aleatoria de los tratamientos y los sujetos que van a realizarlos, igualmente a la selección aleatoria de sujetos dentro de una muestra.
- **Bloqueo:** Permite eliminar el efecto no deseado de un factor dentro del experimento, de esta manera se evita sesgos al momento de procesar los datos obtenidos.
- **Balanceo o Equilibrado**: Útil al momento de la asignación de tratamientos, se cumple cuando a cada tratamiento se le asigna un número igual de sujetos.

Para establecer un correcto diseño experimental se debe considerar dos premisas fundamentales: el número de las variables independientes y la cantidad de los tratamientos que cada sujeto experimental será asignado.

Selección de Sujetos

La elección de los distintos sujetos se encuentra definida en una de las secciones que se abordaron en la selección del contexto. El sujeto es un individuo sobre el cual se ejecutara un experimento, así mismo esto influye en los resultados del experimento ya que este forma parte de la realización del mismo (Genero Bocco et al., 2015).

Instrumentación

La instrumentación intenta que se establezca una guía para ejecutar un experimento sin que se vea afectado el nivel de control que se tiene sobre el mismo.

La validez de los experimentos varía de acuerdo con como este haya sido establecida su instrumentación, dando como resultado que lo que se obtiene del experimento deba ser lo mismo independientemente de cómo se haya instrumentado. Ya que si los instrumentos hacen que el resultado del experimento varie, este no será válido (Jedlitschka & Pfahl, 2005).

Evaluación de la Validez

Cualquier experimento necesita ser validado antes de su ejecución, para conocer la probabilidad de incurrir en un sesgo que afectaría a sus resultados. De esta manera se puede planear tomando en cuenta los riesgos y reduciéndolos de manera considerable.

(Shadish et al., 2001) sugieren la existencia de tres tipos de amenazas en este tipo de investigación:

- **A la Validez Interna:** afectan el grado de confianza que se tiene en la relación causa-efecto que mantienen las variables independiente y dependiente.
- **A la Validez Externa:** afectan al grado en el cual los resultados pueden ser generalizados teniendo en cuenta la población seleccionada para el experimento. La validez externa de un experimento se ve afectada por: el diseño experimental, la selección de sujetos, el entorno y temporalidad del experimento.
- **A la Validez de Constructo:** la principal amenaza a la validez de constructo es la inexistencia de trabajos previos, los cuales ayuden a corroborar el diseño y resultados del experimento.
- **A la Validez de la Conclusión:** se determina mediante qué tan estadísticamente significativos fueron los resultados del experimento, además de la fiabilidad de las medidas.

Incluso con lo mencionado anteriormente, (Genero Bocco et al., 2015) explican que la mitigación total de las amenazas a un experimento es muy poco probable debido a la existencia de factores fuera del control de los investigadores, siendo necesario explicar cuáles han sido mitigadas durante el desarrollo del experimento.

Operación

Como su nombre lo indica, en esta fase se realiza la ejecución de la actividad propuesta juntamente con la recolección de los datos durante y después del mismo. Para esto se describen los siguientes pasos:

Preparación

Dentro de esta fase se realizan todas las actividades necesarias para la ejecución del experimento, es decir que principalmente se trabaja con los sujetos experimentales, de manera que estos estén preparados para realizar los tratamientos especificados.

Existen varias consideraciones éticas que se deben tomar en cuenta, ya que por lo general un experimento en ingeniería de software involucra seres humanos:

- Contar con el consentimiento de los sujetos.
- El rendimiento de cada sujeto debe ser confidencial.
- Se debe ofrecer un incentivo por realizar el experimento.
- Es necesario comunicar todos los aspectos del experimento, siempre y cuando estos no introduzcan sesgos en los resultados.

Además, en la fase de preparación se debe entrenar a los sujetos en el tema de estudio para que sean capaces de realizar los tratamientos que les serán asignados. También es recomendable pedir a los sujetos que llenen una encuesta sobre datos personales, experiencia y conocimientos previos. Así se tendrá un análisis más granular de los resultados (Genero Bocco et al., 2015).

Ejecución

Para la ejecución del experimento se debe realizar una planificación previa en la cual se visualice todos los puntos a tratar en el periodo de tiempo establecido. Una vez iniciado el experimento, para reducir amenazas a su validez, se debe realizar tanto la capacitación como la evaluación a todos los sujetos al mismo tiempo, de preferencia en una misma zona, donde los investigadores deberán comunicar el proceso a seguir, así como despejar dudas mientras se encuentre el experimento activo.

Validación de los Datos

Después de la recolección de los datos es necesario tomar en cualquier anomalía detectada al momento de la ejecución del experimento, de esta manera en la etapa de procesamiento se podrán filtrar estos datos de ser necesario.

Estado del Arte

La revisión de literatura acerca de la comparativa de frameworks, herramientas de Front End y conceptos de tecnologías web en general, se realizó basado en las guías de revisión sistemática de literatura propuestas por (B. A. Kitchenham, 2012). Se considera las actividades descritas a continuación:

Planteamiento de la Revisión de Literatura

Considerando los objetivos de la investigación, así como su problemática, hace falta la realización de una indagación sobre artículos e investigaciones científicas relacionadas con herramientas de Front End y comparativas de, definir un objetivo de búsqueda y plantear preguntas de investigación.

Construcción de la Cadena de Búsqueda

Para completar la construcción de cadena de búsqueda se consideraron varias combinaciones de palabras clave que permitieran encontrar la mayor cantidad de artículos con una relación al tema a investigar, se conformó la siguiente cadena de búsqueda: (FRAMEWORKS) AND ((FRONT END) OR (SOFTWARE COMPARISON) OR (PERFORMANCE) OR (WEB TECHNOLOGIES)) AND ((FRONT END TOOLS) OR(MEASUREMENT) AND (SOFTWARE). Esta cadena está enfocada a los principales estudios de tecnologías de desarrollo web modernas, de comparación de frameworks y los procedimientos utilizados para determinar las diferencias entre las herramientas en estudio.

Elaboración del Estado del Arte

La tecnología web ha tenido un gran impacto en la vida diaria de los desarrolladores, enfocándose especialmente en la educación profesional, abriendo posibilidades tanto para estudiantes como para profesores/instructores, debido a la adquisición del conocimiento facilitada por el acceso a bibliotecas virtuales (Sh.A.Abduraxmanova & Jo‘rayev, 2022). Además de las bibliotecas, Abduraxmanova et al., proponen el uso de Webinars para la educación, mediante el formato de conferencias varios expertos de un tema realizan disertaciones y presentaciones, se puede decir que, mediante el uso de estas tecnologías, la motivación y entendimiento de los estudiantes aumentan de manera significativa.

(Tekdal et al., 2018) en su estudio hace una recapitulación de las tecnologías de la web, desde la Web 1.0 hasta la Web 3.0 y sobre la futura Web 4.0 y 5.0, comparándolas y demostrando los avances que han existido. Así se puede ver como en la actualidad y a futuro se busca una conexión total de todos los dispositivos electrónicos con el Internet, en busca de facilitar cualquier proceso. Mediante una tabla anotan las propiedades comparativas usadas, demostrando los años que abarcan las distintas webs y cómo influyó esto en el ámbito educativo, concluyendo como el progreso de la tecnología web hará que la educación avance a la misma velocidad.

(Basili & Selby, 1991) marcan varios paradigmas a seguir al momento de desarrollar un experimento en la ingeniería de software, especificando pasos concretos a analizar en pos de facilitar el proceso. Especifican cómo dependiendo de la educación y el entrenamiento que haya tenido la persona a cargo del experimento, el resultado de este variará si no se tiene un método estandarizado.

Los pasos establecidos son: caracterizando el medio ambiente, el planeamiento, la ejecución, el análisis y el aprendizaje y feedback, especificando cómo esta estructura ayuda a tener un mejor desarrollo y resultados de un experimento.

(Pikkanen, 2021) en su de proyecto de graduación (tesis) realiza una comparativa entre los frameworks Vue y react, realizando dos tipos de frameworks, siendo una de ellas pruebas de Ranking en las cuales se usaron las métricas de lighthouse. Con esto obtiene resultados comparativos sobre rendimiento, capacidades progresivas de aplicaciones web y accesibilidad, que al ser comparados con el segundo tipo de pruebas arrojan resultados contradictorios. La conclusión a la cual se llega es que se necesita un estudio más exhaustivo con una mayor variación de sujetos de prueba para conseguir resultados más precisos. También se ve la posibilidad de realizar otro experimento usando métodos o herramientas que no dependan del sujeto de prueba.

(Mishra, 2021) explica cómo mediante el uso de frameworks de front end se puede mejorar problemas de complejidad al momento de desarrollar una página web. Es por esto por lo que apelan a tener un mayor entendimiento de las bondades de distintos frameworks y cómo escoger el correcto dependiendo de las características del proyecto a desarrollar ya que esto podría permitir una mayor accesibilidad. La revisión habla como es necesario conocer el desempeño y la escalabilidad que tendrá una aplicación web, realizando dos comparativas distintas, una basada en las respuestas por segundo mediante distintos formatos de datos y la otra comprando las mejores prácticas para el desarrollo web. Con esto concluye que hacer uso de mejores prácticas reduce el posible número de errores y da una mayor percepción a las características propias del framework para el beneficio del desarrollo.

(Ollila et al., 2022) menciona que entre más crezca la complejidad de una aplicación es más difícil usar APIs de navegador imperativas, para resolver este problema la comunidad de desarrollo sugiere el uso de frameworks. Basado en este argumento, realiza una comparativa de los principales frameworks de desarrollo basándose en sus estrategias de renderizado de contenidos. Al comparar los frameworks en varias situaciones, se encontró que estas diferencias teóricas se traducen directamente en diferencias en el desempeño práctico, a menudo provocando varias décimas de segundos de diferencia entre estrategias. Además, descubre diferencias significativas, incluso, en los costos fijos de

memoria y procesamiento en la creación de componentes y elementos, algo que cada framework debe realizar en la misma cantidad de procesamientos cuando se renderiza inicialmente una vista.

Ollila encuentra grandes diferencias entre los frameworks más populares para el front end, particularmente en la forma en que actualizan el contenido existente y la representación de contenido estático, llegando a la conclusión de que una mejor comprensión de las herramientas utilizadas para producir contenido, incluyendo sus características de desempeño, es una necesidad para asegurar que la Web sigue siendo una plataforma accesible para todos.

(Xu, 2021) propone dos experimentos para comparar varios frameworks de front end Vue, Angular, React y Svelte, el primero es desarrollar el mismo sitio web utilizando diferentes frameworks front-end, en base a la comparación del rendimiento usando métricas seleccionadas, el segundo experimento es una prueba de referencia que mide la duración de cada operación de DOM.

Como resultado los dos experimentos, el autor pudo concluir que Vue y Svelte brindan la experiencia de interacción de usuario más rápida al implementar la aplicación, por el tiempo usado en las operaciones del DOM. React por su parte no es tan rápido en estas operaciones puesto que se enfoca más en la estabilidad de las operaciones antes que en su rapidez.

React resultó ser el más lento en términos de tiempo dedicado a las diversas operaciones de DOM. Sin embargo, el análisis subyacente anterior sugiere que la razón por la que es más lento y estable es que el enfoque de React es optimizar la dinámica de JavaScript en lugar de mejorar la velocidad de cambios en el Virtual DOM. Angular, por otro lado, está en el medio del rango en términos de rendimiento de velocidad en ambos experimentos.

(Wohlgethan, 2018) encuentra que Angular y React difieren principalmente en su lenguaje de desarrollo y su filosofía de separación de archivos. Por otro lado, tienen la similitud de que el desarrollo de cada uno de ellos los patrocina grandes empresas de tecnología, lo que garantiza un cierto nivel de

confianza para la base de usuarios o aquellos que planean usarlos. Vue en este sentido es todo lo contrario, ya que se desarrolla teniendo en cuenta principalmente el interés de la comunidad. También tomó algunas de las mejores características de los frameworks ya existentes y logró crear algo nuevo que vio un enorme aumento en popularidad en 2017 y en curso.

(Flipse & van de Loo, 2018) consideran que las herramientas y métodos existentes el desarrollo web, por lo general no contienen explícitamente aspectos relevantes por la innovación responsable. Además, tanto las herramientas de control organizativo como las de calidad son inútiles sin comunicar sus implicaciones en la práctica diaria, retroalimentando las futuras opciones de investigación y desarrollo (I+D). Pero las herramientas destinadas específicamente a estimular y apoyar dicha comunicación funcional, sobre aspectos relacionados con la calidad, parecen estar mucho menos desarrolladas.

En este sentido se puede aprecias que existen varios trabajos en la actualidad que buscan comparar frameworks, enfocados en varios puntos de vista, con distintos objetivos como medir el que tiene mejor rendimiento en memoria, en cuanto a tiempo de respuesta haciendo uso de varios parámetros como son segundos de respuesta, uso de espacio en memoria, tiempo de renderizado, velocidad por parte de los desarrolladores. Este campo se encuentra en constante descubrimiento ya que el software siempre se actualiza, es por esta razón que buscamos un framework que permita comparar estos frameworks enfocado en el ámbito educacional, siendo esta la base para el experimento.

Capítulo III: Diseño y planificación del experimento

Una vez definida la teoría, es necesario planificar el experimento a realizar, para entender los resultados a obtener, así como las amenazas a la validez.

Definición del Alcance

El presente experimento tiene como objetivo comparar la usabilidad y el rendimiento de frameworks de desarrollo Web para el lado del cliente desde el punto de vista del desarrollador, mediante un experimento, con el propósito de verificar la facilidad de aprendizaje, de desarrollo, y el desempeño de las aplicaciones web desarrolladas usando estos frameworks, mediante métricas del desempeño de los desarrolladores, de percepción del aprendizaje y métricas Web Vitals. Esto, en el contexto de estudiantes de pregrado de las carreras de Ingeniería de Software y Tecnologías de la Información de la Universidad de las Fuerzas Armadas ESPE durante el semestre 202251 Oct2022-Feb2023.

Selección del Contexto

Objetos Experimentales

Para el presente experimento se han elegido dos objetos experimentales: dos interfaces de front end, dos servicios REST públicos como Back end que no serán modificados durante el experimento, Se realizarán distintos desarrollos de aplicaciones front-end por diferentes grupos de sujetos experimentales. La aplicación back-end será provista previamente y las encuestas se realizarán antes y después del desarrollo. Las aplicaciones front-end serán analizadas indistintamente por la herramienta Lighthouse.

- Back end 1: Servicio de RestAPI de acceso público Foursquare, el cual será consumido en los desarrollos front-end de Vue.
- Back end 2: Servicio de RestAPI de acceso público RickyMortyAPI, el cual será consumido en los desarrollos front-end de React.
- F_{vue}: Front end desarrollado utilizando el framework de Vue, con HTML5, CSS3 y JS.
- F_{React}: Front end desarrollado utilizando la librería de React, con HTML5, CSS3 y JS.

Cada front end, en conjunto con su respectivo back end forma una aplicación web SPA, con la cual se realizarán consultas a la API y mostrara la información consultada en la pantalla, en búsqueda el rendimiento de los desarrollos de acuerdo con las métricas de lighthouse explicadas previamente.

Sujetos Experimentales

Para la ejecución del experimento se obtuvo la participación voluntaria de treinta y nueve sujetos del departamento de ciencias de la computación ESPE con la distribución que se ve en la Tabla 5.

Tabla 5

Detalles de cursos participantes del experimento

Carrera	NRC	Nombre del curso	Número de alumnos	Número de créditos
Software	8017	Desarrollo Web Avanzado	11	6
Software	8018	Desarrollo Web Avanzado	10	6
TI (En línea)	8516	Desarrollo de Aplicaciones Web	18	4

Nota. Esta tabla describe la información detallada de los cursos a recibir el entrenamiento para el

Para la distribución de los sujetos disponibles para el experimento se analizaron sus niveles de conocimientos, así como el semestre en curso dentro de su respectiva carrera. Esto nos permite tener un grupo homogéneo al momento de realizar la capacitación y realización del experimento.

La participación de los sujetos en el experimento es voluntaria. Para evitar temor a la evaluación que pudiera ocasionar amenazas a la validez, se utilizan como base las medidas sugeridas por (Genero Bocco et al., 2015) las cuales consisten en comunicar a los estudiantes que su desempeño en el experimento no será calificado y únicamente, en las evaluaciones de la asignatura se incluirán temas relacionados al experimento.

Por otro lado, para que todos los sujetos compartan una base de conocimientos común, se planifica una capacitación acerca de ambos frameworks de front end y uso de las herramientas necesarias para ejecutar el experimento, tales como: Vue, React, JavaScript, buenas prácticas de programación, HTML5, CSS3, REST y JSON.

En el contexto de este experimento se considera apropiado trabajar con estudiantes (Höst et al., 2000) debido a que:

- La ejecución de las tareas propuestas no requiere de experiencia profesional.
- Cada grupo de estudiantes tiene un nivel de conocimientos homogéneo.
- Existe una mayor facilidad para monitorear el experimento.
- La gran cantidad de sujetos experimentales disponibles.

Finalmente, consideramos que, durante los primeros años de estudio en la carrera de Ingeniería de Software, el currículo impartido (Departamento de Ciencias de la Computación, 2020) está fuertemente orientado al desarrollo web usando HTML5, CSS3 y JavaScript, sin especificar ningún framework o biblioteca |Front End o Back End. Razón por lo cual, se espera que la mayoría de los sujetos

cuenten con los conocimientos necesarios en el desarrollo web para cumplir los requerimientos del experimento, al no tener mayor experiencia con los frameworks mencionados.

Selección de Variables

Variable Independiente

Elección del framework de desarrollo, la cual es una variable nominal que toma dos valores: Vue y React.

Variable Dependiente

- Rendimiento de los desarrolladores en los laboratorios propuestos al final del entrenamiento, haciendo uso de la siguiente fórmula creada para este experimento, donde el rendimiento del desarrollador está en base al número de funciones desarrollados y al tiempo utilizado para cumplir con los requerimientos:

$$R_{dev-fw} = C * \frac{F_{com} * T_{ttl}}{F_{exi} * T_{usd}}$$

  Donde:

  - R_{dev-fw} es el rendimiento del desarrollador en función del framework usado.
  - C es el condicional si el laboratorio se completó en el tiempo estimado.
  - F_{exi} es el número de funciones existentes.
  - F_{com} es el número de funciones completadas por el desarrollador.
  - T_{ttl} es el tiempo total para completar las funciones solicitadas.
  - T_{usd} es el tiempo usado en las funciones completadas.
- Facilidad de aprendizaje percibida por los desarrolladores luego de recibir el entrenamiento en uno de los frameworks, basándose en las respuestas recibidas en encuestas pre y post entrenamiento, usando la escala de Likert de 5 puntos.

- Rendimiento de un proyecto de software medida respecto al uso de los frameworks de front end, utilizando las métricas Web Vitals explicadas anteriormente:
    1. Largest Contentful Paint (LCP): Esta métrica está relacionada con el tiempo usado por la aplicación web para mostrar el componente o contenido más grande.
    2. First Input Delay (FID): Esta métrica está relacionada con la demora existente hasta el ingreso del primer input.
    3. Cumulative Layout Shift (CLS): Esta métrica está relacionada con el tiempo utilizado en hacer un cambio en el diseño de una aplicación web.

Formulación de Hipótesis

En la Tabla 6 se describen las hipótesis nulas y alternativas relacionadas con la variables dependiente e independiente, el objetivo del análisis estadístico será rechazar las hipótesis nulas y posiblemente aceptar alguna de las hipótesis alternativas, las cuales tienen como objetivo establecer las diferencias en los aspectos de aprendizaje del framework, rendimiento del desarrollador y rendimiento de las aplicaciones desarrolladas con distintos frameworks de front end.

Tabla 6

Hipótesis nulas y alternativas del experimento

Hipótesis Nula	Hipótesis alternativas
$H_{0,1}$: $LCP_{React} = LCP_{Vue}$	$H_{1,1,1}$: $LCP_{React} \neq LCP_{Vue}$ $H_{1,1,2}$: $LCP_{React} > LCP_{Vue}$ $H_{1,1,3}$: $LCP_{React} < LCP_{Vue}$
$H_{0,2}$: $FID_{React} = FID_{Vue}$	$H_{1,2,1}$: $FID_{React} \neq FID_{Vue}$ $H_{1,2,2}$: $FID_{React} > FID_{Vue}$ $H_{1,2,3}$: $FID_{React} < FID_{Vue}$
$H_{0,3}$: $Performance_{React} = Performance_{Vue}$	$H_{1,3,1}$: $Performance_{React} \neq Performance_{Vue}$ $H_{1,3,2}$: $Performance_{React} > Performance_{Vue}$ $H_{1,3,3}$: $Performance_{React} < Performance_{Vue}$
$H_{0,4}$: $Dev\ Performance_{React} = Dev\ Performance_{Vue}$	$H_{1,4,1}$: $Dev\ Performance_{React} \neq Dev\ Performance_{Vue}$ $H_{1,4,2}$: $Dev\ Performance_{React} > Dev\ Performance_{Vue}$ $H_{1,4,3}$: $Dev\ Performance_{React} < Dev\ Performance_{Vue}$
$H_{0,5}$: $Learning_{React} = Learning_{Vue}$	$H_{1,5,1}$: $Learning_{React} \neq Learning_{Vue}$ $H_{1,5,2}$: $Learning_{React} > Learning_{Vue}$ $H_{1,5,3}$: $Learning_{React} < Learning_{Vue}$

Nota. Esta tabla describe las distintas hipótesis nulas y alternativas.

Donde:

- $H_{0,1}$: $LCP_{React} = LCP_{Vue}$ es la hipótesis nula 1, donde el LCP de las aplicaciones desarrolladas con React es igual a el LCP de las aplicaciones desarrolladas con Vue.
- $H_{1,1,1}$: $LCP_{React} \neq LCP_{Vue}$ es una hipótesis alternativa, donde el LCP de las aplicaciones desarrolladas con React es distinto a el LCP de las aplicaciones desarrolladas con Vue.
- $H_{1,1,2}$: $LCP_{React} > LCP_{Vue}$ es una hipótesis alternativa, donde el LCP de las aplicaciones desarrolladas con React es mayor que el LCP de las aplicaciones desarrolladas con Vue.
- $H_{1,1,3}$: $LCP_{React} < LCP_{Vue}$ es una hipótesis alternativa, donde el LCP de las aplicaciones desarrolladas con React es menor que el LCP de las aplicaciones desarrolladas con Vue.
- $H_{0,2}$: $FID_{React} = FID_{Vue}$ es la hipótesis nula 2, donde el FID de las aplicaciones desarrolladas con React es igual al FID de las aplicaciones desarrolladas con Vue.
- $H_{1,2,1}$: $FID_{React} \neq FID_{Vue}$ es una hipótesis alternativa, donde el FID de las aplicaciones desarrolladas con React es distinto al FID de las aplicaciones desarrolladas con Vue.
- $H_{1,2,2}$: $FID_{React} > FID_{Vue}$ es una hipótesis alternativa, donde el FID de las aplicaciones desarrolladas con React es mayor que el FID de las aplicaciones desarrolladas con Vue.
- $H_{1,2,3}$: $FID_{React} < FID_{Vue}$ es una hipótesis alternativa, donde el FID de las aplicaciones desarrolladas con React es menor que el FID de las aplicaciones desarrolladas con Vue.
- $H_{0,3}$: $Performance_{React} = Performance_{Vue}$ es la hipótesis nula 3, donde el rendimiento de las aplicaciones desarrolladas con React es igual al rendimiento de las aplicaciones desarrolladas con Vue.
- $H_{1,3,1}$: $Performance_{React} \neq Performance_{Vue}$ es una hipótesis alternativa, donde el rendimiento de las aplicaciones desarrolladas con React es distinto al rendimiento de las aplicaciones desarrolladas con Vue.
- $H_{1,3,2}$: $Performance_{React} > Performance_{Vue}$ es una hipótesis alternativa, donde el

rendimiento de las aplicaciones desarrolladas con React es mayor que el rendimiento de las aplicaciones desarrolladas con Vue.

- $H_{1,3,3}$: Performance $_{React}$ < Performance $_{Vue}$ es una hipótesis alternativa, donde el rendimiento de las aplicaciones desarrolladas con React es menor que el rendimiento de las aplicaciones desarrolladas con Vue.
- $H_{0,4}$: Dev Performance $_{React}$ = Dev Performance $_{Vue}$ es la hipótesis nula 4, donde el rendimiento de los desarrolladores al resolver el laboratorio usando React es igual al rendimiento de los desarrolladores al resolver el laboratorio usando Vue.
- $H_{1,4,1}$: Dev Performance $_{React}$ ≠ Dev Performance $_{Vue}$ es una hipótesis alternativa, donde el rendimiento de los desarrolladores al resolver el laboratorio usando React es distinto al rendimiento de los desarrolladores al resolver el laboratorio usando Vue.
- $H_{1,4,2}$: Dev Performance $_{React}$ > Dev Performance $_{Vue}$ es una hipótesis alternativa, donde el rendimiento de los desarrolladores al resolver el laboratorio usando React es mayor que el rendimiento de los desarrolladores al resolver el laboratorio usando Vue.
- $H_{1,4,3}$: Dev Performance $_{React}$ < Dev Performance $_{Vue}$ es una hipótesis alternativa, donde el rendimiento de los desarrolladores al resolver el laboratorio usando React es menor que el rendimiento de los desarrolladores al resolver el laboratorio usando Vue.
- $H_{0,5}$: Learning $_{React}$ = Learning $_{Vue}$ es la hipótesis nula 5, donde el nivel de aprendizaje percibido por los desarrolladores al ser capacitados en React es igual al nivel de aprendizaje percibido por los desarrolladores al ser capacitados en Vue.
- $H_{1,5,1}$: Learning $_{React}$ ≠ Learning $_{Vue}$ es una hipótesis alternativa, donde el nivel de aprendizaje percibido por los desarrolladores al ser capacitados en React es distinto al nivel de aprendizaje percibido por los desarrolladores al ser capacitados en Vue.
- $H_{1,5,2}$: Learning $_{React}$ > Learning $_{Vue}$ es una hipótesis alternativa, donde el nivel de

aprendizaje percibido por los desarrolladores al ser capacitados en React es mayor que el nivel de aprendizaje percibido por los desarrolladores al ser capacitados en Vue.

- $H_{1,5,3}$: $Learning_{React} < Learning_{Vue}$ es una hipótesis alternativa, donde el nivel de aprendizaje percibido por los desarrolladores al ser capacitados en React es menor que el nivel de aprendizaje percibido por los desarrolladores al ser capacitados en Vue.

Elección del Diseño

Debido a la cantidad de sujetos participantes del experimento, y teniendo en cuenta la cantidad de variables independientes seleccionadas, se planteó los siguientes tratamientos:

- Tratamiento I: Ejecutar para el grupo de React las tareas de desarrollo y consumo especificadas en el instructivo de desarrollo.
- Tratamiento II: Ejecutar para el grupo de Vue las tareas de desarrollo y consumo especificadas en el instructivo de desarrollo.

Para los grupos de Desarrollo web Avanzado de la carrera Ingeniería de Software (21 sujetos), se utilizará un diseño intra-sujetos como lo describe (Dolado et al., 2003), con las siguientes características: al tener dos grupos separados de 11 y 10 sujetos, cada subgrupo empezó con un framework diferente, para después del tiempo necesario para el primer tratamiento, ser entrenados en el otro framework, resultando en que cada sujeto experimental realizará ambos tratamientos.

El experimento dura dos semanas. La primera semana el subgrupo I desarrollará el tratamiento I y el subgrupo II, el tratamiento II. La segunda semana los subgrupos realizarán el tratamiento restante que no hayan realizado.

Es necesario recalcar que fueron consideradas varias medidas propuestas por (Wohlin, 2014) para reducir el efecto de aprendizaje como evitar las opiniones personales de los experimentadores, no

mencionar el objetivo final del experimento y realizar el segundo tratamiento inmediatamente luego del primero que podrían experimentar los sujetos una vez ya desarrollados los laboratorios del primer framework. Las tareas y los frameworks se asignarán en orden aleatorio.

Para el grupo de Desarrollo Web de la carrera Ingeniería en Tecnologías de la Información de la modalidad en línea (18 sujetos), se usa un diseño balanceado inter-sujetos como en (Dolado et al., 2003), con las siguientes características: durante la tercera semana, el grupo se capacitará en uno de los frameworks escogido de manera aleatoria, para posteriormente en la cuarta semana aplicar el otro tratamiento.

La distribución de los sujetos, así como los periodos de tiempo de los tratamientos pueden ser observados en la Tabla 7.

Tabla 7

Distribución de los sujetos y tratamientos para el experimento

Semana 1 (12 al 16 de diciembre del 2022)	**Semana 2 (19 al 23 de diciembre del 2022)**	**Semana 3 (2 al 7 de enero del 2023)**	**Semana 4 (9 al 14 de enero del 2023)**
DWA (8017) – T1	DWA (8017) – T2	DAW – T2	DAW – T1
DWA (8018) – T2	DWA (8018) – T1		

Nota. Esta tabla describe las distribuciones realizadas en las semanas de entrenamiento.

Donde:

- DWA: Desarrollo Web Avanzado.
- DAW: Desarrollo de Aplicaciones Web.

- T1: Framework de Front end React.
- T2: Framework de Front end Vue.

Instrumentación

Para la valoración de las métricas de rendimiento del front end desarrollado por los sujetos se usará los siguientes instrumentos:

- Dos laboratorios front-end, cada uno utilizando uno de los frameworks propuestos: Vue y React, ambos empleando HTML5, CSS3 y JS.
- Dos REST Apis de acceso gratuito que servirán como fuentes de datos para los desarrollos de Front end.

Para cada versión de Front end se escribe un laboratorio prediseñado, las cuales permitirán comprobar si los desarrollos realizados por los sujetos son exitosos y cumplen con los requerimientos propuestos para las métricas del experimento.

El laboratorio que será usado para la recolección de los datos de las variables dependientes consta de las siguientes tareas:

- Crear entradas de datos (inputs) para hacer una búsqueda.
- Crear un componente para el formato de los datos a recibir.
- Crear una conexión con la API a consumir.
- Guardar los datos recibidos en un objeto para ser usado por el componente creado anteriormente (JSON).
- Mostrar la información conseguida en pantalla de acuerdo con el input de búsqueda.

Cada tarea tiene como único objetivo la creación de código front end. Finalmente, como parte de la instrumentación, se ejecutarán tres encuestas:

- Previo al entrenamiento, con módulos demográfico y de conocimiento previos.
- Post primer entrenamiento.
- Post segundo entrenamiento.

Las preguntas planteadas en la encuesta sirven como medida de la percepción de aprendizaje y hacen uso de una escala de Likert con un rango de 5 puntos. Esto nos ayuda a visualizar la relación entre el rendimiento y la percepción de aprendizaje de los desarrolladores.

Los enlaces a los repositorios de GitHub con las soluciones obtenidas por los sujetos y la solución propuesta por los capacitadores se encuentran en el Apéndice A, la encuesta preentrenamiento en el Apéndice B, la encuesta post entrenamiento en el Apéndice C, y la encuesta post experimento se encuentra en el Apéndice D.

Operación

Preparación

Para el desarrollo del experimento se preparó una lista que describe las actividades a realizar:

a) Solicitar a los sujetos que realicen la encuesta de conocimientos previos.
b) Realizar una capacitación durante una semana antes de la ejecución del experimento, en el cual los sujetos seguirán capacitados en cada framework front end, conceptos generales y herramientas necesarias.
c) Ejecutar el experimento, solicitando a los sujetos que realicen las tareas definidas en el instructivo previamente mencionado, para el framework correspondiente.
d) Una vez finalizado cualquiera de los tratamientos se pedirá que una encuesta diseñada específicamente para esa esta sea llenada.

Ejecución

Como parte de la ejecución del experimento se llevarán a cabo las siguientes actividades:

a) Se distribuirá el material necesario para realizar el desarrollo del experimento (entrenamiento previo, laboratorio detallado y API a usar).
b) Se distribuirá el material necesario para realizar el desarrollo del experimento (entrenamiento previo, laboratorio detallado y API a usar).
c) Se darán las indicaciones sobre la ejecución del experimento:
    - Durante el experimento no se permitirá la comunicación entre los sujetos.
    - Cualquier duda debe ser consultada con los experimentadores.
    - Cuando los experimentadores lo indiquen los sujetos podrán ejecutar empezar el desarrollo de la aplicación web.
    - Las tareas de desarrollo deberán ser realizadas en el orden detallado en su manual.
d) Una vez concluido el tiempo estimado para el experimento, los sujetos deberán subir sus cambios en el código a en su respectiva rama del repositorio GitHub.

Para evitar posibles sesgos en los resultados no se comunica a los sujetos las hipótesis bajo estudio, solo se les dice que el presente experimento es parte de una investigación sobre el desarrollo front-end en el ámbito académico.

Validación de los Datos

Cuando se complete el período de los tratamientos se juntarán todos los datos de acuerdo con las variables dependientes propuestas, usando las métricas y métodos de evaluación descritos anteriormente. Cabe destacar que cualquier actividad que no haya sido completada en su totalidad o de

manera distinta a la requerida, o datos atípicos encontrados en las encuestas, estos serán descartados sin ningún efecto negativo para el análisis estadístico.

Con todo lo anterior definido, podemos ejecutar el experimento siguiendo reglas y lineamientos que nos permitan recolectar datos de manera más segura y con un mayor nivel de confianza, por lo que en el siguiente capítulo se procede a detallar estos mismos, con su respectivo análisis estadístico.

Capítulo IV: Análisis e interpretación de resultados

En este capítulo se realizará el análisis de todos los datos obtenidos posterior a su limpieza, así como la interpretación de estos en relación con las hipótesis planteadas. Estos resultados se concentran en el desempeño y en la medición de sus métricas: FID, LCP, y CLS, adicionalmente, se midió las percepciones de los desarrolladores mediante encuestas. Se analizan las diferencias entre los frameworks a comparar (React y Vue), así también cómo la enseñanza de los distintos frameworks impactó en la muestra de estudio. A continuación, se mencionan los pasos que se siguieron para realizar el análisis de los resultados:

- Estudio de los datos estadísticos descriptivos para las métricas de la variable dependiente: las métricas de Web Vitals (LCP, FID, CLS).
- Comprobación del tipo de distribución de los datos por medio del test de normalidad Kolmogorov-Smirnov para decidir qué tipo de test estadístico utilizar: paramétrico o no paramétrico.
- Rechazar o aceptar de las hipótesis nulas de acuerdo con los resultados obtenidos de los análisis estadísticos.
- Comparación los resultados obtenidos de las encuestas para evaluar el impacto del entrenamiento y como se percibieron los distintos frameworks por los programadores.

Para la aceptación o rechazo de las hipótesis nulas se observó que los resultados arrojados de los test estadísticos lleguen a un nivel de significancia de 95% o más. Estos resultados fueron obtenidos mediante el uso del software estadístico SPSS.

Análisis de los Estadísticos Descriptivos, usando Web Vitals

Con el fin de obtener varias perspectivas sobre los resultados obtenidos se realizaron múltiples análisis, considerando distintos contextos como pueden ser general, por curso y por orden en que fueron aplicados los tratamientos.

Análisis General

En primera instancia, se muestran los estadísticos descriptivos generales en la Tabla 8 para las distintas métricas de Web Vitals que se utilizaron, aplicados a los distintos frameworks empleados en el experimento. Donde se observa que en general, las métricas de los programas desarrollados que se analizaron son superior en React. Es decir, el desempeño de aplicaciones web que se implementan usando React es mejor que las aplicaciones desarrolladas con Vue.

Tabla 8

Estadísticos generales descriptivos

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
Performance React	39	80	88	84,46	2,105
Performance Vue	39	61	75	66,33	3,002

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
FID React	39	0,50	0,73	0,6183	0,07551
FID Vue	39	0,50	0,69	0,6136	0,07632
LCP React	39	2,08	3,00	2,495	0,35544
LCP Vue	39	2,05	2,90	2,4754	0,32806
CLS React	39	0,05	0,10	0,0771	0,02545
CLS Vue	39	0,05	0,10	0,864	0,2335
Best Practices React	39	88	98	92,2917	2,5104
Best Practices Vue	39	90	96	91,9091	2,0226

Nota. Esta tabla muestra los estadísticos generales del desempeño de las aplicaciones web desarrolladas por todos los sujetos participantes

Análisis Por Métrica

En las Tabla 9 y Tabla 10, considerando las tendencias observadas podemos decir que en FID, teniendo en cuenta que un valor menor es mejor, React posee una ventaja en tiempo considerable frente a Vue. Los valores de LCP son un tanto similares, pero con mejores números aquí continúa sobresaliendo React con menos diferencia respecto a la anterior métrica.

Tabla 9

Estadísticos descriptivos de Vue

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
Performance	39	61	75	66,33	3,002
FID	39	0,70	1,11	0,8342	0,17362
LCP	39	2,80	3,70	3,3792	0,22063
CLS	39	0.05	0.10	0,0792	0,02518
Best Practices	39	90	100	95,542	1,0632

Nota. Esta tabla muestra los estadísticos del desempeño de las aplicaciones web desarrolladas en Vue por todos los sujetos participantes

Tabla 10

Estadísticos descriptivos de React

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
Performance	39	80	88	84,15	2,075

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
FID	39	0,50	0,73	0,6183	0,07551
LCP	39	2,08	3,00	2,4950	0,35544
CLS	39	0.05	0.10	0,0771	0,02545
Best Practices	39	88	98	92,2917	2,51049

Nota. Esta tabla muestra los estadísticos del desempeño de las aplicaciones web desarrolladas en React por todos los sujetos participantes

Análisis Por Orden de Capacitación

En las siguientes tablas se encuentran los estadísticos descriptivos por orden de capacitación, los cursos que primero tuvieron una capacitación con Vue y luego React, o viceversa. Esta agrupación se realizó para observar el efecto del aprendizaje que pueda experimentar el sujeto al conocer previamente una de las dos herramientas utilizadas en el experimento, esto es, el orden de la capacitación.

En la Tabla 11 y Tabla 14, se observó que quienes tuvieron su capacitación primero con React, al tener una capacitación posterior con Vue, tuvieron un mejor desempeño en su desarrollo a diferencia que quienes iniciaron su capacitación con Vue, contrastando con la Tabla 12 y Tabla 13, la tendencia indica que quienes empezaron su capacitación con Vue poseen un mejor uso de prácticas a diferencia de quienes iniciaron con React.

Tabla 11

Estadísticos de sujetos que tuvieron React como primer entrenamiento

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
Performance	13	80	87	84,15	2,075
FID	13	0,50	0,73	0,6223	0,07769
LCP	13	2,08	3,00	2,4754	0,35806
CLS	13	0,05	0,10	0,0692	0,02532
Best Practices	13	88,00	98,00	92,6154	2,90225

Nota. Esta tabla muestra los estadísticos del desempeño de las aplicaciones web desarrolladas en React por los sujetos que empezaron con React.

Tabla 12

Estadísticos de sujetos que tuvieron React como segundo entrenamiento

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
Performance	26	80	88	84,82	2,183

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
FID	26	0,50	0,73	0,6136	0,07632
LCP	26	2,10	3,00	2,5182	0,36829
CLS	26	0,05	0,10	0,0864	0,02335
Best Practices	26	90,00	96,00	91,9091	2,02260

Nota. Esta tabla muestra los estadísticos del desempeño de las aplicaciones web desarrolladas en React por sujetos que empezaron con Vue.

Tabla 13

Estadísticos de sujetos que tuvieron Vue como primer entrenamiento

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
Performance	26	62	75	66,55	3,643
FID	26	1,70	2,50	2,2545	0,22074
LCP	26	2,80	3,70	3,3364	0,25796
CLS	26	0,05	0,10	0,0909	0,02023

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
Best Practices	26	97,00	100,00	99,0909	1,04447

Nota. Esta tabla muestra los estadísticos del desempeño de las aplicaciones web desarrolladas en Vue por sujetos que empezaron con Vue.

Tabla 14

Estadísticos de sujetos que tuvieron Vue como segundo entrenamiento

Categorías	Casos Válidos	Valor Mínimo	Valor Máximo	Media	Desviación Estándar
Performance	13	61	70	66,15	2,478
FID	13	2,10	2,50	2,2769	0,13009
LCP	13	3,00	3,70	3,4154	0,18640
CLS	13	0,05	0,10	0,0692	0,02532
Best Practices	13	97,00	100,00	98,9231	1,11516

Nota. Esta tabla muestra los estadísticos del desempeño de las aplicaciones web desarrolladas en Vue por sujetos que empezaron con React.

Comprobación del tipo de distribución de los datos

Para discernir la distribución de los datos obtenidos y por consiguiente el tipo de prueba estadística a realizar, se hizo uso de la prueba de normalidad de Kolmogórov-Smirnov en las siguientes métricas: Performance, FID, LCP, CLS y Best Practices. Los resultados de la prueba de normalidad se presentan en la Tabla 15.

Tabla 15

Prueba de normalidad de una muestra de Kolmogórov-Smirnov

		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Cantidad		39	39	39	39	39	39	39	39	39	39
Parm nor.	Med.	66,33	2,266	3,379	,0792	99	84,46	0,618	2,495	0,077	92,29
	Desv est.	3,002	0,173	0,220	0,025	1,063	2,105	0,075	0,355	0,025	2,510
Dif Ext.	Abs.	0,128	0,184	0,246	0,379	0,243	0,185	0,100	0,189	0,358	0,213
	Pos.	0,128	0,174	0,167	0,293	0,173	0,107	0,100	0,189	0,315	0,213
	Neg.	0,081	0,184	0,246	0,379	0,243	0,185	0,085	0,138	0,358	0,139

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Kolmogórov-Smirnov Z	0,128	0,184	0,246	0,379	0,243	0,185	0,100	0,189	0,358	0,213
Nivel de significancia	0,200	0,035	<,001	<,001	<,001	0,124	,0200	0,027	<,001	0,006

Nota. Esta tabla muestra los resultados hallados luego de aplicar la prueba de normalidad de una muestra de Kolmogórov-Smirnov.

Donde

- **C1** es el performance de las aplicaciones desarrolladas en Vue
- **C2** es el FID de las aplicaciones desarrolladas en Vue
- **C3** es el LCP de las aplicaciones desarrolladas en Vue
- **C4** es el CLS de las aplicaciones desarrolladas en Vue
- **C5** es las best practices de las aplicaciones desarrolladas en Vue
- **C6** es el performance de las aplicaciones desarrolladas en React
- **C7** es el FID de las aplicaciones desarrolladas en React
- **C8** es el LCP de las aplicaciones desarrolladas en React
- **C9** es el CLS de las aplicaciones desarrolladas en React
- **C10** es las best practices de las aplicaciones desarrolladas en React
- **Par. nor.** son los parámetros normales considerados para el test estadístico
- **Med** es la media de los diferentes criterios usados para el test estadístico
- **Desv est.** es la desviación estándar de los criterios.

- **Dif. ext.** son las diferencias extremas halladas en los criterios.
- **Abs.** es la diferencia extrema absoluta.
- **Pos.** es la diferencia extrema positiva.
- **Neg.** es la diferencia extrema negativa.

El test estadístico se decidió en base a dos distintas premisas si los niveles de significancia eran menores a 0.05, esto permite determinar que los datos no siguen una distribución normal. Para la mayoría de las variables dependientes se consiguió un nivel de significancia menor al 0,05, por consiguiente, estos datos siguen una distribución normal.

El cálculo y valor que se va a considerar con un nivel de significancia del 95%, dado esto para contrastar las hipótesis se decidió realizar lo siguiente:

- Las medidas de rendimiento tienen una distribución no normal, siendo necesario el uso de un test no paramétrico para el procesamiento de sus datos. Para esta investigación se consideró el test de Wilcoxon para estos casos.
- Las medidas de FID, CLS, LCP y Best Practices siguen una distribución normal, por lo cual es necesario usar un test paramétrico para el procesamiento de sus datos. Para esta investigación se consideró la prueba T Student para estos casos.

De acuerdo con los resultados, se decidió usar los métodos previamente descritos tal como se detalla en la siguiente sección.

Pruebas de hipótesis

Pruebas de hipótesis relacionada al FID

La prueba T de Student para las medidas FID muestra los resultados detallados en la Tabla 16, que permiten rechazar la hipótesis nula $H_{0,2:}$ $FID_{React} = FID_{Vue}$, en vista de que el p-valor fue menor a

0.05. Esto, además, permite aceptar la hipótesis alternativa: $H_{1,2,3:}$ $FID_{React} < FID_{Vue,}$ de acuerdo a los valores descritos.

Tabla 16

Estadísticos de muestras emparejadas FID

	Media	Cantidad	Desviación estándar	Media de error estándar
FID Vue	2,2667	39	0,17362	0,03544
FID React	0,6183	39	0,07551	0,01541

Nota. Esta tabla muestra los estadísticos de muestras emparejadas del FID de todas las aplicaciones desarrolladas en cada tecnología.

Pruebas de hipótesis relacionada al LCP

La prueba T de Student para las medidas LCP muestra los resultados detallados en la Tabla 17, que permiten rechazar la hipótesis nula $H_{0,1:}$ $LCP_{React} = LCP_{Vue,}$ en vista de que el p-valor fue menor a 0.05. Esto, además, permite aceptar la hipótesis alternativa: $H_{1,1,3:}$ $LCP_{React} < LCP_{Vue,}$ de acuerdo con los valores descritos.

Tabla 17

Estadísticos de muestras emparejadas LCP

	Media	Cantidad	Desviación estándar	Media de error estándar
LCP Vue	3,3792	39	0,22063	0,04504
LCP React	2,4950	39	0,35544	0,07255

Nota. Esta tabla muestra los estadísticos de muestras emparejadas del LCP de todas las aplicaciones

Pruebas de hipótesis relacionada a Best Practices

La prueba T de Student para la medida de Best Practices muestra los resultados detallados en la Tabla 18, que permiten rechazar la hipótesis nula $H_{0,1:}$ BestPractices $_{React}$ = BestPractices $_{Vue,}$ en vista de que el p-valor fue menor a 0.05. Esto, además, permite aceptar la hipótesis alternativa: $H_{1,1,3:}$ BestPractices $_{React}$ < BestPractices $_{Vue,}$ de acuerdo con los valores descritos.

Tabla 18

Estadísticos de muestras emparejadas Best Practices

	Media	Cantidad	Desviación estándar	Media de error estándar
Best Practices Vue	99,0000	39	1,06322	0,21703
Best Practices React	92,2917	39	2,51049	0,51245

Pruebas con el desempeño

El test no paramétrico de Wilcoxon aplicado al rendimiento de las aplicaciones desarrolladas muestra los resultados descritos en la Tabla 19, donde se percibe que el nivel de significancia obtenido es menor al 0.05. Esto permite rechazar la hipótesis nula $H_{0,3}$: Performance $_{React}$ = Performance $_{Vue}$ y aceptar una de las hipótesis alternativas.

Tabla 19

Test estadístico de Wilcoxon para el performance de las aplicaciones desarrolladas

	Performance React – Performance Vue
Z	- 4,290
Nivel de Significancia	<0,001

Nota. Esta tabla muestra el resultado del test estadístico de Wilcoxon para los datos del rendimiento, los cuales no seguían una distribución normal.

Resultados de las encuestas

Análisis de las encuestas pre y post experimento

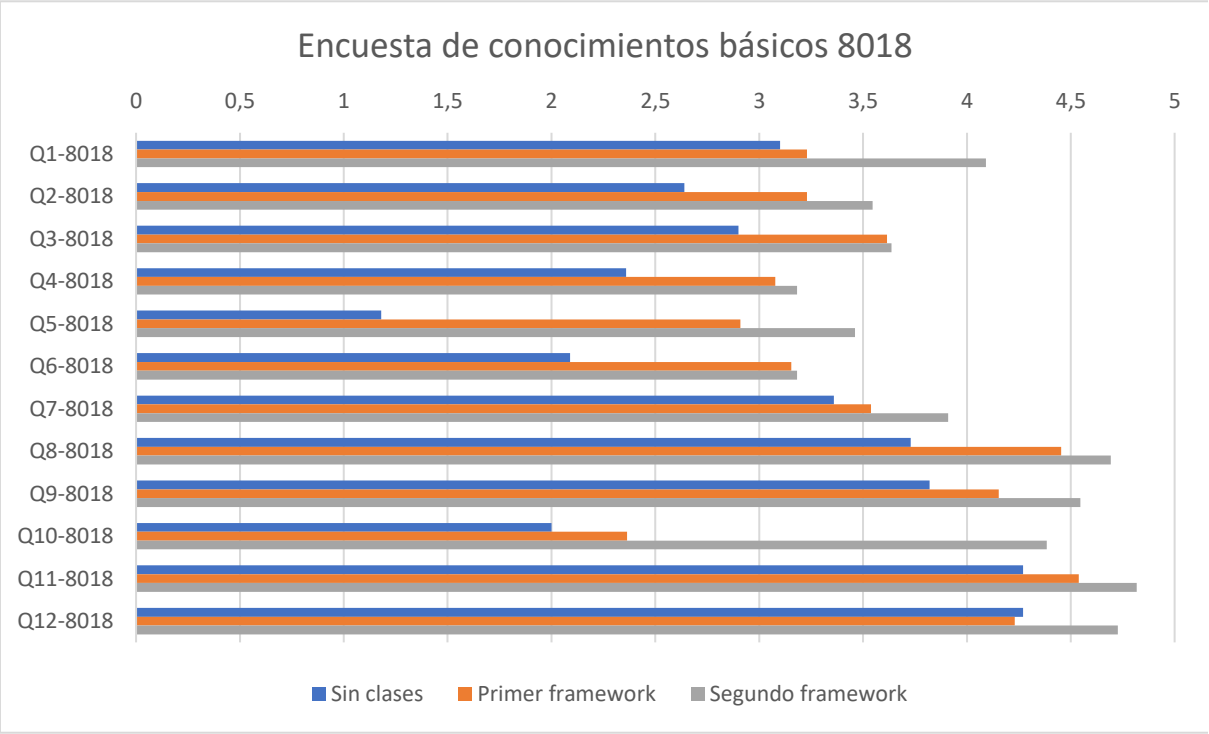
Una vez completado los tratamientos y las capacitaciones, mediante el uso de los valores llenados en las encuestas, se determina el impacto que tuvo este proceso en el nivel de conocimiento de los sujetos participantes, estos datos se encuentran en el Apéndice E.

Se puede observar en la Figura 5, Figura 6 y Figura 7 que después del entrenamiento todos los grupos de estudio experimentaron crecimiento en sus niveles de conocimiento, indicando un impacto positivo del entrenamiento en los sujetos experimentales.

La encuesta cubre el conocimiento previo de los frameworks, por ejemplo, esta cubre el conocimiento previo acerca de las bases de los frameworks, es decir, HTML, CSS y JavaScript, a más de conocimientos sobre rendimiento y acerca de Front End.

Figura 5

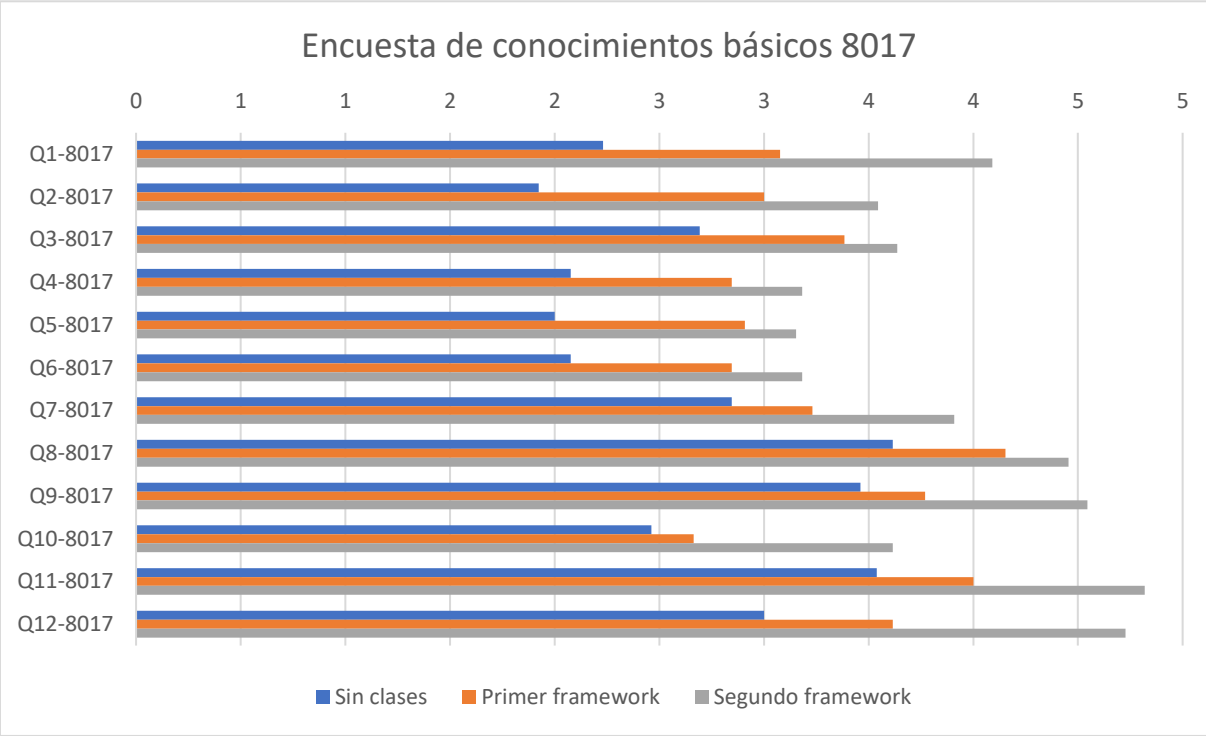
Encuesta de conocimientos previos del NRC 8018



Nota. Esta figura muestra la diferencia de los conocimientos básicos percibida por los sujetos experimentales del grupo bajo el NRC 8018.

Figura 6

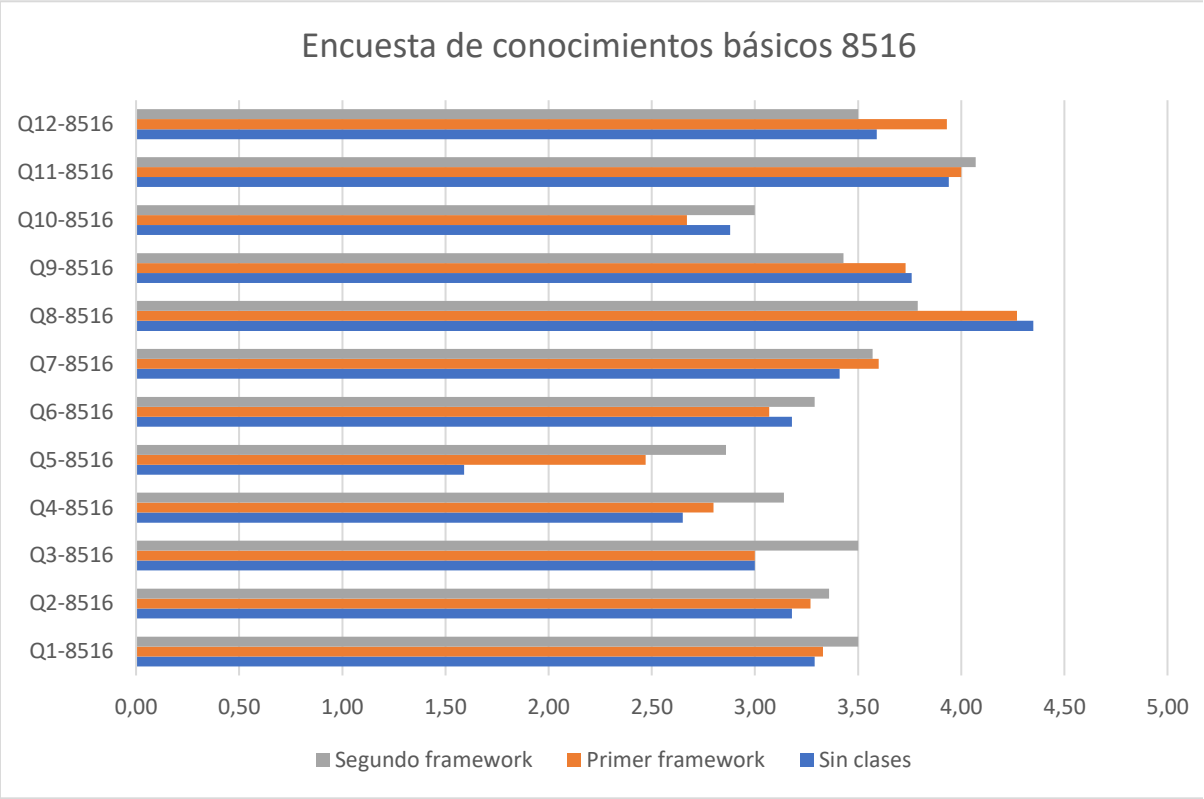
Encuesta de conocimientos previos del NRC 8017



Nota. Esta figura muestra la diferencia de los conocimientos básicos percibida por los sujetos experimentales del grupo bajo el NRC 8017.

Figura 7

Encuesta de conocimientos previos del NRC 8516

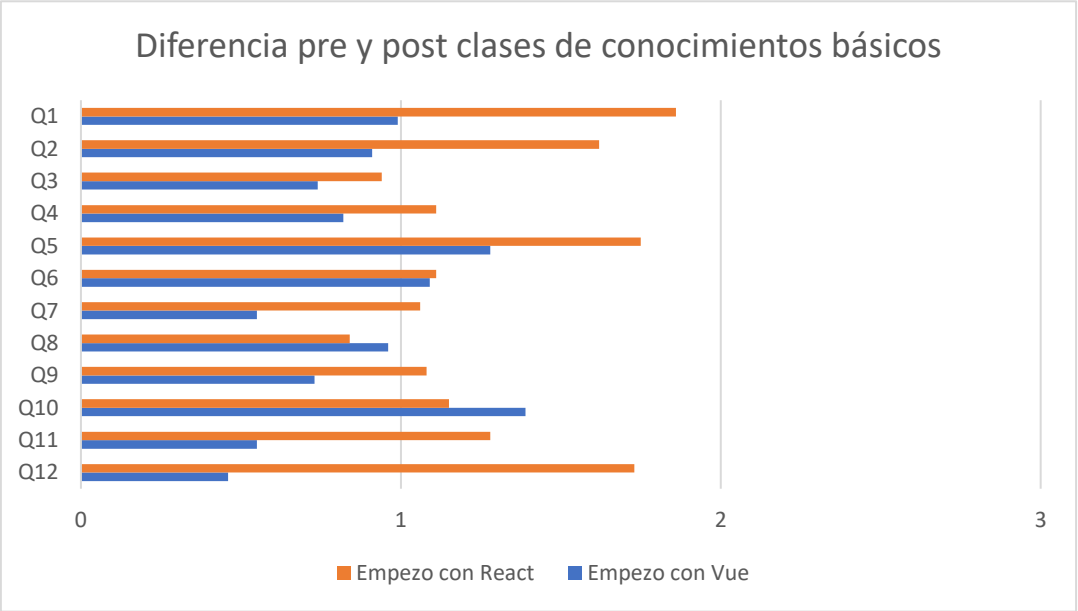


Nota. Esta figura muestra la diferencia de los conocimientos básicos percibida por los sujetos experimentales del grupo bajo el NRC 8516.

Para observar de manera más clara la influencia que ejerció el entrenamiento en el conocimiento de los sujetos, se realizó el cálculo de la diferencia entre los valores obtenidos pre y post experimento, considerando con cuál tratamiento iniciaron. Esto se detalla en la Figura 8.

Figura 8

Media de conocimientos pre y post entrenamiento



Nota. Esta figura muestra el crecimiento de los conocimientos básicos percibida por los sujetos

Análisis de la encuesta post experimento

El propósito de este análisis es hallar las diferencias percibidas en el aprendizaje de los sujetos en los distintos tratamientos y como estos pudieron y podrían influir en sus conocimientos y habilidades de desarrollo.

Esto permite rechazar la hipótesis nula, Ho: $\text{Aprendizaje}_{\text{React}} = \text{Aprendizaje}_{\text{Vue}}$ (No existe diferencia entre el aprendizaje de Vue y React). Para esto se hace uso de 8 preguntas de la encuesta aplicada post tratamiento.

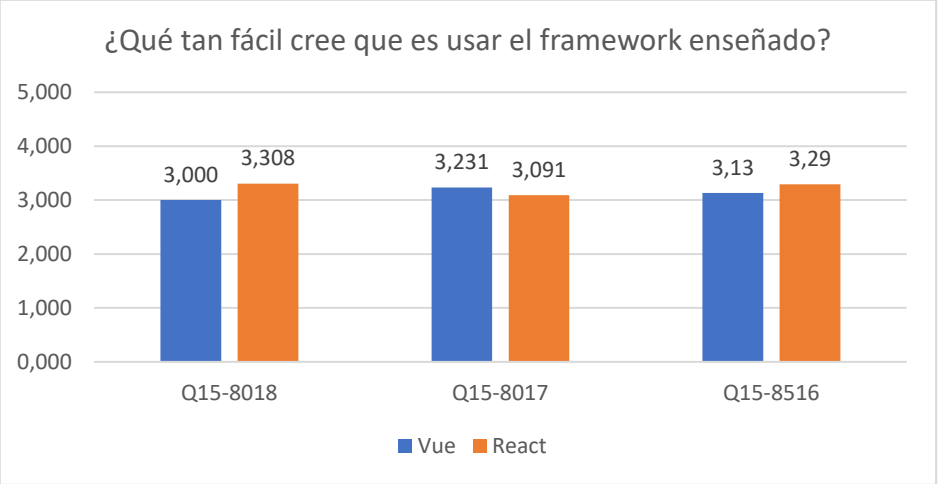
Pregunta 1: Acerca del nivel de facilidad de uso del framework enseñado

En esta pregunta "¿Qué tan fácil cree que es usar el framework enseñado?", donde el valor 5 representa el mayor nivel de facilidad percibida al usar el framework y el valor 1 representa el menor nivel de facilidad percibida al usar el framework por el sujeto experimental.

En la Figura 9 se observa que los sujetos consideran que React es un framework mucho más amigable y fácil de usar indiferentemente del orden de capacitación. Por otro lado, Vue fue considerado un framework no tan fácil de usar a diferencia de React.

Figura 9

Media de nivel de facilidad de uso de cada framework por curso



Nota. Esta figura muestra la diferencia de la facilidad de uso percibida por los sujetos experimentales de acuerdo con la tecnología usada.

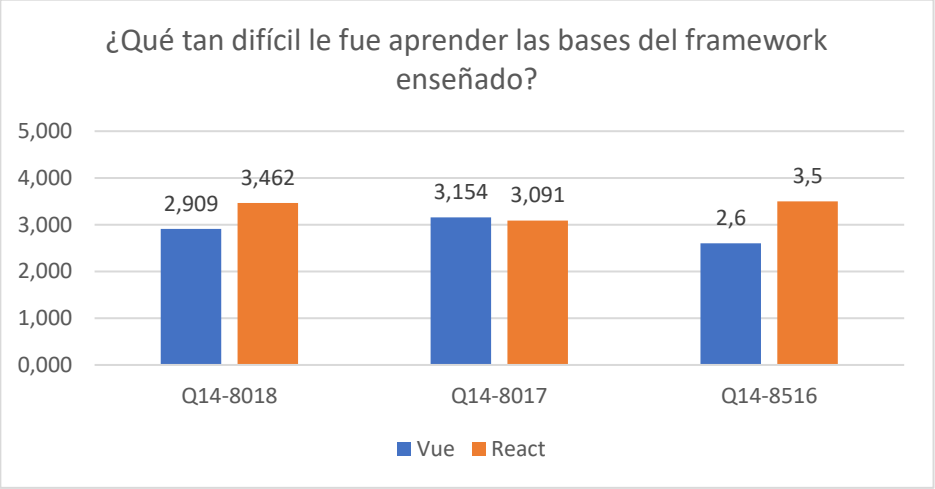
Pregunta 2: Acerca de la percepción de la dificultad de aprendizaje del framework enseñado

En esta pregunta "¿Qué tan difícil le fue aprender las bases del framework enseñado?", donde el valor 5 representa el menor nivel de dificultad de aprendizaje percibida al experimentar el framework y el valor 1 representa el mayor nivel de dificultad de aprendizaje percibida al experimentar el framework por el sujeto experimental.

En la Figura 10 se observa que los sujetos consideran que React es un framework mucho más amigable y fácil de aprender indiferentemente si este fue abordado antes o después de las primeras capacitaciones, aunque cabe recalcar que en uno de los cursos los valores son similares. Por otro lado, Vue fue considerado un framework no tan fácil de aprender a diferencia de React, vale considerar que la diferencia en ninguno de los casos de los distintos cursos es mayor a un punto.

Figura 10

Media de dificultad percibida en los frameworks por cada curso



Nota. Esta figura muestra la diferencia de la dificulta de uso percibida por los sujetos experimentales de acuerdo con la tecnología usada.

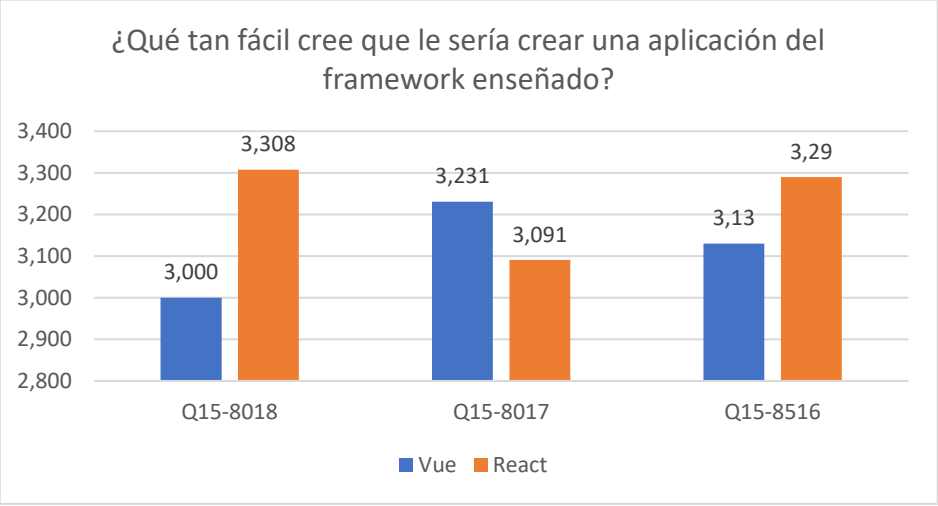
Pregunta 3: Acerca de la percepción de la facilidad de desarrollo del framework enseñado

En esta pregunta "¿Qué tan fácil cree que le sería crear una aplicación del framework enseñado?", donde el valor 5 representa el mayor nivel de facilidad de desarrollo percibida al usar el framework y el valor 1 representa el menor nivel de facilidad de desarrollo percibida al usar el framework por el sujeto experimental.

En la Figura 11 se observa que los sujetos consideran que React es un framework mucho más fácil de desarrollar, aunque cabe recalcar que en uno de los cursos los valores son mayores en Vue que en React. Por otro lado, Vue fue considerado un framework no tan fácil de desarrollar a diferencia de React, vale considerar que la diferencia en ninguno de los casos de los distintos cursos es mayor a un punto.

Figura 11

Media de facilidad percibida en los frameworks por cada curso



Nota. Esta figura muestra la diferencia de la facilidad percibida por los sujetos experimentales de acuerdo con la tecnología usada.

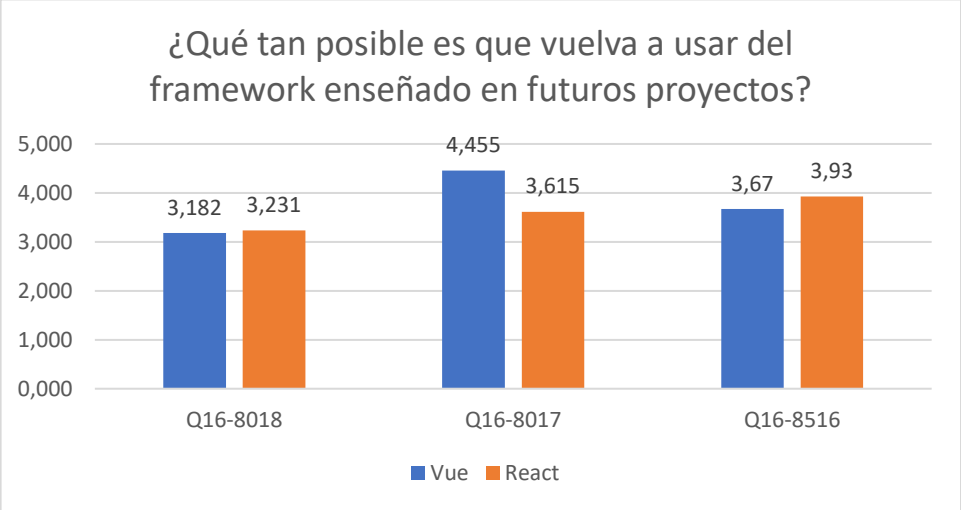
Pregunta 4: Acerca de la posibilidad de volver a usar el framework enseñado

En esta pregunta "¿Qué tan posible es que vuelva a usar el framework enseñado en futuros proyectos?", donde el valor 5 representa el mayor nivel de posibilidad de volver a usar el framework y el valor 1 representa el menor nivel de posibilidad de volver a usar el framework por el sujeto experimental.

En la Figura 12 se observa que los sujetos consideran que React es un Framework el cual volverían a utilizar, en ambos casos bastante parejas las cifras, pero si revisamos las cifras del curso que tuvo clases de Vue primero, ellos consideran a Vue su favorito para volver a utilizar.

Figura 12

Media de posibilidad de volver a usar el framework por curso



Nota. Esta figura muestra la diferencia de la posibilidad de volver a usar la tecnología enseñada de los sujetos experimentales.

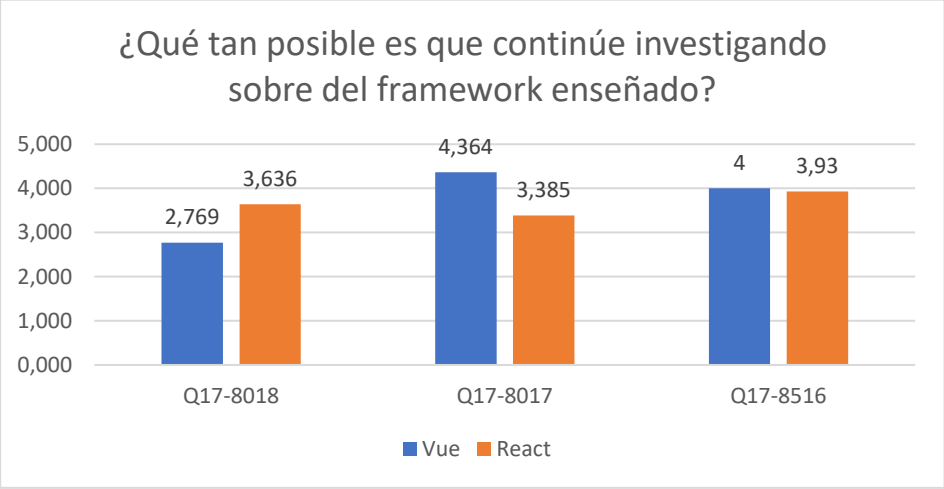
Pregunta 5: Acerca de la posibilidad de seguir investigando el framework enseñado

En esta pregunta "¿Qué tan posible es que continúe investigando sobre el framework enseñado?", donde el valor 5 representa el mayor nivel de posibilidad de investigar sobre el framework y el valor 1 representa el menor nivel de posibilidad de investigar sobre el framework por el sujeto experimental.

En la Figura 13 se observa que los sujetos consideran que React es un Framework el cual los desarrolladores desean seguir investigando, en ambos casos de los distintos cursos las cifras son un tanto parejas, pero si revisamos las cifras del curso que tuvo clases de Vue primero, ellos consideran a Vue su favorito para seguir investigando, así mismo con el curso que tuvo su primera capacitación con React.

Figura 13

Media de posibilidad de investigar el framework por curso



Nota. Esta figura muestra la diferencia de la posibilidad de investigar más sobre la tecnología enseñada por los sujetos experimentales.

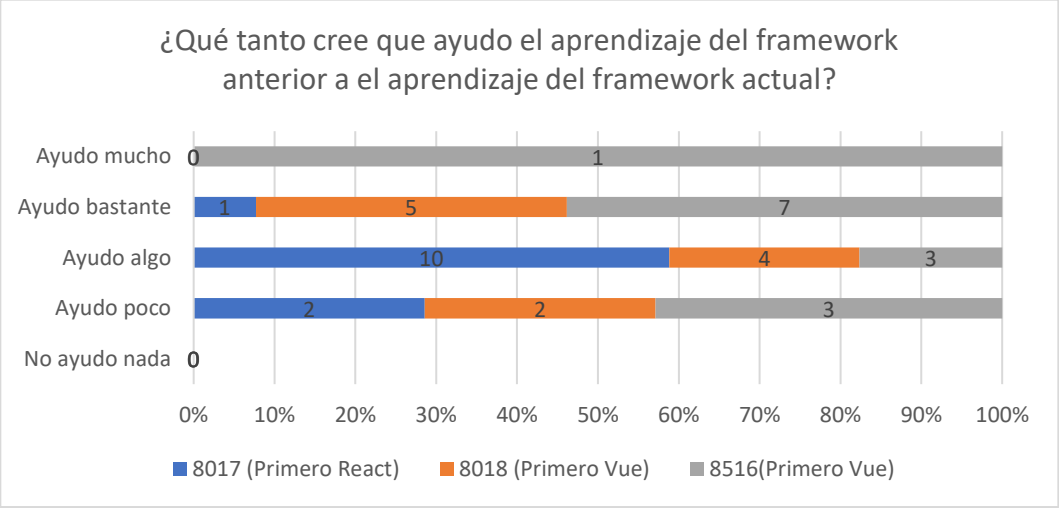
Pregunta 6: Acerca de qué tanto ayudó la primera capacitación a una mejor comprensión de la segunda capacitación

En esta pregunta "¿Qué tanto cree que ayudo el aprendizaje del framework anterior a al aprendizaje del framework actual?", los valores son mostrados de acuerdo con una escala en la que se encuentra caracterizado desde que ayudó mucho, hasta que no ayudó nada la capacitación del primer framework para entender el segundo framework.

En la Figura 14 se observa que los sujetos a los cuales se les preparó con React primero consideran que esta capacitación ayudó a la mejor comprensión del siguiente framework, en cambio los que fueron capacitados primero con Vue, mencionan que su comprensión mejoró en mayor forma teniendo Vue como antecedente a React.

Figura 14

Nivel de ayuda percibida al recibir una capacitación previa al segundo tratamiento.



Nota. Esta figura muestra la percepción del nivel de ayuda recibido para el segundo tratamiento por el primer tratamiento.

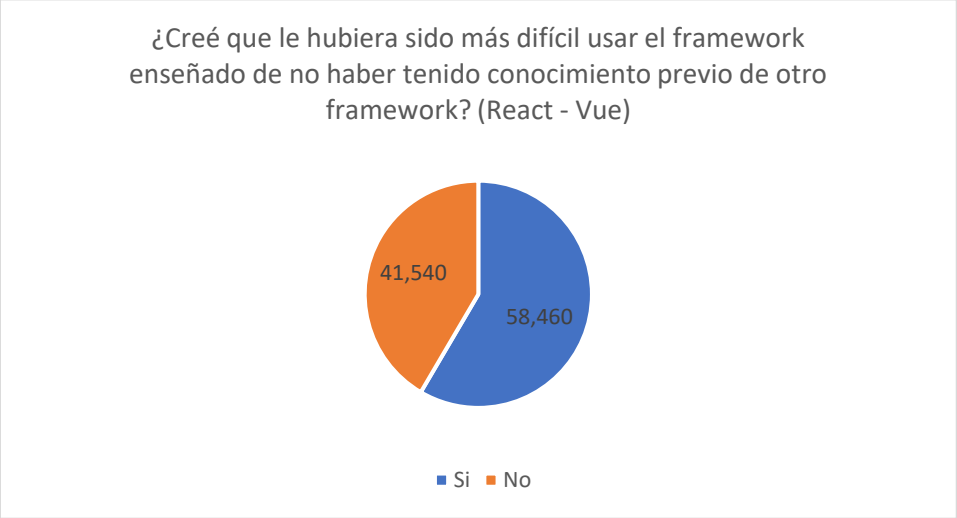
Pregunta 7: Acerca de qué tan difícil habría resultado usar otro framework de no haber tenido conocimiento

En la pregunta "¿Creé que le hubiera sido más difícil usar el framework enseñado de no haber tenido conocimiento previo de otro framework?", se realizó una medición en base a una respuesta de sí o no.

En la Figura 15 se observa que los sujetos, en su gran mayoría consideran que tener conocimiento en un framework de JavaScript les permite tener cierta familiaridad con otros frameworks, es por esto por lo que sin conocimiento usar un framework de JavaScript se les dificultaría.

Figura 15

Porcentaje de dificultad del segundo tratamiento



Nota. Esta figura muestra el porcentaje de los sujetos experimentales que piensan que el segundo tratamiento hubiera sido más difícil si no hubiera recibido la capacitación del primer tratamiento.

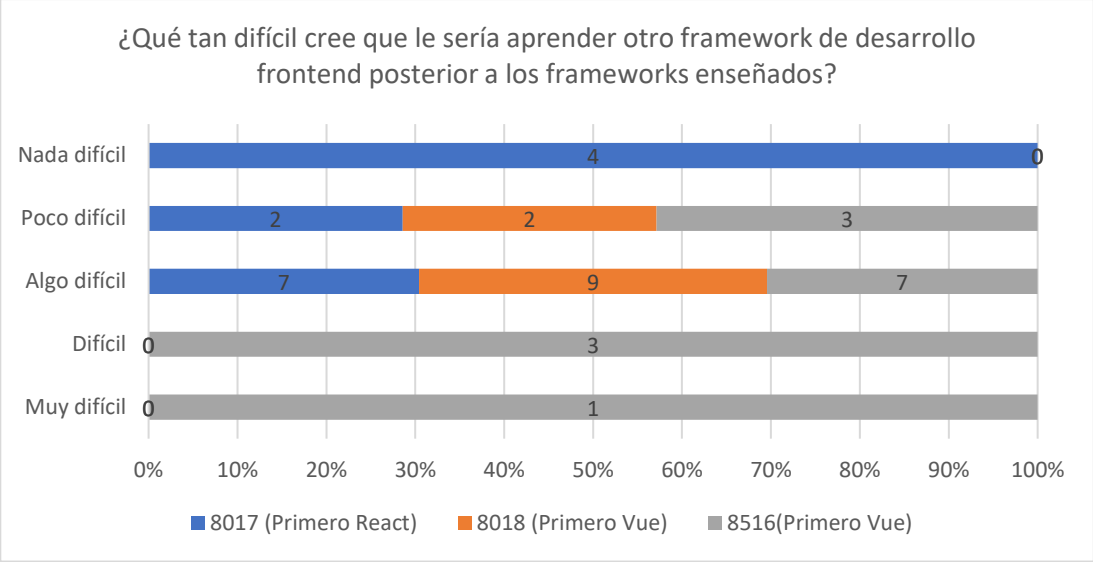
Pregunta 8: Acerca de qué tan difícil habría resultado aprender otro framework de después de las capacitaciones recibidas

En la pregunta "¿Qué tan difícil cree que le sería aprender otro framework de desarrollo front-end posterior a los frameworks enseñados?", se realizó una medición en base una escala que parte desde nada difícil hasta muy difícil.

En la Figura 16 se observa que los sujetos, en su gran mayoría consideran que posterior al conocimiento adquirido con las distintas capacitaciones el aprender un nuevo framework les resultaría algo difícil.

Figura 16

Nivel de dificultad para aprender nuevos frameworks post experimento



Nota. Esta figura muestra que tan difícil creen los sujetos que les será aprender un nuevo framework o tecnología post experimento.

Amenazas a la validez

Validez externa

La validez está directamente relacionada a la probabilidad de generalizar los resultados más allá de los frameworks usados. Existen varias amenazas en este aspecto, mismas que se describen a continuación:

- Los sujetos experimentales, al ser estudiantes en niveles superiores de las carreras de Ingeniería de Software e Ingeniería en Tecnologías de la Información pueden contar con conocimientos básicos sobre los frameworks usados en el experimento. Sin embargo, en el experimento se pudo constatar durante la capacitación que estos casos eran muy raros.
- Los laboratorios usados en los experimentos pueden no ser representativos de una aplicación real, debido a su enfoque en simplicidad.

Validez interna

Las amenazas a la validez interna fueron mitigadas dentro de las posibilidades de la investigación. Se usaron grupos de sujetos balanceados, seleccionados de manera aleatoria y con niveles de conocimiento similares. Los cuestionarios post experimento revelaron que no existió una diferencia significativa en la dificultad de comprensión del código percibida por los sujetos entre ambos frameworks.

Validez de constructo

De acuerdo con (Pikkanen, 2021), para obtener una mejor comparación del rendimiento de los frameworks es necesario contar con más criterios de contraste que el funcionamiento de la aplicación

web desarrollada. Para esto se compararon otros aspectos como la percepción de la facilidad de aprendizaje y el rendimiento del desarrollador.

Validez de la conclusión

Se emplearon test estadísticos apropiados tomando en consideración la distribución de datos para corroborar las conclusiones del experimento: el test no paramétrico de Wilcoxon para datos que sin distribución normal y T de Student para datos con distribución normal.

Capítulo V: Conclusiones, Recomendaciones y Líneas de Trabajo

Conclusiones

Se realizó una revisión preliminar de literatura sobre lo relacionado a Frameworks de desarrollo y a comparativas entre diferentes frameworks web. Se encontró varios estudios previos que realizaron comparaciones entre tecnologías web y frameworks. La elección de los criterios para la comparación, relacionados al rendimiento del desarrollador y del rendimiento de un software fue de utilidad en el presente trabajo.

Se diseñó el experimento en base a guías mencionadas (Genero Bocco et al., 2015). Se considera la selección de la muestra, el tipo de investigación, la definición de alcance, la selección de variables y la ejecución del experimento. El diseño permitió una comparación objetiva del framework Vue versus la biblioteca React. Se minimizó las amenazas a su validez con una buena distribución de cursos y un entrenamiento uniforme da los sujetos del experimento.

Se demostró la validez del experimento en base a la selección de una muestra significativa de 39 sujetos experimentales que contaban con un nivel similar de conocimientos, el registro de datos usando instrumentos de medición usados en ambientes de desarrollo e investigación como fue Web Vitals y encuestas usando la escala de Likert. Esto nos llevó a obtener resultados similares a estudios previos y que concuerdan con la teoría existente y poder rechazar la hipótesis nula.

Se logró implementar exitosamente el experimento. Se uso tres grupos de alumnos, dos de la carrera de Ingeniería de Software y uno de la carrera de Ingeniería en Tecnologías de Información en línea, todos con niveles de conocimiento similares. Se realizó capacitaciones en ambas tecnologías, React y Vue. Posteriormente se ejecutó el laboratorio donde la mayoría de los sujetos fueron capaces de desarrollar el ejercicio en su totalidad obteniendo los resultados de las variables independientes. La realización del experimento se realizó sin contratiempos y siguiendo la planificación establecida.

Los resultados del experimento recibieron un tratamiento para limpiar datos atípicos. Al interpretar los resultados, se pudo encontrar diferencias significativas en el rendimiento de las aplicaciones desarrolladas, en la percepción de aprendizaje y en el desempeño de los desarrolladores. Esto confirmó lo teorizado al inicio de la investigación, y nos permitió rechazar las hipótesis nulas, encontrando diferencias en todos los criterios de investigación. Estos datos fueron corroborados mediante el análisis estadístico, demostrando un nivel de significancia menor que 0.05, lo cual permite determinar que el framework A es mejor que el framework B en los tres aspectos, rendimiento de las aplicaciones desarrolladas, rendimiento del desarrollador y facilidad de aprendizaje.

Se logró definir un framework para comparar distintas tecnologías web con la finalidad de establecer ciertas diferencias entre las principales el desempeño y aspectos relacionados al desarrollador, siendo este el enfoque principal del framework desarrollado. No se puede concluir la utilidad de este framework en la presente investigación, por lo tanto, será necesario estudios adicionales, basados en distintas muestras y con un rango mayor de frameworks de desarrollo web.

Recomendaciones

Se recomienda realizar una revisión de literatura más profunda que analice en detalle las métricas, su significado en el contexto de la comparación de herramientas de desarrollo de aplicaciones web front-end, y como dichas métricas podrían influir al momento de seleccionar un framework, biblioteca o librería.

Se debería realizar un experimento más amplio con relación a los sujetos experimentales, por ejemplo, en carreras relacionadas a las ciencias de la computación en otras universidades, y en empresas de desarrollo de software. También se podría considerar variables de los sujetos experimentales, realizando el mismo experimento a grupos con diferente edad, género y nivel de conocimientos.

Realizar un entrenamiento en temas diferentes por parte de distintos instructores, podría ocasionar un sesgo en el experimento dependiendo del nivel de conocimiento de los sujetos, se recomienda que los instructores trabajen en conjunto para el desarrollo del material y el contenido del entrenamiento.

Se recomienda desarrollar una comparación más amplia incluyendo frameworks como Next.js, Remix, Angular, así mismo se debería considerar otros criterios de evaluación.

Se recomienda variar la metodología de análisis estadístico para así contrastar los distintos resultados variando el método estadístico empleado.

Bibliografía

Abdullah, H. M., & Zeki, A. M. (2014). Frontend and Backend Web Technologies in Social Networking Sites: Facebook as an Example. *2014 3rd International Conference on Advanced Computer Science Applications and Technologies*, 85–89. https://doi.org/10.1109/ACSAT.2014.22

Aleti, A., Trubiani, C., van Hoorn, A., & Jamshidi, P. (2018). An efficient method for uncertainty propagation in robust software performance estimation. *Journal of Systems and Software*, *138*, 222–235. https://doi.org/10.1016/j.jss.2018.01.010

Angular—What is Angular? (n.d.). Retrieved May 1, 2022, from https://angular.io/guide/what-is-angular

Baida, R., Andriienko, M., & Plechawska-Wójcik, M. (2020). Perfomance analysis of frameworks Angular and Vue.js. *Journal of Computer Sciences Institute*, *14*, 59–64. https://doi.org/10.35784/jcsi.1577

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). *THE GOAL QUESTION METRIC APPROACH*. 10.

Basili, V. R., & Selby, R. W. (1991). Paradigms for experimentation and empirical studies in software engineering. *Reliability Engineering & System Safety*, *32*(1), 171–191. https://doi.org/10.1016/0951-8320(91)90053-A

Boduch, A., & Derks, R. (2020). *React and React Native: A complete hands-on guide to modern web and mobile development with React.js*. Packt Publishing Ltd.

Boehm, B. W., & Papaccio, P. N. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, *14*(10), 1462–1477. https://doi.org/10.1109/32.6191

Chen, T. (2008). The Application of the Function Point Analysis in Software Developers' Performance Evaluation. *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, 1–4. https://doi.org/10.1109/WiCom.2008.1722

Cortellessa, V., Di Marco, A., & Inverardi, P. (2011). *Model-Based Software Performance Analysis*. Springer. https://doi.org/10.1007/978-3-642-13621-4

Deitel, P. (2007). *Internet & World Wide Web: How to Program*.

Dolado, J. J., Harman, M., Otero, M. C., & Hu, L. (2003). An empirical investigation of the influence of a type of side effects on program comprehension. *IEEE Transactions on Software Engineering*, *29*(7), 665–670. https://doi.org/10.1109/TSE.2003.1214329

Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., & Abrahamsson, P. (2015). Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments. *Information and Software Technology*, *64*, 132–147. https://doi.org/10.1016/j.infsof.2015.01.010

Flipse, S. M., & van de Loo, C. J. (2018). Responsible innovation during front-end development: Increasing intervention capacities for enhancing project management reflections on complexity. *Journal of Responsible Innovation*, *5*(2), 225–240. https://doi.org/10.1080/23299460.2018.1465168

Gackenheimer, C. (2015). What Is React? In C. Gackenheimer (Ed.), *Introduction to React* (pp. 1–20). Apress. https://doi.org/10.1007/978-1-4842-1245-5_1

Genero Bocco, M. (2015). *Metodos de investigacion en ingenieria del software*.

Genero Bocco, M., Cruz Lemus, J. A., & Piattini Velthuis, M. (2015). *MÉTODOS DE INVESTIGACIÓN EN INGENIERÍA DEL SOFTWARE*. EDICIONES DE LA U LTDA. http://190.57.147.202:90/xmlui/handle/123456789/2525

Graziotin, D., Wang, X., & Abrahamsson, P. (2014). Software Developers, Moods, Emotions, and Performance. *IEEE Software*, *31*(4), 24–27. https://doi.org/10.1109/MS.2014.94

Höst, M., Regnell, B., & Wohlin, C. (2000). Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. *Empirical Software Engineering*, *5*(3), 201–214. https://doi.org/10.1023/A:1026586415054

Jedlitschka, A., & Pfahl, D. (2005). Reporting guidelines for controlled experiments in software engineering. *2005 International Symposium on Empirical Software Engineering, 2005.*, 10 pp.-. https://doi.org/10.1109/ISESE.2005.1541818

Kaluža, M., & Vukelić, B. (2018). Comparison of front-end frameworks for web applications development. *Zbornik Veleučilišta u Rijeci*, *6*(1), 261–282. https://doi.org/10.31784/zvr.6.1.19

Kitchenham, B. A. (2012). Systematic review in software engineering: Where we are and where we should be going. *Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies*, 1–2. https://doi.org/10.1145/2372233.2372235

Kitchenham, B., & Pfleeger, S. L. (1996). Software quality: The elusive target [special issues section]. *IEEE Software*, *13*(1), 12–21. https://doi.org/10.1109/52.476281

Makker, S., & Rathy, R. K. (2011). *Web Server Performance Optimization using Prediction Prefetching Engine*.

McCarthy, S., O'Raghallaigh, P., Fitzgerald, C., & Adam, F. (2019). Towards a framework for shared understanding and shared commitment in agile distributed ISD project teams. *ECIS 2019, Proceedings of the 27th European Conference on Information Systems*, 1–15. https://cora.ucc.ie/handle/10468/7960

Mishra. (2021). Web development frameworks and its performance analysis—A review. In *Smart Computing* (pp. 337–343). CRC Press. https://doi.org/10.1201/9781003167488-39

NORMAS ISO 25000. (n.d.). Retrieved May 1, 2022, from https://iso25000.com/index.php/normas-iso-25000

Ollila, R., Mäkitalo, N., & Mikkonen, T. (2022). Modern Web Frameworks: A Comparison of Rendering Performance. *Journal of Web Engineering*, 789–814. https://doi.org/10.13052/jwe1540-9589.21311

Pikkanen, M. (2021). *React and Vue performance comparison*.

Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach*. Palgrave Macmillan.

Ramírez, M. O. G., De-la-Torre, M., & Monsalve, C. (2019). Methodologies for the design of application frameworks: Systematic review. *2019 8th International Conference On Software Process Improvement (CIMPS)*, 1–10. https://doi.org/10.1109/CIMPS49236.2019.9082427

Rebitzer, G., Ekvall, T., Frischknecht, R., Hunkeler, D., Norris, G., Rydberg, T., Schmidt, W.-P., Suh, S., Weidema, B. P., & Pennington, D. W. (2004). Life cycle assessment. *Environment International*, *30*(5), Article 5. https://doi.org/10.1016/j.envint.2003.11.005

Rodríguez, M., Pedreira, Ó., & Fernández, C. M. (2015). Certificación de la Mantenibilidad del Producto Software: Un Caso Práctico. *Revista Latinoamericana de Ingenieria de Software*, *3*(3), Article 3. https://doi.org/10.18294/relais.2015.127-134

Saks, E. (2019). *JavaScript Frameworks: Angular vs React vs Vue.* [Fi=AMK-opinnäytetyö|sv=YH-examensarbete|en=Bachelor's thesis|]. http://www.theseus.fi/handle/10024/261970

Sh.A.Abduraxmanova, & Jo'rayev, X. (2022). MODERN WEB TECHNOLOGIES USED IN PROFESSIONAL EDUCATION. *Conference Zone*, 178–179.

Shadish, W. R., Cook, T. D., & Campbell, D. T. (2001). *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*.

Shaw, J. (2000). Web Application Performance Testing—A Case Study of an On-line Learning Application. *BT Technology Journal*, *18*(2), 79–86. https://doi.org/10.1023/A:1026732502654

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6th Edition. *CCE Faculty Books and Book Chapters*. https://nsuworks.nova.edu/gscis_facbooks/18

Smith, C. U., & Williams, L. G. (1993). Software performance engineering: A case study including performance comparison with design alternatives. *IEEE Transactions on Software Engineering*, *19*(7), 720–741. https://doi.org/10.1109/32.238572

Sommerville, I. (2015). *Software Engineering*.

Srivastava, N., Kumar, U., & Singh, P. (2021). Software and Performance Testing Tools. *Journal of Informatics Electrical and Electronics Engineering (JIEEE)*, *2*(1), Article 1. https://doi.org/10.54060/JIEEE/002.01.001

Tekdal, M., SAYGINER, Ş., & Baz, F. (2018). *DEVELOPMENTS OF WEB TECHNOLOGIES AND THEIR REFLECTIONS TO EDUCATION: A COMPARATIVE STUDY*.

Uluca, D. (2018). *Angular 6 for Enterprise-Ready Web Applications: Deliver production-ready and cloud-scale Angular web apps*. Packt Publishing Ltd.

Valenciano López, J. (2015). *Auditoría [de] mantenibilidad [para] aplicaciones según la ISO/IEC 25000* [Info:eu-repo/semantics/bachelorThesis]. https://eprints.ucm.es/id/eprint/37485/

Web Vitals. (2023, January 28). web.dev. https://web.dev/i18n/es/vitals/

Wohlgethan, E. (2018). *Bachelorarbeit Comparing Three Major JavaScript Frameworks :*

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 1–10. https://doi.org/10.1145/2601248.2601268

Woodside, M., Franks, G., & Petriu, D. C. (2007). The Future of Software Performance Engineering. *Future of Software Engineering (FOSE '07)*, 171–187. https://doi.org/10.1109/FOSE.2007.32

Xu, W. (2021). *Benchmark Comparison of JavaScript Frameworks React, Vue, Angular and Svelte*. http://www.tcd.ie/calendar

Apéndices