UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

MAESTRÍA EN

POSGRADOS

SOFTWARE CON MENCIÓN EN DESARROLLO WEB Y MÓVIL

RPC-SO-34-NO.778-2021

OPCIÓN DE TITULACIÓN:

PROYECTO DE TITULACIÓN CON COMPONENTES DE INVESTIGACIÓN APLICADA Y/O DE DESARROLLO

TEMA:

DISEÑO DE UN DESIGN SYSTEM RD EN VUE.JS PARA LA ESTANDARIZACIÓN VISUAL MEDIANTE UN ENFOQUE UI Y DE REUTILIZACIÓN QUE APORTE AL DESARROLLO DE APLICACIONES WEB

AUTORA

DIANA RAQUEL DIAZ TAPIA

DIRECTOR:

FAUSTO STANLEY ALMEIDA CAMPOS

QUITO – ECUADOR

2024

Autor(es):

Diana Raquel Diaz Tapia
Ingeniera en Informática y Sistemas Computacionales
Candidata a Magíster en Software por la Universidad Politécnica Salesiana – Sede Quito.
ddiazt1@est.ups.edu.ec

Dirigido por:

Fausto Stanley Almeida Campos
Ingeniero en Sistemas Computacionales
Magister en Sistemas de Información Gerencial
fausto.almeidac@gmail.com

Todos los derechos reservados.

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra para fines comerciales, sin contar con autorización de los titulares de propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual. Se permite la libre difusión de este texto con fines académicos investigativos por cualquier medio, con la debida notificación a los autores.

DERECHOS RESERVADOS
2024 © Universidad Politécnica Salesiana.
QUITO– ECUADOR – SUDAMÉRICA
Diana Raquel Diaz Tapia
Diseño de un Design System RD en Vue.js para la estandarización visual mediante un enfoque UI y de reutilización que aporte al desarrollo de aplicaciones web

DEDICATORIA

Mi trabajo, si bien es cierto ha requerido de mucho esfuerzo y dedicación, no hubiese sido posible sin el apoyo incondicional de mi madre Mariana Tapia que con sus consejos y el ejemplo de ser una persona de bien e inculcarme el sabio don de la responsabilidad desde muy pequeña, eso me ha motivado a no desmayar, gracias por nunca cortarme las alas y apoyarme en todo momento, a mi padre Hugo Diaz por ser siempre auténtico y de corazón dócil, y mi hermana "ña" Denisse que me han apoyado incondicionalmente. A ti querida "Abue" Cristina Proaño por el amor que me ha brindado y por su ejemplo de ser una mujer valiente e independiente. A ti, Pau que me apoyaste en el proceso y también eres un ejemplo de superación. A un angelito que considero me ha dado la fuerza espiritual en cada día, a ti Danielito, a mis "Ñas" Vale y Naty por ser un apoyo en todas las etapas de mi vida. A Bryam por encontrarnos y compartir este sueño y por ser una fuente constante de inspiración y amor real. A toda mi familia que de una u otra forma me han brindado su apoyo y se han involucrado en mi vida y en el proyecto. Y finalmente y muy especial, a mis "Pequeños" que no son mis hijos, pero casi, Caramelo, Chocolate, Mimí y Estrellita, por ser mis compañeros de aventuras, y muestra de un amor incondicional.

AGRADECIMIENTO

Al finalizar con éxito mi trabajo quiero expresar mi agradecimiento sincero a la Universidad Politécnica Salesiana por brindarme la oportunidad de ser parte de esta noble institución, a los docentes que impartieron sus conocimientos, al tutor de mi proyecto por su apoyo y recomendación técnica durante el desarrollo de la propuesta tecnológica, a mis padres, mi hermana y mi hermano en el cielo, “ñas”, mi familia que me han apoyado , finalmente, a mí, por tener la convicción y perseverancia que me permitió resistir y persistir en los momentos de duda hasta alcanzar mis objetivos en cada asignatura recibida y ahora con el proyecto lograr mi titulación.

Tabla de Contenido

Resumen 8

Abstract 9

1. Introducción 10

2. Determinación del Problema 11

3. Marco teórico referencial 13

3.1 Design Systems y su Importancia 13

3.1.1 Definición de Design System 14

3.1.2 Beneficios de la Implementación de Design Systems 14

3.2 Vue.js en el Desarrollo Web 15

3.2.1 Descripción de Vue.js 15

3.2.2 Características y Ventajas de Vue.js 15

3.3 Usabilidad, UX y UI 16

3.3.1 Importancia de la Usabilidad 16

3.3.2 Experiencia de usuario - ux 16

3.3.3 INTERFAZ DE USUARIO - UI 16

3.4 Metodologías de Desarrollo y Herramientas 17

3.4.1 Revisión de Metodologías de Desarrollo (Software) 17

3.4.2 Revisión de metodologías para el desarrollo (Interfaz) 17

3.4.3 Herramientas en el Desarrollo de Design Systems 18

3.5 Casos de Estudio Relevantes 19

3.5.1 Implementación Exitosa de Design Systems 19

3.5.2 Casos Específicos de Aplicaciones Vue.js: 20

3.6 Transformación Digital y Desarrollo Empresarial 20

3.6.1 Transformación Digital Fundamental en las Empresas 20

3.6.2 Papel Clave de Design Systems en Empresas 20

4. Materiales y metodología 21

4.1 Marco Metodológico 21

4.2 Proceso de Desarrollo Scrum 21

4.2.1 Roles 21

4.2.2 Ceremonias 21

4.2.3 Manejo del Proyecto 22

4.3 Herramientas y Materiales Utilizados: 22

5. Resultados y discusión 23

5.1 Eficacia de la metodología scrum 24
5.1.1 Resultados (Tareas backlog) 24
5.1.2 Resultados sprints y tiempo 27
5.1.3 discusión 30
5.2 Contribución de herramientas de diseño (figma) 31
5.2.1 User Persona 31
5.2.2 Mapa de empatía 32
5.2.3 Análisis de competencia 32
5.2.4 Site map 33
5.2.5 Journey map 34
5.2.6 Tareas (Userflows documentación) 36
5.3 Contribución de herramientas de desarrollo (componentes-rd) 40
5.3.1 GitHub 40
5.3.2 Dependencias 42
5.3.3 Estructura del proyecto 43
5.3.4 Desarrollo de un componente (Button) 44
5.4 Impacto de las pruebas 47
5.4.1 Jest y TDD 47
5.4.1 Ejemplo aplicado (Button-fix) 47
5.4.2 Ejemplo aplicado (Button-feat) 48
5.4.1 Ejemplo aplicado (Button-refactor) 48
5.5 Impacto de la documentación 49
5.4.2 Documentación con Markdown (VuePress) 49
3.1.1 Estructura basada en el site map 50
5.5 Publicación y uso práctico 54
5.6.1 Configuración rollup (build) 54
5.6.2 Npm 55
5.6.3 Ejemplo aplicado 56
6. Conclusiones 58
7. Referencias 59

Diseño de un Design System RD en Vue.js para la estandarización visual mediante un enfoque UI y de reutilización que aporte al desarrollo de aplicaciones web

Autora:

Diana Raquel Diaz Tapia

Resumen

El proyecto se enfoca en desarrollar un Design System en Vue.js para establecer directrices de diseño de interfaz de usuario, con el objetivo de mejorar la eficiencia y coherencia de las aplicaciones web. A través de la creación de componentes reutilizables y su documentación detallada, se busca garantizar una estandarización visual que facilite y agilice el proceso de desarrollo.

Los objetivos incluyen investigar las mejores prácticas en Design Systems, crear una biblioteca de componentes reutilizables en Vue.js, y elaborar documentación clara que guíe su aplicación efectiva en proyectos web. Además, se implementará una pantalla de demostración para evidenciar la uniformidad visual conseguida.

El proyecto se limitará a la creación de componentes visuales en Vue.js, sin incursionar en el desarrollo backend ni en integraciones externas. Se demostrará el uso de los componentes en una pantalla de demostración, destacando la coherencia visual sin comparaciones de tiempo de desarrollo.

Se documenta y aplica un conjunto de componentes reutilizables, proporcionando un recurso valioso para el desarrollo coherente y eficiente de aplicaciones web. La documentación facilitará la adopción de los componentes, mejorando la uniformidad visual en proyectos futuros.

El proyecto se centra en fortalecer el diseño de componentes, proporcionando un Design System que sirva como base para desarrollos futuros y asegure una experiencia de usuario consistente y eficiente en diversas aplicaciones web.

Palabras clave:

Sistema de Diseño, Diseño Web, Vue.js, Interfaz de Usuario, Componentes Reutilizables

ABSTRACT

The project focuses on developing a Design System in Vue.js to establish user interface design guidelines, with the goal of improving the efficiency and coherence of web applications. Through the creation of reusable components and detailed documentation, it aims to ensure visual standardization that facilitates and accelerates the development process.

The objectives include researching best practices in Design Systems, creating a library of reusable components in Vue.js, and producing clear documentation that guides their effective application in web projects. Additionally, a demonstration screen will be implemented to showcase the achieved visual uniformity.

The project will be limited to creating visual components in Vue.js, without delving into backend development or external integrations. The use of the components will be demonstrated on a demonstration screen, highlighting visual coherence without comparisons of development time.

A set of reusable components is documented and applied, providing a valuable resource for the consistent and efficient development of web applications. The documentation will facilitate the adoption of the components, enhancing visual uniformity in future projects.

The project focuses on strengthening component design, providing a Design System that serves as a foundation for future developments and ensures a consistent and efficient user experience across various web applications.

Keywords:

Design System, Web Design, Vue.js, User Interface, Reusable Components

1. Introducción

La evolución constante del desarrollo de aplicaciones web, caracterizada por su creciente dinamismo y complejidad, ha llevado a la popularización de las aplicaciones de una sola página (SPA). En decir, uno de los principales desafíos es mantener la coherencia y calidad de la interfaz de usuario, lo cual puede ser difícil sin un sistema estandarizado. La falta de un enfoque uniforme para el diseño y desarrollo de componentes de UI puede resultar en experiencias de usuario inconsistentes y procesos de desarrollo ineficientes.

Los Design Systems emergen como una solución esencial para estos desafíos, proporcionando pautas detalladas para la presentación y comportamiento de elementos fundamentales de la UI. Garantiza la coherencia y calidad de la experiencia del usuario en productos digitales. Con la implementación de Design Systems, especialmente utilizando bibliotecas como Vue.js, se pueden crear componentes reutilizables que mejoren la eficiencia y coherencia en el desarrollo de aplicaciones web.

El objetivo general desarrollar un Design System en Vue.js que establezca directrices claras y efectivas de diseño de interfaz de usuario, para mejorar la apariencia y eficiencia de las aplicaciones web. El sistema buscará garantizar la estandarización visual.

Los objetivos específicos se centran investigar sobre System Designs para entender y aplicar las mejores prácticas en el desarrollo de directrices de diseño que incluyan elementos visuales clave como colores, tipografía y componentes de interfaz; el primer proyecto es crear una biblioteca básica de componentes reutilizables en Vue.js que sigan estos principios de diseño, facilitando el desarrollo y manteniendo uniformidad visual; el segundo proyecto se basa en elaborar documentación detallada del Design System, proporcionando pautas claras para su aplicación en aplicaciones web, lo que ayudará al equipo de desarrollo a implementar el sistema de manera efectiva; y finalmente el tercer proyecto es la aplicación de dichos componentes creados, dando como resultado la estandarización y reutilización de componentes.

El proyecto se centrará exclusivamente en la creación y documentación de componentes visuales reutilizables usando Vue.js, sin abordar el desarrollo de funcionalidades backend ni integraciones con servicios externos. La implementación demostrará el uso de los componentes desarrollados en una pantalla de demostración para ilustrar la estandarización y coherencia visual. El enfoque garantiza que el principal objetivo del proyecto permanezca en el diseño de componentes, crucial para guiar desarrollos futuros y asegurar uniformidad en aplicaciones. No se realiza ninguna comparación en tiempo de desarrollo, el proyecto profundizará en la relevancia de la estandarización del uso de componentes en las aplicaciones, resultando en una aplicación con uniformidad visual y funcional en sus componentes.

2. Determinación del Problema

El desarrollo de aplicaciones web se vuelve cada vez más dinámico y complejo, y que las aplicaciones de una sola página ya han alcanzado una gran popularidad, un Design System ofrece pautas minuciosas acerca de cómo se deben presentar y comportar los elementos de la interfaz de usuario, y actúa como la guía primordial para el proceso de desarrollo de interfaces [1], para lo cual en [2], [3], [4] detallan la importancia de la consistencia y la cohesión del UX en productos digitales que en cierto punto mide la calidad de la Interfaz de Usuario y afecta o beneficia a la usabilidad, diseño ergonómico en la cual estos criterios pueden ser evaluados mediante cuestionarios, entrevistas, prototipos de alta fidelidad y pruebas de usabilidad.

En [5] se discuten factores importantes a considerar en el diseño de interacción de aplicaciones, como el diseño de los elementos de la interfaz y los colores. Además, se menciona que el diseño de la interfaz determina la calidad del producto final.

Además, se menciona el uso de enfoque de Design Thinking para el desarrollo de aplicaciones de billetera digital [6] la cual define que se inicia identificando a los usuarios para determinar las características prioritarias de la aplicación y luego se pasa por las etapas de definición, ideación, prototipado y prueba, siempre centrando la

atención en la experiencia del usuario. Además, en [7] en la cual desarrollan una biblioteca de componentes desarrollada utilizando Svelte y Tailwind CSS, y el segundo es una aplicación Vue que utiliza los componentes de la biblioteca mencionada anteriormente y lo demuestran con algunos ejemplos.

Además, en [8] aplica el método de diseño enfocado en el usuario (UCD) para mejorar la apariencia y la eficiencia del sistema de información académica AIS, entre las metodologías y técnicas que se aplicaron incluyen la recopilación de datos a través de estudios bibliotecarios, la planificación del proceso centrado en el ser humano, la especificación del contenido de uso, la especificación de los requerimientos, la producción de una solución de diseño y la evaluación del diseño, los resultados muestran un aumento significativo en la apariencia y la eficiencia, mientras que en [9] se analiza la influencia de factores como los principios de diseño, la participación del usuario y las percepciones de los evaluadores en la usabilidad mediante instrumentos de Evaluación Heurística en el aporte de la calidad de la UI/UX.

Finalmente, en [10] enfatiza que la transformación digital es imprescindible para que una empresa prospere y siga siendo competitiva, y que la competencia del mercado impone exigencias sin precedentes a la colaboración de los diferentes perfiles de los equipos de desarrollo profesional, así como Low Code, y los Design Systems desempeñan un papel relevante siendo herramientas de diseño y su integración efectiva con plataformas de código bajo y reutilización son aspectos clave para mejorar la calidad y la consistencia en las soluciones digitales en el ámbito empresarial.

3. Marco teórico referencial

3.1 Design Systems y su Importancia

La implementación de un sistema de diseño es importante porque proporciona un conjunto de pautas, principios y estándares para crear experiencias de usuario consistentes y cohesivas en diferentes productos y plataformas. Ayuda a agilizar el proceso de diseño, reducir la deuda de diseño y mejorar la colaboración entre diseñadores y desarrolladores. Un sistema de diseño también garantiza que el diseño sea accesible, escalable y mantenible, lo que puede ahorrar tiempo y recursos a largo plazo. Además, puede ayudar a establecer una identidad de marca sólida y mejorar la confianza y lealtad de los usuarios.

Los sistemas de diseño son esenciales en el desarrollo de interacción del usuario, pero la investigación sobre herramientas que respalden su creación, mantenimiento y uso es limitada, como en [11] lo menciona y aborda los desafíos, valores de los líderes de proyectos de sistemas de diseño, destacando la necesidad de herramientas que se centren en el comportamiento de los componentes de interfaz. Se destaca la preferencia por un enfoque ascendente en la creación de sistemas de diseño, donde los componentes evolucionan desde productos específicos. Estos hallazgos sientan las bases para diseñar técnicas y herramientas que apoyen la práctica de sistemas de diseño y enfoques modernos de diseño de interacción con el usuario.

Además, en [12] menciona la importancia del usuario en la interacción con los sistemas, especialmente en el contexto de la creciente automatización de la interacción y el papel cada vez mayor de la inteligencia artificial. Explora cómo la automatización de la interacción está cambiando la dinámica de la interacción humano-máquina y cómo estos cambios afectan la experiencia del usuario.

3.1.1 Definición de Design System

En [13] menciona que es una librería de patrones perceptuales y funcionales que se utiliza para estandarizar la implementación de los elementos de la interfaz en un producto digital. Es un conjunto de reglas y pautas que definen un lenguaje compartido entre los equipos de la empresa, con el fin de lograr una comunicación más consistente y efectiva. El objetivo principal de un Design System es visualizar coherencia, optimizar el tiempo de diseño y desarrollo de componentes/pantallas.

Es importante mencionar que un Design System tiene principios sociotécnicos, los cuales en [14] afirma que los principios se enfocan en el diseño de nuevos productos de software, incorporando tecnologías de la información y prácticas de gestión modernas. El documento destaca tres tipos de principios: meta, contenido y proceso, destinados a ser utilizados por gestores de sistemas, usuarios, diseñadores y profesionales de la tecnología y ciencias sociales. El estudio aborda la necesidad de una perspectiva integrada en el diseño sociotécnico y reconoce la importancia de principios autónomos, respaldados por métodos y herramientas relevantes.

3.1.2 Beneficios de la Implementación de Design Systems

Según [15] destaca la importancia fundamental de los sistemas de diseño en la creación de experiencias de usuario efectivas y consistentes, así como en la eficiencia y la colaboración en el desarrollo de software. Se enfatiza que los sistemas de diseño permiten establecer una apariencia y comportamiento coherentes en todas las interfaces de usuario, garantizando una experiencia unificada para los usuarios. Además, facilitan la escalabilidad al permitir la reutilización de componentes y patrones de diseño, agilizando el desarrollo y manteniendo la consistencia en diferentes partes de una aplicación. Asimismo, se resalta su papel en la facilitación del trabajo en equipo, al proporcionar un marco común para que diseñadores, desarrolladores y otros interesados colaboren de manera efectiva.

El desarrollo eficaz de sistemas depende de la capacidad para diseñar software de manera confiable. Hasta ahora, el diseño de software ha sido ad hoc, lento y costoso, como lo menciona en [16] y propone una base científica para el diseño de software

utilizando el enfoque axiomático, basado en los axiomas de Independencia e Información. El enfoque proporciona herramientas para la toma de decisiones y representa sistemáticamente conocimientos, datos y estructuras de software.

3.2 Vue.js en el Desarrollo Web

3.2.1 Descripción de Vue.js

En [17] describe a Vue.js como un "framework progresivo para la construcción de interfaces de usuario", es incremental, enfocado en la visualización y de fácil integración con otras bibliotecas o proyectos existentes. Además, se menciona que tiene un sistema de reactividad que permite la creación de instancias, definición de propiedades con getters y setters para observar cambios dinámicamente, también se elige por su amigabilidad para los usuarios, su velocidad, sus paquetes oficiales y su capacidad de ser adoptado progresivamente en una página web, como concuerda en [18] y añade que Vuejs ofrece soporte para la internacionalización y la seguridad contra XSS y clickjacking.

3.2.2 Características y Ventajas de Vue.js

En [19] realiza un análisis de Vue.js y se destaca por su bajo umbral de entrada, ofreciendo una curva de aprendizaje accesible y amigable, además, su implementación del DOM virtual contribuye significativamente al rendimiento, la alta flexibilidad en proyectos web de cualquier tamaño y complejidad, proporcionando versatilidad. Su facilidad de integración con otras bibliotecas y frameworks agrega un valor adicional, permitiendo su incorporación sin complicaciones en proyectos existentes, es decir, Vue.js se posiciona como una herramienta poderosa y adaptable en el arsenal de desarrollo web.

Además, un estudio de caso [20] destaca que el uso de los frameworks Laravel y VueJs proporcionó varias ventajas significativas en el desarrollo del sistema web, estas ventajas incluyen la simplificación del desarrollo, la separación de la lógica de negocio y la capa de presentación, la mejora de la navegabilidad y funcionalidad del sistema a través de componentes reutilizables, la comunicación eficiente con otras aplicaciones y hardware, y la implementación exitosa de la metodología ágil SCRUM la cual mejora la comunicación y cumplimiento de actividades en los tiempos establecidos. En conjunto,

estas ventajas contribuyeron a la creación de un sistema eficiente y fácil de mantener, optimizando la gestión por procesos.

3.3 Usabilidad, UX y UI

3.3.1 Importancia de la Usabilidad

En [21] resalta que la usabilidad enfocada a los startups de software, reconocen que los resultados producidos por las prácticas de UX pueden generar valor para los inversores y conducir al desarrollo de estrategias para identificar y comprometer a los usuarios. Además, se destaca la necesidad de comprender el valor del trabajo de UX y la conexión entre la UX y los modelos de negocio para crear valor para el cliente

3.3.2 Experiencia de usuario - UX

Existe una deuda técnica como lo menciona en [22], debido a estos dos aspectos en el desarrollo de software tienen una falta de atención y puede generar UXDebt, que es un tipo de deuda técnica que se acumula cuando se ignoran o posponen las mejoras, además se menciona que la gestión de UXDebt es importante para garantizar la satisfacción del usuario final y la calidad del software.

3.3.3 INTERFAZ DE USUARIO - UI

Ahora, en [23] presenta un estudio para la adaptación dinámica de UI en aplicaciones móviles, haciendo uso de patrones de diseño y modelos de ontología, el objetivo es ajustar las interfaces a las necesidades, preferencias y contexto de uso de los usuarios, con un impacto directo en la usabilidad. Las pruebas de usabilidad realizadas mediante una aplicación móvil desarrollada y subrayan la eficacia de las interfaces adaptativas, evaluando aspectos cruciales como la satisfacción y la facilidad de aprendizaje.

En [24] aborda una perspectiva integradora y multidisciplinaria en el desarrollo de aplicaciones, es fundamental de vincular los principios de diseño con el proceso y el personal, proponiendo un sistema aplicable a diversas metodologías y tipos de aplicaciones. El enfoque se centra en comprender las relaciones entre los principios de diseño y el proceso de desarrollo, considerando aspectos organizativos, metodológicos, semiótico-visuales y tecnológicos.

3.4 Metodologías de Desarrollo y Herramientas

3.4.1 Revisión de Metodologías de Desarrollo (Software)

Según [25] [26] Scrum destacada como la metodología ágil líder, se presenta como una estrategia de gestión de proyectos que enfoca la entrega incremental e iterativa de valor, mencionan algunos beneficios:

- Equipo y Colaboración: la cohesión del equipo y la colaboración destaca la importancia de Scrum para promover relaciones sólidas y eficaces entre los miembros del equipo, priorizando individuos e interacciones sobre procesos y tareas.
- Control Empírico e Incremental: es mencionado en la segunda referencia como un pilar fundamental de Scrum. Dicho método fundamentado en transparencia, inspección y adaptación contribuye a un control permanente del estado del software.
- Flexibilidad y Adaptabilidad: La referencia adicional destaca la flexibilidad de Scrum al permitir organización de responsabilidades y tareas entre el equipo, adaptándose a las necesidades cambiantes del proyecto.

En conclusión, Scrum destaca como una metodología ágil líder, centrándose en la entrega incremental de valor, el control empírico e incremental para un seguimiento constante del estado del software y su flexibilidad en la gestión de proyectos.

3.4.2 Revisión de metodologías para el desarrollo (Interfaz)

En [27] menciona las siguientes metodologías para el desarrollo de UI:

- Diseño orientado a metas (Goal-oriented design): Se basa en el análisis de las metas del usuario y la comprensión de las intenciones del usuario para crear productos más óptimos y de alta calidad.

- Diseño centrado en el usuario (User-centered design): Se enfoca en analizar las necesidades y capacidades del usuario final y adaptar los productos a estas necesidades, a través de la comprensión de la psicología de los usuarios y la evolución del diseño basado en retroalimentación de los usuarios.
- Diseño emocional (Emotional design): Se refiere a la consideración de las experiencias emocionales y percepciones de los usuarios al diseñar productos, con el objetivo de convertir estas experiencias en propiedades específicas del producto.

3.4.3 Herramientas en el Desarrollo de Design Systems

- Herramientas de Diseño (Figma)

En [28] afirma la importancia de la implementación de conceptos y técnicas de UI/UX en el diseño de interfaces web, utilizando herramientas como Figma para crear interfaces consistentes y armoniosas, es decir, se destaca la necesidad de optimizar el diseño, lo que resulta en experiencias satisfactorias y accesibles para los usuarios en el entorno tecnológico actual.

- Gestión de Versiones y Colaboración

Queda en evidencia en [29] un estudio sobre la productividad de los desarrolladores al utilizar herramientas de programación en pareja con inteligencia artificial, como GitHub Copilot, destaca la importancia de la aceptación de sugerencias de la IA como predictor de la productividad percibida. Se enfatiza que la utilidad de las sugerencias recibidas, más que su corrección, juega un papel crucial en mejorar la eficiencia y fluidez del trabajo de los desarrolladores, independientemente de su nivel de experiencia.

- Desarrollo de Componentes

Vue.js es un framework de JavaScript de código abierto que ha ganado popularidad por su arquitectura adaptable, documentación detallada y facilidad de aprendizaje. A diferencia de Angular, Vue.js se destaca por su simplicidad y claridad, lo que lo hace ideal para diseñadores de UI que buscan construir componentes flexibles y reutilizables. Además, Vue.js utiliza un Virtual DOM que agiliza el rendimiento de la interfaz de usuario de las aplicaciones [30].

a) Estilos y Temas:

   a. Sass o Less: Preprocesadores CSS que permiten organizar y reutilizar estilos de manera eficiente.
   b. PostCSS: Herramienta para transformar estilos CSS mediante plugins.

b) Documentación:
   a. Markdown: Para escribir documentación simple y clara.
   b. VuePress o Docusaurus: Frameworks de documentación que facilitan la creación de sitios web estáticos para documentar tu Design System.

c) Pruebas (Unitarias):
   a. Jest: Framework de pruebas que puedes utilizar no solo para probar la lógica de los componentes, sino también para realizar pruebas de snapshots que ayudan a mantener la consistencia visual

d) Empaquetar
   a. Rollup: Empaquetar tus componentes de UI en un bundle que luego puede ser fácilmente integrado en diferentes proyectos o plataformas

e) Automatización de Tareas:
   a. npm o Yarn: Para gestionar las dependencias y scripts de construcción.

f) Ejemplo aplicado

3.5 Casos de Estudio Relevantes

3.5.1 Implementación Exitosa de Design Systems

En [31] desarrolló un nuevo sitio web de comercio electrónico y el desafío fue construirlo desde cero, crearon un Design System que mejoró la eficiencia con Invision Design System Manager como la herramienta adecuada para gestionar el sistema por su adecuada integración con su flujo de trabajo existente.

En [32] resalta y analiza los aspectos monetarios de diferentes desarrollos de software empresarial relacionados con UX y cómo se pueden calcular los beneficios económicos de una buena UX utilizando el Análisis Costo-Beneficio (CBA) para el apoyo a la toma de decisiones la cual aplican diferentes modelos y enfoques que se pueden utilizar para analizar el costo-beneficio de la UX en el desarrollo de software empresarial como evidencias teóricas y empíricas.

3.5.2 Casos Específicos de Aplicaciones Vue.js:

En [33] aborda los desafíos en el sistema médico chino y presenta el desarrollo de CHAVE TECH Intelligent Clinic Software, una solución centrada en la gestión inteligente de clínicas con Vue.js como marco de desarrollo frontal. Se destacan características clave de Vue.js, como el patrón MVVM y usar componentes, y se proporciona un ejemplo de implementación utilizando Vuex. Los resultados parciales de las pruebas funcionales, enfatizando la mejora de eficiencia y calidad en las clínicas mediante la aplicación de tecnologías inteligentes.

3.6 Transformación Digital y Desarrollo Empresarial

3.6.1 Transformación Digital Fundamental en las Empresas

En finanzas, es clave para mejorar productos, optimizar procesos y obtener ventaja competitiva. Va más allá de la adaptación al entorno digital, permitiendo estrategias centradas en el cliente y la creación de valor. Además, facilita la adaptación a cambios del mercado, fomentando una cultura innovadora para respuestas ágiles y diferenciadas como afirma [34].

Es decir, La transformación digital no solo es esencial para adaptarse al entorno empresarial digital, sino que también impulsa mejoras en productos, procesos y competencias, el sector financiero, esta transformación no solo significa mantenerse a la vanguardia, sino también adoptar estrategias centradas en el cliente y fomentar una cultura innovadora. La capacidad de respuesta a cambios y la creación de valor se convierten en elementos clave para obtener y mantener una ventaja competitiva

3.6.2 Papel Clave de Design Systems en Empresas

[35] resalta la importancia de los sistemas de diseño en las empresas, según un estudio de Forrester y Adobe. Señala que las empresas con sistemas de diseño maduros obtienen beneficios notables, como experiencias de cliente más cohesivas y mayor

velocidad de lanzamiento al mercado. Identifica cinco elementos clave de un sistema de diseño efectivo e insta a establecer roles dedicados y métricas de éxito centradas en la adopción y resultados comerciales.

4. Materiales y metodología

4.1 Marco Metodológico

El proyecto se basa en Scrum para gestionar eficientemente el desarrollo del Design System centrado en Vue.js. Scrum proporciona una estructura flexible que permite entregas iterativas y priorización constante.

4.2 Proceso de Desarrollo Scrum

Utilizar la metodología Scrum como desarrollador individual implica adaptar sus roles y ceremonias para un entorno unipersonal.

4.2.1 Roles

- Product Owner, Scrum Master, y Desarrollador: definiendo objetivos, priorizando tareas, y ejecutando el desarrollo.

4.2.2 Ceremonias

- Planificación del Sprint, establecer objetivos claros y selecciona tareas del backlog para el sprint.
- Daily Scrum: evalúa el progreso diariamente, ajustando tus planes según sea necesario.
- Revisión del Sprint: Al final del sprint, revisa lo completado.
- Retrospectiva del Sprint: Reflexionar sobre qué mejoró y qué podría cambiar para los próximos sprints.

4.2.3 Manejo del Proyecto

- Backlog: Organizar y priorizar las tareas utilizando Clickup.
- Estimaciones: estimar el esfuerzo para cada tarea para planificar mejor.

Los resultados de la aplicación de la metodología se muestran a detalle en el siguiente capítulo que permitió obtener los resultados planteados en el proyecto.

4.3 Herramientas y Materiales Utilizados:

- Vue.js

Descripción: Vue.js es un framework basado en JavaScript utilizado, es altamente adaptable y permite el desarrollo de componentes reutilizables que pueden ser integrados en diversas partes de una aplicación para mantener la coherencia y la funcionalidad.

Uso Principal: En el contexto de un sistema de diseño, Vue.js sería usado para desarrollar y estructurar los componentes del sistema, tales como botones, formularios, y otros elementos de UI que se mantienen consistentes en diversas aplicaciones.

- Figma

Descripción: Figma es una herramienta de diseño de interfaces basada en la nube que facilita la colaboración en tiempo real. Permite a los diseñadores crear, prototipar, y obtener feedback de manera eficiente, haciendo que sea más fácil iterar y mejorar los diseños.

Uso Principal: Figma sería utilizada para diseñar los componentes visuales del sistema de diseño, definir la guía de estilo, y crear prototipos que muestren cómo deben interactuar y verse los componentes UI. La integración de Figma con herramientas de desarrollo permite exportar los diseños directamente a código, lo que ayuda a acelerar el proceso de desarrollo.

- GitHub

Descripción: GitHub es una plataforma de hosting para control de versiones y colaboración. Permite a los equipos de desarrollo colaborar en proyectos desde cualquier lugar, facilitando el seguimiento de cambios, la gestión de problemas, y la revisión de código.

Uso Principal: GitHub se utilizaría para alojar el repositorio de código del sistema de diseño, gestionar el control de versiones de todos los componentes, y facilitar la colaboración entre diseñadores y desarrolladores. También sería crucial para implementar procesos de integración continua y despliegue continuo (CI/CD), asegurando que las actualizaciones del sistema de diseño se apliquen de manera eficiente y consistente en todos los proyectos que lo utilizan.

- Semver

El versionamiento semántico (semver) en NPM permite a los desarrolladores comunicar cambios de forma clara y consistente, facilitando la gestión de actualizaciones importantes como parches de seguridad y mejorando la estabilidad de las dependencias. En [36] menciona que el uso del versionamiento semántico (semver) en NPM permite a los desarrolladores etiquetar claramente las actualizaciones de sus paquetes, diferenciando entre cambios importantes, adiciones de funcionalidades y correcciones de errores. Esto facilita la gestión ágil de actualizaciones críticas como parches de seguridad, mejorando la estabilidad y seguridad del ecosistema de NPM.

5. Resultados y discusión

5.1 Eficacia de la metodología scrum

Como se mencionó se aplicó Scrum, sin embargo, se lo realizó como un único desarrollador, aunque no es típico para el que se diseñó esta metodología, generalmente se utiliza en equipos para fomentar la colaboración, pero un desarrollador solo también puede adaptar los principios y prácticas de Scrum para mejorar la gestión del tiempo y la productividad de su proyecto. Aquí te explico cómo podría hacerlo:

5.1.1 Resultados (Tareas backlog)

- Implementación de la metodología Scrum adaptada para un solo desarrollador.
- Uso de ClickUp para gestionar el backlog de productos y visualizar claramente las tareas.
- Desarrollo iterativo del proyecto, con sprints que permitieron evaluar y ajustar continuamente el enfoque del desarrollo.

La Figura 1 muestra el backlog para el diseño, detalla la realización de tareas como la creación de User Persona, Journey Map, Site Map, User Flow, y el desarrollo de Wireframes de baja y alta fidelidad dentro de un sistema de diseño trae múltiples beneficios, principalmente, permite establecer un marco coherente y estandarizado que garantiza la consistencia en todo el proyecto, esto es crucial para mantener una experiencia de usuario uniforme y profesional.

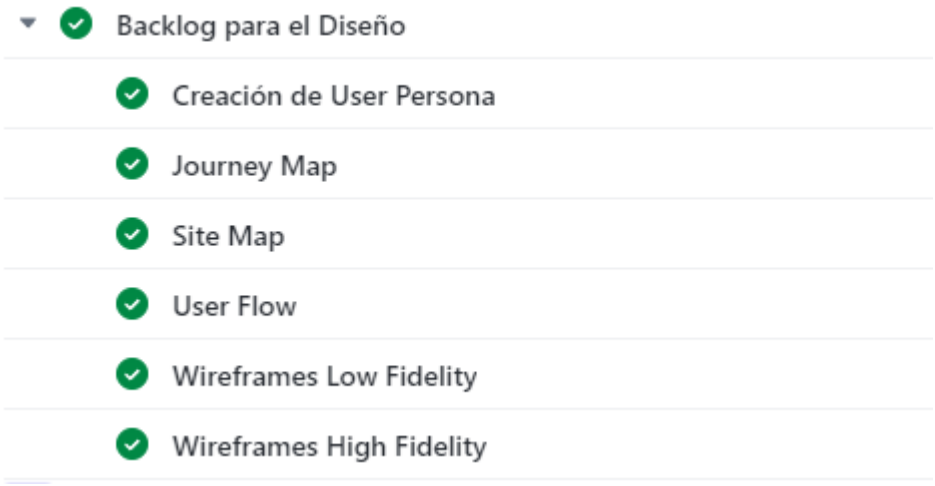

Figura 1. Backlog para el Diseño

Además, esto aporta a tener una referencia clara y compartida para todos los elementos del diseño y la interactividad, en cuestión a la documentación. Esto se traduce en una mayor eficiencia y cohesión durante el proceso de desarrollo, permitiendo que los equipos se concentren en innovar y mejorar la usabilidad en lugar de resolver inconsistencias o malentendidos.

La Figura 2 demuestra el backlog para el Desarrollo de los componentes, detalla las tareas a realizar, utilizar Vue.js permite la creación de componentes dinámicos y reutilizables que mejoran la eficiencia del desarrollo. La estilización con SCSS facilita la gestión de estilos más complejos, manteniendo la coherencia visual a través del proyecto. Implementar pruebas unitarias con Jest asegura que cada componente funcione correctamente antes de su integración, minimizando los riesgos de errores en producción. La documentación en GitHub es esencial para el mantenimiento y la escalabilidad del proyecto, permitiendo que cualquier desarrollador entienda y contribuya efectivamente al código. Publicar componentes en NPM y realizar pruebas de usabilidad son pasos finales clave para garantizar que los componentes sean fácilmente accesibles y funcionen como se espera en diversos contextos de uso.

Figura 2 Backlog para el Desarrollo (Components).

Utilizar Vue.js permite la creación de componentes dinámicos y reutilizables que mejoran la eficiencia del desarrollo. La estilización con SCSS facilita la gestión de estilos más complejos, manteniendo la coherencia visual a través del proyecto. Implementar pruebas unitarias con Jest asegura que cada componente funcione correctamente antes de su integración, minimizando los riesgos de errores en producción. La documentación en GitHub es esencial para el mantenimiento y la escalabilidad del proyecto, permitiendo que cualquier desarrollador entienda y contribuya efectivamente al código. Publicar componentes en NPM y realizar pruebas de usabilidad son pasos finales clave para garantizar que los componentes sean fácilmente accesibles y funcionen como se espera en diversos contextos de uso.

La Figura 3 muestra las tareas para el desarrollo de la documentación,

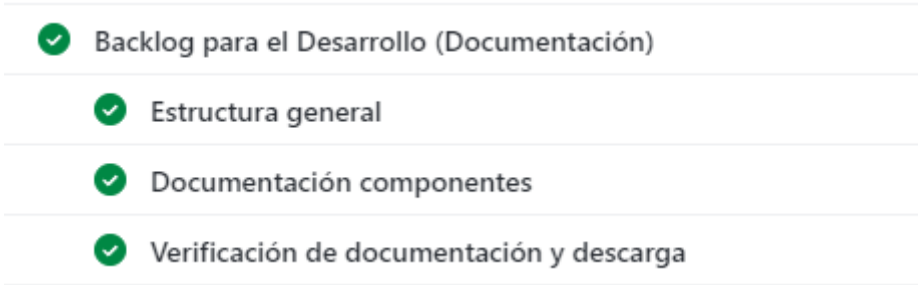

Figura 3. Backlog para el Desarrollo (Documentación)

La Figura 4 muestra las tareas para la publicación y uso de los componentes

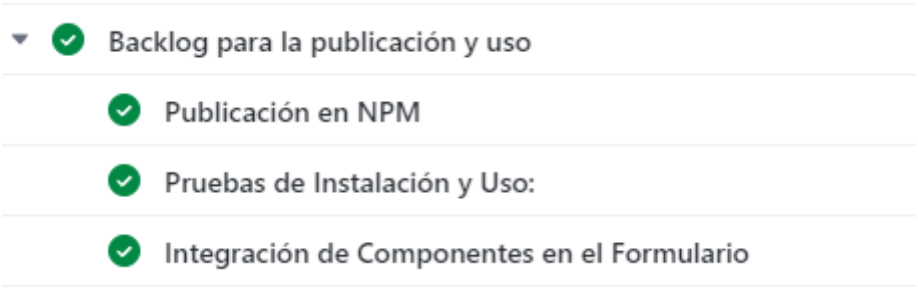

Figura 4 Bakclog para la publicación y uso

El proceso se llevo con fechas como se puede verificar en la plataforma mencionada: https://sharing.clickup.com/9013129020/g/h/8ckjvtw-533/ffc9c0ba104c0bb

5.1.2 Resultados sprints y tiempo

El proyecto se desarrolló siguiendo la metodología Scrum, adaptada para ser implementada por un único desarrollador. Se dividió en cuatro sprints semanales, con un total de 140 horas de trabajo distribuidas de manera equitativa entre los sprints los cuales fueron cuidadosamente planificados, con el objetivo de optimizar tanto el proceso de desarrollo como los resultados. Es importante mencionar que en el desarrollo se realizaron tres proyectos: componentes, documentación e implementación, a continuación, se detalla el enfoque y los resultados obtenidos en cada sprint.

- Sprint 1

La Tabla "Distribución de Horas para el Sprint 1: Planificación y Diseño" detalla la fase inicial del proyecto, asignando horas a la creación de User Persona, el desarrollo de Journey Map, la elaboración de Site Map, el diseño de User Flow, y el desarrollo de wireframes. El primer sprint es fundamental establece la estructura y dirección del diseño de los componentes.

Tabla 1 "Distribución de Horas para el Sprint 1: Planificación y Diseño

Sprint	Objetivo	Tarea	Horas
1	Planificación y Diseño	Crear User Persona	4
1	Planificación y Diseño	Desarrollar Journey Map	4
1	Planificación y Diseño	Elaborar Site Map	4

1	Planificación y Diseño	Diseñar User Flow	6
1	Planificación y Diseño	Desarrollo de Wireframes de baja y alta fidelidad	10

Se centra exclusivamente en el desarrollo inicial de los componentes bajo "Componentes-RD". Incluye tareas de diseño preliminar como la creación de User Persona, el desarrollo de Journey Map, y la elaboración de Site Map, hasta llegar al diseño de User Flow y el desarrollo de wireframes de la “Documentacion-RD”. El sprint establece las bases para los componentes que se desarrollarán y documentarán en sprints posteriores.

- Sprint 2

La Tabla "Distribución de Horas para el Sprint 2: Desarrollo de Componentes" resume las actividades de implementación en "Componentes-RD", incluyendo el desarrollo de componentes en Vue.js, la estilización con SCSS, pruebas unitarias, y la documentación en GitHub, asegurando una ejecución y documentación detallada de los componentes desarrollados.

Tabla 2 Distribución de Horas para el Sprint 2: Desarrollo de Componentes

Sprint	Objetivo	Tarea	Horas
2	Desarrollo de Componentes	Desarrollo de componentes básicos con Vue.js	20
2	Desarrollo de Componentes	Aplicación de estilos SCSS a los componentes	10
2	Desarrollo de Componentes	Escribir y ejecutar pruebas unitarias para cada componente	5
2	Desarrollo de Componentes	Documentación de componentes en GitHub	10

Sigue dentro de "Componentes-RD", enfocándose en la materialización de los componentes diseñados previamente. Se llevan a cabo actividades como el desarrollo de los componentes básicos en Vue.js, la aplicación de estilos SCSS, la ejecución de pruebas unitarias y la documentación inicial en GitHub, asegurando que los componentes están bien documentados sobre su uso.

- Sprint 3

La Tabla "Distribución de Horas para el Sprint 3: Publicación de Componentes y Documentación" detalla un sprint clave donde se publican los componentes en NPM y

se documentan en VuePress. Incluye tareas de preparación, pruebas de usabilidad, y redacción detallada, asegurando la correcta implementación y documentación de los componentes.

Tabla 3 Distribución de Horas para el Sprint 3: Publicación de Componentes y Documentación

Sprint	Objetivo	Tarea	Horas
3	Publicación de Componentes	Preparación para publicación en NPM	10
3	Publicación de Componentes	Realización de pruebas de usabilidad	5
3	Publicación de Componentes	Publicación de componentes en el registro de NPM	2
3	Documentación en VuePress	Integración de componentes en DocumentationRD	5
3	Documentación en VuePress	Redacción de documentación detallada	10
3	Documentación en VuePress	Revisión y mejora de documentación	5

Es un punto crucial donde convergen dos áreas: se finaliza la publicación de los componentes de "Componentes-RD" y se inicia la documentación intensiva en "DocumentationRD". Las tareas incluyen la preparación y publicación de los componentes en NPM, complementadas con pruebas de usabilidad. Paralelamente, se comienza a integrar estos componentes en la documentación con VuePress, redactando detalles técnicos y realizando revisiones para asegurar precisión y claridad.

- Sprint 4

La Tabla 4 "Distribución de Horas para el Sprint 4: Uso de Componentes RD" refleja un sprint enfocado exclusivamente en la "Implementación", donde los componentes desarrollados se utilizan para crear un formulario práctico. El sprint es crucial para validar la funcionalidad de los componentes en un entorno real, garantizando la estandarización del producto final.

Tabla 4 Distribución de Horas para el Sprint 4: Uso de Componentes RD

Sprint	Objetivo	Tarea	Horas
4	Uso de components RD	Creación de formulario utilizando los componentes	30

Se dedica enteramente a "Implementacion", donde se integran los componentes desarrollados en un formulario práctico. El sprint es vital para validar la funcionalidad completa de los componentes en un entorno real, enfocándose en asegurar que el producto final sea estandarizado.

5.1.3 DISCUSIÓN

- Adaptación de Roles de Scrum: En el proyecto, asumí los roles de Product Owner, Scrum Master y desarrollador simultáneamente. Esta adaptación me permitió manejar eficazmente todo el ciclo de vida del desarrollo, manteniendo un enfoque disciplinado y objetivo en la priorización y evaluación de las tareas a lo largo de los sprints.
- Uso de ClickUp: Utilicé ClickUp para segmentar el trabajo en tareas manejables y para gestionar los backlogs específicos de cada sprint. Esta herramienta fue fundamental para mantener una estructura organizada y permitió una visibilidad clara del progreso del proyecto.
- Proceso Iterativo y Aprendizaje Continuo: La metodología Scrum facilitó un enfoque iterativo que se adaptó a cada fase del proyecto:
    - Sprint 1: Se centró en la planificación y diseño inicial, estableciendo una base sólida para los componentes y documentación futuros.
    - Sprint 2: Avancé en el desarrollo de los componentes reutilizables, lo que permitió empezar a ver la funcionalidad de las piezas creadas.
    - Sprint 3: Me enfoqué en la publicación de los componentes y en la elaboración de la documentación detallada, preparando todo para su uso efectivo.
    - Sprint 4: Se dedicó a la implementación y prueba de los componentes en un formulario práctico, demostrando su funcionalidad en un entorno real.

Esta estructura de sprints facilitó no solo el desarrollo técnico del proyecto sino también el aprendizaje personal sobre la gestión y aplicación efectiva de un Design System en un entorno de desarrollo real.

5.2 Contribución de herramientas de diseño (figma)

- Resultados: Facilitó una palpación más real del producto final enfocada a la documentación, sus objetivos, sus flujos.
- Discusión: Figma fue esencial para mantener la cohesión visual a través del proyecto, aunque se enfrentaron desafíos al mantener la sincronización entre los archivos de diseño y el código fuente cuando se realizaban cambios significativos.

5.2.1 User Persona

La Figura 5 muestra un "user persona" en la cual se detalla información de Laura Gómez, una desarrolladora frontend de 32 años, basada en Quito y casada, el perfil incluye su edad, ubicación, estado civil, y detalla su profesión junto con un conjunto de aspectos clave como sus metas, necesidades, motivaciones y frustraciones en su campo de trabajo.

Figura 5. User Persona

La creación de user personas como esta es crucial para el desarrollo de productos y servicios más efectivos y orientados al usuario. Al simular las características y comportamientos de los usuarios reales, los equipos pueden diseñar soluciones más

adaptadas a las necesidades específicas, mejorando la experiencia del usuario y aumentando la probabilidad de éxito del producto en el mercado.

5.2.2 Mapa de empatía

La Figura 6 muestra el mapa de empatía en la cual se detalla el perfil y perspectiva de un desarrollador web al considerar la adopción de un nuevo sistema de diseño. El mapa segmenta la experiencia del usuario en varias dimensiones: lo que piensan y sienten, lo que ven, lo que escuchan, lo que dicen y hacen, junto con sus dolores y ganancias esperadas. Cada sección proporciona insights valiosos sobre las expectativas y retos del usuario frente a innovaciones en su entorno de trabajo.

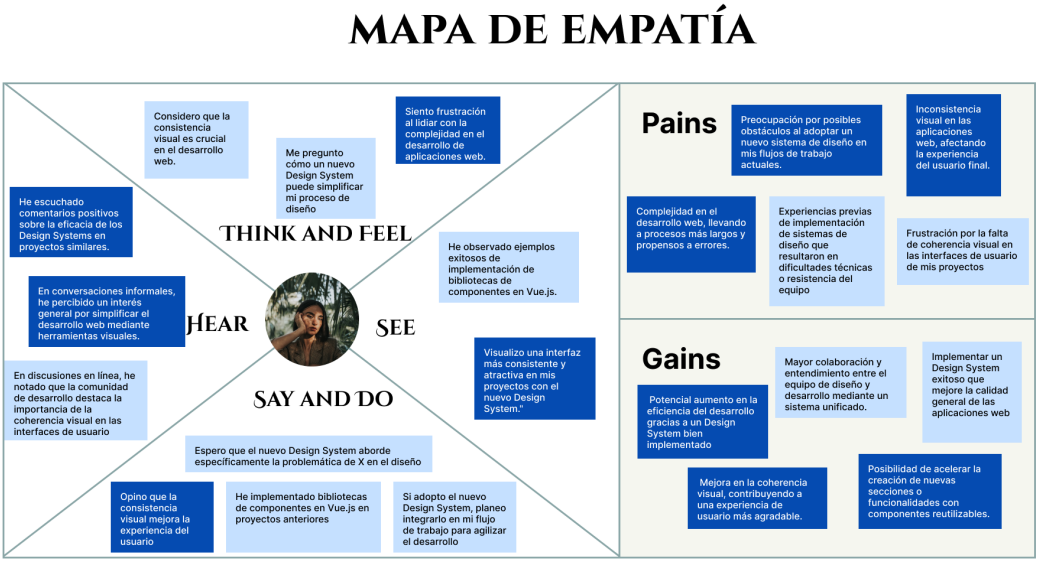

Figura 6 Mapa de empatía

El mapa de empatía se subraya la importancia de abordar las preocupaciones visuales y la necesidad de eficiencia en el proceso de desarrollo a través de la implementación de un sistema de diseño coherente, se espera mejorar acelerar la producción y elevar la calidad general de las interfaces de usuario, lo que resulta en una experiencia de usuario optimizada y más satisfactoria.

5.2.3 Análisis de competencia

La Figura 7 presenta un análisis de competencia entre tres sistemas de diseño diferentes: Carbon Design System, VueDS y PrimeVue. Se compara características clave como la facilidad de navegación, cantidad de componentes y la calidad de la documentación.

Figura 7 Análisis de competencia

Se evidencia que Carbon Design System ofrece un menú con jerarquía profunda y una amplia gama de componentes, VueDS se centra en lo esencial, ideal para proyectos que necesitan rapidez y eficiencia con componentes básicos. PrimeVue, con su organización clara y documentación moderada, ofrece un equilibrio entre ambos, enfocado en mejorar la visualización de la interfaz de usuario.

5.2.4 Site map

La Figura 8 proporcionada muestra un mapa del sitio detallado para el "Design System RD Vue". El mapa del sitio está organizado en cuatro secciones principales: "Getting Started", "Design Principles", "Design Tokens", y "Components". Cada sección principal está subdividida en subsecciones más específicas que abordan aspectos tales como resúmenes, documentación esencial, uso de componentes y clases de diseño, entre

otros. Además, la sección "Components" detalla varios elementos, lo cual permite a los desarrolladores acceder rápidamente a recursos específicos.

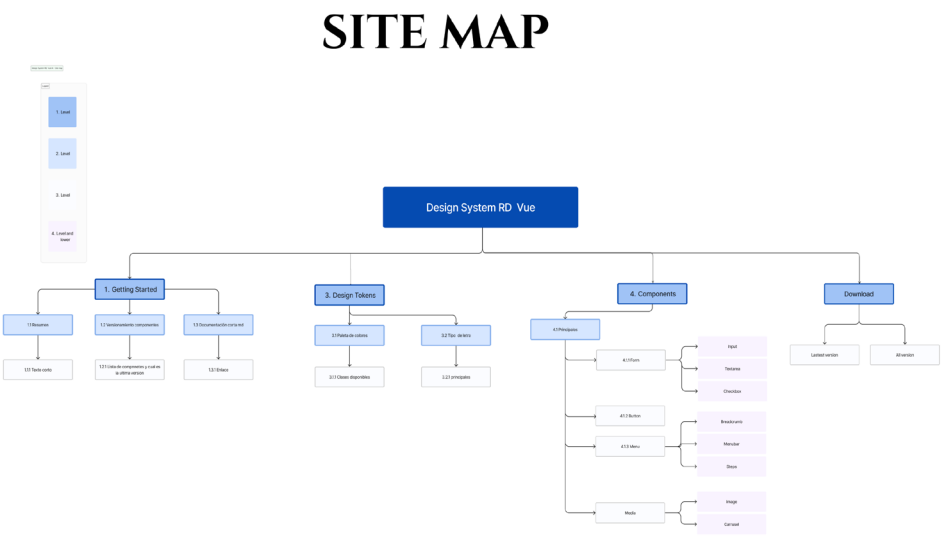

Figura 8 SIte map

Esto proporciona el uso coherente del sistema de diseño en proyectos de desarrollo, asegurando que todos los componentes y directrices están fácilmente accesibles y son comprensibles.

5.2.5 Journey map

La Figura 9 muestra un Journey Map que detalla varias fases del proceso de usuario, desde el descubrimiento hasta el compromiso continuo, pasando por la incorporación y la personalización. Cada fase incluye acciones específicas del usuario, puntos de contacto, sentimientos y pensamientos, puntos de dolor y posibles soluciones.

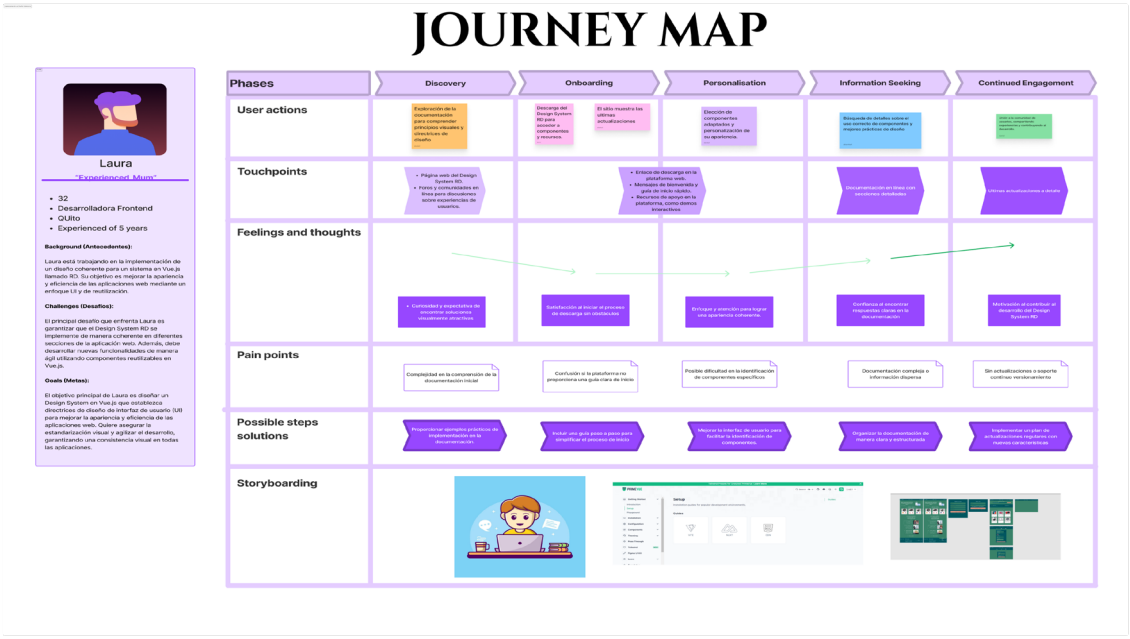

Figura 9 Journey Map

En un enfoque de relación entre “User person”, se detallada del "Journey Map" en el caso de Laura indica que para optimizar la experiencia de usuario en la implementación y uso del sistema de diseño en Vue.js, es crucial abordar las preocupaciones específicas identificadas en cada etapa del proceso:

- Durante la fase de descubrimiento y onboarding, Laura enfrenta dificultades con la complejidad de la documentación y la falta de claridad en la plataforma, lo que sugiere una necesidad urgente de simplificar y mejorar los recursos informativos disponibles.
- En las fases de personalización e información, la confusión sobre componentes específicos y la documentación incompleta son barreras significativas que impiden una integración fluida y eficiente.
- Por último, para fomentar un compromiso continuo, es esencial proporcionar soporte continuo y actualizaciones que mantengan el sistema relevante y alineado con las necesidades cambiantes de los usuarios como Laura.

Estos ajustes no solo mejorarán la satisfacción y eficiencia de Laura sino que también potenciarán la adopción general del sistema de diseño dentro de su entorno de trabajo.

5.2.6 Tareas (Userflows documentación)

La Figura 10 proporciona una visión comprensiva de cuatro flujos de usuario cruciales dentro de un sistema de diseño enfocado en Vue.js, destacando tareas específicas como la inspección visual, el desarrollo ágil con componentes reutilizables, la consulta de documentación, y la evaluación del impacto en usabilidad. Cada flujo de usuario está diseñado para facilitar tareas específicas dentro del ciclo de desarrollo de software, desde el inicio del proceso con la inspección visual hasta la evaluación final de la usabilidad y el rendimiento del sistema implementado.

Figura 10 Tareas

A continuación, se detalla un User Flow de una de las tareas.

a) User Flow 1: Inspección Visual

La Figura 11 muestra el User Flow 1, Inspección Visual, ilustra el proceso por el cual un usuario interactúa con una aplicación web para buscar y localizar información específica a través de la navegación visual o el uso del buscador. El flujo inicia cuando el usuario accede a la aplicación y tiene la opción de explorar categorías de documentación disponibles o utilizar el buscador. Si elige usar el buscador, ingresa un criterio de

búsqueda y el sistema muestra los resultados pertinentes; si no, navega a través del menú para encontrar la información deseada.

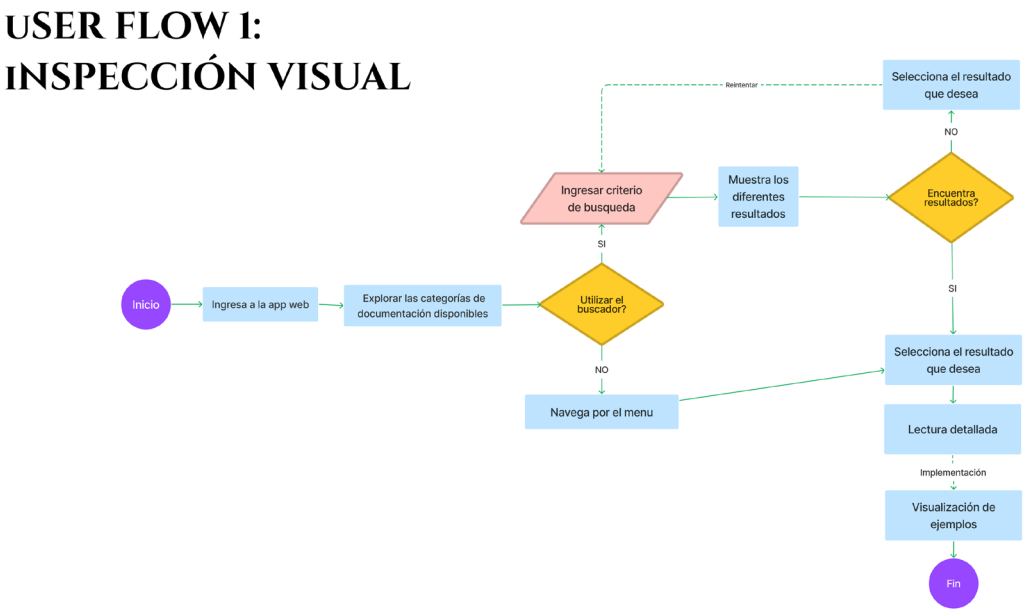

Figura 11 UserFlow1: Inspección visual

El flujo está diseñado para facilitar el acceso rápido y eficiente a la información, permitiendo al usuario elegir el método que mejor se ajuste a sus preferencias o necesidades inmediatas. La posibilidad de reiniciar la búsqueda o de navegar por diferentes resultados antes de seleccionar uno específico para una lectura detallada y la visualización de ejemplos, subraya la flexibilidad del sistema para adaptarse a varios estilos de interacción del usuario, asegurando que el proceso de búsqueda de información sea lo más intuitivo y productivo posible.

b) User Flow 2: Desarrollo ágil con componentes reutilizables

La Figura 12 el User Flow 2, "Desarrollo Ágil con Componentes Reutilizables", detalla el proceso que sigue un usuario para integrar y personalizar componentes dentro de un sistema de desarrollo, el flujo inicia con la selección de un componente específico que el usuario desea implementar en su proyecto. Posteriormente, el usuario tiene la opción de importar directamente el componente o modificarlo según sus necesidades específicas. Si elige personalizar el componente, ingresa los cambios deseados y el sistema muestra el resultado. Si está satisfecho con la personalización, procede a integrar el componente personalizado; de lo contrario, tiene la opción de revertir a la versión default del componente.

USER FLOW 2:
DESARROLLO ÁGIL CON COMPONENTES REUTILIZABLES

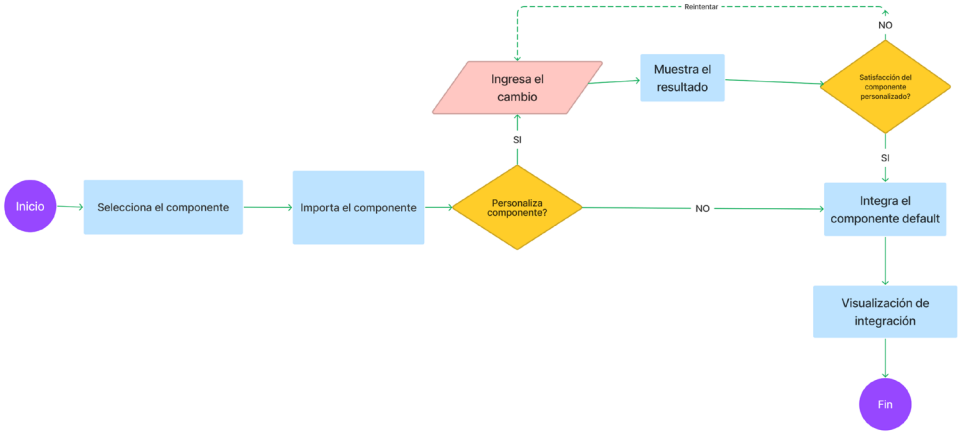

Figura 12 Desarrollo ágil con componentes reutilizables

c) User Flow 3: Consultar documentación

La Figura 13 muestra el User Flow 3, Consulta de Documentación, facilita el acceso eficiente a la información sobre componentes dentro de un sistema de diseño, permitiendo al usuario seleccionar entre documentación detallada o simplificada según sus necesidades específicas.

USER FLOW 3:
CONSULTA DE DOCUMENTACIÓN

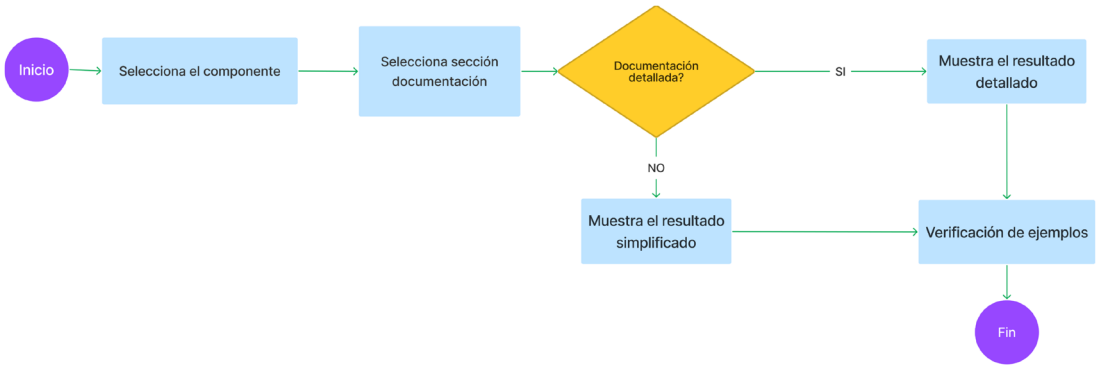

Figura 13 User Flow 3 Consulta de Documentación

El flujo comienza con la selección de un componente, seguido de la elección de la profundidad de la documentación, y culmina con la verificación de ejemplos prácticos para asegurar una comprensión clara y aplicación efectiva del componente, el diseño estratégico del flujo de usuario permite una personalización rápida y eficiente del aprendizaje, mejorando significativamente la usabilidad y la eficacia en la implementación de los sistemas de diseño.

d) User Flow 4: Evaluación del impacto en usabilidad

La Figura 14 muestra el User Flow 4, Evaluación del Impacto en Usabilidad, detalla el proceso sistemático para evaluar cómo las implementaciones del sistema de diseño afectan la usabilidad de una aplicación. El flujo comienza con la selección de una sección específica de la aplicación que ha integrado el sistema de diseño. A continuación, se llevan a cabo pruebas de usabilidad para evaluar la efectividad de estas implementaciones. Basándose en los resultados de estas pruebas, el usuario determina si el impacto ha sido positivo. Si las evaluaciones no son satisfactorias, se sugieren ajustes basados en los hallazgos de las pruebas para mejorar la interfaz.

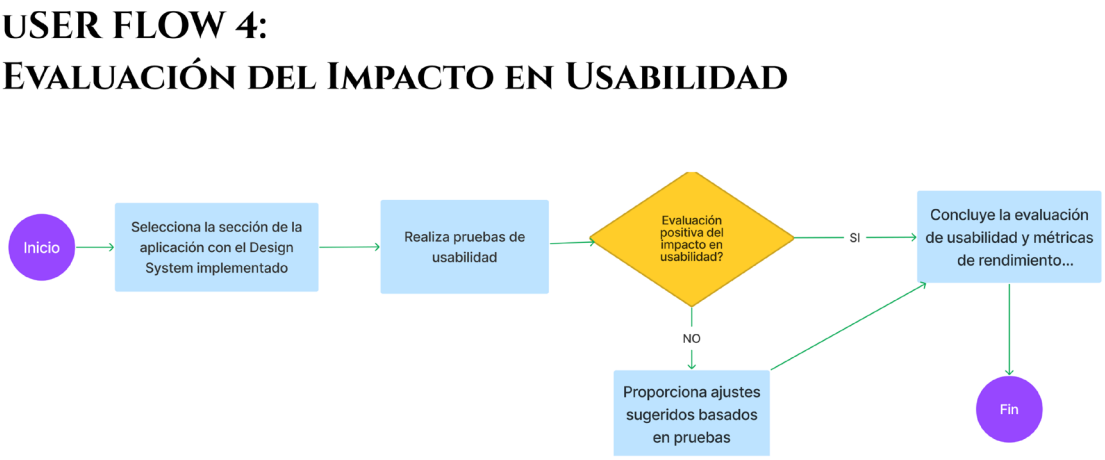

Figura 14 Evaluación del impacto en usabilidad

El flujo es crucial para el desarrollo iterativo y centrado en el usuario para realizar mejoras continúas basadas en feedback real y medido. Dicho proceso no solo ayuda a identificar problemas de usabilidad, sino que también garantiza que las soluciones aplicadas mejoren efectivamente la experiencia del usuario final. Al final del flujo, se

concluyen las evaluaciones, proporcionando un punto de reflexión y ajuste antes de la finalización y el lanzamiento de la sección o producto evaluado.

5.3 Contribución de herramientas de desarrollo (componentes-rd)

- Resultados: Vue.js permitió un desarrollo rápido y eficiente de componentes reutilizables que se integraron perfectamente en varias aplicaciones, manteniendo la coherencia visual y funcional.
- Discusión: Se desarrollo los componentes básicos que requiere un Design System, siendo Vue.js el framework basado en JavaScript con una curva de aprendizaje menor y una capacidad de adaptabilidad a cualquier proyecto.

5.3.1 GitHub

- Resultados: Proveyó una plataforma robusta para la gestión de versiones, asegurando que todos los cambios estuvieran bien documentados y que el código fuente fuera accesible.
- Discusión: La integración con herramientas de CI/CD facilitó despliegues consistentes y automatizados, aunque se requirió configuración inicial y supervisión para optimizar estos procesos.

La Figura 15 evidencia el repositorio de GitHub "components-rd" se presenta como un desarrollo activo, evidenciado por su actividad reciente en commits y la organización de su estructura de archivos.

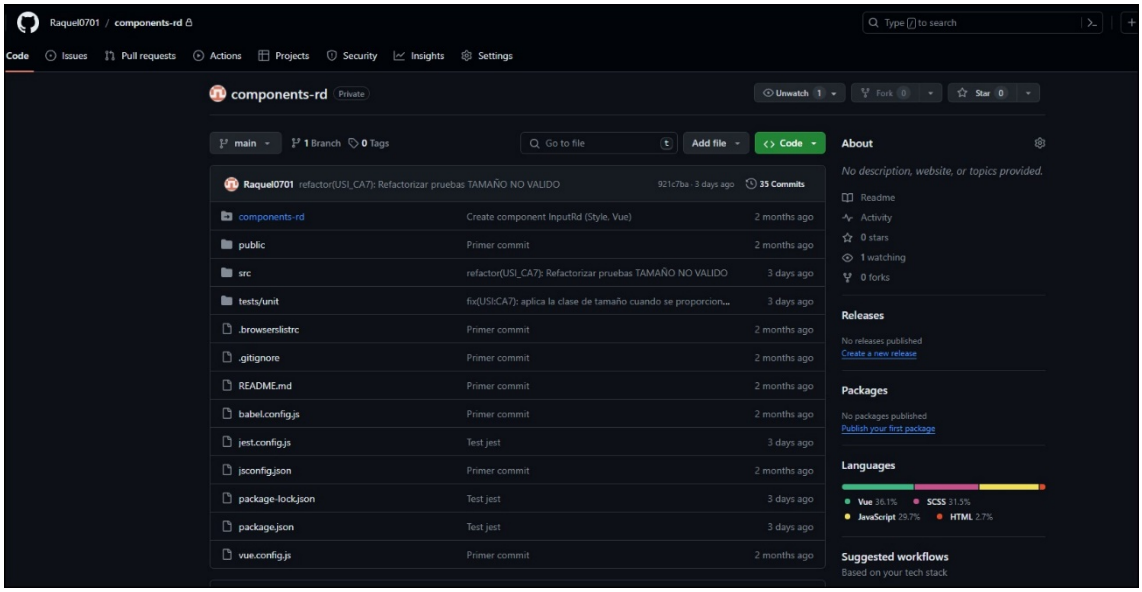

Figura 15 Reposotorio GitHub: components-rd

- Accesso de git al proyecto “Componentes-RD”: https://github.com/Raquel0701/components-rd

A continuación, se detalla aspectos de versionamiento, commits, y herramientas utilizadas en el proyecto:

- Versionamiento y Commits
  - Commits: Hay un total de 35 commits, mostrando una actividad regular en el desarrollo. Los mensajes de commit varían desde "Primer commit" hasta ajustes específicos como "Refactorizar pruebas TAMAÑO NO VALIDO" y "fix(USI-CA7): aplica la clase de tamaño cuando se proporciona...", lo que sugiere una iteración constante y correcciones en las funcionalidades.
  - Branches: un único branch principal llamado 'main', lo que evidencia que es proyecto está en sus fases iniciales y gestionado individualmente.
- Herramientas y Configuración
  - .gitignore: Utilizado para excluir archivos del seguimiento de Git, esencial para mantener limpio el repositorio de archivos no necesarios o sensibles.

- babel.config.js: Configuración de Babel que ayuda en la transpilación del código JavaScript moderno para que sea compatible con versiones antiguas de navegadores.
- jest.config.js: Proporciona la configuración para Jest, que es la herramienta usada para las pruebas unitarias, indicando cómo se deben ejecutar estas pruebas dentro del proyecto.
- vue.config.js: Configura aspectos específicos de Vue CLI, que es crucial para personalizar el proceso de compilación y desarrollo del proyecto.

- Uso de Tecnologías
  - Lenguajes: El proyecto utiliza principalmente Vue.js (36.1%), SCSS (31.5%), y JavaScript (29.7%), con un pequeño componente de HTML (2.7%). Esta distribución sugiere un fuerte enfoque en el desarrollo frontend con componentes interactivos y estilos avanzados.
- Archivos y Directorios Claves
  - src: Contiene el código fuente del proyecto, incluyendo componentes Vue y estilos SCSS.
  - tests/unit: Referente a pruebas unitarias, verifica la calidad y funcionamiento a medida que el proyecto evoluciona.
  - package.json y package-lock.json: Estos archivos son esenciales para la gestión de dependencias del proyecto, asegurando la consistencia entre diferentes entornos de desarrollo.

5.3.2 Dependencias

La Tabla 1 detalla las diferentes dependencias utilizadas lo cual aprovecha la robustez de Vue.js para el desarrollo, incorporando herramientas cruciales como Vue CLI para la gestión y construcción del entorno de desarrollo, Jest para pruebas unitarias y Sass para estilos avanzados.

Tabla 5 Dependencias y DevDependencias

Tipo	Nombre del Paquete	Versión	Descripción
Dependencias	vue	^3.2.13	Framework progresivo para construir interfaces de usuario.

Dependencias	core-js	^3.8.3	Proporciona polyfills necesarios para la compatibilidad de ECMAScript en navegadores antiguos.
Dev Dependencias	@vue/cli-service	~5.0.0	Herramienta estándar de Vue CLI para el desarrollo y la construcción de proyectos Vue.
Dev Dependencias	@vue/test-utils	^2.0.0-0	Utilidades oficiales para probar componentes Vue, compatibles con la versión 3 de Vue.
Dev Dependencias	jest	^27.0.5	Framework de pruebas para JavaScript utilizado para garantizar la calidad del código a través de pruebas automatizadas.
Dev Dependencias	sass	^1.32.7	Preprocesador CSS que permite el uso desintaxis avanzada de SASS en estilos.
Dev Dependencias	rollup	^2.79.1	Empaquetador de módulos JavaScript utilizado para la creación de bundles de producción eficientes.
Dev Dependencias	@rollup/plugin-node-resolve	^15.2.3	Plugin de Rollup para resolver módulos de Node.js durante el proceso de empaquetado.
Dev Dependencias	@rollup/plugin-commonjs	^25.0.7	Plugin de Rollup para convertir módulos CommonJS a ES6, facilitando la compatibilidad.

Estas herramientas son esenciales para mantener un flujo de trabajo eficiente y garantizar la calidad del software en desarrollos modernos de interfaces de usuario.

5.3.3 Estructura del proyecto

La Figura 16 describe un diagrama que representa la estructura del proyecto ilustra la organización de los directorios del proyecto, mostrando claramente la distribución de los componentes Vue, estilos SCSS y pruebas Jest en el directorio src y las pruebas unitarias bajo tests.

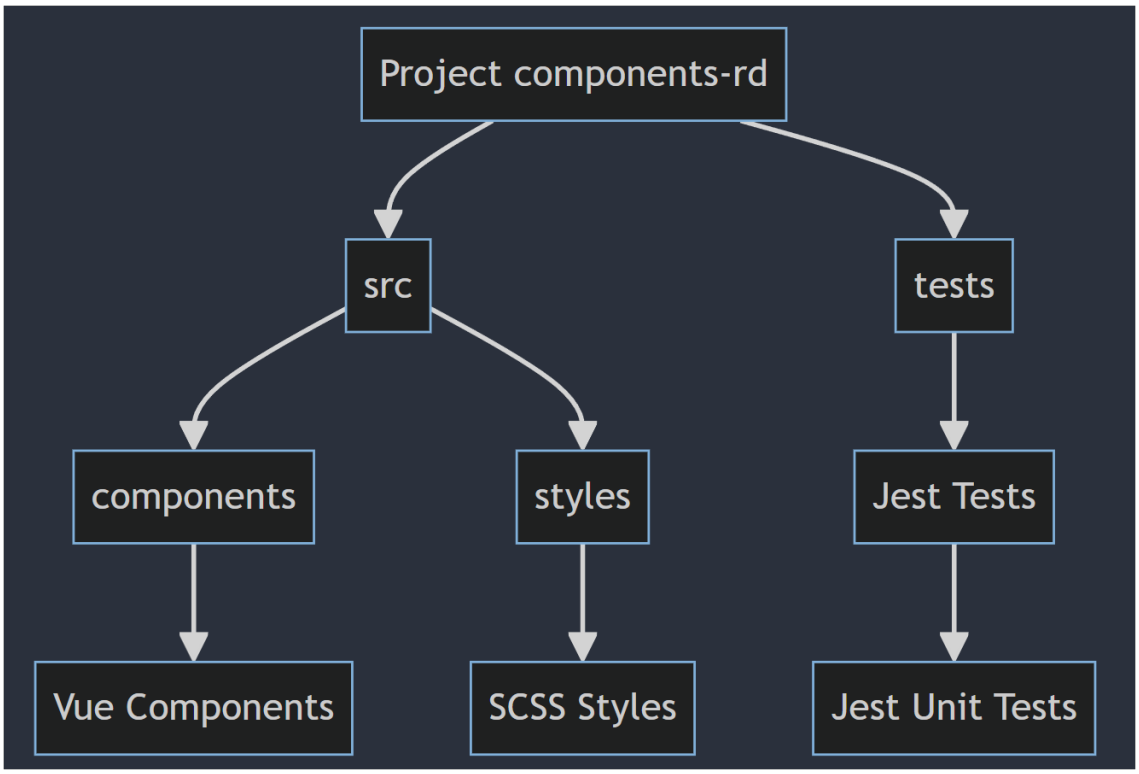

Figura 16 Estructura del Proyecto components-rd

5.3.4 Desarrollo de un componente (Button)

La Figura 17 describe el componente ButtonRd que está estructurado para utilizar las funcionalidades avanzadas de Vue.js y proporciona una forma flexible y reutilizable de implementar botones con diferentes estilos y comportamientos.

Figura 17 Creación del componente ButtonRd

A continuación, se detalla la estructura que fue aplicada:

- Template
  - El template del componente contiene un único elemento button que utiliza:

  - :class="computedClasses": Aplica clases dinámicas basadas en una propiedad computada, lo que permite que el estilo del botón cambie según la propiedad severity.
  - @click="handleClick": Define un manejador de eventos que se ejecuta cuando el botón es clickeado. El manejador llama a la función handleClick() definida en las opciones del componente.
- Script

El script del componente define varias opciones:

  - props:

        - severity: Una propiedad que acepta un String. Su valor por defecto es "primary". También incluye un validador que asegura que el valor asignado esté dentro de un conjunto permitido de valores ("primary", "secondary", "success", "info", "warning", "help", "danger", "contrast").
- methods:
    - handleClick(): Método que se activa cuando el usuario hace clic en el botón. El método a su vez llama a emitAction().
    - emitAction(): Emite un evento personalizado llamado 'action', que puede ser escuchado por componentes padres o en otros lugares donde se utilice dicho botón.
- computed:
    - computedClasses(): Una propiedad computada que retorna una cadena de clases CSS basada en la severity. El resultado es una clase en el formato "button-rd button-${this.severity}", donde ${this.severity} es reemplazado por el valor actual de la propiedad severity.

Vue.js puede ser utilizado para crear interfaces de usuario dinámicas y reutilizables, es decir, se evidencia la lógica encapsulada, lo que facilita su mantenimiento y reutilización en diferentes partes de una aplicación. Además, la utilización de propiedades computadas y el manejo de eventos hacen que el componente sea interactivo y adaptable a las necesidades específicas del entorno en el que se utilice.

Además, la Figura 18 muestra el estilo para el componente Button-rd

ButtonRd.vue M _button.scss M

src > styles > _button.scss > .button-rd

```scss
// _button-rd.scss

@import "variables";
@import "sizes";

.button-rd {
  border-radius: 6px;
  padding: 0.85rem 1.2rem;
  margin: 2px;
  font-size: $text-medium;
  cursor: pointer;
  border: 0px solid $color-contrast;
  transition: background-color 0.2s, color 0.2s, border-color 0.2s,
    box-shadow 0.2s, outline-color 0.2s;
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
  outline-color: transparent;

  &:hover {
    background-color: $color-contrast-hover;
    border-color: $color-contrast-hover;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
  }
}

.button-primary {
  background-color: $color-primary;
  color: #fff;

  &:hover {
    background-color: $color-primary-hover;
  }
}
```

Figura 18 Estilo Scss componente ButtonRd

5.4 Impacto de las pruebas

5.4.1 Jest y TDD

Resultados: La aplicación de TDD y el uso de Jest para pruebas unitarias aseguraron que los componentes del sistema de diseño fueran robustos y funcionaran según lo esperado antes de su publicación.

Discusión: Aunque las pruebas unitarias aumentaron la calidad del código, el tiempo necesario para escribir y mantener estas pruebas fue significativo.

5.4.1 Ejemplo aplicado (Button-fix)

A continuación, se evidencia la aplicación al componente ya mencionado:

La Figura 19 muestra una prueba fallida donde se esperaba que un botón emitiera un evento 'action' al hacer clic en él, pero no lo hizo, resultando en un valor undefined.

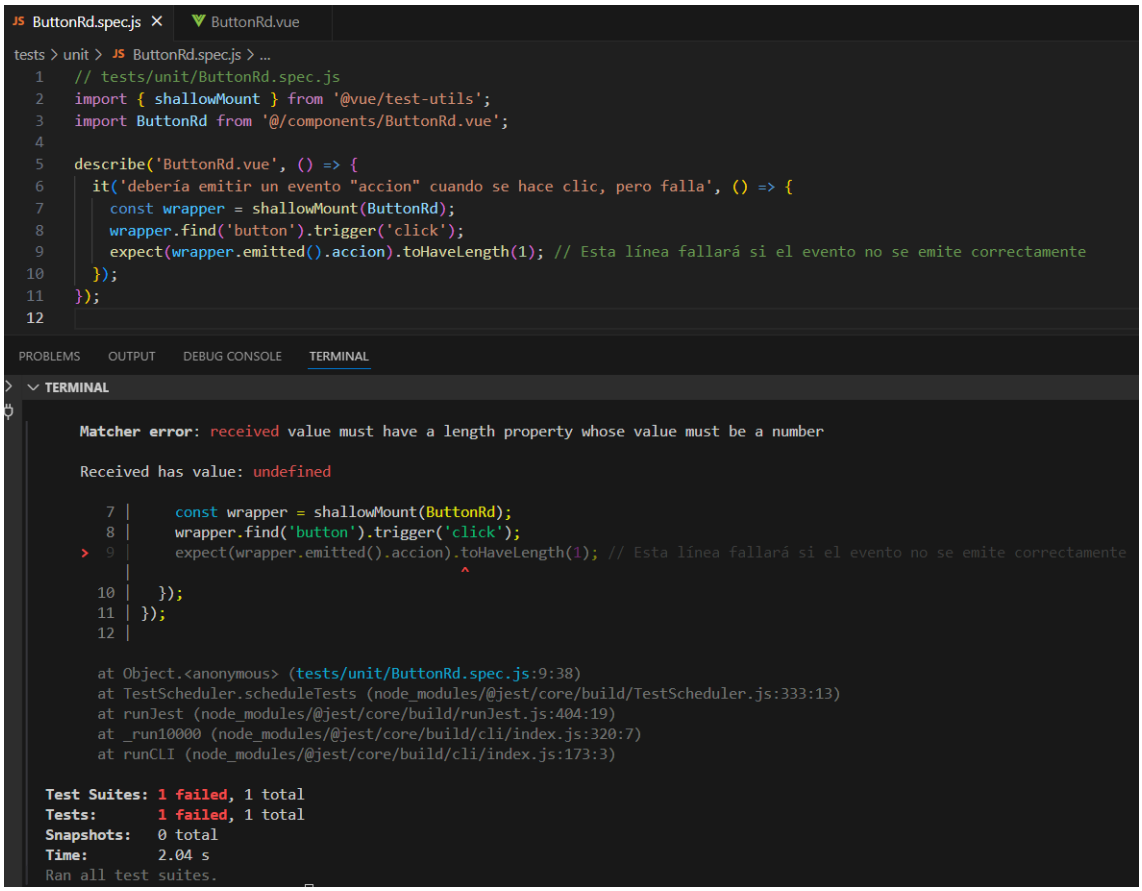

Figura 19 Prueba Fallida

Esta imagen ilustra un problema común en el desarrollo de software: un componente que no funciona como se espera. La necesidad de ajustar el código para que emita el evento correctamente es fundamental antes de proceder.

5.4.2 Ejemplo aplicado (Button-feat)

La Figura 20 muestra que después de hacer los ajustes necesarios en el código, las pruebas pasan exitosamente, indicando que el botón ahora emite el evento 'action' correctamente y aplica la clase correcta basada en una propiedad

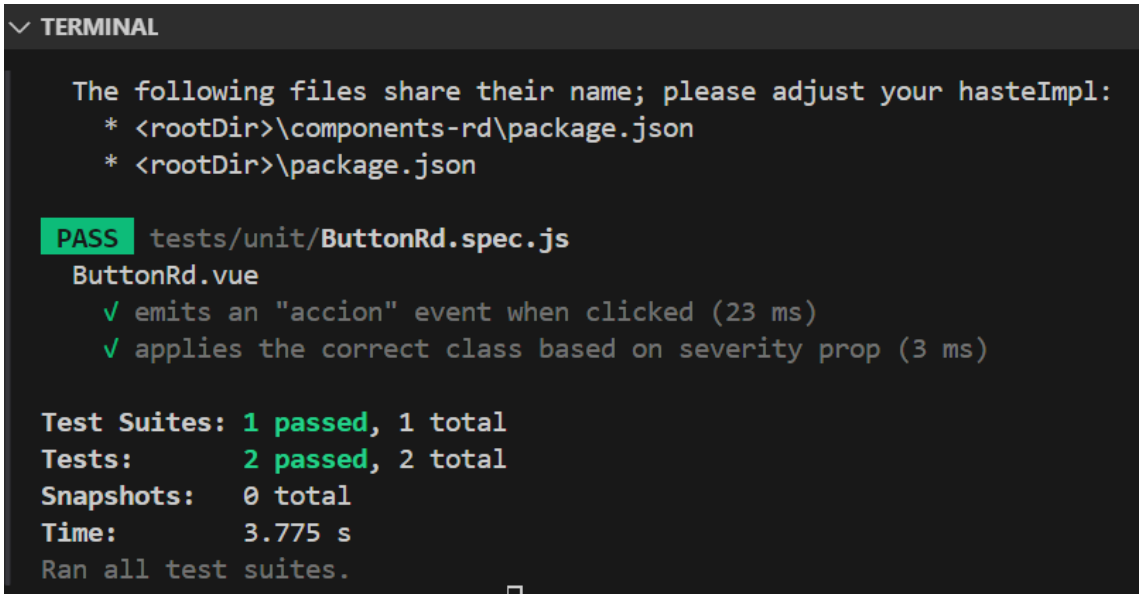

Figura 20 Prueba superada con éxito

Es decir, el enfoque de TDD lleva a la resolución de errores y refuerza la calidad del software, asegurando que el componente cumpla con las especificaciones deseadas.

5.4.1 Ejemplo aplicado (Button-refactor)

La Figura 21 muestra el código final del componente ButtonRd.vue tras aplicar un proceso de refactorización dentro de un contexto de Desarrollo Guiado por Pruebas (TDD), en esta versión del código, se aplica una estructura clara y bien organizada del componente, que ha sido diseñada para ser eficiente y fácil de mantener.

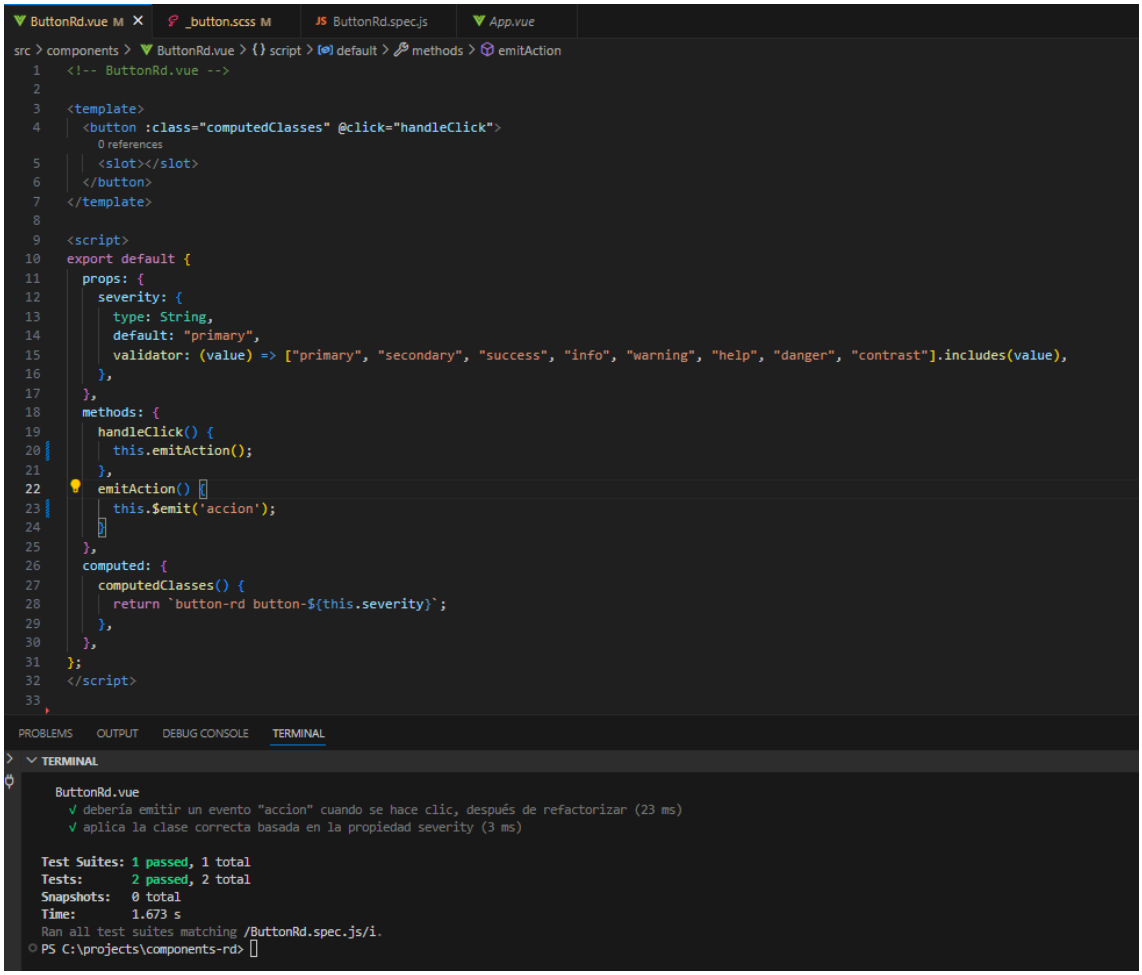

Figura 21 Refactor

Esta refactorización asegura que el componente no solo funcione como se espera en términos de funcionalidad, sino que también sea escalable y fácil de gestionar a largo plazo, reflejando los principios clave del TDD de escribir código limpio y mantenible desde el inicio.

5.5 Impacto de la documentación

5.4.2 Documentación con Markdown (VuePress)

- Resultados: La documentación fue fácil de mantener y actualizar, proporcionando un recurso valioso para los desarrolladores y usuarios finales del sistema de diseño.

- Discusión: La necesidad de mantener la documentación actualizada con cada cambio en el sistema fue un desafío, especialmente en fases de desarrollo rápido.
- Acceso al proyecto "Documentación RD": https://github.com/Raquel0701/documentation-rd

3.1.1 Estructura basada en el site map

La estructura de la documentación ya fue realizada en la fase de diseño, lo cual, se ha aplicado esa visión con algunos cambios solo de forma, como la visualización horizontal del menú, sin embargo, las secciones son las mismas.

- **Página principal**

La Figura 22 muestra una visión general de la página principal. Es ideal para que los usuarios entiendan rápidamente el contexto y el propósito de la página al verla.

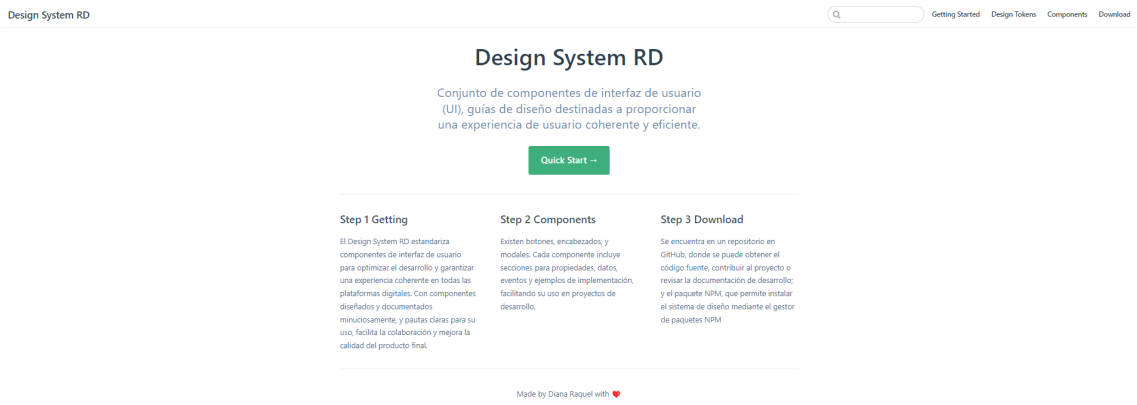

Figura 22 Documentación Design System RD, página Principal: Una Visión General

- **Getting Started**

La Figura 23 muestra la sección de introducción o de inicio de la documentación del Design System RD, abordando tanto los conceptos básicos del sistema como una visión general de sus componentes y filosofía de diseño.

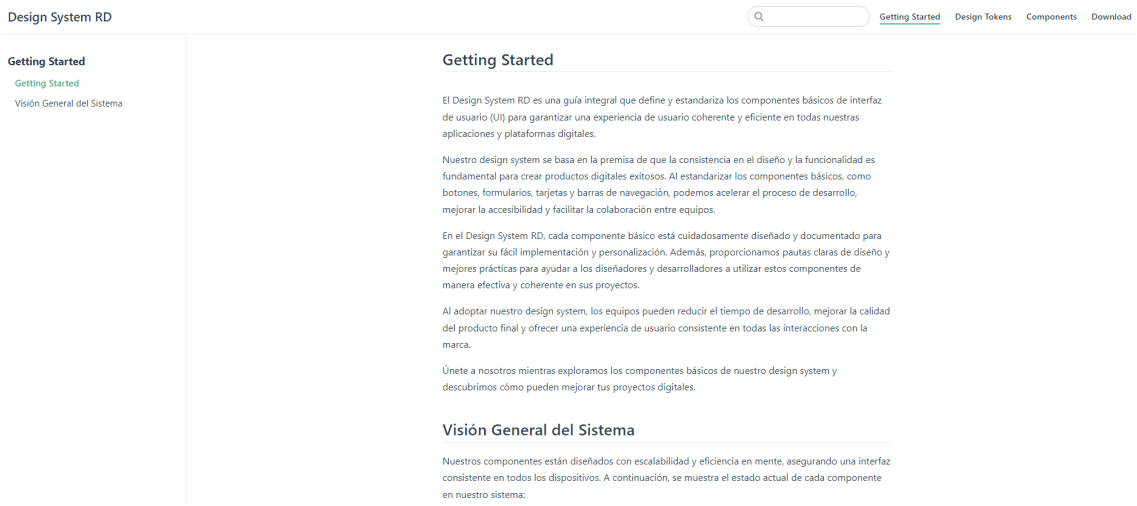

Figura 23 Documentación Design System RD, Sección de Inicio: Detalles y Visión General

- **Design Tokens**

La Figura 24 detalla aspectos fundamentales como colores, espaciados y tipografía que son cruciales para mantener la coherencia visual en todas las interfaces de usuario.

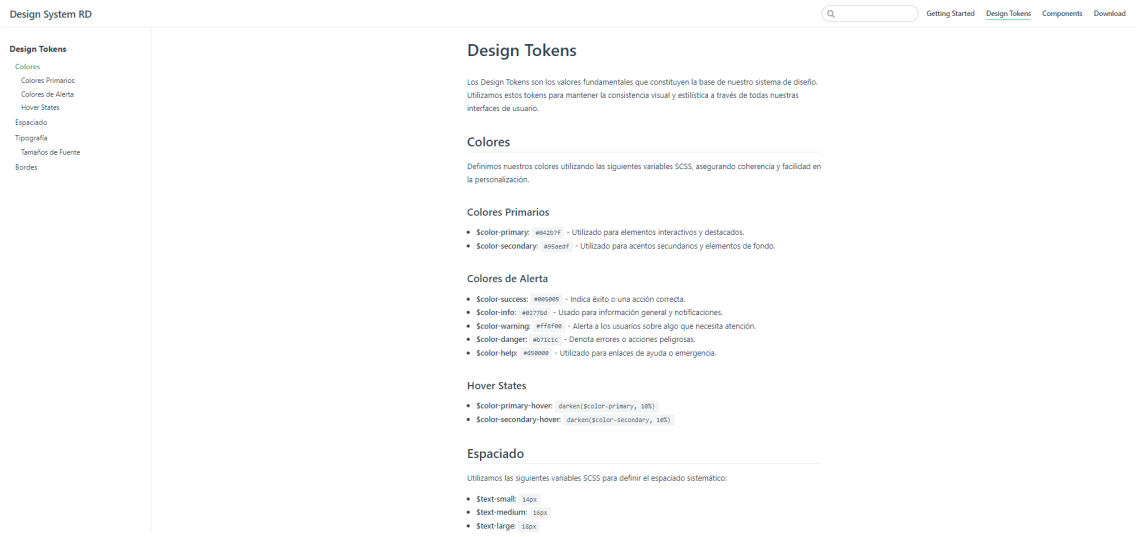

Figura 24 Documentación Design System RD, Sección de Design Tokens: Definiciones y Usos

- **Componentes**

La Figura 25 muestra la sección específica de la documentación que aborda los detalles de los componentes, incluyendo sus propiedades y ejemplos de implementación. Esto ofrece una visión clara del uso y las opciones de personalización.

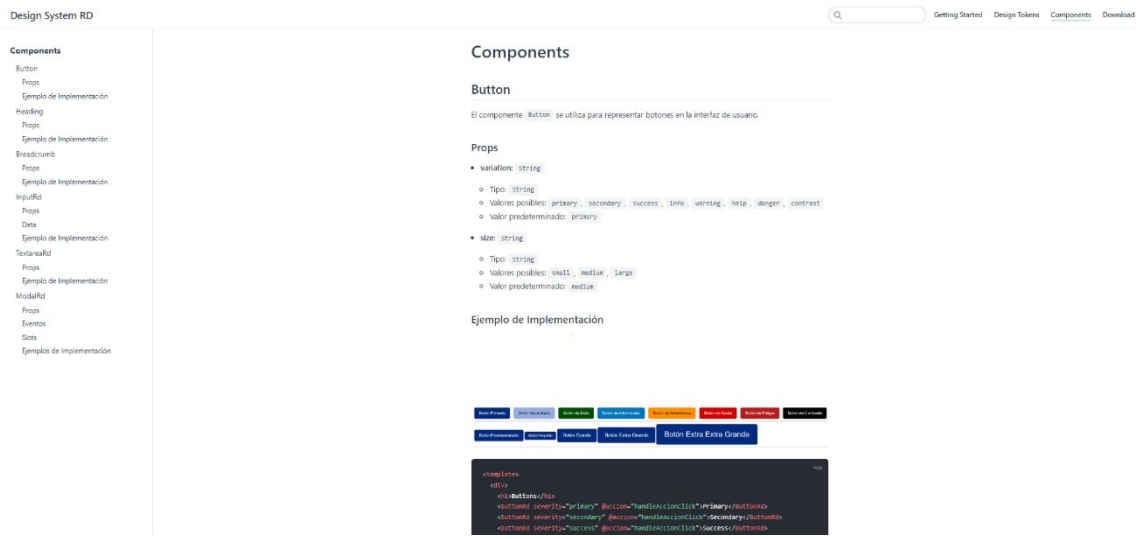

Figura 25 Documentación Design System RD, Sección de Componentes: Detalles

- **Componente Button:** Descripción y propiedades

La Figura 26 muestra la documentación específica para el componente de botón, incluyendo detalles sobre las propiedades variation y size que definen su apariencia y comportamiento.

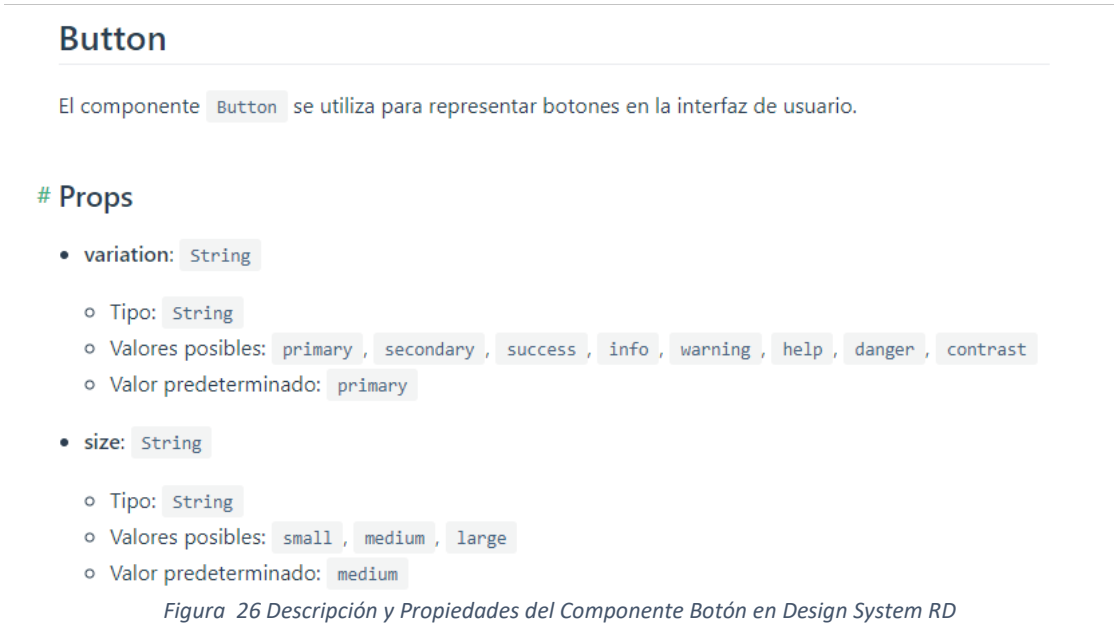

Figura 26 Descripción y Propiedades del Componente Botón en Design System RD

- **Componente Button:** Ejemplo de implementación

La Figura 27 detalla el código utilizado para implementar varios tipos de botones con diferentes propiedades de severidad y tamaños, proporcionando una visión clara de cómo se pueden utilizar estos componentes en el desarrollo práctico

```vue
<template>
  <div>
    <h1>Buttons</h1>
    <ButtonRd severity="primary" @accion="handleAccionClick">Primary</ButtonRd>
    <ButtonRd severity="secondary" @accion="handleAccionClick">Secondary</ButtonRd>
    <ButtonRd severity="success" @accion="handleAccionClick">Success</ButtonRd>
    <ButtonRd severity="info" @accion="handleAccionClick">Info</ButtonRd>
    <ButtonRd severity="warning" @accion="handleAccionClick">Warning</ButtonRd>
    <ButtonRd severity="help" @accion="handleAccionClick">Help</ButtonRd>
    <ButtonRd severity="danger" @accion="handleAccionClick">Danger</ButtonRd>
    <ButtonRd severity="contrast" @accion="handleAccionClick">Contrast</ButtonRd>
    <br />
    <button class="button-rd button-primary">Botón Predeterminado</button>
    <button class="button-rd button-primary button-sm">Botón Pequeño</button>
    <button class="button-rd button-primary button-lg">Botón Grande</button>
    <button class="button-rd button-primary button-xl">
      Botón Extra Grande
    </button>
    <button class="button-rd button-primary button-xxl">
      Botón Extra Extra Grande
    </button>
  </div>
</template>

<script>
import ButtonRd from "@/components/ButtonRd.vue";

export default {
  components: {
    ButtonRd,
  },
  methods: {
    handleAccionClick() {
      console.log("Botón con acción clicado");
    },
  },
};
</script>
```

Figura 27 Ejemplo de Código para Botones en Design System RD

- **Download**

La Figura 28muestra la sección de descarga del Design System, proporcionando detalles sobre cómo obtener los componentes a través del repositorio de GitHub o mediante el paquete NPM, facilitando a los usuarios el acceso y la implementación del sistema en sus proyectos.

Figura 28 Documentación Design System RD, opciones de descarga: GitHub y NPM

5.5 Publicación y uso práctico

5.6.1 Configuración rollup (build)

La Figura 29 muestra la configuración de Rollup parece bien diseñada para un proyecto de librería de componentes Vue que utiliza SCSS para estilos y desea exportar tanto en formatos ESM (ECMAScript Modules) como UMD (Universal Module Definition.

```
// Importar todos los plugins necesarios
import vue from 'rollup-plugin-vue';  // Manejo de archivos .vue
import commonjs from '@rollup/plugin-commonjs';  // Convertir módulos CommonJS a ES6
import scss from 'rollup-plugin-scss';  // Compilar SCSS a CSS
import { terser } from 'rollup-plugin-terser';  // Minificar archivos JS

export default {
  input: 'src/components/index.js', // Archivo de entrada para tu biblioteca
  output: [
    {
      file: 'dist/components-rd.esm.js',
      format: 'esm', // Formato de módulo ES
      sourcemap: true // Generar mapas de código fuente
    },
    {
      file: 'dist/components-rd.min.js',
      format: 'umd', // Formato UMD para uso en navegadores y Node
      name: 'ComponentsRd', // Nombre de la variable global en UMD
      globals: {
        vue: 'Vue'
      },
      sourcemap: true,
      plugins: [terser()] // Aplicar minificación
    }
  ],
  plugins: [
    vue({
      css: false, // No inyectar CSS directamente en JavaScript
      template: {
        isProduction: true,
      },
      style: {
        preprocessOptions: {
          scss: {
            additionalData: `@import "src/styles/variables.scss";`
          }
        }
      }
    }),
    scss({
      output: 'dist/components-rd.css',
      outputStyle: 'compressed',
      failOnError: true,
      verbose: true
    }),
    commonjs(),
  ],
  external: ['vue'], // No incluir Vue en el bundle, usarlo como externo
};
```

Figura 29 Configuración de Rollup

La Figura 30 muestra la configuración del Pakcage.json se detalla información para su publicación en npm. La Figura muestra la configuración final del proyecto.

```
{
  "name": "components-rd",
  "version": "1.0.6",
  "private": false,
  "keywords": [
    "vue",
    "vuejs",
    "design-system",
    "ui-kit",
    "component-library",
    "frontend",
    "css"
  ],
  "author": "Diana Diaz",
  "license": "MIT",
  "main": "dist/components-rd.esm.js",
  "files": [
    "dist/*.js",
    "dist/*.css"
  ],
  ▷ Debug
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "rollup -c --verbose",
    "test:unit": "vue-cli-service test:unit"
  },
  "dependencies": {
    "core-js": "^3.8.3",
    "vue": "^3.2.13"
  },
  "peerDependencies": {
    "vue": "^3.2.13"
  },
  "devDependencies": {
    "@rollup/plugin-commonjs": "^25.0.7",
```

Figura 30 Configuración package.json

5.6.2 NPM

- Resultados: Publicar el sistema de diseño en npm facilitó su distribución y adopción por parte de otros proyectos y desarrolladores.
- Discusión: Se enfrentaron desafíos en la gestión de versiones y dependencias, especialmente cuando se introdujeron actualizaciones incompatibles con versiones anteriores.

La Figura 31 muestra la última versión publicada 1.0.6 para el paquete components-rd, en un enfoque de versionamiento SemVer (Semantic Versioning), se ha realizado un incremento de PATCH.

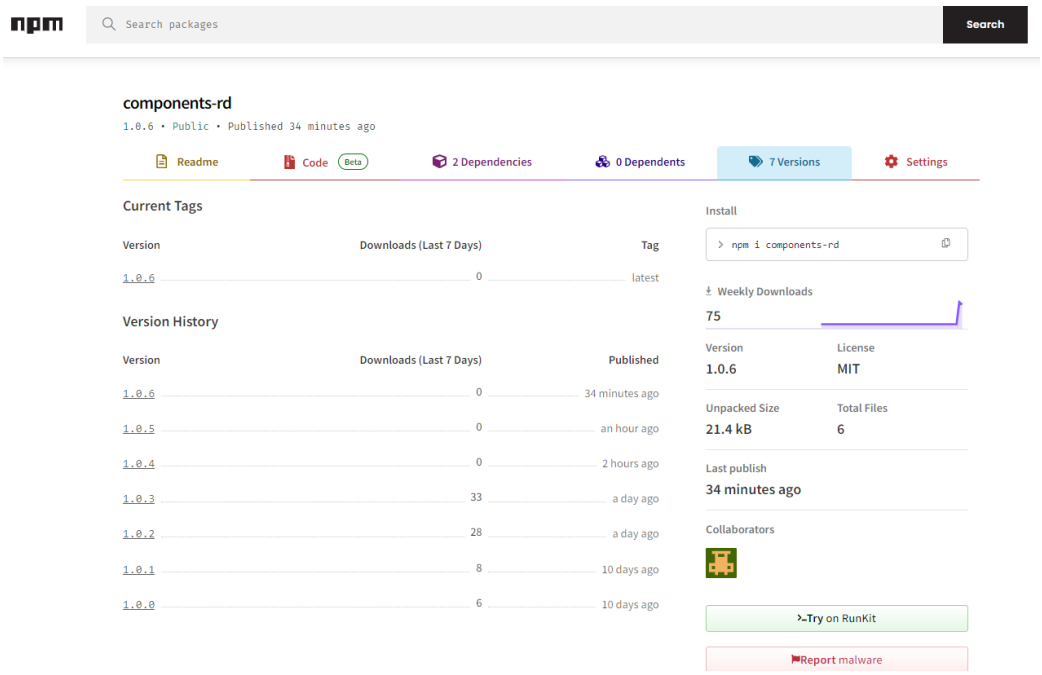

Figura 31 Publicado como librería NPM (incremento de PATCH)

5.6.3 Ejemplo aplicado

- Resultados: La aplicación práctica del sistema de diseño para construir un formulario demostró su eficacia y la facilidad de uso.
- Discusión: Esta implementación sirvió como un caso de prueba vital para validar la funcionalidad completa del sistema en un entorno de producción real.
- Acceso al proyecto "Implementación RD": https://github.com/Raquel0701/implementacion-rd

La Figura 32 muestra la aplicación de algunos de los componentes disponibles en el Design System RD, un formulario básico, la implementación de los demás se encuentra en la documentación.

Figura 32 Ejemplo aplicado con Design System RD

Se evidencia una reutilización y uniformidad en el diseño, esto optimiza tiempo en el diseño de componentes y tiempo de los desarrolladores, teniendo una guía clara de implementación.

6. Conclusiones

El desarrollo se dividió en tres proyectos interconectados: "Componentes-RD", "DocumentationRD" y "Implementacion". Esta división facilitó la especialización en cada área, permitiendo un enfoque detallado y profundo en la creación de componentes, su documentación y la implementación práctica. La sincronización entre estos proyectos aseguró que las mejoras en uno se reflejaran adecuadamente en los otros.

El proyecto "Componentes-RD" demostró cómo una biblioteca de componentes bien diseñada y modular puede mejorar significativamente la eficiencia del desarrollo web. La arquitectura escalable y la reusabilidad de los componentes permitieron una integración fluida y una coherencia visual en todas las aplicaciones que los utilizan.

El proyecto "DocumentationRD" fue crucial para el éxito del sistema de diseño. La documentación detallada y accesible creada en VuePress facilitó a los desarrolladores entender y utilizar los componentes de manera efectiva. Esta documentación es un recurso valioso para la capacitación y el soporte continuo del equipo de desarrollo.

El proyecto "Implementacion" sirvió como un campo de pruebas vital para los componentes desarrollados, permitiendo la validación de su funcionalidad en un entorno real. Esta fase no solo probó la eficacia de los componentes sino que también destacó la importancia de la iteración y el ajuste basado en el feedback real de uso.

La implementación de SCSS para la estilización y el uso de herramientas de desarrollo como Babel y Jest subrayaron el compromiso del proyecto con las buenas prácticas y la calidad del código. Estas herramientas mejoraron la eficiencia del desarrollo y aseguraron que los componentes funcionaran de manera óptima en varios entornos.

Mirando hacia el futuro, el Design System RD está preparado para expandirse y adaptarse a nuevas tecnologías y requisitos del mercado. La base sólida establecida por los tres proyectos facilita la inclusión de mejoras y la expansión del sistema, asegurando su relevancia y utilidad continua en el desarrollo de aplicaciones web.

En conclusión, se destaca la importancia de una planificación meticulosa, la colaboración entre diferentes proyectos y el compromiso con la calidad y la coherencia, que juntas forman la base para el éxito continuo del Design System RD en un entorno de desarrollo web dinámico.

7. Referencias

[1] Y. Goldenberg and N. Tractinsky, "Prioritizing UI Items for Bidirectional Design System Development: A Suggested Set of Criteria and its Implementation," *Conference on Human Factors in Computing Systems - Proceedings*, Apr. 2022, doi: 10.1145/3491101.3519868.

[2] J. Edelberg and J. Kilrain, "Design systems: Consistency, efficiency & collaboration in creating digital products," *SIGDOC 2020 - Proceedings of the 38th ACM International Conference on Design of Communication*, Oct. 2020, doi: 10.1145/3380851.3416743.

[3] L. M. Hasani, D. I. Sensuse, Kautsarina, and R. R. Suryono, "User-Centered Design of e-Learning User Interfaces: A Survey of the Practices," *2020 3rd International Conference on Computer and Informatics Engineering, IC2IE 2020*, pp. 299–305, Sep. 2020, doi: 10.1109/IC2IE50715.2020.9274623.

[4] S. Chandran, A. Al-Sa'di, and E. Ahmad, "Exploring User Centered Design in Healthcare: A Literature Review," *4th International Symposium on Multidisciplinary Studies and Innovative Technologies, ISMSIT 2020 - Proceedings*, Oct. 2020, doi: 10.1109/ISMSIT50672.2020.9255313.

[5] F. Miao, "Website UI Design Interactivity and Interface Visual Design Based on Improved DFA Algorithm," *2nd IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics, ICDCECE 2023*, 2023, doi: 10.1109/ICDCECE57866.2023.10151237.

[6] A. S. Addany, N. I. Pradana, S. P. Putra Prabowo, and I. S. Widiati, "UI/UX Design of e-Wallet Appllication Using Design Thinking Approach," *2022 4th International Conference on Cybernetics and Intelligent System, ICORIS 2022*, 2022, doi: 10.1109/ICORIS56080.2022.10031309.

[7] M. Christian Klimm and S. Bente, "Design Systems for Micro Frontends - An Investigation into the Development of Framework-Agnostic Design Systems using Svelte and Tailwind CSS," Feb. 2021, Accessed: Dec. 11, 2023. [Online]. Available: https://epb.bibl.th-koeln.de/frontdoor/index/index/docId/1666

[8] A. M. B, C. S. Lumingkewas, and A. Rofi'i, "The Implementation of User Centered Design Method in Developing UI/UX," *Journal of Information System, Technology and Engineering*, vol. 1, no. 2, pp. 26–31, Jun. 2023, doi: 10.61487/JISTE.V1I2.13.

[9] N. T. Ramadhanti, C. W. Budiyanto, and R. A. Yuana, "The use of heuristic evaluation on UI/UX design: A review to anticipate web app's usability," *AIP Conf Proc*, vol. 2540, no. 1, p. 080008, Jan. 2023, doi: 10.1063/5.0105701.

[10] M. Bexiga, S. Garbatov, and J. C. Seco, "Closing the gap between designers and developers in a low code ecosystem," *Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems,*

MODELS-C 2020 - Companion Proceedings, pp. 413–422, Oct. 2020, doi: 10.1145/3417990.3420195.

[11] Y. Lamine and J. Cheng, "Understanding and supporting the design systems practice," *Empir Softw Eng*, vol. 27, no. 6, Nov. 2022, doi: 10.1007/S10664-022-10181-Y.

[12] M. Wiberg and E. Stolterman Bergqvist, "Automation of interaction—interaction design at the crossroads of user experience (UX) and artificial intelligence (AI)," *Pers Ubiquitous Comput*, vol. 27, no. 6, pp. 2281–2290, Dec. 2023, doi: 10.1007/S00779-023-01779-0/FIGURES/2.

[13] M. José, M. Fernández, I. Stefani, and H. Sanabria, "Design system: estandarización de un lenguaje de diseño compartido para el sistema de facturación Billo," 2022, Accessed: Jan. 13, 2024. [Online]. Available: https://repositoriotec.tec.ac.cr/handle/2238/14382

[14] C. W. Clegg, "Sociotechnical principles for system design," *Appl Ergon*, vol. 31, no. 5, pp. 463–477, 2000, doi: https://doi.org/10.1016/S0003-6870(00)00009-0.

[15] M. Christian Klimm and S. Bente, "Design Systems for Micro Frontends An Investigation into the Development of Framework-Agnostic Design Systems using Svelte and Tailwind CSS".

[16] S. J. Kim, N. P. Suh, and S. G. Kim, "Design of Software System Based on Axiomatic Design," *CIRP Annals*, vol. 40, no. 1, pp. 165–170, Jan. 1991, doi: 10.1016/S0007-8506(07)61959-3.

[17] P. Pšenák and M. Tibenský, "The usage of Vue JS framework for web application creation," *Mesterséges intelligencia*, vol. 2, no. 2, pp. 61–72, 2020, doi: 10.35406/MI.2020.2.61.

[18] J. Swacha and A. Kulpa, "Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks," *Electronics (Switzerland)*, vol. 12, no. 17, Sep. 2023, doi: 10.3390/ELECTRONICS12173563.

[19] R. Baida, M. Andriienko, and M. Plechawska-Wójcik, "Perfomance analysis of frameworks Angular and Vue.js," *Journal of Computer Sciences Institute*, vol. 14, pp. 59–64, Mar. 2020, doi: 10.35784/JCSI.1577.

[20] S. Avilés Matute, D. Avila-Pesantez, and M. Avila, "Desarrollo de sistema Web basado en los frameworks de Laravel y VueJs, para la gestión por procesos: Un estudio de caso," *Revista peruana de computación y sistemas*, Riobamba, Dec. 17, 2020.

[21] L. Zaina *et al.*, "What do software startups need from UX work?," *Empir Softw Eng*, vol. 28, no. 3, pp. 1–45, May 2023, doi: 10.1007/S10664-023-10322-X/TABLES/21.

[22] A. Rodriguez, J. C. Gardey, J. Grigera, G. Rossi, and A. Garrido, "UX debt in an agile development process: evidence and characterization," *Software Quality Journal*, vol. 31, no. 4, pp. 1467–1498, Dec. 2023, doi: 10.1007/S11219-023-09652-2/FIGURES/3.

[23] A. Braham, F. Buendía, M. Khemaja, and F. Gargouri, "User interface design patterns and ontology models for adaptive mobile applications," *Pers Ubiquitous Comput*, vol. 26, no. 6, pp. 1395–1411, Dec. 2022, doi: 10.1007/S00779-020-01481-5/FIGURES/12.

[24] S. M. Labrada, "Principios del proceso de diseño de interfaz de usuario," *Revista Cubana de Transformación Digital*, vol. 1, no. 3, pp. 143–155, Dec. 2020,

Accessed: Jan. 13, 2024. [Online]. Available: https://rctd.uic.cu/rctd/article/view/96

[25] J. Morales-Carrillo *et al.*, “Metodologías de desarrollo de software y su ámbito de aplicación: Una revisión sistemática”.

[26] C. Técnicas Aplicadas, M. I. Vinicio Estrada-Velasco, J. I. Alexandra Núñez-Villacis, and W. I. Clemente Cunuhay-Cuchipe, “Revisión Sistemática de la Metodología Scrum para el Desarrollo de Software,” *Dominio de las Ciencias, ISSN-e 2477-8818, Vol. 7, Nº. Extra 4, 2021 (Ejemplar dedicado a: AGOSTO ESPECIAL), pág. 54*, vol. 7, no. 4, p. 54, 2021, doi: 10.23857/dc.v7i4.2429.

[27] V. N. Lukin, A. L. Dzyubenko, and Y. B. Chechikov, “Approaches to User Interface Development,” *Programming and Computer Software*, vol. 46, no. 5, pp. 316–323, Sep. 2020, doi: 10.1134/S0361768820050059/FIGURES/1.

[28] M. F. Santoso, “Implementation Of UI/UX Concepts And Techniques In Web Layout Design With Figma,” *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 6, no. 2, pp. 279–285, Apr. 2024, doi: 10.47233/JTEKSIS.V6I2.1223.

[29] A. Ziegler *et al.*, “Measuring GitHub Copilot’s Impact on Productivity,” *Commun ACM*, vol. 67, no. 3, pp. 54–63, Feb. 2024, doi: 10.1145/3633453.

[30] O. C. Novac, D. E. Madar, C. M. Novac, G. Bujdoso, M. Oproescu, and T. Gal, “Comparative study of some applications made in the Angular and Vue.js frameworks,” *2021 16th International Conference on Engineering of Modern Electric Systems, EMES 2021 - Proceedings*, Jun. 2021, doi: 10.1109/EMES52337.2021.9484150.

[31] Y.-H. Huang, “Developing a design system for an e-commerce website”, Accessed: Dec. 11, 2023. [Online]. Available: www.aalto.fi

[32] F. Erdos, “Economical Aspects of UX Design and Development,” *10th IEEE International Conference on Cognitive Infocommunications, CogInfoCom 2019 - Proceedings*, pp. 211–214, Oct. 2019, doi: 10.1109/COGINFOCOM47531.2019.9089992.

[33] M. Li, J. Hu, and X. Lin, “The Development of Web Application Front-End of Intelligent Clinic Based on Vue.js,” *Lecture Notes in Electrical Engineering*, vol. 586, pp. 683–690, 2020, doi: 10.1007/978-981-32-9050-1_77/COVER.

[34] L. Inés, A. Arbildo, and W. Lafosse Quintana, “El rol de la transformación digital en la formulación de la estrategia de negocio de empresas del sector financiero en el Perú,” 2020, Accessed: Jan. 13, 2024. [Online]. Available: http://repositorio.up.edu.pe/handle/11354/2793

[35] “Best practices to scale design with Design Systems,” Adobe Blog. Accessed: Jan. 13, 2024. [Online]. Available: https://blog.adobe.com/en/publish/2021/05/26/best-practices-to-scale-design-with-design-systems

[36] D. Pinckney, F. Cassano, A. Guha, and J. Bell, “A Large Scale Analysis of Semantic Versioning in NPM,” *Proceedings - 2023 IEEE/ACM 20th International Conference on Mining Software Repositories, MSR 2023*, pp. 485–497, 2023, doi: 10.1109/MSR59073.2023.00073.