UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Programa de Apoyo para alumnos con Experiencia Profesional

TÍTULO: Librería de componentes para agilizar el desarrollo inicial de un proyecto

AUTOR: Edith Ester Sosa

DIRECTOR/A ACADÉMICO: Claudia Banchoff Tzancoff

DIRECTOR/A PROFESIONAL: Mariano Rivas

CODIRECTOR/A ACADÉMICO: Matias Pagano

CARRERA: Licenciatura en Sistemas

RESUMEN

La tesina aborda el desarrollo de una biblioteca de componentes en React para estandarizar y optimizar interfaces web en TrueNorth. Para mejorar la eficiencia y coherencia visual de las aplicaciones, se emplearon herramientas como Storybook, facilitando la reutilización y adaptabilidad de componentes en diversos proyectos. La colaboración entre los equipos de UX/UI y frontend fue clave para garantizar que los componentes fueran técnicamente sólidos, intuitivos y accesibles. Este trabajo explora las metodologías, el impacto en la productividad y su contribución al ecosistema de desarrollo web.

Palabras Claves

Frontend, Biblioteca de Componentes, React, Desarrollo Web, UX/UI, Estándares de Diseño, Accesibilidad

Conclusiones

La creación de una biblioteca de componentes en React ha permitido a TrueNorth consolidar un entorno de desarrollo más eficiente y coherente, optimizando tiempos de desarrollo y garantizando una experiencia de usuario uniforme en todos los productos. La implementación de esta biblioteca facilitó la colaboración entre diseño y desarrollo, mejorando la escalabilidad y mantenibilidad del código. La adopción de herramientas de documentación como Storybook y

Trabajos Realizados

- Desarrollo de la biblioteca de componentes en React, basada en un Design System de TrueNorth.
- Creación de documentación interactiva de los componentes mediante Storybook.
- Implementación de un sitio web de documentación con ejemplos y guías detalladas.
- Colaboración con los equipos de UX/UI para asegurar la accesibilidad y consistencia de la biblioteca

Trabajos Futuros

- Expansión de la biblioteca de componentes a otros frameworks y tecnologías para ampliar su aplicabilidad.
- Integración de inteligencia artificial en el desarrollo de componentes para mejorar la personalización de la experiencia de usuario.
- Investigación de nuevas metodologías de pruebas de usabilidad para optimizar el diseño centrado en el usuario.

Fecha de la presentación: MES Y AÑO


UNIVERSIDAD
NACIONAL
DE LA PLATA

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

TESINA PAEP DE LICENCIATURA EN SISTEMAS

TÍTULO: *Librería de componentes para agilizar el desarrollo inicial de un proyecto*

AUTORA: Edith Ester Sosa

DIRECTORA: Claudia Banchoff Tzancoff

CODIRECTOR: Matias Pagano

CARRERA: Licenciatura en Informática

Fecha de Presentación
Noviembre 2024

Agradecimientos

A mi mamá y a mi papá, mis pilares inquebrantables, por su amor y por enseñarme a nunca bajar los brazos. Gracias por ser siempre ese faro, esa guía que me impulsó a creer en mí y seguir avanzando.

A mis amigos y amigas, quienes, aun cuando la facultad me alejaba, siempre estuvieron ahí, esperándome con los brazos abiertos y alentándome sin condición. Ustedes son el refugio y la fuerza en cada paso de este viaje. Gracias por su paciencia, su apoyo y por celebrar cada logro conmigo como si fuera propio.

A mi querido grupo de estudio: Pochy, Eze, Fer, Pato, Seba, Nico y Cami. Juntos, transformamos los días de esfuerzo en recuerdos imborrables, llenos de mates, risas y un compañerismo inquebrantable. Gracias por hacer de este camino una aventura y por convertir cada desafío en un momento de amistad y aprendizaje compartido.

A mis directores de tesis, quienes no solo fueron guías en este proceso, sino también inspiración constante. Gracias por su paciencia, sus enseñanzas y por mostrarme el valor de la dedicación y el esfuerzo en cada etapa de este proyecto.

A la universidad pública y gratuita, y a esos profesores y ayudantes que están ahí por puro amor a la carrera y a la enseñanza. Ustedes son el alma de esta institución y representan la entrega y la vocación en su forma más pura. Gracias por darnos la posibilidad de crecer y aprender.

A mi querido amigo Guti, que físicamente ya no está, pero cuyo espíritu me acompaña cada día. Este logro también es tuyo, Guti. Te extraño y siempre llevaré tu recuerdo en el corazón.

Y, a mí misma, por no rendirme, por aprender a abrazar cada obstáculo y seguir adelante a pesar del cansancio. Como decía Osvaldo Zubeldía, "A la gloria no se llega por un camino de rosas." Hoy, celebro cada caída y cada tropiezo, porque fueron los que me hicieron más fuerte. Este logro es el resultado de ese esfuerzo constante, de cada vez que elegí levantarme con más ganas.

Gracias a todos los que me acompañaron en este camino. Sin ustedes, este sueño no habría sido posible. Hoy, este logro es nuestro, y cada uno de ustedes tiene un lugar especial en este capítulo de mi vida.

Índice

Índice **3**
Capítulo 1. Introducción **5**
1.1 Motivación y objetivo de la tesina 7
1.1.1 Motivación 7
1.1.2 Objetivo 8
1.2 Trabajo realizado 10
1.2.1 Exhibición a través de Storybook 10
1.2.2 Desarrollo de un sitio web de documentación 10
1.2.3 Objetivos específicos del trabajo realizado 10
1.2.4 Conclusión 11
1.3 Organización de la tesina 11
Capítulo 2. Desarrollo frontend **14**
2.1 Fundamentos del desarrollo frontend: HTML, CSS y JavaScript 14
2.2 Diseño responsivo y accesibilidad 16
2.3 Experiencia del usuario (UX) 17
2.4 Importancia de las librerías de componentes 17
2.4.1 Ejemplos de librerías de componentes 18
2.5 Importancia en el desarrollo frontend 18
2.6 Revisión de frameworks: Vue, Angular y React 19
2.6.1 Vue.js 20
2.6.2 Angular 21
2.6.3 React 22
2.6.4 Comparativa de frameworks 22
2.7 Tendencias tecnológicas 25
2.7.1 Inteligencia artificial (AI) y aprendizaje automático (ML) 25
2.8 Resumen del capítulo 26
Capítulo 3. Diseño centrado en el usuario (UX/UI) **28**
3.1 Principios de UX y UI 28
3.1.1 Usabilidad 29
3.1.2 Accesibilidad 29
3.1.3 Diseño centrado en el usuario 30
3.1.4 Consistencia en el diseño 30
3.1.5 Importancia de la estética en la interfaz 31
3.2 Integración de UX/UI en el desarrollo de componentes 31
3.2.1 Diseño de componentes reutilizables 32
3.2.2 Desarrollo con un enfoque modular 32
3.2.3 Prototipado y pruebas de usabilidad en componentes 33
3.2.4 Implementación y mantenimiento de componentes UX/UI 33
3.3 Sinergia entre diseño y funcionalidad 34
3.3.1 Equilibrio entre diseño estético y funcionalidad técnica 34
3.3.2 La colaboración entre diseñadores y desarrolladores 35

3.3.3 Manejo de compromisos en el desarrollo frontend..... 35
3.3.4 La importancia de la colaboración interdisciplinaria.....36
3.4 Resumen del capítulo.....37
Capítulo 4. El Desarrollo realizado..... 38
4.1 Creación de la biblioteca de componentes..... 38
4.1.2 Diseño y planificación.....39
4.1.3 Desarrollo de componentes..... 39
4.2 Elección de React y herramientas utilizadas.....52
4.2.1 Ventajas de React..... 53
4.2.2 Herramientas utilizadas en el desarrollo..... 53
4.2.3 Integración y beneficios.....55
4.3 Proceso colaborativo y ejemplos prácticos..... 55
4.3.1 Gestión del proyecto y metodología de trabajo.....55
4.3.2 Proceso colaborativo entre equipos..... 56
4.4 Participación de la tesista en el equipo de desarrollo..... 57
Componentes desarrollados..... 58
Coordinación con el equipo.....58
4.5 Resumen del capítulo.....58
Capítulo 5. Evaluación de la biblioteca de componentes..... 60
5.1 Introducción.....60
5.2 Ejemplo práctico de componentes en acción.....60
5.2.1 Dashboard.....60
5.3 Validación de la biblioteca de componentes..... 69
Accesibilidad..... 70
Otras validaciones (rendimiento, mejores prácticas y optimización para motores de búsqueda)..... 73
Mejoras que se podrían realizar.....74
Reutilización..... 74
Mantenibilidad..... 75
Consistencia visual.....75
5.4 Conclusión.....75
5.5 Resumen del capítulo.....75
Capítulo 6: Conclusiones..... 77
6.1 Resumen del trabajo realizado.....77
6.2 Impacto de la biblioteca de componentes.....77
6.3 Lecciones aprendidas..... 77
6.4 Contribuciones y futuro..... 78
6.5 Reflexión final.....79
Capítulo 7. Referencias bibliográficas..... 80

Capítulo 1. Introducción

En el escenario actual, la innovación tecnológica y el desarrollo de software son imperativos clave para la evolución empresarial. Dentro de este marco, se ha observado un movimiento significativo hacia la estandarización y mejora de la consistencia en la creación de productos digitales, una tendencia que refleja la necesidad de respuestas ágiles y eficientes en un mercado en constante cambio. Según su web oficial TrueNorth[1], es reconocida como una de las compañías de desarrollo de software de servicios financieros más exitosas a nivel mundial, ha sido pionera en este frente. Especializada en soluciones digitales innovadoras, TrueNorth ha liderado 45 empresas originales, 120 transformaciones digitales y 40 renovaciones de productos, marcando un precedente en la aplicación de tecnologías de vanguardia en sectores críticos como el crédito, los pagos, la gestión de patrimonios y los seguros, expandiendo recientemente su experiencia a industrias vitales como la salud y la educación.

En respuesta a esta necesidad imperante de innovación y excelencia, el presente trabajo se adentra en la presentación de una biblioteca de componentes desarrollada en el ámbito de TrueNorth. El objetivo fundamental de esta tesina es sumergirse en un análisis de las herramientas, procesos y funcionalidades centrales de esta biblioteca de componentes. A lo largo de este trabajo, se buscará no solo ofrecer una visión integral de las herramientas y metodologías aplicadas, sino también destacar su trascendencia y contribución al diseño de interfaces en aplicaciones web.

En este camino hacia la comprensión y exposición del proceso de creación de la biblioteca de componentes, se otorgará especial énfasis a las herramientas empleadas y su relevancia en el contexto del desarrollo web. A través de esta exploración, se pretende esclarecer y profundizar en la manera en que estos componentes se convierten en pilares fundamentales para el desarrollo de aplicaciones robustas y escalables.

La motivación que impulsó este trabajo proviene de la necesidad latente de establecer un entorno de desarrollo coherente y eficiente dentro de la empresa. Esta iniciativa se presenta como una respuesta estratégica para optimizar y acelerar los procesos de desarrollo de software, solidificando un enfoque uniforme y eficaz en todos los proyectos a abordar.

La consolidación de una librería de componentes en el contexto de React, apoyada en un Design System (Sistema de diseño) elaborado por el equipo de UX (Experiencia de usuario)/UI (Interfaz de usuario), representa un paso crucial en la transformación integral de

[1] Sitio oficial de Truenorth https://www.truenorth.co/our-company

cómo se conceptualizan y materializan los productos digitales. Este enfoque estratégico no solo se centra en reducir los tiempos de desarrollo, sino también en promover la coherencia visual y funcional en todos los proyectos.

Asimismo, se busca contextualizar el papel esencial del equipo de UX/UI en este proceso de desarrollo. Ambas disciplinas, trabajaron en estrecha colaboración, desde la concepción hasta la implementación final, se esforzaron por comprender las necesidades de los usuarios, diseñar soluciones efectivas y garantizar la calidad y satisfacción del usuario final.

Además, este trabajo dedica un capítulo específico al análisis detallado del equipo de UX/UI de TrueNorth, enfatizando su contribución indispensable en el proceso de desarrollo de la biblioteca de componentes. Este segmento profundizará en cómo la colaboración entre las disciplinas de experiencia de usuario e interfaz de usuario ha sido crucial para asegurar que los componentes no solo sean técnicamente sólidos sino también intuitivos y accesibles para los usuarios finales. Se abordarán las metodologías, herramientas y estrategias implementadas por el equipo, ilustrando su papel en la creación de soluciones que satisfacen las necesidades de los usuarios y mejoran su interacción con los productos digitales.

Destacando el papel crucial del desarrollo frontend en este ecosistema de innovación, es importante señalar que, si bien el equipo de UX/UI sienta las bases del diseño y la experiencia de usuario, la realización práctica y técnica de estas visiones recae en manos del desarrollo frontend. Como programadora frontend, mi contribución a este proyecto fue más allá de la simple implementación de diseños; se trató de una tarea que implicaba traducir los conceptos y prototipos de UX/UI en aplicaciones web funcionales y eficientes. Este proceso no solo exigió un profundo entendimiento de las tecnologías de frontend como React (Facebook, 2023) y TypeScript (Microsoft Corporation, 2023), sino también una colaboración continua con el equipo de UX/UI para asegurar que el producto final no solo sea técnicamente sólido, sino que también cumpla con las expectativas de diseño y usabilidad. Este trabajo profundiza en la sinergia entre el diseño y el desarrollo frontend, explorando cómo las decisiones técnicas y estéticas se entrelazan para crear productos digitales que son tanto innovadores como accesibles para el usuario final.

En los capítulos siguientes, se detalla este delicado equilibrio entre la visión creativa y la realización técnica, resaltando mi papel en la interpretación y materialización de los componentes de la biblioteca. Se examinará cómo la estrecha interacción entre el equipo de UX/UI y el desarrollo frontend facilitó una implementación que no solo es fiel al diseño original, sino que también optimiza la experiencia del usuario en el entorno digital.

En este sentido, esta tesina se establece como un puente entre la innovación tecnológica, el desarrollo de software y la creación de productos digitales. A través de un análisis y una contextualización detallada, se aspira a proporcionar una visión ampliada de la importancia y la aplicación práctica de la biblioteca de componentes en el marco del desarrollo web y su relevancia en el entorno empresarial actual.

1.1 Motivación y objetivo de la tesina

1.1.1 Motivación

La motivación para el desarrollo de esta biblioteca de componentes en TrueNorth se arraiga profundamente en la visión estratégica de la empresa por establecer un ecosistema de desarrollo robusto, eficiente y cohesivo. Este impulso estratégico nace de un reconocimiento crítico de los desafíos inherentes al desarrollo de software en el ámbito empresarial moderno, donde la velocidad, la coherencia y la escalabilidad no son meramente deseables, sino requisitos esenciales para el éxito y la sostenibilidad.

La decisión de invertir en la creación de una librería de componentes especializada surge de varias necesidades y observaciones clave:

Optimización de procesos: en el núcleo de esta iniciativa está el objetivo de optimizar y agilizar los procesos de desarrollo de software dentro de TrueNorth. La empresa reconoce que la eficiencia en el desarrollo no solo reduce los tiempos de entrega sino que también mejora la calidad del software producido. Esto implica consolidar un enfoque de desarrollo que sea a la vez coherente en todos los proyectos y flexible para adaptarse a las necesidades cambiantes.

Estandarización y consistencia: la librería de componentes responde a la necesidad crítica de estandarizar la creación de productos digitales, asegurando una consistencia tanto en la funcionalidad como en la interfaz de usuario a través de todos los proyectos de desarrollo. Los componentes modulares, reutilizables no solo promueven la eficiencia y la mantenibilidad sino que también facilitan una coherencia visual y funcional, elementos clave para la identidad de marca y la experiencia del usuario.

Fundamentación en un Design System: la creación de la librería está intrínsecamente ligada a un Design System meticulosamente desarrollado por el equipo de UX/UI de TrueNorth. Este sistema representa una estructura comprensiva que dicta cómo los componentes deben ser diseñados, desarrollados y utilizados. Al seguir los principios establecidos por este sistema, la librería de componentes no solo optimiza el proceso de

desarrollo sino que también mantiene la coherencia en todos los desarrollos, alineándose con la visión de Kholmatova (2016) sobre la importancia crítica de los sistemas de diseño para la cohesión del producto.

Colaboración entre UX y UI: la empresa alberga un equipo de UX/UI altamente calificado, cuya experiencia cubre un amplio espectro de funciones esenciales en el diseño y desarrollo de productos digitales. Este equipo no solo se enfoca en el análisis profundo de las necesidades y comportamientos de los usuarios finales sino que también se dedica al diseño visual y funcional de la interfaz de usuario (Schneiderman, 2005). La colaboración estrecha entre UX y UI asegura que cada componente de la librería no solo sea funcional y eficiente sino también estéticamente atractivo y accesible, reflejando la interdependencia entre la usabilidad y la estética en el diseño de productos exitosos.

Adopción de principios multidisciplinarios: la filosofía detrás del desarrollo de la biblioteca de componentes y el Design System refleja una comprensión profunda de los principios multidisciplinarios de diseño de producto, vigentes desde su formulación por Norman en 1988 y aún aplicados en el contexto de UX/UI y desarrollo digital. Estos principios subrayan la importancia de diseñar con un entendimiento de cómo los usuarios interactúan con los productos en su vida cotidiana, enfatizando la necesidad de crear soluciones que se adapten intuitivamente a las necesidades y capacidades de los usuarios actuales..

La confluencia de estos factores motiva la creación de una biblioteca de componentes en React que no solo sirve como una herramienta práctica para el desarrollo de software sino que también representa los valores de innovación, eficiencia y excelencia en diseño de TrueNorth. A través de este trabajo, se buscó proporcionar un recurso valioso que simplifique y estandarice la creación de productos digitales, promoviendo la agilidad y facilitando el trabajo colaborativo en todos los proyectos de desarrollo de la empresa.

1.1.2 Objetivo

El objetivo principal de esta tesina es brindar una presentación exhaustiva y detallada de una biblioteca de componentes específicamente desarrollada durante mi periodo laboral en TrueNorth. Esta biblioteca representa un esfuerzo significativo por estandarizar y mejorar la eficiencia en el desarrollo de aplicaciones web, abordando tanto la necesidad de coherencia en el diseño como la optimización en el proceso de desarrollo.

Para lograr este objetivo, se realizó un análisis profundo de varias dimensiones clave del proyecto:

Herramientas tecnológicas: en este informe se describen las herramientas tecnológicas seleccionadas para el desarrollo de la biblioteca de componentes, justificando su elección. Esto incluye una evaluación de lenguajes de programación, frameworks, librerías y otras tecnologías auxiliares. Se explora cómo estas herramientas contribuyen a los objetivos de eficiencia, escalabilidad y mantenibilidad del código.

Proceso de desarrollo: se detalla el proceso de desarrollo adoptado, desde la concepción inicial hasta la implementación final de la biblioteca. Este análisis incluye fases de diseño, prototipado, desarrollo iterativo, pruebas y despliegue. Se presta especial atención a las metodologías ágiles de desarrollo, la integración continua, y las prácticas de entrega continua, si fueran aplicadas.

Funcionalidades clave: se proporciona una descripción de las funcionalidades clave que ofrece la biblioteca, destacando cómo éstas responden a desafíos específicos del desarrollo web. Se discuten casos de uso reales donde estas funcionalidades han demostrado ser particularmente valiosas, así como su impacto en la eficiencia del desarrollo y la experiencia del usuario final.

Diseño e interfaz de usuario: se examina el enfoque adoptado para el diseño de interfaces, incluyendo la filosofía detrás de la selección de componentes, la coherencia visual, la accesibilidad y la usabilidad. Se describe cómo la biblioteca facilita la creación de interfaces ricas y responsivas, y cómo se alinea con las mejores prácticas y estándares de la industria.

Contribución al desarrollo web: finalmente, se discute la contribución global de la biblioteca al campo del desarrollo web. Se evalúa cómo la adopción de esta biblioteca puede influir en la productividad de los desarrolladores, la calidad del software, y la innovación en el diseño de aplicaciones web.

A través de este análisis detallado, este trabajo aspira a proporcionar una visión clara y completa de la biblioteca de componentes desarrollada en TrueNorth. Se espera que este estudio no solo resalte la importancia de las herramientas y procesos empleados sino también que sirva como recurso valioso para desarrolladores web, diseñadores de interfaces y profesionales del software interesados en la estandarización y mejora del desarrollo de aplicaciones web.

1.2 Trabajo realizado

La presente tesina tiene como fin último presentar y examinar en profundidad una biblioteca de componentes innovadora, desarrollada durante mi trabajo en TrueNorth. Esta biblioteca, construida utilizando React y TypeScript, se erige sobre el sólido fundamento del Design System desarrollado por el competente equipo de UX/UI de la empresa. Este sistema no sólo guía la coherencia visual y funcional de los componentes sino que también asegura su adaptabilidad y reutilización en una diversidad de proyectos de desarrollo web.

1.2.1 Exhibición a través de Storybook

Una faceta central de esta propuesta es la utilización de Storybook (Storybook, 2023) como plataforma principal para la exhibición y documentación de la biblioteca de componentes. Storybook, reconocido por su capacidad para permitir a los equipos de desarrollo visualizar, interactuar y documentar componentes de UI de manera eficaz, servirá como el medio a través del cual los componentes serán presentados. La integración de Storybook en este proyecto no solo facilitará la visualización interactiva y el entendimiento profundo de cada componente, incluyendo sus variantes y casos de uso específicos, sino que también proveerá un entorno práctico para demostrar la flexibilidad y la funcionalidad de la biblioteca en situaciones reales de desarrollo.

1.2.2 Desarrollo de un sitio web de documentación

Complementando la presentación en Storybook, la propuesta incluye el desarrollo de un sitio web de documentación dedicado. Este sitio ofrecerá ejemplos prácticos y guías detalladas sobre la implementación de los componentes en proyectos reales, proporcionando así una herramienta invaluable para los equipos de diseño y desarrollo. La documentación abordará desde la configuración inicial hasta complejas implementaciones, destacando cómo los componentes pueden ser personalizados y combinados para crear aplicaciones web robustas y visualmente atractivas.

1.2.3 Objetivos específicos del trabajo realizado

Demostrar la eficiencia y la eficacia de la biblioteca: a través de la integración con Storybook y el sitio web de documentación, esta propuesta busca demostrar cómo la biblioteca de componentes puede agilizar significativamente el proceso de desarrollo de software en TrueNorth, facilitando una mayor uniformidad y coherencia en todos los proyectos de la empresa.

Facilitar la adopción y la reutilización de componentes: al proporcionar una documentación exhaustiva y accesible, así como ejemplos interactivos de uso, se pretende fomentar la adopción de la biblioteca por parte de los equipos de desarrollo y diseño, maximizando la reutilización de componentes y promoviendo prácticas de desarrollo más eficientes y sostenibles.

Resaltar la colaboración entre diseño y desarrollo: presentando el proceso colaborativo entre los equipos de UX/UI y desarrollo que dio lugar a la creación de la biblioteca, esta tesina subraya la importancia de una estrecha cooperación interdisciplinaria en el desarrollo de soluciones digitales innovadoras y centradas en el usuario.

1.2.4 Conclusión

La implementación y documentación detallada de esta biblioteca de componentes en el contexto de TrueNorth no solo sirven como un testimonio del compromiso de la empresa con la excelencia en el desarrollo de software, sino que también establecen un precedente para futuras iniciativas de desarrollo dentro de la organización. Esta tesina, por lo tanto, no solo contribuye al cuerpo académico en el campo del desarrollo web y diseño de UI/UX sino que también ofrece una guía práctica y un recurso valioso para profesionales y empresas que buscan optimizar sus procesos de desarrollo mediante la estandarización y reutilización de componentes.

1.3 Organización de la tesina

Con el fin de proporcionar una comprensión integral del desarrollo y la implementación de la biblioteca de componentes en TrueNorth, así como su impacto en el desarrollo de software y diseño de productos digitales, esta tesina se organiza en los siguientes capítulos:

Capítulo 1: Introducción

Este capítulo establece el escenario para la investigación, destacando la importancia de la innovación tecnológica y el desarrollo de software en el contexto empresarial actual. Se discute el movimiento hacia la estandarización y mejora de la consistencia en la creación de productos digitales, con TrueNorth como un caso de estudio ejemplar debido a su éxito en el desarrollo de soluciones digitales innovadoras. Se introduce la motivación detrás de la creación de la biblioteca de componentes, el objetivo de la tesina, y se anticipa la importancia del desarrollo frontend en este proceso, marcando una distinción clara entre el papel del desarrollo y el diseño UX/UI.

Capítulo 2: Desarrollo frontend

Este capítulo se sumerge en los fundamentos del desarrollo frontend, explorando qué implica esta disciplina y la importancia de las librerías de componentes dentro de este contexto. Se hará una revisión de los principales frameworks utilizados en el desarrollo web, como Angular, Vue y React, incluyendo una comparativa de estos para destacar sus ventajas y aplicaciones específicas. Se incluirán gráficos de tendencias en la utilización de estas tecnologías para proporcionar un panorama actual del desarrollo frontend.

Capítulo 3: Diseño centrado en el usuario (UX/UI)

Aquí, se aborda la importancia del diseño centrado en el usuario, detallando cómo los principios de UX y UI se integran en el proceso de desarrollo de la biblioteca de componentes. Este capítulo subraya la sinergia entre el diseño y la funcionalidad, fundamentales para crear interfaces que satisfagan y excedan las expectativas de los usuarios.

Capítulo 4: El desarrollo realizado

Este capítulo ofrece una visión detallada del proceso de creación de la biblioteca de componentes, desde la concepción inicial hasta su estado actual. Se discutirán los aspectos técnicos del desarrollo, incluyendo la elección de React como tecnología subyacente, la conformación del equipo de trabajo, las herramientas utilizadas como Jira, y el proceso colaborativo. También se presentarán ejemplos prácticos y capturas de pantalla de ejemplos de posibles desarrollos que utilizan los componentes creados, acompañados de fragmentos de código relevante.

Capítulo 5: Evaluación

En este capítulo, se evalúa cómo se ha utilizado la biblioteca de componentes en un proyecto real, presentando ejemplos específicos de su uso.

Capítulo 6: Conclusiones

Se recogerán las principales conclusiones derivadas del estudio, reflexionando sobre la contribución de la biblioteca de componentes al desarrollo de software y diseño de productos digitales en TrueNorth. Se discutirán las lecciones aprendidas y el valor agregado por la biblioteca al ecosistema de desarrollo.

Capítulo 7: Referencias

Este capítulo listará todas las fuentes bibliográficas y recursos consultados durante la elaboración de la tesina, proporcionando crédito adecuado y permitiendo a los lectores profundizar en los temas de interés.

Capítulo 2. Desarrollo frontend

El desarrollo frontend constituye una rama fundamental de la informática dedicada a la construcción de la interfaz de usuario y la experiencia de interacción en aplicaciones web y móviles. Esta disciplina se centra en la implementación de diseños visuales y estructuras interactivas que los usuarios finales encuentran al visitar sitios web o utilizar aplicaciones, utilizando para ello un conjunto de tecnologías y prácticas específicas.

Para la creación de estas interfaces, los desarrolladores frontend emplean lenguajes de marcado como HTML (Lenguaje de Marcado de Hipertexto), CSS (Hojas de Estilo en Cascada), y JavaScript como lenguaje de programación, los cuales son pilares fundamentales en el desarrollo web (Duckett, 2014). Estas tecnologías permiten definir la estructura, presentación y dinámicas de interacción de las páginas web, respectivamente. Además, el uso de frameworks modernos como React, Angular, y Vue.js facilita la creación de aplicaciones ricas y complejas con mejor manejo de estado y reactividad (Meigs, 2019).

La evolución de las prácticas de desarrollo frontend ha incorporado también principios de diseño responsivo, asegurando que las aplicaciones se adapten y funcionen adecuadamente en una diversidad de dispositivos y tamaños de pantalla. Asimismo, la optimización de la velocidad de carga de las páginas y la accesibilidad web se han convertido en aspectos cruciales, buscando ofrecer una experiencia de usuario óptima para todos, incluyendo aquellos con discapacidades (Lazar, Feng, & Hochheiser, 2017).

El desarrollo frontend juega un rol crítico en el diseño e implementación de interfaces digitales, siendo un campo de estudio y trabajo en constante evolución que demanda una actualización y aprendizaje continuos de parte de sus practicantes.

2.1 Fundamentos del desarrollo frontend: HTML, CSS y JavaScript

Los fundamentos del desarrollo frontend se construyen sobre tres pilares tecnológicos esenciales: HTML, CSS y JavaScript. Estas tecnologías, aunque poseen funciones distintas, trabajan en conjunto para crear interfaces web que sean accesibles, dinámicas y visualmente atractivas (MDN Web Docs, 2023).

1. **HTML: la estructura de la web**
   HTML, siglas en inglés de *Lenguaje de Marcado de Hipertexto (HyperText Markup Language)*, constituye el pilar fundamental de la estructura de una página web. Este lenguaje permite organizar el contenido mediante etiquetas que asignan un propósito

semántico a cada elemento de la página, como <header>, <footer>, <article> y <section>, lo cual mejora la accesibilidad y la navegabilidad, especialmente para usuarios que dependen de tecnologías de asistencia (Gómez, 2022). HTML5, la última revisión importante de este estándar, amplió las capacidades de HTML al incluir etiquetas semánticas y elementos multimedia como <video> y <audio>, los cuales permiten integrar contenido audiovisual sin necesidad de complementos externos, haciendo las aplicaciones más interactivas y accesibles (Díaz-Jorge, 2022).

2. **CSS: estilización y diseño visual**
   CSS, siglas de inglés de *Hojas de Estilo en Cascada* (*Cascading Style Sheets*), define la capa de presentación de un sitio web, permitiendo a los desarrolladores controlar el diseño, los colores, las tipografías y otros aspectos visuales de la página (Revista Social Fronteriza, 2021). CSS3 introdujo propiedades avanzadas que revolucionaron el diseño web, como transiciones y animaciones, que permiten agregar efectos visuales sin el uso de JavaScript. Además, los modelos de diseño como Flexbox y CSS Grid han facilitado la creación de layouts responsivos y adaptables, que se ajustan eficientemente a diversos tamaños de pantalla (MDN Web Docs, 2023). Recientemente, se ha introducido container queries, una característica que permite adaptar el diseño en función de las dimensiones de su contenedor, proporcionando mayor control y flexibilidad en la creación de layouts complejos y adaptativos (Mozilla Developer Network, 2023).
3. **JavaScript y su ecosistema: interactividad y dinamismo**
   JavaScript permite incorporar interactividad y lógica en las páginas web, transformándolas de simples documentos estáticos a aplicaciones interactivas. Este lenguaje posibilita la manipulación del DOM (Document Object Model) y la implementación de funcionalidades asincrónicas, como la API Fetch y async/await, las cuales facilitan la carga y actualización de datos sin recargar la página completa (ECMA International, 2023). La evolución de ECMAScript, el estándar que regula JavaScript, ha traído consigo innovaciones que mejoran tanto la eficiencia como la mantenibilidad del código, como las promesas y módulos (Simpson, 2015).

Además, el surgimiento de frameworks y bibliotecas basados en JavaScript, como React, Angular y Vue.js, ha transformado el desarrollo de aplicaciones web. Estas herramientas ofrecen estructuras y patrones de diseño que simplifican la creación de interfaces de usuario complejas y reactivas, gestionan el estado de la aplicación de manera eficiente y promueven el desarrollo modular y reutilizable del código. React, por ejemplo, introduce un modelo basado en componentes que facilita la construcción de interfaces de usuario

interactivas y eficientes mediante la actualización selectiva de partes de la página. Angular proporciona un marco robusto para el desarrollo de aplicaciones de página única (SPA) con inyección de dependencias y enlace de datos bidireccional, mientras que Vue.js destaca por su sencillez y su enfoque progresivo, permitiendo a los desarrolladores adoptar el framework pieza por pieza según sea necesario (Mead, 2020).

En conjunto, HTML, CSS y JavaScript, junto con su ecosistema de frameworks y bibliotecas, forman una base robusta para el desarrollo frontend, adaptándose continuamente a los avances tecnológicos y los estándares de la web moderna para ofrecer aplicaciones interactivas, accesibles y escalables (MDN Web Docs, 2023).

2.2 Diseño responsivo y accesibilidad

El diseño responsivo y la accesibilidad web representan pilares fundamentales en el desarrollo frontend moderno, respaldando una web universalmente accesible y usable en cualquier dispositivo o contexto de navegación. El diseño responsivo se basa en la flexibilidad y adaptabilidad del diseño de las páginas web, permitiendo que estas se ajusten y funcionen de manera óptima en una amplia gama de dispositivos, desde teléfonos móviles hasta pantallas de gran tamaño. Esta adaptabilidad se logra mediante el uso de consultas de medios en CSS, que permiten modificar el estilo de los elementos de la página en función de las características específicas del dispositivo, como el tamaño de la pantalla, la orientación o la resolución. El objetivo es proporcionar una experiencia de usuario coherente y satisfactoria, independientemente del medio a través del cual se acceda al contenido web.

Por otro lado, la accesibilidad web se enfoca en eliminar las barreras que impiden que las personas con discapacidades puedan interactuar y acceder a los sitios web. La accesibilidad es una consideración esencial que abarca diversos aspectos, desde el diseño visual y la navegación hasta la compatibilidad con tecnologías de asistencia, como lectores de pantalla y teclados adaptativos. Las Directrices de Accesibilidad para el Contenido Web (WCAG)[2] del Consorcio World Wide Web (W3C) establecen un conjunto de criterios y técnicas para crear contenido web accesible. El W3C es una organización internacional que desarrolla estándares abiertos para asegurar el crecimiento a largo plazo de la Web, y sus WCAG son reconocidas como el principal estándar internacional en accesibilidad web, proporcionando tres niveles de conformidad: A (el nivel más bajo), AA y AAA (el nivel más alto) (World Wide Web Consortium, 2018).

[2] Recomendación disponible en https://www.w3.org/TR/WCAG22/. Último acceso 30 de octubre de 2024

La combinación del diseño responsivo y la implementación de prácticas de accesibilidad asegura que los sitios web y aplicaciones sean inclusivos y accesibles para todos los usuarios, incluidos aquellos con discapacidades visuales, auditivas, motoras o cognitivas. Esta inclusividad no solo es una responsabilidad ética para los desarrolladores y diseñadores web, sino que también se traduce en beneficios legales y económicos, ampliando el alcance del contenido a una audiencia más amplia y cumpliendo con las legislaciones de accesibilidad web en varios países.

2.3 Experiencia del usuario (UX)

La experiencia del usuario (UX) en el desarrollo frontend implica diseñar sitios web que no solo sean accesibles y estéticamente agradables, sino también fáciles de usar y navegar. Los principios de UX se centran en entender las necesidades y comportamientos de los usuarios, creando interfaces que faciliten la consecución de sus objetivos de manera intuitiva. Esto incluye la organización lógica de la información, la carga rápida de las páginas y la minimización de la fricción en la interacción del usuario (Norman & Nielsen, 2013).

2.4 Importancia de las librerías de componentes

En el desarrollo frontend, la modularidad, reusabilidad y mantenibilidad del código son aspectos críticos para la construcción eficiente y efectiva de aplicaciones web modernas. Las librerías de componentes, que encapsulan y estandarizan elementos de la interfaz de usuario, emergen como herramientas esenciales en este contexto, facilitando el desarrollo ágil y la coherencia visual a través de proyectos de software. Estas librerías ofrecen un conjunto de componentes predefinidos y estilizados que pueden ser utilizados y personalizados según las necesidades específicas de un proyecto, optimizando el flujo de trabajo de desarrollo y asegurando una experiencia de usuario consistente.

2.4.1 Ejemplos de librerías de componentes

Si bien existen numerosos frameworks para el desarrollo frontend[3], se plantean, como ejemplo, los tres más populares en la actualidad (StackOverflow, 2024)(State Of JavaScript, 2023)

[3] Un resumen muy completo y actualizado puede encontrarse en la página de WIkipedia: https://en.wikipedia.org/wiki/Comparison_of_JavaScript-based_web_frameworks, donde no solo se enumeran dichos frameworks, sino que se presenta una tabla comparativa muy útil. Si bien no se trata de una referencia académica, es interesante de mencionar por la actualización y cantidad de herramientas incluidas.

React.js y Material-UI: React es una biblioteca de JavaScript conocida por su capacidad para construir interfaces de usuario de manera declarativa. Este enfoque declarativo facilita el entendimiento del estado de la interfaz de usuario en cualquier momento, lo cual es un beneficio ampliamente reconocido por su claridad y mantenibilidad del código. React optimiza la actualización del DOM, lo que la hace eficiente, especialmente en aplicaciones web dinámicas que requieren frecuentes actualizaciones de la UI (React Documentation, 2023). Material-UI, por otro lado, ofrece componentes React que implementan Google's Material Design, proporcionando un conjunto extenso de herramientas visuales y de interacción listas para usar. Esta combinación permite a los desarrolladores construir aplicaciones web interactivas y visualmente atractivas con una experiencia de usuario coherente y moderna. La eficiencia de Material-UI se deriva de su integración bien optimizada con React, facilitando el desarrollo rápido y consistente de interfaces sin sacrificar el rendimiento (Material-UI Documentation, 2023).

Vue.js y Vuetify: Vue.js es un framework progresivo de JavaScript utilizado para construir interfaces de usuario y aplicaciones de una sola página. Vuetify es una librería de componentes Vue que proporciona una amplia gama de elementos de interfaz de usuario siguiendo las especificaciones de Material Design, facilitando el desarrollo de aplicaciones ricas y responsivas (Vuetify, 2020).

Angular y Angular Material: Angular es un plataforma y framework para construir aplicaciones web de una sola página con HTML, CSS y TypeScript. Angular Material ofrece un conjunto de componentes reutilizables, bien probados y accesibles basados en Material Design, permitiendo a los desarrolladores implementar rápidamente interfaces consistentes y accesibles (Angular Material, 2020).

2.5 Importancia en el desarrollo frontend

Las librerías de componentes juegan un papel crucial en el desarrollo frontend por varias razones:

- **Eficiencia en el desarrollo**: permiten a los desarrolladores construir aplicaciones de manera más rápida y eficiente, reutilizando componentes que ya han sido probados y optimizados.
- **Consistencia visual**: aseguran una coherencia visual y funcional a lo largo de toda la aplicación, lo que es esencial para la identidad de marca y la experiencia del usuario.

- **Mantenibilidad**: facilitan la mantenibilidad del código al centralizar la lógica y el diseño de los componentes, haciendo que las actualizaciones y los cambios sean más manejables.
- **Accesibilidad**: muchas librerías de componentes están diseñadas con consideraciones de accesibilidad, asegurando que las aplicaciones sean utilizables por una audiencia más amplia.

2.6 Revisión de frameworks: Vue, Angular y React

En el ámbito del desarrollo frontend, la selección de un framework adecuado es crucial para el éxito de cualquier proyecto web. Los frameworks no solo proporcionan las herramientas necesarias para el desarrollo de aplicaciones eficientes y escalables, sino que también establecen un conjunto de prácticas estandarizadas que guían a los desarrolladores a lo largo del proceso de construcción. De acuerdo a varios análisis y relevamientos [(Indianic, 2023) (Simform, 2024) (Linkedin, 2024) (Devjobsscanner, 2023)], entre la multitud de opciones disponibles, Vue, Angular y React se destacan por su robustez, comunidad activa y capacidad para adaptarse a diferentes necesidades de proyectos.

La Figura 1 ilustra la demanda de empleo en el mercado para desarrolladores especializados en estos frameworks durante el período comprendido entre noviembre de 2022 y diciembre de 2023. Según los datos proporcionados por Devjobsscanner (2023), React lidera con un total de 225,821 ofertas de empleo, lo que refleja su amplio uso y preferencia en el sector. Le sigue Angular, con 172,693 ofertas, lo que lo posiciona como una opción sólida para proyectos de gran escala, especialmente en empresas que buscan estructuras más robustas y completas. Por otro lado, Vue, aunque cuenta con una menor cantidad de ofertas de empleo (48,078), sigue siendo una opción viable, especialmente en proyectos que requieren mayor flexibilidad y simplicidad en la estructura del código. Finalmente, otros frameworks menos populares cuentan con 7,205 ofertas de empleo.

Este análisis resalta cómo la elección de un framework no solo depende de las características técnicas, sino también de la demanda laboral y la comunidad que respalda a cada tecnología. Las empresas y desarrolladores deben tener en cuenta estas tendencias a la hora de decidir qué herramienta utilizar en sus proyectos, ya que el conocimiento en un framework con mayor demanda puede mejorar las oportunidades laborales y la escalabilidad de los proyectos a largo plazo.

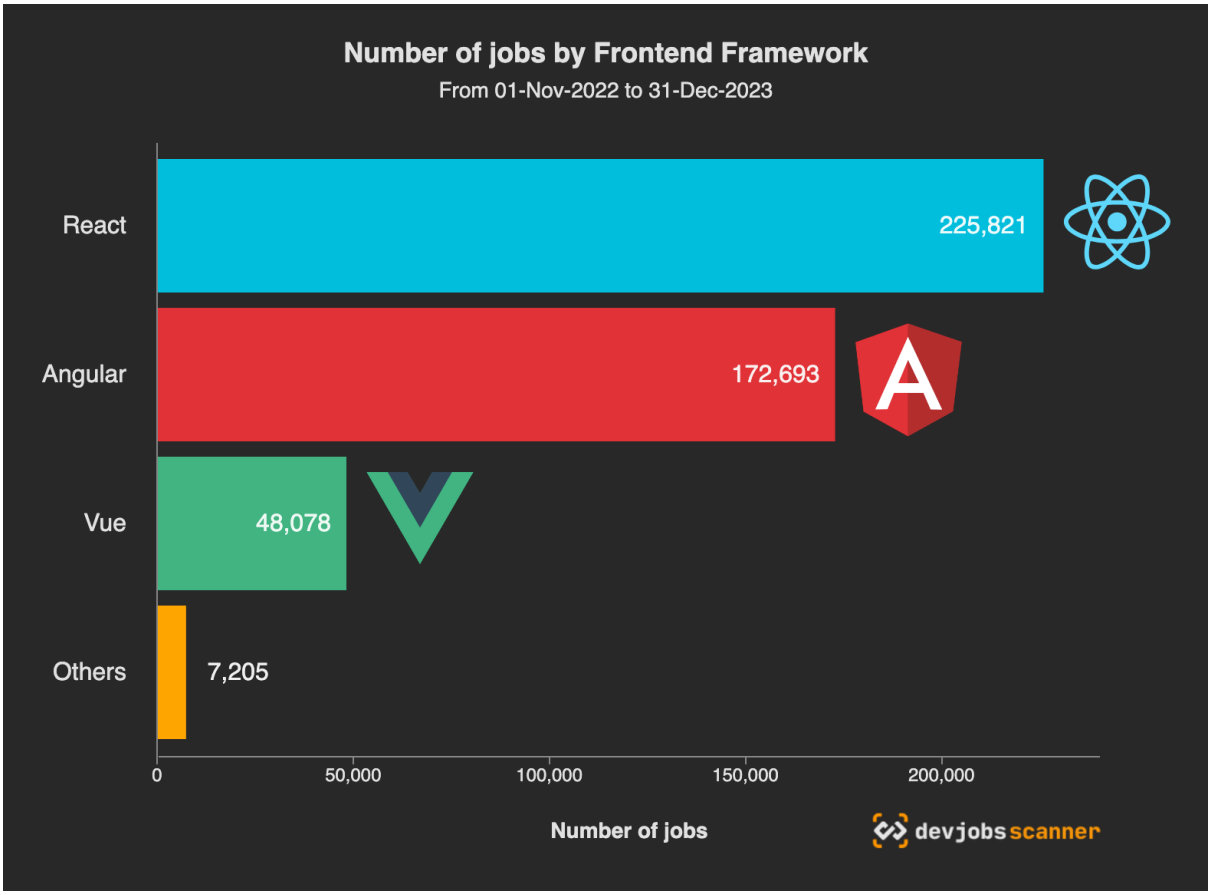

Figura 1: Number of jobs by Frontend Framwworkse [Devjobsscanner, 2023].

2.6.1 Vue.js

Historia y desarrollo

Vue.js, creado por Evan You en 2014, se ha distinguido por su enfoque minimalista y su capacidad para facilitar el desarrollo de interfaces de usuario complejas de una manera declarativa y reactiva. Originado como un proyecto individual, Vue ha crecido exponencialmente, ganándose la admiración de la comunidad de desarrolladores por su sencillez y eficacia.

Características principales

Vue.js se caracteriza por su curva de aprendizaje suave, lo que lo hace accesible para principiantes, mientras que su sistema reactivo y la arquitectura basada en componentes satisfacen las necesidades de aplicaciones más complejas. La documentación extensa y comprensible es frecuentemente citada como una de las principales razones de su popularidad. Además, Vue ofrece flexibilidad en la integración con otras bibliotecas o

proyectos existentes, lo que permite a los desarrolladores adoptar Vue gradualmente (You, 2014).

Ventajas y desventajas

Entre las ventajas de Vue.js se incluye su tamaño ligero, su sistema de reactividad eficiente, y la facilidad de integración. Sin embargo, a pesar de su creciente comunidad, puede enfrentarse a limitaciones en términos de disponibilidad de plugins específicos en comparación con sus competidores más establecidos como Angular o React.

2.6.2 Angular

Historia y desarrollo

AngularJS, lanzado por Google en 2010, evolucionó a Angular 2 en 2016, marcando un avance significativo con la adopción de TypeScript. Esta transformación reflejó un compromiso con el desarrollo web moderno, mejorando la modularidad, el rendimiento y la seguridad. Angular continúa actualizándose, adaptándose a las tendencias actuales como las SPA y el desarrollo móvil.

Características principales

Angular es conocido por su enfoque holístico, ofreciendo una solución completa para el desarrollo de aplicaciones. Con características como el enlace de datos bidireccional, inyección de dependencias, y un sistema de módulos, Angular busca facilitar el desarrollo de aplicaciones robustas y escalables. La adopción de TypeScript mejora la calidad del código mediante la adición de tipos estáticos (Google, 2023).

Ventajas y desventajas

La naturaleza completa de Angular lo hace ideal para proyectos grandes y complejos, pero su curva de aprendizaje puede ser desafiante para los nuevos desarrolladores. Además, la necesidad de adherirse a las convenciones específicas de Angular puede limitar la flexibilidad en comparación con frameworks más ligeros.

2.6.3 React

Historia y desarrollo

React fue desarrollado por Facebook y lanzado en 2013. Se introdujo como una solución para construir interfaces de usuario de alto rendimiento y dinámicas, especialmente para

aplicaciones web de gran escala. A diferencia de otros frameworks, React se centra en la capa de vista en el modelo MVC (Modelo-Vista-Controlador), lo que permite a los desarrolladores construir componentes reutilizables de UI.

Características principales

Una de las características distintivas de React es su Virtual DOM, que optimiza las actualizaciones del DOM real al minimizar el número de manipulaciones necesarias, resultando en un rendimiento mejorado. React también promueve la programación declarativa, lo que hace que el código sea más predecible y fácil de depurar. Además, su enfoque basado en componentes facilita la reutilización de código y mejora la organización del proyecto (Facebook, 2023).

Ventajas y desventajas

React es altamente valorado por su rendimiento y flexibilidad. La amplia adopción y la vasta comunidad ofrecen una extensa variedad de recursos, incluidos tutoriales, componentes, y herramientas de desarrollo. Sin embargo, React es solo una biblioteca para la interfaz de usuario y puede requerir la integración con otras bibliotecas para manejar funcionalidades como el enrutamiento y la gestión del estado, lo cual puede aumentar la complejidad del proyecto.

2.6.4 Comparativa de frameworks

Para realizar una comparativa efectiva entre Vue, Angular y React, es importante considerar varios criterios clave que son esenciales para el desarrollo frontend moderno. Estos criterios incluyen el rendimiento, la curva de aprendizaje, la comunidad y soporte, la flexibilidad y la escalabilidad. A continuación, se presenta un análisis comparativo basado en estos criterios. Esta comparativa surge tanto de experiencias propias de desarrollo como de algunos informes publicados. (Openinnova, 2024) (BrowserStack, 2024).

Rendimiento

El rendimiento en el desarrollo de aplicaciones web es crítico, y Angular, React y Vue ofrecen distintos enfoques en su manejo del Document Object Model (DOM), afectando su eficiencia.

Angular utiliza el DOM real, lo que puede ralentizar aplicaciones complejas debido a la manipulación directa del DOM en el navegador. Este enfoque puede complicar el desarrollo y la depuración en proyectos dinámicos.

React mejora el rendimiento mediante un DOM virtual, permitiendo actualizaciones eficientes al comparar cambios en un entorno virtual antes de aplicarlos al DOM real. Este método reduce la carga sobre el navegador, optimizando la experiencia del usuario.

Vue combina las ventajas del DOM real y virtual, adoptando un enfoque similar al de React para ofrecer reactividad y eficiencia, manteniendo una experiencia de usuario suave en aplicaciones complejas.

Cada tecnología tiene sus fortalezas y desafíos, y la elección depende de las necesidades específicas del proyecto y las preferencias del equipo de desarrollo.

Se puede observar en la Figura 2 la evaluación del rendimiento basado en las manipulaciones del DOM, según Openinnova (2024).

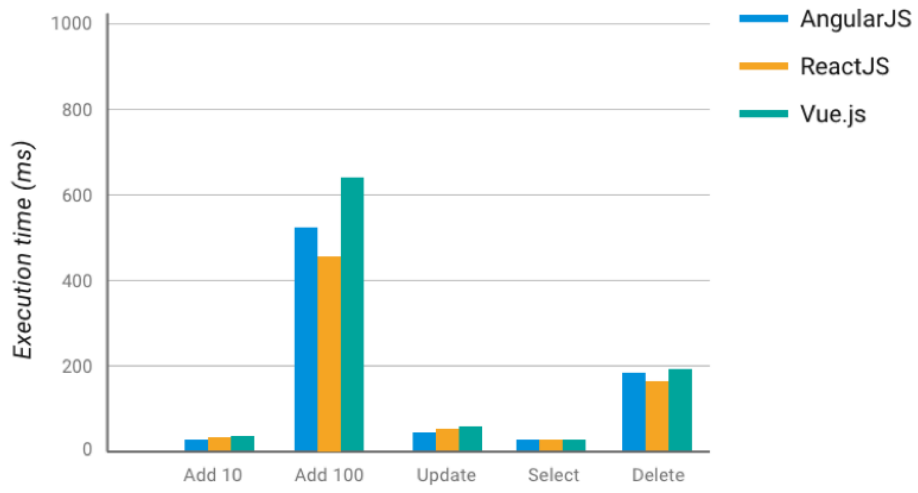

Figura 2: Performance evaluation based on DOM manipulations of table [Openinnova, 2024].

Curva de Aprendizaje

Vue es ampliamente reconocido por su curva de aprendizaje suave, lo que lo hace accesible para principiantes, mientras que Angular requiere una comprensión más profunda de conceptos como TypeScript y MVC. React se sitúa en un punto intermedio, con una curva de aprendizaje moderada pero una rápida adaptación gracias a su enfoque en componentes.

Se puede observar en la Figura 3 la evaluación de la curva de aprendizaje según Openinnova (2024).

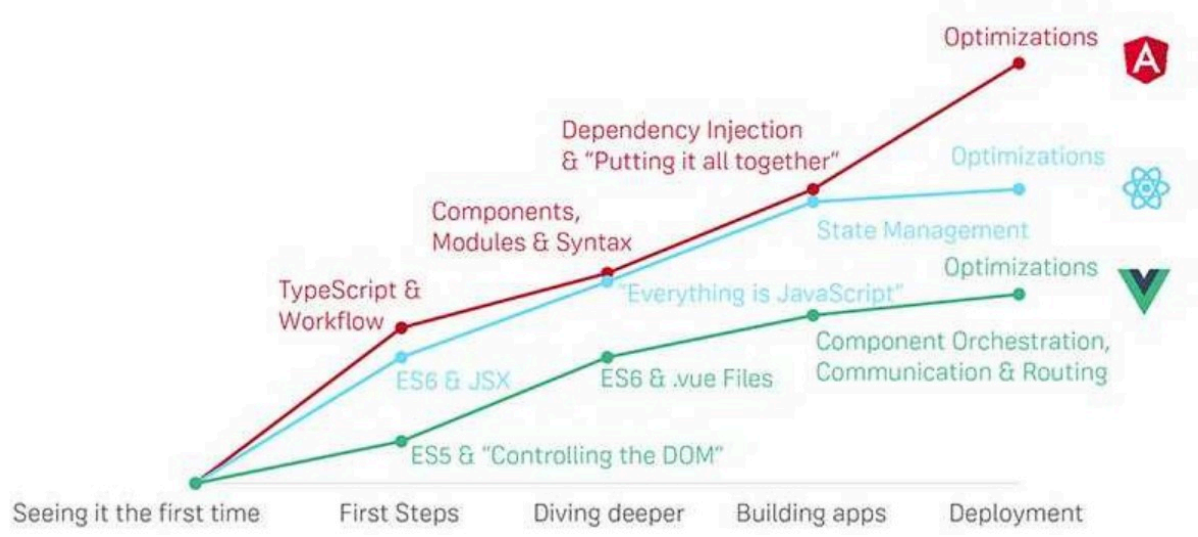

Figura 3: (Possible) Learning Curve [Openinnova, 2024].

Comunidad y soporte

React y Angular disfrutan del respaldo de grandes empresas (Facebook y Google, respectivamente) y tienen vastas comunidades de desarrolladores. Vue, aunque es un proyecto independiente, ha construido una comunidad fuerte y dedicada. La amplia disponibilidad de recursos de aprendizaje, bibliotecas complementarias y soporte comunitario es un factor crucial para los desarrolladores al elegir un framework.

Flexibilidad

Angular, React y Vue varían en flexibilidad. Angular es completo pero menos flexible, ideal para quienes prefieren un paquete integrado sin necesidad de herramientas adicionales. React destaca por su flexibilidad, permitiendo elegir libremente herramientas y bibliotecas, lo cual es preferido por desarrolladores que valoran la personalización. Vue se sitúa en un punto medio, ofreciendo flexibilidad con opciones convencionales para enrutamiento y gestión del estado, pero sin ser tan restrictivo como Angular ni tan abierto como React.

Escalabilidad

En cuanto a escalabilidad, Angular es reconocido por su robustez en aplicaciones empresariales grandes, gracias a su estructura y funcionalidades integradas. React, con su flexibilidad y el ecosistema rico en bibliotecas, es altamente escalable para proyectos de cualquier tamaño, adaptándose fácilmente a las necesidades cambiantes. Vue, equilibrado entre flexibilidad y convencionalismo, es escalable y adecuado tanto para proyectos pequeños como grandes, ofreciendo soluciones eficientes para la expansión. La elección entre estos depende de la preferencia entre un marco integral, la libertad de personalización

o un equilibrio de ambos, siempre considerando la capacidad de crecer y adaptarse del proyecto.

2.7 Tendencias tecnológicas

El desarrollo frontend está en constante evolución, adaptándose a las nuevas necesidades y expectativas de los usuarios. A medida que avanzamos, varias tendencias tecnológicas se destacan por su potencial para transformar la forma en que se diseñan y desarrollan las aplicaciones web.

2.7.1 Inteligencia artificial (AI) y aprendizaje automático (ML)

La integración de AI y ML en el desarrollo frontend está revolucionando la productividad y la eficiencia. Herramientas basadas en AI, como Sketch2Code de Microsoft, están automatizando tareas que antes eran manuales, transformando bocetos a mano en código HTML funcional. Este enfoque no solo mejora la productividad de los desarrolladores sino que también abre nuevas posibilidades en la personalización y la interacción del usuario (LambdaTest, 2023).

Dominio de JavaScript

JavaScript continúa siendo el lenguaje de programación dominante en el desarrollo frontend, con un uso casi universal en la web. La adopción unánime de JavaScript y su ecosistema de bibliotecas y frameworks, como ReactJS, AngularJS y Vue.js, subraya su importancia para el desarrollo de aplicaciones web modernas y dinámicas (LambdaTest, 2023).

Aplicaciones de página única (SPA)

Las SPA proporcionan una experiencia de usuario fluida y coherente, actualizando dinámicamente el contenido de la página web sin necesidad de recargar. Este enfoque mejora significativamente la velocidad de carga y la interactividad, siendo adoptado ampliamente en aplicaciones web modernas como Gmail, Netflix, y Pinterest (LambdaTest, 2023).

Micro frontends y arquitectura sin servidor (Serverless)

La adopción de micro frontends y arquitectura sin servidor refleja un movimiento hacia sistemas más modulares y eficientes. Estos enfoques facilitan el desarrollo escalable y

mantenible de aplicaciones, permitiendo despliegues independientes y reduciendo la sobrecarga operativa (FrontTribe, 2023).

Headless CMS y Jamstack

El uso de Headless CMS (Sistema de gestión de contenidos) y la arquitectura Jamstack están ganando popularidad por su flexibilidad y eficiencia. Al separar la gestión del contenido de la capa de presentación, estos enfoques permiten una entrega de contenido más rápida y adaptable a través de múltiples plataformas (FrontTribe, 2023).

Desarrollo Low-Code y No-Code

Las plataformas de desarrollo low-code y no-code están democratizando la creación de aplicaciones web, permitiendo a usuarios no técnicos participar en el desarrollo. Esta tendencia acelera el proceso de desarrollo y facilita la innovación al reducir las barreras técnicas (Codemotion, 2023).

Animaciones y diseño de movimiento

Las animaciones y el diseño de movimiento están enriqueciendo la experiencia del usuario, haciendo que las interacciones sean más atractivas e intuitivas. La capacidad de añadir animaciones complejas sin conocimientos avanzados de JavaScript destaca la evolución de CSS y las herramientas de diseño web (Fireart Studio, 2023).

2.8 Resumen del capítulo

En el presente capítulo se ha explorado detalladamente el desarrollo frontend, una disciplina esencial dentro del campo de la informática dedicada a la creación de interfaces de usuario y la experiencia de interacción en aplicaciones web y móviles. El capítulo comienza abordando los fundamentos del desarrollo frontend, destacando el papel crucial que desempeñan HTML, CSS y JavaScript como pilares tecnológicos. HTML se presenta como la estructura fundamental de la web, CSS como la herramienta clave para la estilización y diseño visual, y JavaScript como el motor que aporta interactividad y dinamismo a las páginas web. Además, se ha subrayado la evolución de estas tecnologías, como la transición a HTML5, CSS3 y las mejoras en ECMAScript, que han ampliado significativamente las capacidades del desarrollo web.

El capítulo también ha profundizado en la importancia de las librerías de componentes y frameworks en el desarrollo frontend moderno. Se ha llevado a cabo un análisis de herramientas populares como React, Angular y Vue.js, comparando sus características,

ventajas y desventajas. React se destaca por su enfoque en la eficiencia del DOM virtual y la modularidad del desarrollo, mientras que Angular ofrece una solución integral con TypeScript, y Vue.js se presenta como una opción flexible y accesible tanto para principiantes como para desarrolladores experimentados.

Finalmente, se han discutido las tendencias actuales en el desarrollo frontend, incluyendo el diseño responsivo, la accesibilidad web, y la creciente relevancia de tecnologías emergentes como la inteligencia artificial y el aprendizaje automático en la automatización y personalización de la experiencia de usuario. Estas tendencias reflejan la evolución constante de este campo y subrayan la necesidad de que los desarrolladores frontend mantengan un aprendizaje continuo para adaptarse a las nuevas demandas y tecnologías.

En conclusión, el capítulo proporciona una visión integral del desarrollo frontend, subrayando su importancia no solo en la creación de aplicaciones web estéticamente atractivas, sino también en la mejora de la accesibilidad, usabilidad y eficiencia en la interacción del usuario. El conocimiento profundo de estas tecnologías y prácticas es fundamental para cualquier desarrollador que aspire a construir aplicaciones modernas y escalables, capaces de ofrecer experiencias de usuario óptimas en un entorno web en constante cambio.

Capítulo 3. Diseño centrado en el usuario (UX/UI)

El diseño centrado en el usuario (UX/UI) es una disciplina esencial dentro del desarrollo de software, especialmente en el ámbito frontend, donde la interacción directa con los usuarios ocurre a través de interfaces visuales. Este enfoque se basa en la comprensión profunda de las necesidades, expectativas y comportamientos de los usuarios para crear interfaces que no solo sean funcionales, sino también intuitivas y agradables de utilizar (Brown, 2020). En este sentido, UX (Experiencia de Usuario) se refiere a cómo se siente un usuario al interactuar con un sistema, mientras que UI (Interfaz de Usuario) se centra en el aspecto visual y la disposición de los elementos dentro de la aplicación (Interaction Design Foundation, 2019).

El diseño UX/UI abarca una variedad de principios y prácticas que tienen como objetivo final mejorar la usabilidad, accesibilidad y satisfacción del usuario al interactuar con productos digitales (Gothelf, 2021). A través de una combinación de investigación de usuarios, diseño iterativo y pruebas de usabilidad, los desarrolladores y diseñadores pueden crear soluciones que no solo cumplen con los requisitos funcionales, sino que también proporcionan una experiencia de usuario positiva y memorable (Moran & Nielsen, 2020).

En el contexto del desarrollo frontend, la implementación efectiva de UX/UI es crucial para garantizar que las aplicaciones web y móviles no solo sean técnicamente competentes, sino también atractivas y fáciles de usar (Babich, 2019). Esto implica la integración de principios de diseño desde las primeras etapas de desarrollo, asegurando que cada componente y cada interacción sean diseñados teniendo en cuenta la perspectiva del usuario final (Miller, 2021).

Este capítulo tiene como objetivo establecer una base teórica sólida que guíe las decisiones de diseño durante el desarrollo frontend, mostrando cómo un enfoque centrado en el usuario puede llevar a la creación de productos más eficientes, accesibles y satisfactorios para el usuario final.

3.1 Principios de UX y UI

El diseño centrado en el usuario (UX/UI) es un aspecto crítico del desarrollo frontend, donde se busca no solo la creación de interfaces visualmente atractivas, sino también que estas sean intuitivas, accesibles y efectivas en la satisfacción de las necesidades del usuario. A continuación, se detallan los principios fundamentales que guían el diseño UX/UI y cómo estos impactan directamente en la experiencia del usuario final.

3.1.1 Usabilidad

La usabilidad es un pilar central en el diseño de interfaces de usuario. Se refiere a la facilidad con la que un usuario puede aprender a utilizar una interfaz y completar tareas específicas con eficiencia y satisfacción (Nielsen & Budiu, 2020). Los principios de usabilidad incluyen:

- **Simplicidad**: una interfaz debe ser lo más simple posible, eliminando elementos innecesarios que puedan distraer al usuario (Krug, 2020). Un diseño simple no solo reduce la carga cognitiva, sino que también facilita la navegación.
- **Claridad**: los elementos deben estar claramente identificados y organizados para que el usuario pueda entender de inmediato cómo interactuar con la interfaz. El uso de iconografía, etiquetas y diseños consistentes es esencial para lograr claridad (Babich, 2019).
- **Retroalimentación**: es fundamental proporcionar respuestas inmediatas a las acciones del usuario. Esto puede incluir cambios visuales en los botones cuando son presionados o mensajes que confirmen la realización de una acción (Miller, 2021).

Una interfaz altamente usable es crucial para evitar la frustración del usuario, lo cual puede derivar en la deserción del uso de la aplicación.

3.1.2 Accesibilidad

La accesibilidad en el diseño de interfaces se refiere a la creación de productos que pueden ser utilizados por la mayor cantidad de personas posible, incluidas aquellas con discapacidades (Cecchini & Roderick, 2020). Este principio asegura que todos los usuarios, independientemente de sus limitaciones, puedan interactuar eficazmente con la aplicación. Algunos aspectos clave incluyen:

- **Contraste de color adecuado**: asegurar que los textos y elementos interactivos sean legibles para usuarios con discapacidades visuales mediante el uso de un contraste de color fuerte (W3C, 2018).
- **Soporte para tecnologías de asistencia**: proveer alternativas textuales para imágenes y videos, y garantizar que todos los elementos interactivos sean accesibles mediante el teclado, lo cual es crucial para usuarios que no pueden usar un mouse (McNulty, 2021).
- **Diseño inclusivo**: considerar diferentes necesidades desde el inicio del diseño para asegurar que la interfaz sea usable por un público diverso (Clarkson, 2020).

En este proyecto, se emplearon herramientas y prácticas específicas para cumplir con estos estándares de accesibilidad. En particular, se utilizó Radix UI Primitive (Radix UI, s.f.) como base para los componentes. Esta biblioteca proporciona componentes preconfigurados que son accesibles desde su diseño inicial, facilitando la navegación con teclado y la compatibilidad con lectores de pantalla. Además, se implementó Class Variance Authority (CVA Docs, s.f.) para gestionar variantes de estilo en los componentes, permitiendo mantener la coherencia visual sin sacrificar la accesibilidad en distintas configuraciones de diseño. Estas herramientas ayudan a crear una experiencia inclusiva y aseguran que cada componente cumpla con las mejores prácticas de accesibilidad.

La accesibilidad no solo cumple con las normativas legales en muchos países, sino que también amplía el alcance potencial de una aplicación y contribuye a una experiencia de usuario inclusiva para todos.

3.1.3 Diseño centrado en el usuario

El diseño centrado en el usuario (DCU) es un enfoque iterativo que pone al usuario final en el centro del proceso de diseño. Este método se basa en la investigación de usuarios para identificar sus necesidades y expectativas, y en la validación continua a través de pruebas de usabilidad (Norman, 2020). Los pasos fundamentales incluyen:

- **Investigación del usuario**: comprender a fondo quiénes son los usuarios, sus comportamientos, y cómo interactúan con productos similares (Baxter, Courage, & Caine, 2019).
- **Prototipado y pruebas**: crear prototipos que permitan evaluar y mejorar la interfaz antes del lanzamiento final, basándose en la retroalimentación directa de los usuarios (Righi & James, 2020).
- **Iteración**: refinar continuamente el diseño en función de las pruebas de usabilidad y la retroalimentación del usuario, asegurando que el producto final cumpla con las expectativas y necesidades del usuario (Moran, 2020).

Este enfoque garantiza que la interfaz desarrollada no solo cumpla con los requisitos técnicos, sino que también ofrezca una experiencia satisfactoria y efectiva.

3.1.4 Consistencia en el diseño

La consistencia en el diseño UX/UI implica mantener uniformidad en todos los elementos de la interfaz, lo que facilita al usuario aprender y recordar cómo interactuar con la aplicación (Babich, 2019). Los aspectos clave incluyen:

- **Diseño visual consistente**: mantener una paleta de colores, tipografía y estilos gráficos uniformes a lo largo de toda la aplicación para proporcionar una experiencia coherente (Johnson, 2021).
- **Comportamiento interactivo consistente**: asegurar que los elementos interactivos (como botones y menús) funcionen de la misma manera en todas las partes de la aplicación, lo que reduce la confusión del usuario (Garrett, 2020).
- **Terminología consistente**: usar un lenguaje uniforme en todas las áreas de la aplicación para evitar malentendidos (Rogers, Sharp, & Preece, 2020).

La consistencia es clave para construir una experiencia de usuario fluida y predecible, lo que a su vez aumenta la eficiencia y satisfacción del usuario.

3.1.5 Importancia de la estética en la interfaz

Aunque la funcionalidad es primordial, la estética juega un papel crucial en la percepción del usuario sobre la calidad de una aplicación. Un diseño visualmente atractivo no solo capta la atención, sino que también puede influir en la satisfacción y el disfrute del usuario (Hassenzahl & Tractinsky, 2019). Elementos importantes incluyen:

- **Primera impresión**: un diseño estéticamente agradable puede crear una impresión inicial positiva, lo que es crucial para la retención del usuario (Moshagen & Thielsch, 2020).
- **Credibilidad**: los usuarios tienden a asociar un buen diseño con profesionalismo y confiabilidad, lo que puede aumentar la credibilidad de la aplicación (Fogg, 2019).
- **Emoción y placer**: un diseño atractivo puede evocar emociones positivas, haciendo que el uso de la aplicación sea más placentero y memorable (Norman, 2020).

Los principios de UX y UI no solo son esenciales para la creación de interfaces que sean funcionales y accesibles, sino también para garantizar que los usuarios tengan una experiencia agradable y satisfactoria. La implementación de estos principios en el desarrollo frontend asegura que las aplicaciones no solo cumplan con los estándares técnicos, sino que también ofrezcan un valor real y percibido a los usuarios finales.

3.2 Integración de UX/UI en el desarrollo de componentes

El desarrollo de componentes frontend es una práctica clave en la construcción de interfaces web modernas. Los componentes son bloques de construcción reutilizables que encapsulan tanto la lógica como la presentación de una parte específica de la interfaz de usuario. La integración de principios de UX/UI en el desarrollo de estos componentes es

fundamental para garantizar que cada elemento no solo funcione correctamente desde un punto de vista técnico, sino que también ofrezca una experiencia de usuario coherente y agradable.

3.2.1 Diseño de componentes reutilizables

Uno de los aspectos más importantes en el desarrollo frontend es la creación de componentes que puedan ser reutilizados en diferentes partes de una aplicación. Estos componentes deben ser diseñados de manera que puedan integrarse de forma flexible en diversos contextos sin perder su funcionalidad ni su consistencia visual (Feldman, 2019). Al integrar principios de UX/UI desde el inicio, los desarrolladores pueden asegurar que estos componentes sean:

- **Consistentes:** la consistencia en el diseño de componentes asegura que los usuarios no tengan que aprender nuevas interacciones cada vez que utilizan un componente similar en diferentes partes de la aplicación. Esto incluye mantener una uniformidad en la tipografía, colores, y comportamiento de los elementos interactivos (Wolfe, 2020).
- **Intuitivos:** un componente bien diseñado debe ser fácil de entender y usar desde el primer momento. Esto implica una disposición clara de los elementos, etiquetas descriptivas, y señales visuales que guíen al usuario (Babich, 2020).
- **Accesibles:** la accesibilidad debe ser incorporada en el desarrollo de componentes desde el inicio, garantizando que cada componente cumpla con las pautas de accesibilidad web (WCAG). Esto incluye, por ejemplo, la capacidad de ser navegado con teclado, y el uso de texto alternativo para imágenes dentro de los componentes (Clarkson 2020).

3.2.2 Desarrollo con un enfoque modular

El enfoque modular en el desarrollo frontend permite a los desarrolladores crear componentes que pueden ser ensamblados para formar interfaces más complejas. Este enfoque facilita la reutilización de código y la consistencia en el diseño (Garrett, 2020). Integrar UX/UI en este enfoque implica:

- **Modularidad y Escalabilidad:** los componentes deben ser lo suficientemente modulares para ser combinados en diferentes configuraciones sin perder la coherencia de la experiencia del usuario. Al diseñar componentes que son tanto

modulares como escalables, se facilita la expansión y mantenimiento de la interfaz a lo largo del tiempo (Gothelf, 2021).

- **Flexibilidad:** los componentes deben ser flexibles para adaptarse a diferentes dispositivos y tamaños de pantalla, lo que es crucial en un entorno donde el diseño responsivo es la norma. Esto incluye el uso de diseños fluidos y la capacidad de los componentes para adaptarse a diferentes contextos sin necesidad de modificaciones importantes (Wolfe, 2020).

3.2.3 Prototipado y pruebas de usabilidad en componentes

Antes de que un componente sea implementado en producción, es crucial que sea probado para asegurar que cumple con los estándares de UX/UI. El prototipado es una etapa clave en este proceso, permitiendo a los desarrolladores y diseñadores visualizar y evaluar cómo se comportarán los componentes en un entorno real (Brown, 2020).

- **Prototipado rápido:** herramientas como Figma o Sketch permiten la creación de prototipos interactivos que simulan el comportamiento de los componentes en una interfaz real. Estos prototipos pueden ser utilizados para realizar pruebas de usabilidad con usuarios reales, identificando posibles problemas de interacción antes de que el componente sea finalizado (Rogers, Sharp, & Preece, 2020).
- **Pruebas iterativas:** la retroalimentación obtenida de las pruebas de usabilidad debe ser utilizada para iterar en el diseño del componente, refinando su funcionalidad y mejorando la experiencia del usuario. Este proceso iterativo es esencial para asegurar que cada componente no solo cumple su función técnica, sino que también proporciona una experiencia de usuario óptima (Moran & Nielsen, 2020).

3.2.4 Implementación y mantenimiento de componentes UX/UI

La implementación de componentes que integran principios de UX/UI no es el final del proceso. Es fundamental mantener estos componentes actualizados y alineados con los estándares de diseño y usabilidad a medida que la aplicación evoluciona (Miller, 2021).

- **Actualizaciones regulares:** los componentes deben ser revisados y actualizados regularmente para asegurarse de que siguen cumpliendo con las necesidades del usuario y las mejores prácticas de UX/UI. Esto incluye ajustar el diseño para adaptarse a nuevas tendencias o tecnologías, así como mejorar la accesibilidad y la usabilidad basándose en nuevas investigaciones o retroalimentación de los usuarios (McNulty, 2021).

- **Documentación y guías de estilo:** mantener una documentación clara y detallada para cada componente es crucial para asegurar su correcta reutilización y adaptación en diferentes contextos. Las guías de estilo, que incluyen reglas sobre el uso de colores, tipografía, y comportamiento interactivo, son herramientas valiosas para mantener la coherencia en el diseño de una aplicación (Babich, 2020).

La integración de principios de UX/UI en el desarrollo de componentes frontend es esencial para crear interfaces que sean no solo funcionales, sino también coherentes y centradas en el usuario. Al considerar estos principios desde el inicio, los desarrolladores pueden asegurarse de que cada componente contribuye a una experiencia de usuario positiva, lo que en última instancia mejora la calidad y el éxito de la aplicación.

3.3 Sinergia entre diseño y funcionalidad

El desarrollo frontend es un proceso que involucra la fusión de dos áreas clave: el diseño y la funcionalidad técnica. Lograr un equilibrio adecuado entre estas dos dimensiones es fundamental para crear productos que no solo sean visualmente atractivos, sino también eficientes y usables. Esta sección explora cómo se logra esta sinergia en el desarrollo frontend, analizando los compromisos necesarios y la importancia de la colaboración interdisciplinaria.

3.3.1 Equilibrio entre diseño estético y funcionalidad técnica

El diseño estético y la funcionalidad técnica a menudo son percibidos como fuerzas en tensión. Un diseño que es visualmente impresionante puede requerir implementaciones técnicas complejas que podrían afectar la velocidad de carga o la usabilidad. Por otro lado, una solución técnicamente eficiente podría carecer del atractivo visual necesario para captar y mantener la atención del usuario (Coyle, 2019).

- **Prioridades en el diseño:** el primer paso para lograr un equilibrio es establecer las prioridades del proyecto. Esto incluye decidir qué aspectos del diseño son esenciales para la experiencia del usuario y cuáles pueden ser sacrificados en favor de una mejor funcionalidad técnica. Por ejemplo, se podría optar por un diseño menos elaborado en términos de animaciones para mejorar la velocidad de carga en dispositivos móviles (Wroblewski, 2019).
- **Compromisos técnicos:** a veces, es necesario hacer compromisos técnicos para preservar la integridad del diseño. Esto podría implicar el uso de tecnologías más avanzadas o técnicas de optimización para mantener un alto rendimiento sin

sacrificar la estética (Baxter, 2020). El uso de frameworks y herramientas que permiten una integración más fluida entre el diseño y la funcionalidad, como React o Vue.js, puede facilitar este proceso (Feldman, 2020).

3.3.2 La colaboración entre diseñadores y desarrolladores

La colaboración entre diseñadores y desarrolladores es esencial para lograr un producto final que sea tanto estéticamente agradable como técnicamente robusto. Esta colaboración implica una comunicación constante y un entendimiento mutuo de las limitaciones y posibilidades de cada disciplina.

- **Proceso de diseño iterativo:** un proceso iterativo en el cual diseñadores y desarrolladores trabajan juntos desde las primeras etapas del proyecto puede ayudar a identificar y resolver problemas antes de que se conviertan en barreras significativas. Por ejemplo, los desarrolladores pueden proporcionar retroalimentación sobre la viabilidad técnica de ciertos elementos de diseño, lo que permite a los diseñadores ajustar sus propuestas en consecuencia (Rogers et al., 2020).
- **Herramientas de colaboración:** el uso de herramientas de colaboración como Figma o Adobe XD, que permiten a los diseñadores crear prototipos interactivos que pueden ser directamente inspeccionados y comentados por los desarrolladores, facilita la transición del diseño a la implementación. Estas herramientas también permiten a los diseñadores y desarrolladores compartir un lenguaje común y una comprensión clara de los objetivos del proyecto (Kholmatova, 2016).
- **Guías de estilo y design systems:** la creación y uso de guías de estilo y sistemas de diseño (Design Systems) son prácticas que aseguran la coherencia entre el diseño y la implementación técnica. Estos recursos proporcionan a los desarrolladores un conjunto de componentes y patrones de diseño predefinidos que pueden ser reutilizados a lo largo de la aplicación, asegurando que el producto final mantenga una apariencia y comportamiento consistentes (Feldman, 2020).

3.3.3 Manejo de compromisos en el desarrollo frontend

El manejo de compromisos es una parte inevitable del desarrollo frontend. Las decisiones de diseño y funcionalidad a menudo deben ser balanceadas para satisfacer tanto las expectativas del usuario como las limitaciones técnicas.

- **Optimización del rendimiento:** un diseño visualmente complejo puede afectar negativamente el rendimiento de la aplicación, especialmente en dispositivos con recursos limitados. Para mitigar esto, los desarrolladores pueden implementar técnicas como el lazy loading, minificación de archivos, y optimización de imágenes para mejorar el tiempo de carga sin sacrificar la calidad visual (Marcotte, 2019).
- **Accesibilidad versus estética:** en algunos casos, el compromiso entre accesibilidad y estética puede surgir cuando un diseño visual atractivo no cumple con las pautas de accesibilidad. En tales situaciones, es crucial priorizar la accesibilidad, ajustando el diseño para asegurar que todos los usuarios puedan interactuar con el producto de manera efectiva (Clarkson 2020).
- **Escalabilidad y mantenibilidad:** los compromisos en la escalabilidad y mantenibilidad del código también son comunes. Un diseño altamente personalizado puede dificultar la implementación de nuevas características o el mantenimiento a largo plazo del código. Por lo tanto, es importante que los desarrolladores consideren la sostenibilidad del proyecto al decidir sobre la implementación técnica (Garrett, 2020).

3.3.4 La importancia de la colaboración interdisciplinaria

La sinergia entre diseño y funcionalidad no solo depende de las decisiones técnicas y de diseño, sino también de una colaboración interdisciplinaria efectiva. Diseñadores y desarrolladores deben trabajar juntos como un equipo cohesionado, comprendiendo y respetando las habilidades y perspectivas de cada uno.

- **Comunicación abierta:** mantener una comunicación abierta y constante entre diseñadores y desarrolladores es clave para identificar y resolver problemas rápidamente. Reuniones regulares, retroalimentación continua, y la disposición para adaptarse a los cambios son elementos esenciales de esta colaboración (Moran & Nielsen, 2020).
- **Educación mutua:** fomentar la educación mutua entre diseñadores y desarrolladores ayuda a cerrar la brecha entre diseño y tecnología. Los diseñadores pueden beneficiarse de un mayor conocimiento técnico, mientras que los desarrolladores pueden aprender más sobre los principios de diseño, lo que en última instancia mejora la calidad del producto final (Babich, 2020).

Lograr un equilibrio entre diseño y funcionalidad en el desarrollo frontend es un desafío que requiere un enfoque equilibrado y colaborativo. Al manejar cuidadosamente los compromisos y fomentar una colaboración interdisciplinaria, es posible crear productos que

no solo cumplan con los estándares técnicos, sino que también ofrezcan una experiencia de usuario visualmente atractiva y altamente funcional.

3.4 Resumen del capítulo

En este capítulo, se ha explorado la importancia fundamental del diseño centrado en el usuario (UX/UI) en el desarrollo frontend, subrayando cómo los principios de usabilidad, accesibilidad, consistencia y estética son esenciales para crear interfaces que no solo sean funcionales, sino también intuitivas y visualmente atractivas. A través del análisis de los principios de UX/UI, se ha destacado cómo estos guían las decisiones de diseño para mejorar la experiencia del usuario final, asegurando que las interfaces sean fáciles de usar y accesibles para una audiencia diversa.

Asimismo, se ha profundizado en la integración de estos principios en el desarrollo de componentes frontend, resaltando la necesidad de diseñar componentes reutilizables que mantengan la coherencia visual y funcional a lo largo de una aplicación. Este enfoque modular no solo facilita el mantenimiento y la escalabilidad, sino que también garantiza que cada componente contribuya positivamente a la experiencia del usuario.

Por último, se ha analizado la sinergia entre el diseño y la funcionalidad técnica, destacando la importancia de la colaboración interdisciplinaria entre diseñadores y desarrolladores. El equilibrio entre un diseño estéticamente agradable y una implementación técnica eficiente es crucial para el éxito de cualquier proyecto frontend. Este capítulo concluye enfatizando que, al integrar eficazmente los principios de UX/UI desde el inicio y fomentar una colaboración continua entre equipos, se pueden crear productos digitales que no solo cumplan con los requisitos técnicos, sino que también proporcionen una experiencia de usuario excepcional.

Capítulo 4. El Desarrollo realizado

El desarrollo de una librería de componentes para agilizar el inicio de proyectos de desarrollo frontend fue una iniciativa clave dentro de la empresa Truenorth. Este capítulo se centra en detallar el proceso de creación de esta librería, la elección de las herramientas y tecnologías empleadas, y el enfoque colaborativo adoptado por los equipos de diseño y desarrollo.

El propósito principal de esta librería es estandarizar y optimizar el proceso de desarrollo de interfaces de usuario, facilitando la reutilización de componentes visuales y funcionales a lo largo de diferentes proyectos. Esto no solo permite una mayor coherencia visual y eficiencia en el desarrollo, sino que también mejora la mantenibilidad y escalabilidad de las aplicaciones creadas.

A lo largo del capítulo, se exploran los pasos tomados para la creación de la biblioteca de componentes, las razones detrás de la elección de React como tecnología base, y cómo se integraron las herramientas adecuadas para asegurar que cada componente fuera robusto, reutilizable y alineado con las necesidades del usuario. Además, se analiza el proceso colaborativo que permitió que diseñadores y desarrolladores trabajaran de manera sinérgica para asegurar que la librería no solo cumpliera con los estándares técnicos, sino que también proporcionara una experiencia de usuario de alta calidad.

Finalmente, se presentan ejemplos prácticos a través del sitio de documentación creado para la librería, donde se pueden observar los componentes trabajando en conjunto. Este sitio permite visualizar cómo los diferentes componentes pueden ser integrados y reutilizados en diversas configuraciones, demostrando su versatilidad y eficacia. Además, el sitio de documentación sirve como una guía interactiva para los equipos de desarrollo, facilitando la adopción y aplicación de la librería en futuros proyectos. Este capítulo ofrece una visión detallada del desarrollo realizado, destacando las decisiones clave y los resultados obtenidos a partir de este esfuerzo colaborativo.

4.1 Creación de la biblioteca de componentes

La creación de la librería de componentes fue un esfuerzo estratégico para estandarizar el desarrollo frontend dentro de la empresa. Este proceso comenzó con una fase de investigación y planificación, donde se definieron los requisitos técnicos y se alinearon con las necesidades del equipo de diseño. La intención era crear una colección de componentes

reutilizables que pudieran ser implementados de manera consistente en diferentes proyectos, facilitando la coherencia visual y funcional.

4.1.2 Diseño y planificación

El proceso de creación comenzó con una serie de reuniones entre los equipos de desarrollo y diseño para definir los requisitos de la librería. Se establecieron criterios para la creación de componentes que fueran modulares, fácilmente personalizables, y alineados con el Design System de la empresa. El resultado de esta fase fue un conjunto de guías que dictaron las pautas para el desarrollo de cada componente.

En esta fase, también se definieron las categorías de componentes (como botones, inputs, etc.) y se planificó el flujo de trabajo para su implementación. Para gestionar y priorizar la creación de los componentes, se utilizó Jira, una herramienta que permitió mantener una visión clara y estructurada de las tareas. A través de Jira, se crearon y asignaron tickets para cada componente, lo que facilitó la coordinación entre los equipos y aseguró que los componentes más críticos para las necesidades inmediatas de la empresa fueran abordados primero.

4.1.3 Desarrollo de componentes

El desarrollo de la librería de componentes se llevó a cabo tras una fase exhaustiva de planificación, en la cual se seleccionaron React y TypeScript como tecnologías base. Además, se incorporó Radix UI Primitive para construir sobre sus componentes base, que están diseñados para ser accesibles y altamente modulares. Esta decisión estratégica permitió asegurar que cada componente fuera modular, reutilizable y fácilmente personalizable según los requerimientos de cada proyecto, además de cumplir con las mejores prácticas de accesibilidad y consistencia en el diseño. Radix UI Primitive facilitó la creación de una estructura de componentes sólida y optimizada para la accesibilidad desde el inicio, permitiendo a los desarrolladores concentrarse en la lógica específica y la personalización de los componentes. Gracias a este enfoque, los desarrolladores pudieron ajustarse a las necesidades de los distintos proyectos, manteniendo una consistencia en las interfaces de usuario a lo largo del tiempo.

Para organizar mejor la librería y facilitar su uso, los componentes se agruparon en categorías, destacando las características que se aplican de manera general a todos ellos. Estos aspectos incluyen validaciones en los campos de entrada, personalización de los

estilos, opciones de interacción y accesibilidad. A continuación, se presenta una clasificación de los componentes:

1. **Componentes de Entrada de Datos:**
    - **Form Inputs**: campos de entrada de texto, checkbox, radio button, date picker, multiselector, y selector.
      Descripción: permiten al usuario ingresar datos, seleccionar opciones o fechas, y realizar entradas múltiples.
    - **Toggle**: interruptores que permiten cambiar entre dos estados.
      Descripción: útil para alternar configuraciones simples de activado/desactivado.
2. **Componentes Interactivos**
    - **Botones**: button, icon button.
      Descripción: botones estándar y de icono para iniciar acciones en la interfaz.
    - **Dropdown**: Menús desplegables que muestran opciones adicionales.
      Descripción: Permite al usuario seleccionar una opción de una lista desplegable.
    - **Modal y drawer**: ventanas emergentes y paneles laterales.
      Descripción: usados para mostrar contenido adicional sin cambiar de página.
    - **Toast**: mensajes breves que aparecen temporalmente.
      Descripción: usado para notificaciones rápidas y temporales.
    - **Tooltip**: descripciones emergentes para elementos.
      Descripción: ofrece información adicional al pasar el ratón sobre un elemento.
3. **Componentes de Visualización**
    - **Alert**: mensajes de alerta.
      Descripción: proporciona información importante que puede requerir acción o reconocimiento.
    - **Progress**: indicadores de progreso.
      Descripción: muestra el progreso de una tarea en ejecución.
    - **Badge y tag**: etiquetas para mostrar estado o cantidades.
      Descripción: indica estados, categorías o cantidades asociadas a un elemento.
    - **Avatar**: representación visual de usuarios o entidades.
      Descripción: muestra una miniatura de usuario, a menudo con una imagen o iniciales.

4. **Componentes de Navegación**
    - **Breadcrumb**: navegación jerárquica.
      Descripción: muestra la ubicación actual en la estructura de navegación.
    - **Tabs**: organización en pestañas para secciones de contenido.
      Descripción: facilita la navegación entre distintas secciones sin cambiar de página.
    - **Link**: enlaces de navegación.
      Descripción: redirige a otras páginas o secciones de la interfaz.
5. **Componentes de organización y estructura**
    - **Accordion**: expande o colapsa secciones de contenido.
      Descripción: permite mostrar u ocultar contenido adicional según sea necesario.
    - **Card**: contenedor de contenido con estilo.
      Descripción: usado para agrupar información o acciones en una interfaz, como en el dashboard.
    - **Table**: presentación de datos tabulares.
      Descripción: organiza datos en filas y columnas, con opciones de clasificación.
    - **Header**: encabezado de secciones o páginas.
      Descripción: marca el título o encabezado principal de una sección.
6. **Componentes de selección y filtros**
    - **Selector**: desplegable de selección simple.
      Descripción: permite seleccionar una opción de una lista.
    - **Date picker**: selección de fechas.
      Descripción: facilita la entrada de fechas a través de un calendario emergente.
    - **Checkbox y radio button**: opciones de selección múltiple y única.
      Descripción: selección de opciones en formularios y filtros.

Características generales aplicadas a todos los componentes

En todos los casos, los componentes desarrollados respetan las siguientes normas y características generales, lo que garantiza una coherencia y facilidad de uso en cualquier proyecto frontend:

1. **Validaciones de entrada**: todos los campos de entrada, como los inputs y áreas de texto, están equipados con validaciones automáticas. Estas validaciones se adaptan

al tipo de dato ingresado, asegurando que el usuario solo pueda ingresar datos correctos y en el formato adecuado.
    a. **Ejemplo de validación**: un campo de correo electrónico solo acepta entradas en formato user@example.com.
    b. **Uso de tooltips**: los inputs incluyen tooltips que guían al usuario en caso de error o cuando se requiere una aclaración adicional sobre el tipo de dato esperado.
2. **Personalización de estilos**: los componentes son altamente personalizables en términos de estilos, lo que permite a los desarrolladores adaptar los colores, tamaños, y formas de los elementos para integrarse sin problemas en cualquier proyecto.
3. **Accesibilidad y usabilidad**: todos los componentes cumplen con las normas de accesibilidad, garantizando que las personas con discapacidades puedan navegar y utilizar las interfaces sin dificultad.

Propiedades y personalización de los componentes

Todos los componentes desarrollados dentro de la librería siguen un patrón de personalización basado en props que permiten la modificación de aspectos como estilo, tamaño, comportamiento y estado.

A modo de ejemplo se describen las propiedades de uno de los componentes realizados.

Propiedades del componente Botón

1. **Estilos personalizados:** el botón permite configurarse con distintos estilos visuales a través de propiedades personalizables aplicadas en el componente. Esto incluye variaciones en colores de fondo, bordes y tipografía, lo que facilita la adaptación de los botones a diferentes temas o esquemas de colores según las necesidades del proyecto. En la Figura 4 se muestra un conjunto de botones con diferentes estilos visuales: el estilo **Primary** en azul, **Secondary** en verde, **Neutral** en gris claro y **Destructive** en rojo, cada uno pensado para representar diferentes acciones o niveles de importancia en la interfaz.

Figura 4: Diferentes estilos de un botón [Figma Mate-UI, 2024].

2. **Tamaños ajustables:** el componente admite la modificación de su tamaño mediante propiedades como size, lo que permite que se ajuste a diferentes contextos de la interfaz como podemos ver en la Figura 5.

Figura 5: Diferentes tamaños de un botón [Figma Mate-UI, 2024].

3. **Incorporación de íconos**: el botón admite la adición de íconos en el lado izquierdo o derecho del texto, mejorando la comprensión del usuario sobre la acción que realiza el botón. La Figura 6 muestra ejemplos de botones con íconos asociados a acciones comunes.

Figura 6:Uso de iconos en un botón [Figma Mate-UI, 2024].

4. **Estados y variantes del botón:** para asegurar una experiencia de usuario coherente y clara, el sistema de diseño define distintos estados y variantes del botón. En la Figura 7, se presentan tres variantes principales del botón: **Filled** (lleno), **Outlined** (con borde) y **Ghost** (transparente), cada una adaptada para diferentes propósitos de diseño e interacción.

   Cada variante incluye una serie de estados específicos:

   - **Default**: el estado estándar del botón, visible cuando el usuario no está interactuando con él.

- **Hover**: este estado se activa al pasar el cursor sobre el botón, proporcionando retroalimentación visual y mejorando la interactividad.
- **Pressed**: muestra el botón cuando se hace clic o se mantiene presionado, indicando la acción en progreso.
- **Focus**: resalta el botón cuando es seleccionado, optimizando la accesibilidad para usuarios que navegan con teclado.
- **Disabled**: el botón aparece inactivo, lo que indica que la acción no está disponible en ese momento.
- **Loading**: este estado muestra un indicador de carga (spinner) en lugar del texto, para señalar al usuario que la acción está en proceso.

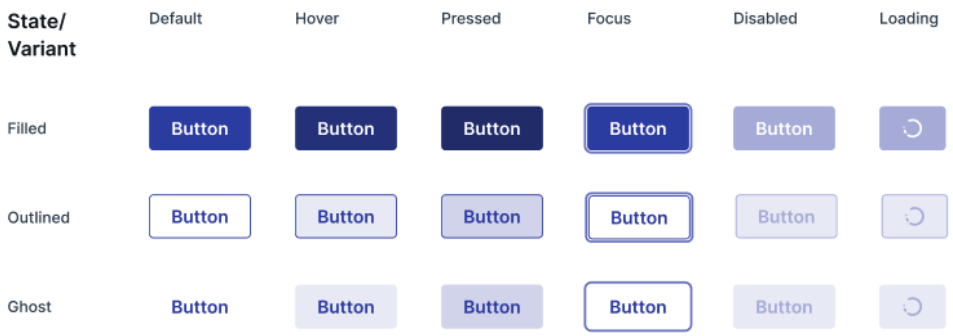

Figura 7: Diferentes variantes y estados del botón [Figma Mate-UI, 2024].

Estos estados y variantes son fundamentales para una interacción intuitiva y consistente en la interfaz, permitiendo a los desarrolladores aplicar los estilos adecuados según la situación y el contexto del botón en la aplicación.

Comportamiento y estado de los botones

El componente también soporta diferentes estados interactivos que mejoran la experiencia de usuario. Algunos de estos incluyen:

- **Estado "Deshabilitado"**: un botón que no responde a interacciones, útil para momentos en que una acción no está disponible.
- **Estado "Cargando"**: un botón que muestra un ícono de carga mientras se realiza una operación en segundo plano, como se muestra en la Figura 6 con el botón de "Sending".

Estas propiedades aseguran que el botón sea completamente flexible, adaptable y alineado con los estándares de accesibilidad y usabilidad. Gracias a este enfoque, los

desarrolladores pueden reutilizar el componente en diferentes partes de las aplicaciones sin perder consistencia en el diseño ni en la funcionalidad.

Ejemplo de implementación del componente Button

El componente Button es uno de los elementos más complejos de la biblioteca. Su estructura está distribuida en varios archivos, cada uno con una responsabilidad específica. Esto permite modularidad y facilita la configuración de variantes, estilos y propiedades adicionales, necesarias para lograr la versatilidad del componente.

A continuación se detallan todos los archivos involucrados en la creación del componente Button:

- **button.stories.tsx**: contiene ejemplos de uso del componente Button para Storybook, permitiendo la visualización y prueba de diferentes configuraciones del botón. Este archivo no es parte de la implementación funcional, sino una herramienta de documentación y pruebas.
- **button.tsx**: este archivo contiene la lógica principal del componente Button. Aquí se define la estructura del botón, el manejo de sus propiedades y la lógica para la asignación de estilos y variantes. También se incluyen configuraciones para íconos (leftIcon y rightIcon) y estados (loading, disabled).
- **index.ts**: este archivo actúa como un punto de entrada para el componente Button, exportando todos los módulos necesarios (Button, styles, y ButtonProps). Al centralizar las exportaciones, permite que el componente pueda ser importado fácilmente en otros módulos de la aplicación, manteniendo el código organizado y accesible.
- **interfaces.ts**: define los tipos e interfaces de las propiedades del componente Button. Este archivo asegura que el componente se utilice correctamente, al especificar los tipos para propiedades como variant, size, themeColor, entre otros.
- **styles.ts**: en este archivo se definen los estilos base y variantes del botón usando Class Variance Authority (CVA). Aquí se configuran las opciones de estilo que se aplicarán al componente en función de las propiedades variant, themeColor y size, facilitando la personalización modular.

Ejemplo de configuración de estilos con CVA

La implementación de Button utiliza CVA para definir y aplicar variantes de estilo, de modo que las propiedades como variant, themeColor, y size se configuran de forma centralizada.

Esto permite que el componente se adapte a diferentes combinaciones de estilo sin necesidad de escribir estilos individuales para cada configuración.

1. **Definición de estilos base y variantes**: en el archivo styles.ts, se utiliza CVA para definir los estilos base y las variantes del botón. Esto incluye:
    - **rootBase**: define las propiedades de estilo generales que se aplican a todos los botones, independientemente de su variante o tema de color.
    - **variant**: permite cambiar el estilo del botón entre filled, outlined y ghost, especificando el tipo de borde y fondo.
    - **themeColor**: controla el color del botón (primary, secondary, neutral, destructive) para que se ajuste al tema visual de la aplicación.
    - **size**: determina el tamaño del botón (sm, md, lg, xl), configurando el espaciado y el tamaño del texto en función del contexto de uso.

El archivo original styles.ts para el componente Button incluye una configuración detallada de variantes de estilo, combinando propiedades como variant, themeColor y size. Dado que el archivo completo es extenso, a continuación veremos en el Código 1 como se presenta una versión simplificada que ilustra los principios de configuración sin mostrar todas las combinaciones posibles.

Código 1: Configuración de variantes y estilos personalizados en componentes con class-variance-authority (CVA).

Esta configuración utiliza CVA para definir variantes modulares y aplicarlas dinámicamente según las propiedades del componente. CVA facilita el mantenimiento de estilos al permitir que cada combinación de propiedades aplique estilos específicos, manteniendo el código organizado y fácilmente extensible.

En el código original, cada combinación posible de themeColor, variant, y size se gestiona con compound variants en CVA, lo cual permite aplicar clases CSS adicionales según las propiedades específicas del componente. Esta versión simplificada ilustra cómo CVA estructura y aplica variantes sin la complejidad completa del archivo real.

Implementación de la lógica del componente

En el archivo button.tsx, el componente Button utiliza las configuraciones de variantes definidas en styles.ts y aprovecha la función cn para gestionar las clases dinámicamente en función de sus propiedades. A continuación se presentan las partes clave de la implementación.

1. Asignación de variantes de estilo

Para aplicar las variantes de estilo, el componente usa la función cn para combinar las clases CSS generadas por **CVA** (desde styles.ts) con clases adicionales según el estado del componente, como podemos ver en el Código 2.

```
const baseIconVariantsClasses = styles.icon({
  themeColor,
  variant,
  size,
});

const iconVariantsClasses = cn(
  baseIconVariantsClasses,
  "group-data-[loading=true]:text-[transparent]",
);
```

Código 2: Combinación de clases CSS con cn para asignar variantes de estilo según el estado del componente.

- **Explicación**: baseIconVariantsClasses utiliza styles.icon de CVA para aplicar clases según themeColor, variant, y size.
- **Función cn**: combina baseIconVariantsClasses con una clase adicional para ocultar el texto del botón (text-[transparent]) cuando loading es true, manteniendo los íconos o spinner visibles.

2. Estados interactivos (loading y disabled)

El componente Button maneja propiedades como loading y disabled para ajustar su presentación y funcionalidad. Si loading es true, se muestra un ícono de carga, y el botón se desactiva, como se observa en el Código 3.

```
const loadingSpinner = loading && (
  <LoaderIcon
    className={cn(baseIconVariantsClasses, "absolute", "animate-spin")}
  />
);

const children = asChild ? (…
) : (
  <>…
  </>
);

return (
  <Comp
    {...props}
    ref={ref}
    data-loading={loading}
    className={cn(
      styles.rootBase({ themeColor, variant }),
      styles.rootSize({ size }),
      "inline-flex",
      "items-center",
      "justify-center",
      className,
    )}
    disabled={loading || disabled}
  >
    {children}
  </Comp>      esosa-tn, 15 months ago • IDMU-151: split button file (#63) …
);
```

Código 3: Implementación de estados interactivos (loading y disabled) en el componente Button.

- **Ícono de carga (loadingSpinner)**: si loading es true, se renderiza un LoaderIcon con una clase animate-spin para indicar que el botón está procesando.
- **Propiedad disabled**: el botón se desactiva si loading o disabled es true, evitando cualquier interacción.

3. Íconos personalizados (leftIcon y righIcon)

El componente acepta leftIcon y rightIcon como propiedades, lo cual permite colocar íconos a la izquierda y derecha del texto del botón, proporcionando flexibilidad en el diseño, como se observa en el Código 4.

```
const leftIcon = _leftIcon && (
  <Slot className={cn(iconVariantsClasses, "mr-2")}>{_leftIcon}</Slot>
);

const rightIcon = _rightIcon && (
  <Slot className={cn(iconVariantsClasses, "ml-2")}>{_rightIcon}</Slot>
);

const loadingSpinner = loading && (…
);

const children = asChild ? (…
) : (
  <>…
  </>
);

return (
  <Comp
    {...props}
    ref={ref}
    data-loading={loading}
    className={cn(
      styles.rootBase({ themeColor, variant }),
      styles.rootSize({ size }),
      "inline-flex",
      "items-center",
      "justify-center",
      className,
    )}
    disabled={loading || disabled}
  >
    {children}
  </Comp>      esosa-tn, 15 months ago • IDMU-151: split button file (#63) …
);
```

Código 4: Gestión de íconos en el componente Button, utilizando leftIcon y rightIcon.

- **Uso de Slot:** Radix UI permite que leftIcon y rightIcon se comporten como elementos secundarios (Slot) que se alinean con el texto y se espacian con mr-2 o ml-2 (clases de Tailwind para definir márgenes laterales).

Ensamblaje final de children

En los ejemplos de código podemos observar un elemento que se llama children, en el render final, children como vemos en el Código 5, integra el ícono de carga (loadingSpinner), el ícono izquierdo (leftIcon), y el ícono derecho (rightIcon). Si loading es true, se oculta el texto del botón temporalmente.

```
const children = asChild ? (
  React.isValidElement(_children) &&
  React.cloneElement(_children, {
    children: (
      <>
        {loadingSpinner}
        {leftIcon}
        {!loadingSpinner ? _children.props.children : null}
        {rightIcon}
      </>
    ),
  } as { children: React.ReactNode })
) : (
  <>
    {loadingSpinner}
    {leftIcon}
    {!loadingSpinner ? _children : null}
    {rightIcon}
  </>      esosa-tn, 15 months ago • IDMU-151: split button file (#63) …
);
```

Código 5: Ensamblaje de children con loadingSpinner, leftIcon y rightIcon.

- **Manejo de children:** si loading es true, el texto se reemplaza con el loadingSpinner. Los íconos izquierdo y derecho se integran según la configuración de leftIcon y rightIcon, permitiendo flexibilidad en el contenido del botón.

Ejemplo de uso del componente Button

El componente Button se puede personalizar ampliamente mediante sus propiedades variant, themeColor, size, loading, leftIcon, y rightIcon. Aquí algunos ejemplos que muestran cómo usar estas opciones:

1. En la Figura 8 podemos observar un ejemplo del código para agregar un componente botón (A) primario con estilo relleno y un ícono a la izquierda (B):

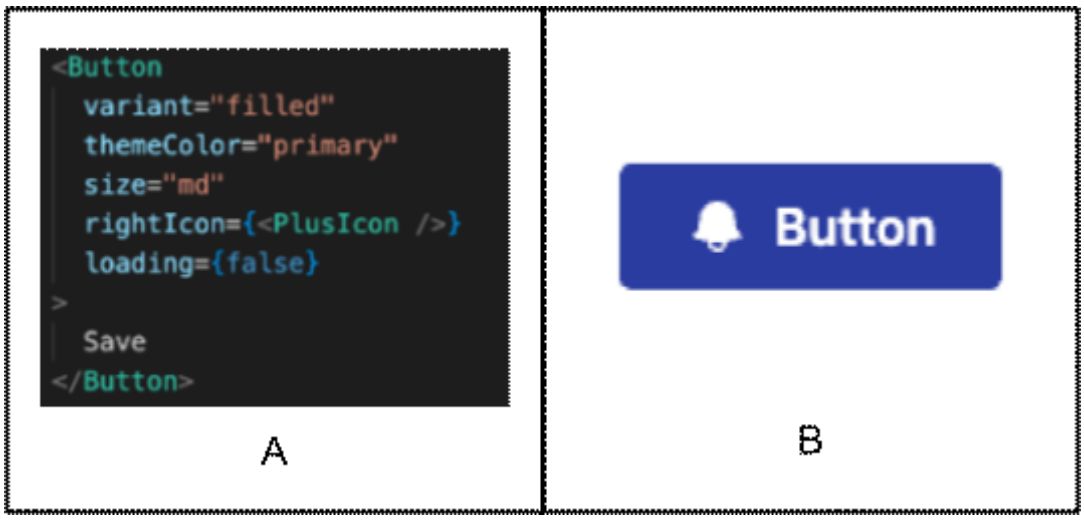

Figura 8: Ejemplo de un botón primario con un ícono a la izquierda.

2. En la Figura 9 podemos observar un ejemplo del código para agregar un componente botón (A) en este caso secundario con estilo outlined y un ícono de campana a la izquierda (B):

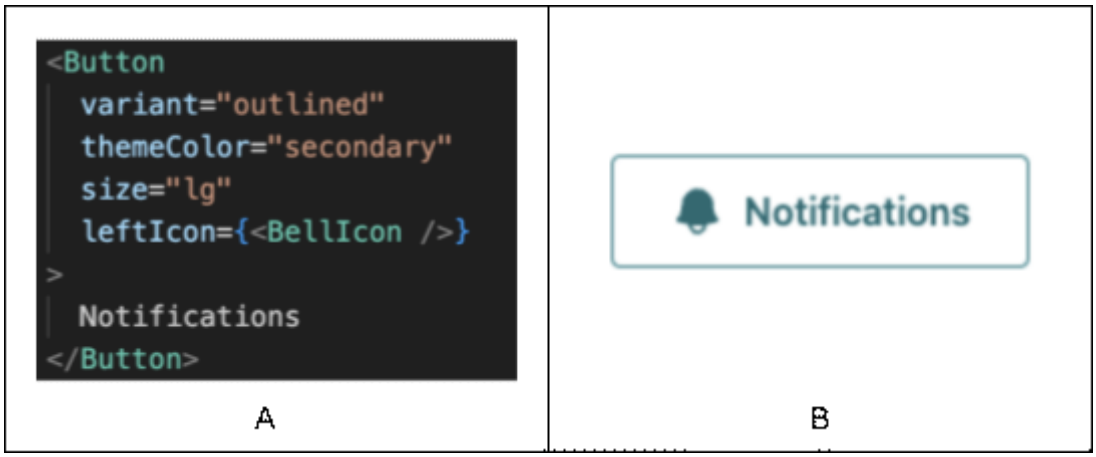

Figura 9: Ejemplo de un botón secundario con un ícono a la izquierda.

3. En la Figura 10 podemos observar un ejemplo de código para agregar un componente botón (A) en este caso destructivo en estilo ghost (B), ideal para acciones como eliminar:

Figura 10: Ejemplo de un botón destructivo.

4. En la Figura 11 podemos observar un ejemplo de código para agregar un componente botón (A) en este caso como enlace, usando asChild para redirigir a una URL externa:

Figura 11: Ejemplo de un botón primario con un link dentro.

4.2 Elección de React y herramientas utilizadas

La elección de React como base tecnológica para la creación de la biblioteca de componentes se fundamenta en varios factores clave que la han convertido en una de las tecnologías más populares para el desarrollo de interfaces de usuario (UI). Como se mencionó en el capítulo 3, React es una biblioteca de JavaScript ampliamente utilizada en el desarrollo web moderno debido a su flexibilidad, eficiencia y la vasta comunidad que respalda su uso. En este apartado, se detallan las razones por las que React fue seleccionado y se describen las herramientas que acompañaron su implementación.

4.2.1 Ventajas de React

Uno de los principales motivos para elegir React es su enfoque basado en componentes, lo cual es altamente beneficioso para la construcción de interfaces modulares y reutilizables (Frost, 2016). React permite dividir la UI en pequeños componentes que pueden ser reutilizados a lo largo de múltiples proyectos, reduciendo significativamente el tiempo de desarrollo y mantenimiento.

Adicionalmente, React emplea un Virtual DOM, lo que mejora el rendimiento de las aplicaciones al minimizar las operaciones costosas en el DOM real (Facebook, 2023). Esto resulta en una experiencia de usuario más fluida, especialmente en aplicaciones complejas o con un alto grado de interacción.

Otra ventaja es su ecosistema maduro y bien soportado. React tiene una gran comunidad y una vasta cantidad de bibliotecas y herramientas que facilitan la integración de funcionalidades avanzadas, como la gestión de estado o la navegación entre vistas (Garrett, 2020). Esto hace que la adopción de React sea una decisión estratégica a largo plazo, ya que garantiza soporte continuo y una evolución constante de la tecnología.

4.2.2 Herramientas utilizadas en el desarrollo

En el desarrollo de esta biblioteca de componentes, React no fue la única herramienta clave. A continuación, se detallan las principales tecnologías y herramientas empleadas en el proceso de desarrollo:

1. **TypeScript**: se eligió TypeScript para agregar tipado estático a JavaScript, lo que contribuye a detectar errores durante la fase de desarrollo y mejorar la calidad del código. TypeScript asegura que los componentes sean más fáciles de mantener y extender, reduciendo la posibilidad de errores imprevistos en proyectos a gran escala (Microsoft, 2023).
2. **Storybook**: para la documentación y visualización de los componentes, se utilizó Storybook. Esta herramienta permite desarrollar y probar los componentes de manera aislada, lo que facilita la revisión y prueba de distintas variantes de los componentes (Storybook, s.f.). Storybook también es un recurso valioso para los equipos de diseño y desarrollo, ya que proporciona una forma clara y visual de cómo funcionan y se integran los componentes en las aplicaciones.
3. **Tailwind CSS**: para agilizar el diseño de los componentes, se implementó Tailwind CSS, un framework de utilidades que permite estilizar los componentes de manera rápida y efectiva (Tailwind, 2017). Tailwind ayuda a mantener la consistencia visual

en toda la interfaz sin necesidad de escribir CSS personalizado desde cero. Además, se utilizó una función de utilidades de clase (cn) para gestionar dinámicamente las clases CSS según las variantes y el estado del componente, facilitando la configuración y personalización de estilos en función de distintas condiciones de diseño. Este enfoque permite aplicar estilos condicionales de manera eficiente, garantizando flexibilidad y adaptabilidad en la presentación visual de los componentes.

4. **Next.js**: como framework de desarrollo basado en React, Next.js fue utilizado para estructurar la aplicación web de manera eficiente, especialmente en términos de renderizado del lado del servidor (SSR) y generación de sitios estáticos (SSG). Esto mejora el rendimiento de las aplicaciones y optimiza su posicionamiento en motores de búsqueda (Vercel, 2016).
5. **Class Variance Authority (CVA)**: para gestionar variantes de estilo y configuraciones de diseño en los componentes, se utilizó CVA. Esta herramienta permite mantener la coherencia visual y simplificar la personalización de estilos en base a múltiples variantes, facilitando el diseño modular en el componente Button (CVA Docs, s.f.).
6. **Radix UI Primitive**: proporcionó componentes base que fueron esenciales para la estructura de la biblioteca, ofreciendo soluciones accesibles y modulares desde el inicio. Gracias a Radix, los desarrolladores pudimos emplear componentes preconfigurados para temas como acordeones, diálogos, y tooltips, que luego fueron personalizados para adaptarse a los requisitos visuales y funcionales del proyecto. Esta elección redujo significativamente el esfuerzo en diseño y desarrollo, permitiendo que el equipo se centrara en la personalización y extensión de componentes específicos (Radix UI, s.f.).
7. **Git y GitHub**: para el control de versiones y la colaboración en equipo, se utilizó Git junto con GitHub. Estas herramientas facilitaron el seguimiento de cambios en el código, permitiendo una colaboración eficiente y organizada entre los miembros del equipo de desarrollo. GitHub, en particular, fue fundamental para gestionar *pull requests*, realizar revisiones de código y mantener un historial detallado del progreso del proyecto. Gracias a este sistema, se garantizó la integridad del código y la facilidad para revertir o comparar versiones anteriores cuando fuera necesario (GitHub, s.f.).
8. **Despliegue con Vercel**: el despliegue de la biblioteca de componentes se realizó en Vercel, una plataforma optimizada para aplicaciones creadas con Next.js y React. Vercel no solo permitió desplegar el proyecto de manera rápida y sencilla, sino que también facilitó la actualización continua mediante integraciones automáticas con

GitHub. Cada cambio en el repositorio desencadenaba un despliegue en Vercel, permitiendo así que las versiones más recientes de la biblioteca estuvieran siempre accesibles para pruebas y revisiones (Vercel, s.f.).

9. **Figma**: en el proceso colaborativo con el equipo de diseño, Figma fue utilizado como la herramienta principal para la creación y revisión de prototipos de interfaz. Figma facilita la colaboración en tiempo real entre desarrolladores y diseñadores, asegurando que los componentes implementados en React se ajusten fielmente a los diseños propuestos (Figma, 2016).

4.2.3 Integración y beneficios

La integración de React con estas herramientas ha permitido un flujo de trabajo eficiente y colaborativo. Gracias a la versatilidad de React y al soporte que ofrecen herramientas como TypeScript y Storybook, se ha logrado crear una biblioteca de componentes robusta, reutilizable y fácil de mantener.

Esta selección tecnológica no solo ha optimizado el desarrollo interno, sino que también ha promovido la colaboración efectiva entre los distintos equipos de trabajo, mejorando así la calidad final del producto.

4.3 Proceso colaborativo y ejemplos prácticos

El desarrollo de la biblioteca de componentes fue un esfuerzo conjunto que involucró a diferentes equipos dentro de la empresa, tales como los equipos de desarrollo, diseño y gestión de producto. La colaboración efectiva y el uso de metodologías ágiles fueron esenciales para garantizar la entrega exitosa de un producto funcional y de alta calidad. Esta sección detalla el proceso colaborativo, las herramientas utilizadas para la organización del equipo y ejemplos prácticos de cómo la biblioteca de componentes podría ser implementada en diferentes proyectos.

4.3.1 Gestión del proyecto y metodología de trabajo

Para organizar y gestionar el trabajo del equipo, se optó por utilizar la metodología **Scrum**, un marco ágil que fomenta la colaboración y la entrega iterativa de funcionalidades. Scrum fue clave para estructurar el trabajo en **sprints** de dos semanas, lo que permitió al equipo enfocarse en la entrega incremental de componentes de la biblioteca.

Las ceremonias de Scrum jugaron un rol fundamental en el proceso de desarrollo:

- **Daily standup**: estas reuniones diarias permitieron al equipo compartir su progreso, identificar obstáculos y ajustar las prioridades según fuera necesario. La transparencia en la comunicación fue vital para detectar problemas temprano y resolverlos de manera ágil (Schwaber & Sutherland, 2017).
- **Sprint planning**: en la planificación de cada sprint, el equipo determinaba qué componentes o funcionalidades serían desarrollados durante las próximas dos semanas. Se utilizaban **user stories** para definir las tareas, que se priorizaban en función de las necesidades del proyecto.
- **Sprint review**: al finalizar cada sprint, el equipo presentaba los componentes desarrollados para recibir feedback tanto de los stakeholders como de los usuarios potenciales. Estas revisiones fueron cruciales para ajustar las funcionalidades de los componentes según las expectativas del cliente y del equipo de diseño.
- **Sprint retrospective**: en esta ceremonia, que se realizaba al final de cada sprint, el equipo reflexionaba sobre lo que había funcionado bien durante el sprint y las áreas en las que se podía mejorar. Esto fomentó un ciclo de mejora continua, donde cada sprint aprendíamos nuevas lecciones para aplicar en el siguiente.

Para gestionar las tareas y el progreso del proyecto, se utilizó **Jira**, una herramienta de gestión de proyectos ampliamente utilizada en entornos ágiles. Jira nos permitió organizar las tareas mediante user stories, hacer seguimiento de su progreso y visualizar el estado del sprint a través de un tablero Kanban. La herramienta también facilitó la asignación de tareas a los desarrolladores y la identificación rápida de bloqueos o problemas (Atlassian, s.f.).

El uso de **Scrum** y **Jira** no solo ayudó a organizar el trabajo del equipo, sino que también permitió una mayor visibilidad y control sobre el progreso del proyecto. Estos procesos permitieron entregar incrementos de valor en cada sprint, manteniendo a los stakeholders informados y asegurando una alineación constante entre los equipos de desarrollo y diseño.

4.3.2 Proceso colaborativo entre equipos

La creación de la biblioteca de componentes también requirió una estrecha colaboración con el equipo de **diseño UX/UI**. Se utilizó **Figma** como la herramienta principal para la creación de prototipos y la comunicación visual. Figma permitió a los diseñadores compartir de manera colaborativa los diseños de los componentes, mientras que los desarrolladores los implementaban en React, asegurando que el resultado final se ajustara a las especificaciones originales.

Este proceso colaborativo fue clave para garantizar la coherencia visual y funcional de los componentes. Los diseñadores trabajaban en los elementos visuales, mientras que los desarrolladores implementaban esos diseños en código. A través de herramientas como **Storybook**, los desarrolladores podían mostrar componentes aislados a los diseñadores, quienes revisaban su implementación y proporcionaban feedback en tiempo real. Este flujo de trabajo garantizó que los componentes fueran desarrollados siguiendo los estándares de diseño definidos, promoviendo la consistencia en toda la biblioteca.

El uso de Storybook también permitió realizar pruebas aisladas de los componentes, permitiendo tanto a diseñadores como a desarrolladores visualizar las variaciones de cada componente (por ejemplo, botones con distintos estados: activo, inactivo, deshabilitado), como se muestra en la Figura 12 donde el Storybook se levantó en un ambiente local para poder obtener la captura. Esto redujo el riesgo de errores visuales y mejoró la comunicación entre ambos equipos.

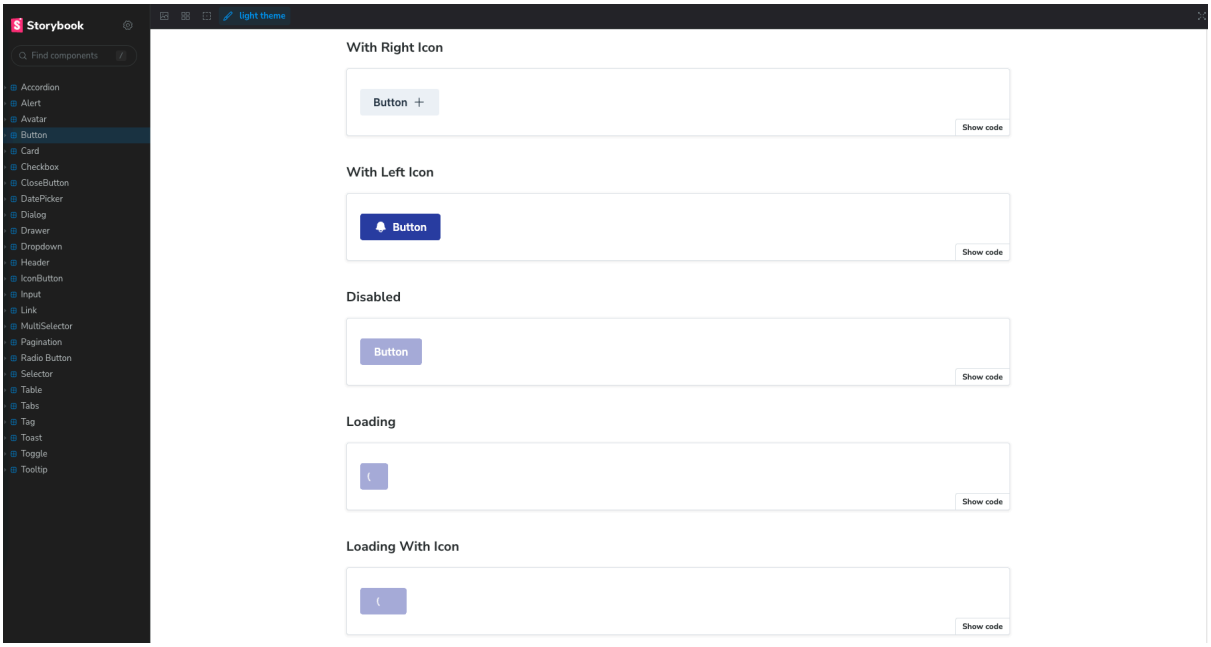

Figura 12: Captura botones de ejemplo del Storybook del proyecto.

4.4 Participación de la tesista en el equipo de desarrollo

En el desarrollo de la biblioteca de componentes, mi rol como desarrolladora frontend se centró en contribuir tanto en la implementación de componentes específicos como en la integración de estos en el sistema de diseño general de la empresa. Dentro del equipo, participé activamente en la fase de diseño técnico de los componentes, colaborando con

diseñadores para asegurar la coherencia visual y funcional, y en la codificación de varios de los elementos más utilizados en la interfaz.

Componentes desarrollados

A lo largo del proyecto, fui responsable de la creación y optimización de componentes clave para la librería. Algunos de estos componentes incluyen:

- **Botones personalizados**: desarrollé un conjunto de botones configurables, incluyendo variaciones en color, tamaño y estado (activo, inactivo, cargando). Este componente se diseñó con el objetivo de adaptarse a diferentes contextos de la interfaz, permitiendo una fácil personalización visual y funcional.
- **Inputs de datos validados**: implementé campos de entrada que incluyen validaciones automáticas y tooltips de ayuda. Estos inputs mejoran la experiencia del usuario al guiarlo sobre el formato y tipo de datos requeridos, lo cual también contribuye a la accesibilidad del sistema.
- **Alertas y notificaciones**: diseñé y programé el componente de alertas, que permite mostrar mensajes informativos, de error y de éxito en distintas partes de la interfaz. Este componente puede personalizarse para adaptarse al tono y color del mensaje, asegurando que el usuario reciba la información de manera clara y efectiva.

Coordinación con el equipo

Además del desarrollo de componentes específicos, participé en la coordinación de tareas mediante Jira, donde realicé el seguimiento de las tareas asignadas y actualicé el estado de los tickets correspondientes a mis componentes. Esta organización permitió una integración ágil con el trabajo de otros miembros del equipo y aseguró que los componentes se entregaran según los plazos y los estándares establecidos en el diseño de la librería.

Mi participación en este proyecto no solo incluyó la creación técnica de componentes, sino también la revisión y prueba de los mismos en **Storybook**, herramienta que facilitó la colaboración con los diseñadores y permitió realizar ajustes visuales y funcionales en tiempo real. Esta experiencia de trabajo colaborativo resultó esencial para la cohesión y calidad final de la librería de componentes.

4.5 Resumen del capítulo

En este capítulo se ha descrito de manera detallada el proceso de desarrollo de una biblioteca de componentes, cuyo objetivo principal fue optimizar el flujo de trabajo en el desarrollo frontend dentro de la empresa Truenorth. La creación de esta biblioteca ayudó a resolver varios desafíos relacionados con la coherencia visual y la eficiencia en la implementación de interfaces de usuario en proyectos web.

Se explicó cómo el diseño modular de los componentes facilitó la reutilización, mejorando la mantenibilidad y escalabilidad de los proyectos. Este enfoque redujo considerablemente el tiempo de desarrollo en comparación con la creación de componentes individuales desde cero. Además, se establecieron pautas claras para garantizar la consistencia en los futuros proyectos, basadas en un sistema sólido de estándares de diseño.

La elección de React como tecnología base fue clave, debido a sus capacidades para construir interfaces dinámicas, respaldadas por herramientas complementarias como TypeScript, que proporciona mayor seguridad en el código gracias al tipado estático. Storybook también jugó un papel importante, permitiendo la visualización y prueba aislada de los componentes, facilitando tanto la colaboración como la revisión continua por parte de los equipos de diseño y desarrollo.

En términos de organización del trabajo, se adoptó la metodología Scrum, que facilitó una entrega ágil y constante de valor a través de sprints regulares. Las herramientas como Jira y Figma permitieron una gestión clara de las tareas y la alineación visual de los componentes, asegurando una comunicación fluida entre los equipos de desarrollo y diseño. Esta colaboración interdisciplinaria garantizó que los componentes fueran tanto técnicamente eficientes como estéticamente consistentes.

Finalmente, como desarrolladora frontend en el equipo, mi participación incluyó el diseño y desarrollo de componentes clave como botones personalizados, campos de entrada con validaciones, y alertas informativas, los cuales se integraron en la biblioteca y pueden adaptarse a distintas aplicaciones. Mi rol también abarcó la coordinación de tareas y la prueba de componentes en Storybook, asegurando que cada elemento cumpliera con los estándares de calidad y funcionalidad establecidos. Esta experiencia colaborativa no solo contribuyó al éxito del proyecto, sino que también reforzó el compromiso con la accesibilidad, la personalización y la usabilidad en el desarrollo frontend.

Capítulo 5. Evaluación de la biblioteca de componentes

5.1 Introducción

En esta sección se presenta una evaluación de la biblioteca de componentes desarrollada en términos de su flexibilidad, facilidad de uso y efectividad para su reutilización en contextos de alto impacto. Para ello, se analizó un ejemplo concreto de su aplicación en una interfaz de usuario real, específicamente un panel de control (dashboard). Este caso de uso permitió observar cómo interactúan los componentes entre sí y cómo se integran en una interfaz de usuario coherente, destacando su adaptabilidad y rendimiento en un entorno de trabajo intensivo. Cabe aclarar que el dashboard fue seleccionado como único caso de prueba para ilustrar la implementación y funcionalidad de la biblioteca en un contexto práctico.

5.2 Ejemplo práctico de componentes en acción

5.2.1 Dashboard

En la Figura 13, el dashboard mostrado a modo de ejemplo simula una interfaz de usuario para una aplicación financiera, diseñada para que el usuario pueda gestionar su cuenta bancaria, revisar su balance, ver su flujo de efectivo y acceder a opciones de transacciones rápidas. Este ejemplo permite observar cómo los componentes de la biblioteca se integran para crear una experiencia de usuario cohesiva y funcional, optimizando la presentación de información y facilitando la interacción. Cabe mencionar que este dashboard fue desplegado en un ambiente local, por lo que no se dispone de una URL pública para su acceso.

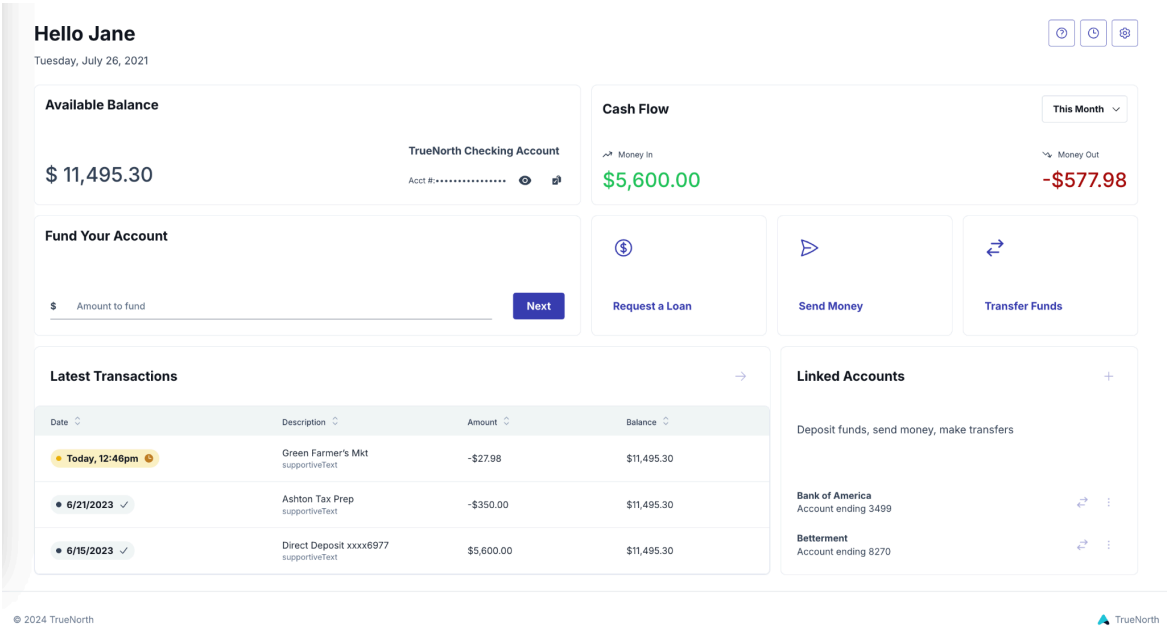

Figura 13: Ejemplo de interfaz de usuario mostrando organización de información financiera y accesos rápidos.

En la Figura 14 podemos observar cómo funciona de manera responsiva para una resolución de 1024 x 1366.

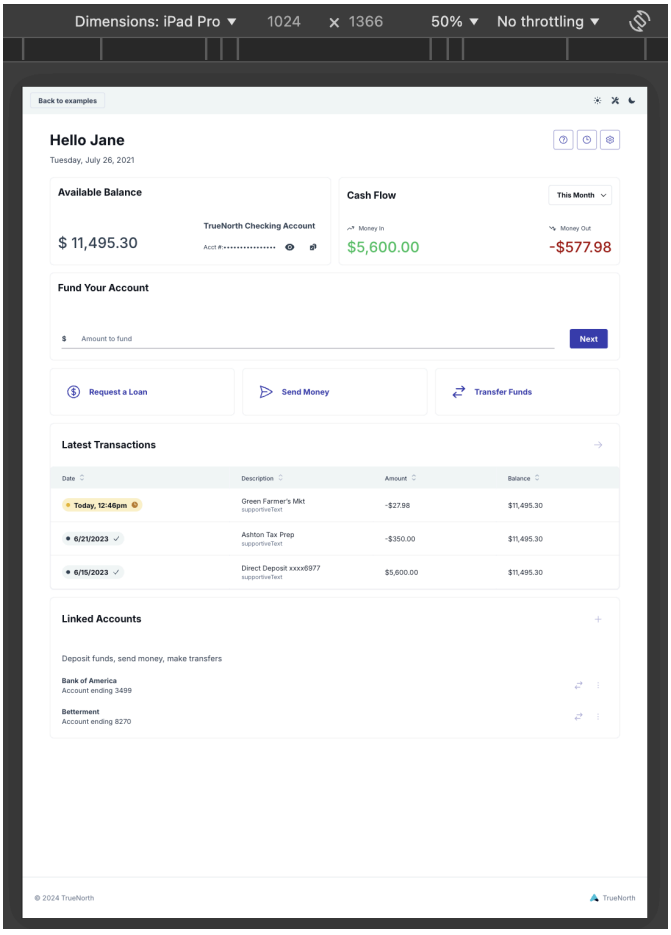

Figura 14: Dashboard resolución 1024 x 1366.

En la Figura 15 podemos observar cómo funciona de manera responsiva para una resolución de 430 x 932, en este caso deberíamos hacer scroll hacia abajo para poder terminar de ver todo el dashboard.

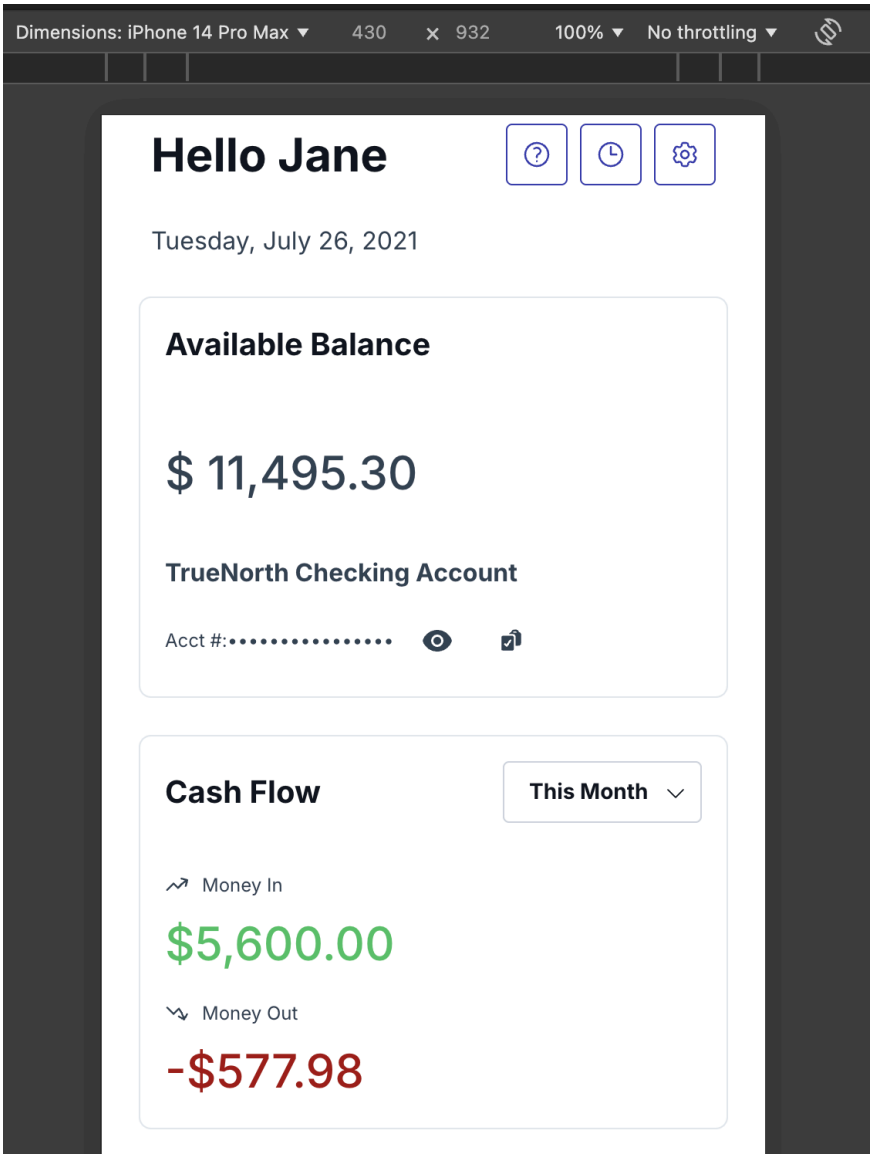

Figura 15: Dashboard resolución 430 x 932.

A continuación se describe cómo se integran los diferentes componentes:

1. Encabezado de bienvenida y fecha

- **Componentes involucrados**: CardHeader, CardTitle, IconButton.
- **Función**: como se muestra en la Figura 16 el encabezado muestra un saludo personalizado al usuario, "Hello Jane", junto con la fecha actual y tres íconos para acceder a ayuda, historial y configuración, este desarrollo lo podemos ver en el Código 6.

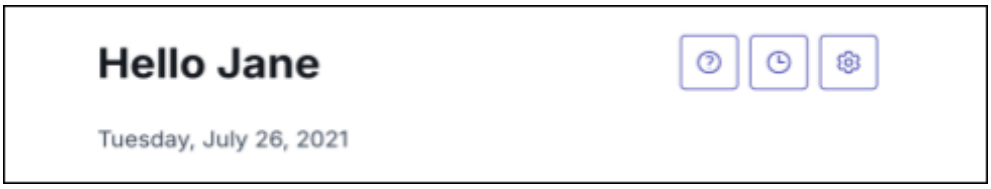

Figura 16: Dashboard - Encabezado de bienvenida y fecha.

```
<CardHeader className="md:flex-wrap-0 flex-wrap p-8 pb-0 md:px-14 md:pt-8">
  <CardTitle className="text-3xl">Hello Jane</CardTitle>
  <div className="my-4 space-x-2 md:my-0">
    <IconButton
      aria-label="Questions?"
      themeColor="primary"
      variant="outlined"
      icon={<QuestionMarkCircleIcon />}
    />
    <IconButton
      aria-label="Questions?"
      themeColor="primary"
      variant="outlined"
      icon={<ClockIcon />}
    />
    <IconButton
      aria-label="Questions?"
      themeColor="primary"
      variant="outlined"
      icon={<Cog6ToothIcon />}
    />
  </div>
</CardHeader>
```

Código 6: Código del encabezado de bienvenida y fecha.

2. Tarjetas de información (Available Balance y Cash Flow)

- **Componentes involucrados**: Card, CardHeader, CardTitle, CardContent, Tooltip, IconButton.
- **Función**: en las tarjetas que vemos en la Figura 17 se muestran el balance disponible y el flujo de efectivo (entradas y salidas de dinero). El balance incluye una opción para ocultar (A) o mostrar (B) el número de cuenta y copiarlo, una vez copiado podemos observar también cómo se muestra el tooltip (C), este desarrollo lo podemos ver en el Código 7.

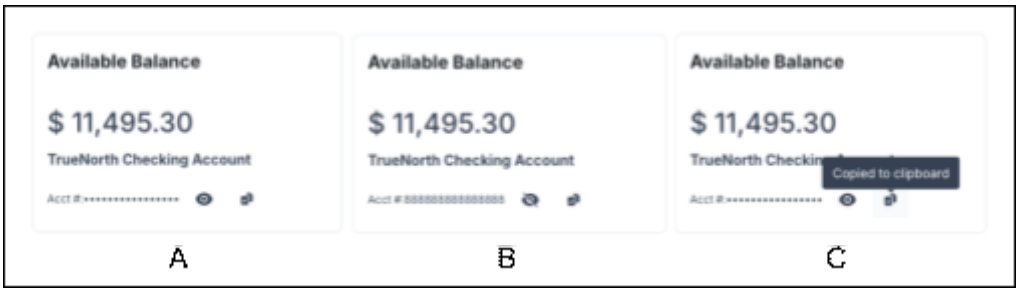

Figura 17: Dashboard - Tarjetas de información.

```tsx
<Card
  cardStyle="outline"
  className="min-h-[180px] basis-full md:basis-1/2"
>
  <CardHeader className="p-4">
    <CardTitle>Available Balance</CardTitle>
  </CardHeader>
  <CardContent className="mt-4 flex flex-col flex-wrap justify-between p-4 md:mt-0 md:flex-row md:items-end">
    <h1 className="mb-2 text-3xl font-medium">$ 11,495.30</h1>
    <div className="flex flex-col items-start space-y-3">
      <h3 className="mt-6 font-semibold md:mt-0">
        TrueNorth Checking Account
      </h3>
      <div className="flex items-center justify-center">
        <span className="text-xs">
          Acct #:
          {showPassword ? "888888888888888" : "••••••••••••••••"}
        </span>
        <Tooltip
          content={!showPassword ? "Show" : "Hide"}
          open={tooltipOpen}
          defaultOpen={false}
          onOpenChange={handleTooltipOpenChange}
          theme="dark"
          placement="top"
          align="center"
        >
          <IconButton
            className="ml-2"
            variant="ghost"
            themeColor="neutral"
            aria-label="show-hide"
            onClick={() => setShowPassword(!showPassword)}
            icon={
              !showPassword ? (
                <EyeIcon className="h-4 w-4" />
              ) : (
                <EyeSlashIcon className="h-4 w-4" />
              )
            }
          />
        </Tooltip>
        <IconButton
          className="ml-2 font-normal"
          variant="ghost"
          themeColor="neutral"
          aria-label="show-hide"
          icon={
            <CopyToClipboard
              tooltipText="copy"
              contentToCopy="888888888888888"
            >
              <ClipboardDocumentCheckIcon className="h-4 w-4" />
            </CopyToClipboard>
          }
        />
      </div>
    </div>
  </CardContent>
</Card>
```

Código 7: Código de las tarjetas de información.

3. Formulario de entrada (Fund Your Account)

- **Componentes involucrados:** FormControl, FormLabel, Input, InputGroup, InputLeftElement, Button.
- **Función:** este formulario que vemos en la Figura 18, permite al usuario ingresar una cantidad para agregar fondos a su cuenta. En el Código 8 podemos ver su desarrollo.

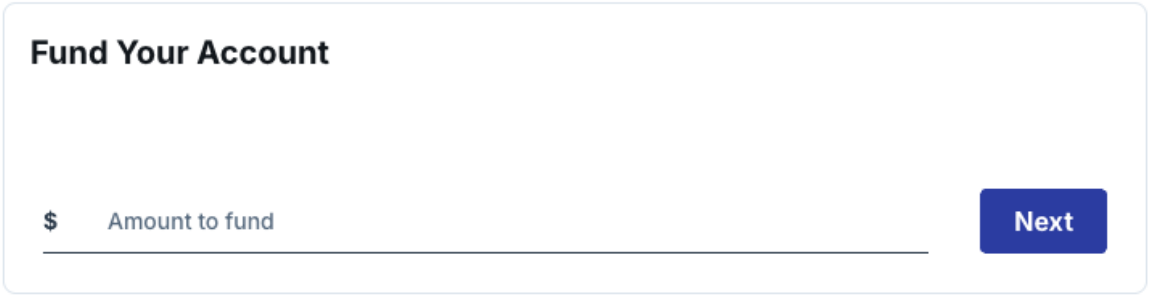

Figura 18: Dashboard - Formulario de entrada.

```
<Card
  cardStyle="outline"
  className="min-h-[180px] basis-full md:basis-1/2"
>
  <CardHeader className="p-4">
    <CardTitle>Fund Your Account</CardTitle>
  </CardHeader>
  <CardContent className="flex flex-col space-y-4 md:flex-row md:items-end md:space-y-0">
    <FormControl inputStyle="underlined" className="relative w-full">
      <FormLabel>Amount to fund</FormLabel>
      <InputGroup>
        <InputLeftElement>
          <span className="text-sm font-bold">$</span>
        </InputLeftElement>
        <Input type="number" pattern="\d*" />
      </InputGroup>
    </FormControl>
    <Button className="md:ml-8">Next</Button>
  </CardContent>
</Card>
```

Código 8: Código del formulario de entrada (Fund Your Account).

4. Acciones rápidas (Request a Loan, Send Money, Transfer Funds)

- **Componentes involucrados:** ActionsCard, IconButton.
- **Función:** en la Figura 19 podemos ver tres tarjetas con íconos permiten al usuario solicitar un préstamo, enviar dinero o transferir fondos rápidamente. En el Código 9 podemos observar como se hizo el desarrollo.

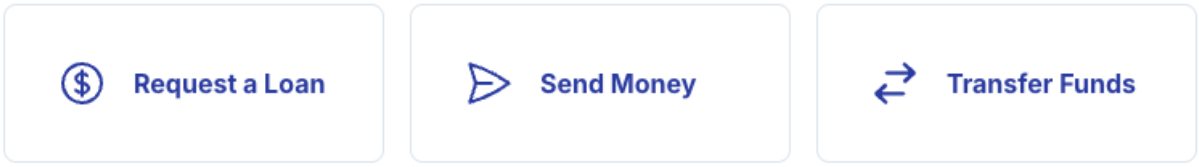

Figura 19: Dashboard - Acciones rápidas.

```
<Card
  cardStyle="outline"
  className="min-h-[180px] basis-full md:basis-1/2"
>
  <CardHeader className="p-4">
    <CardTitle>Fund Your Account</CardTitle>
  </CardHeader>
  <CardContent className="flex flex-col space-y-4 md:flex-row md:items-end md:space-y-0">
    <FormControl inputStyle="underlined" className="relative w-full">
      <FormLabel>Amount to fund</FormLabel>
      <InputGroup>
        <InputLeftElement>
          <span className="text-sm font-bold">$</span>
        </InputLeftElement>
        <Input type="number" pattern="\d*" />
      </InputGroup>
    </FormControl>
    <Button className="md:ml-8">Next</Button>
  </CardContent>
</Card>
<div className="flex basis-full flex-col md:flex-row md:space-x-4  2xl:basis-1/2 2xl:flex-nowrap">
  <ActionsCard
    title="Request a Loan"
    icon={<CurrencyDollarIcon className="h-8 w-8" />}
  />
  <ActionsCard
    title="Send Money"
    icon={<PaperAirplaneIcon className="h-8 w-8" />}
  />
  <ActionsCard
    title="Transfer Funds"
    icon={<ArrowsRightLeftIcon className="h-8 w-8" />}
  />
</div>
```

Código 9: Código de las acciones rápidas (Request a Loan, Send Money, Transfer Funds).

5. Tabla de transacciones recientes

- **Componentes involucrados:** Table, TableHeader, TableBody, TableRow, TableCell, Tag.
- **Función:** en la Figura 20 vemos esta tabla que muestra las transacciones recientes, con una fecha, descripción, monto y balance. Usa un Tag para destacar fechas específicas. En el Código 10 podemos ver su desarrollo.

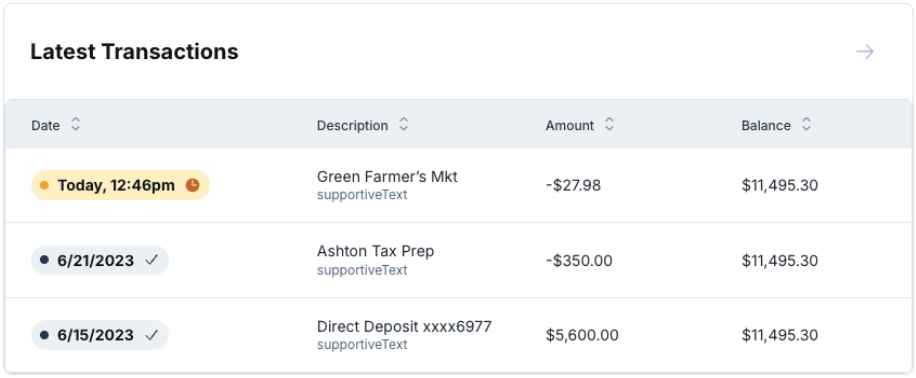

Figura 20: Dashboard - Tabla de transacciones.

```
<Table>
  <TableHeader>
    {headerGroups.map((headerGroup) => (
      <TableRow
        {...headerGroup.getHeaderGroupProps()}
        key={uuidv4()}
      >
        {headerGroup.headers.map((column) => (
          <TableHead
            {...column.getHeaderProps(
              column.sortable && column.getSortByToggleProps(),
            )}
            key={uuidv4()}
            className={column.className}
            desktopOnly={column.desktopOnly}
          >
            {column.render("Header")}
            {column.sortable && <SortColumn />}
          </TableHead>
        ))}
      </TableRow>
    ))}
  </TableHeader>

  <TableBody>
    {rows.map((row) => {
      prepareRow(row);
      return (
        <TableRow {...row.getRowProps()} key={uuidv4()}>
          {row.cells.map((cell) => (
            <TableCell
              {...cell.getCellProps({
                style: {
                  minWidth: cell.column.minWidth,
                  width: cell.column.width,
                  maxWidth: cell.column.maxWidth,
                },
              })}
              key={uuidv4()}
              desktopOnly={cell.column.desktopOnly}
            >
              {cell.render("Cell")}
            </TableCell>
          ))}
        </TableRow>
      );
    })}
  </TableBody>
</Table>
```

Código 10: Código de la tabla de transacciones recientes.

.6. Cuentas enlazadas (Linked Accounts)

- **Componentes involucrados:** Card, CardHeader, CardContent, IconButton.
- **Función:** la Figura 21 muestra las cuentas enlazadas del usuario con opciones para transferir fondos y un menú adicional. En el Código 11 y en el Código 12 (continuación del Código 11) podemos ver el desarrollo.

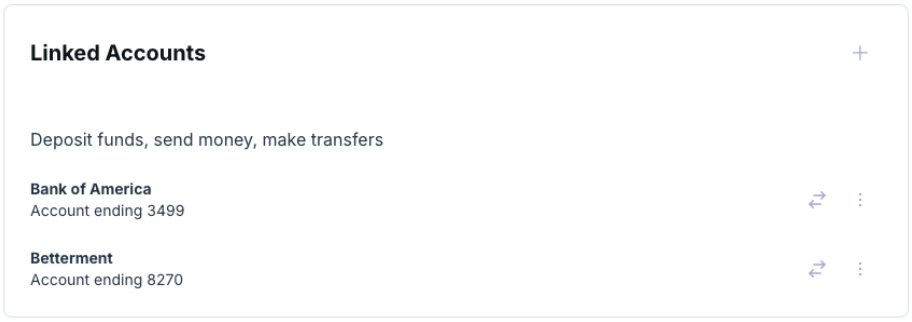

Figura 21: Dashboard - Cuentas enlazadas.

```jsx
<Card cardStyle="outline" className="flex-1">
  <CardHeader>
    <CardTitle>Linked Accounts</CardTitle>
    <IconButton
      aria-label="Questions?"
      themeColor="primary"
      variant="ghost"
      icon={<PlusIcon />}
    />
  </CardHeader>
  <CardContent className="flex flex-col justify-between">
    <h3 className="text-content-pale">
      Deposit funds, send money, make transfers
    </h3>
    <div className="mt-6">
      <div className="flex justify-between">
        <div className="text-sm">
          <h4 className="font-semibold text-content-strong">
            Bank of America
          </h4>
          <span>Account ending 3499</span>
        </div>
        <div>
          <IconButton
            aria-label="Questions?"
            themeColor="primary"
            variant="ghost"
            icon={<ArrowsRightLeftIcon />}
          />
          <IconButton
            aria-label="Questions?"
            themeColor="primary"
            variant="ghost"
            icon={<EllipsisVerticalIcon />}
          />
        </div>
      </div>
```

Código 11: Código de las cuentas enlazadas.

```
      <div className="mt-6 flex justify-between">
        <div className="text-sm">
          <h4 className="font-semibold text-content-strong">
            Betterment
          </h4>
          <span>Account ending 8270</span>
        </div>
        <div>
          <IconButton
            aria-label="Questions?"
            themeColor="primary"
            variant="ghost"
            icon={<ArrowsRightLeftIcon />}
          />
          <IconButton
            aria-label="Questions?"
            themeColor="primary"
            variant="ghost"
            icon={<EllipsisVerticalIcon />}
          />
        </div>
      </div>
    </div>
  </CardContent>
</Card>
```

Código 12: Código de las cuentas enlazadas (continuación del Código 11).

Cada sección del dashboard utiliza componentes de manera modular, mostrando cómo se puede construir una interfaz cohesiva y funcional a partir de una biblioteca de componentes. Cada tarjeta, tabla y botón está diseñado para ser reutilizable y adaptable a diferentes contenidos, mejorando la mantenibilidad y consistencia del diseño en toda la interfaz.

5.3 Validación de la biblioteca de componentes

Este apartado se centra en evaluar los criterios técnicos que se aplicaron a la biblioteca para asegurar su calidad y funcionalidad. Para la validación de aspectos como accesibilidad, rendimiento, mejores prácticas y optimización para motores de búsqueda, se utilizaron **Lighthouse** y **WAVE**.

Lighthouse es una herramienta de auditoría automatizada de Google integrada en el navegador Chrome que genera informes detallados sobre la accesibilidad, rendimiento y prácticas recomendadas de las aplicaciones web, permitiendo optimizarlas para cumplir con estándares de calidad (Google, s.f.).

Por otro lado, WAVE (Web Accessibility Evaluation Tool) es una herramienta de evaluación de accesibilidad desarrollada por WebAIM. WAVE proporciona una representación visual de los elementos de accesibilidad en una página, como etiquetas ARIA, encabezados y

contrastes de color, y permite identificar problemas que podrían afectar a usuarios con discapacidades. A diferencia de Lighthouse, que genera un reporte con puntuaciones y recomendaciones como se muestra en la Figura 22, la versión gratuita de WAVE, utilizada en este proceso, se enfoca en mostrar los elementos accesibles directamente en la interfaz como se observa en la Figura 23, facilitando la comprensión y visualización de cómo un usuario con necesidades especiales podría experimentar la página (WebAIM, 2023). Es importante mencionar que WAVE también cuenta con una versión paga que permite la generación de informes detallados.

A continuación, se explican los criterios evaluados y los resultados obtenidos en el análisis del dashboard implementado con los componentes de la biblioteca.

Accesibilidad

Para validar la accesibilidad, se utilizaron Lighthouse y WAVE en el dashboard construido con componentes de la biblioteca. Lighthouse proporcionó una puntuación de accesibilidad de 93, destacando áreas de mejora como:

- **Nombres accesibles en botones**: algunos botones no tienen nombres accesibles, lo cual dificulta la navegación para usuarios de lectores de pantalla.
- **Estructura de encabezados**: los elementos de encabezado no siguen un orden secuencial descendente, lo cual dificulta la navegación mediante teclado.

WAVE, como complemento a Lighthouse, identificó aspectos adicionales en la estructura y etiquetas ARIA, permitiendo observar visualmente elementos estructurales y atributos de accesibilidad en la interfaz, como se muestra en la Figura 16. La combinación de ambas herramientas aseguró que la biblioteca cumpliera con los estándares de accesibilidad, permitiendo la interacción inclusiva para usuarios con distintas capacidades.

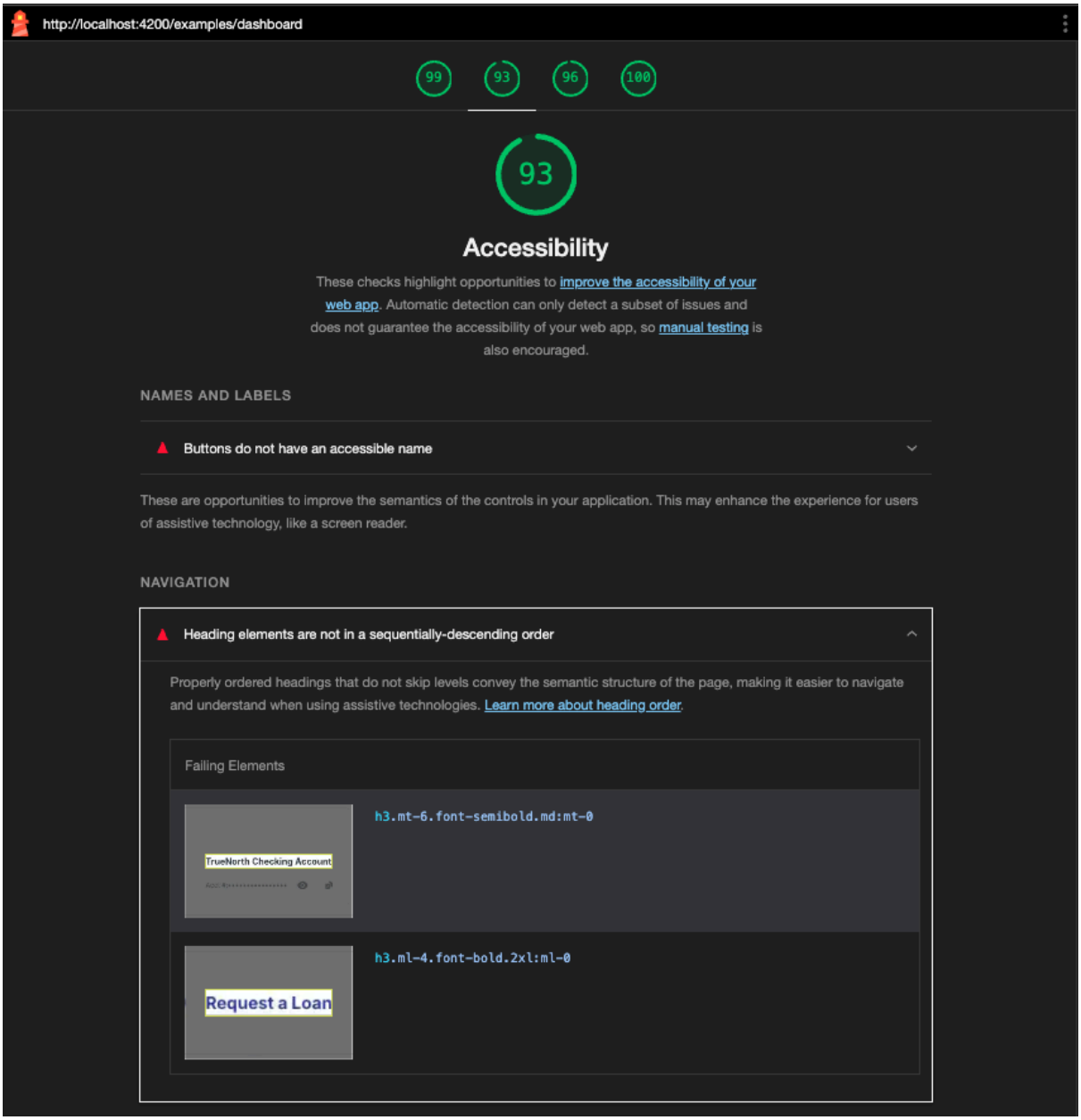

Figura 22: Puntuación de accesibilidad validada por Lighthouse.

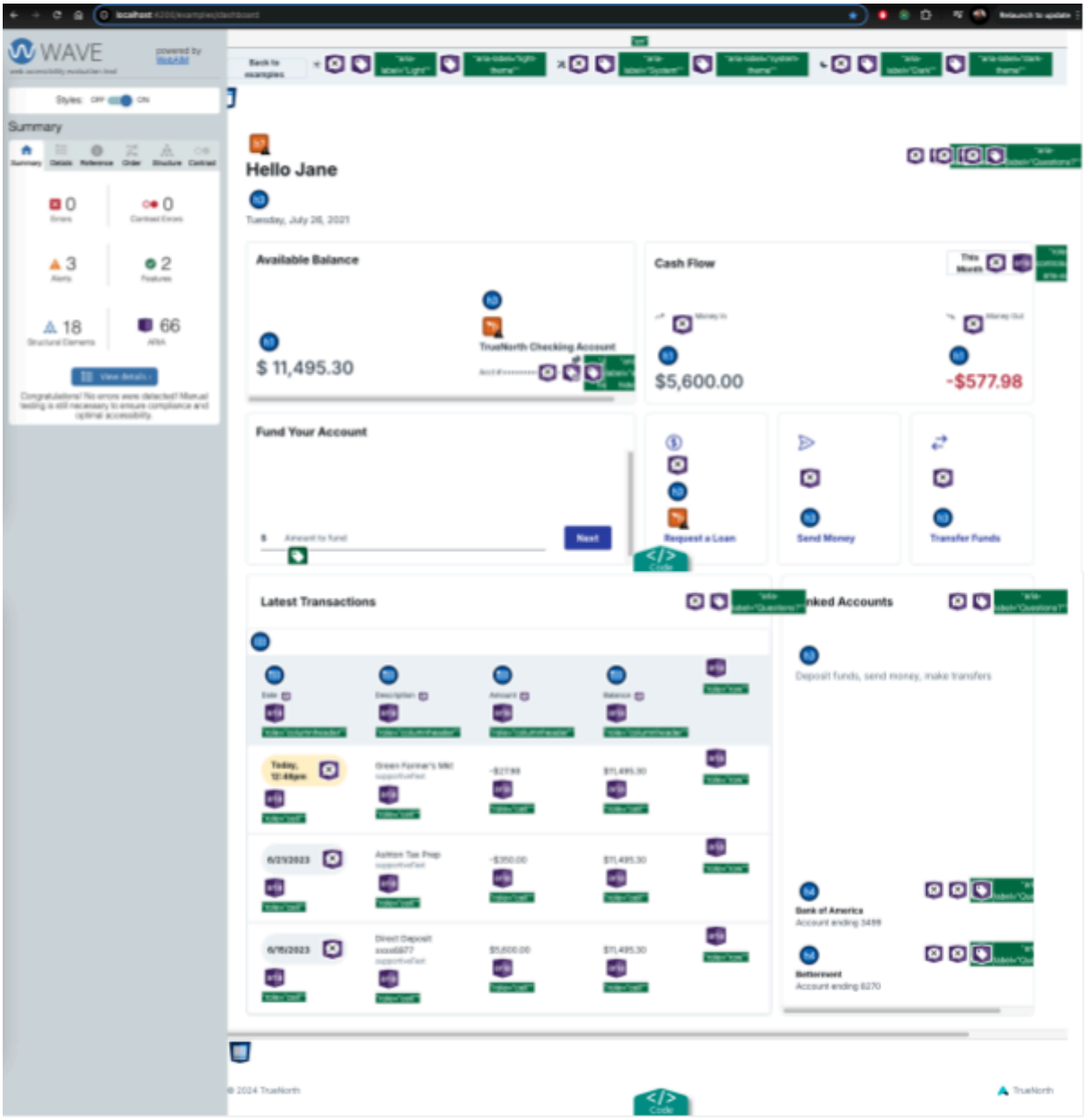

Figura 23: Evaluación de accesibilidad del dashboard usando WAVE, mostrando etiquetas ARIA y estructura de encabezados.

Otras validaciones (rendimiento, mejores prácticas y optimización para motores de búsqueda)

Además de la accesibilidad, **Lighthouse** proporcionó recomendaciones adicionales para mejorar el rendimiento, mejores prácticas y optimización para motores de búsqueda en el dashboard. Para el análisis de rendimiento, Lighthouse evaluó métricas clave que reflejan la rapidez y estabilidad visual de la página web. A continuación, se explican estas métricas para contextualizar su importancia en la experiencia del usuario:

- **First Contentful Paint (FCP)**: representa el tiempo que tarda en aparecer el primer contenido en la pantalla, como texto o imágenes. Este indicador es relevante porque ofrece una primera impresión de la rapidez de la página para los usuarios. Un valor bajo en FCP significa que el contenido principal se muestra rápidamente, mejorando la experiencia inicial del usuario (Google Developers, 2023; Lighthouse, 2023)..
- **Largest Contentful Paint (LCP)**: mide el tiempo que tarda en cargarse el contenido más grande visible en la pantalla, que suele ser una imagen destacada o un bloque de texto principal. El LCP es crucial para evaluar la rapidez con la que los elementos importantes se muestran al usuario, ya que una carga lenta del LCP puede hacer que la página parezca incompleta o lenta (Google Developers, 2023; Lighthouse, 2023)..
- **Cumulative Layout Shift (CLS)**: evalúa la estabilidad visual de la página al medir los cambios de diseño inesperados mientras se carga. Un valor bajo de CLS indica que los elementos no se mueven inesperadamente durante la carga, lo cual es importante para evitar frustraciones en la experiencia de usuario, especialmente en dispositivos móviles (Google Developers, 2023; Lighthouse, 2023).
- **Optimización para motores de búsqueda**: Lighthouse también proporciona recomendaciones para la optimización en motores de búsqueda SEO (Search Engine Optimization), que es fundamental para mejorar la visibilidad del dashboard en los resultados de búsqueda y, por ende, su accesibilidad para los usuarios. La optimización SEO incluye el uso adecuado de etiquetas meta, encabezados bien estructurados, descripciones de imágenes mediante atributos "alt" y una organización semántica del contenido que facilita la indexación por parte de los motores de búsqueda. Un dashboard con una estructura SEO optimizada tiene mayor probabilidad de ser encontrado por los usuarios en plataformas de búsqueda, contribuyendo a su eficacia y alcance (Lighthouse, 2023).

Con estos parámetros en mente, a continuación se presentan los resultados específicos del dashboard:

1. **Rendimiento**:
    - **Puntuación**: 99. el dashboard muestra un rendimiento óptimo, con métricas destacadas como:
        - **First Contentful Paint**: 0.3 s
        - **Largest Contentful Paint**: 0.4 s
        - **Cumulative Layout Shift**: 0.002
    - Lighthouse recomendó eliminar algunos recursos de bloqueo de renderizado para mejorar ligeramente la carga inicial.
2. **Mejores prácticas**:
    - **Puntuación**: 96. La única recomendación fue eliminar errores registrados en la consola del navegador y optimizar el uso de mapas fuente (source maps) en archivos JavaScript grandes.
3. **SEO**:
    - **Puntuación**: 100. El dashboard cumple completamente con las recomendaciones de SEO de Lighthouse, asegurando una estructura optimizada para motores de búsqueda.

Mejoras que se podrían realizar

Aunque los resultados de Lighthouse indican un cumplimiento significativo de los estándares, existen algunas mejoras que podrían aplicarse para optimizar aún más la accesibilidad y el rendimiento del dashboard:

- **Nombres accesibles en componentes interactivos**: asegurar que todos los botones y enlaces cuenten con etiquetas aria-label o descripciones que permitan a los lectores de pantalla interpretar adecuadamente su función.
- **Carga diferida de recursos**: optimizar el uso de recursos de bloqueo mediante la carga diferida de scripts y estilos, especialmente para los elementos que no son críticos en la vista inicial.

Estas mejoras potenciales ayudarían a asegurar que el dashboard mantenga una experiencia de usuario accesible, rápida y optimizada en diferentes contextos y dispositivos.

Reutilización

Los componentes se diseñaron para ser reutilizables en diferentes proyectos sin necesidad de modificaciones extensas. La implementación modular y el uso de *props* permiten una fácil adaptación a distintos contextos de interfaz.

Mantenibilidad

La integración de herramientas como **TypeScript** facilita la mantenibilidad a largo plazo. TypeScript permite identificar errores de tipado durante el desarrollo y proporciona documentación clara mediante la definición de interfaces y tipos.

- **Validación de mantenibilidad**: la estructura tipada de TypeScript en la biblioteca asegura que los desarrolladores puedan modificar o extender componentes sin riesgo de introducir errores en producción, lo cual ha sido verificado en pruebas internas de integración de los componentes.

Consistencia visual

Todos los componentes de la biblioteca siguen guías de estilo comunes, lo que garantiza una coherencia visual en las interfaces donde se implementan.

- **Revisión visual en Storybook**: cada componente fue revisado en Storybook para asegurar la consistencia de estilos en diferentes variantes y estados. Además, se utilizó **Lighthouse** para confirmar que las prácticas de diseño cumplen con estándares de rendimiento y optimización para motores de búsqueda.

5.4 Conclusión

La evaluación de la biblioteca de componentes muestra su eficacia en la creación de interfaces de usuario intuitivas y adaptables. El ejemplo analizado demuestra cómo los componentes desarrollados son capaces de ajustarse a diferentes contextos, garantizando flexibilidad y coherencia. Esta biblioteca no solo mejora la eficiencia del desarrollo, sino que también asegura una experiencia de usuario de alta calidad.

5.5 Resumen del capítulo

En este capítulo, se realizó una evaluación detallada de la biblioteca de componentes desarrollada, centrándose en su implementación dentro de un entorno de dashboard. Este caso de aplicación permitió examinar a fondo la funcionalidad, accesibilidad y usabilidad de los componentes en un contexto de uso real.

Los resultados de las pruebas en el dashboard demostraron que la biblioteca cumple con los requisitos técnicos fundamentales de modularidad, consistencia visual y rendimiento. El dashboard, como caso de prueba, evidenció la capacidad de los componentes para

integrarse de manera armoniosa y para soportar diversas interacciones sin comprometer la experiencia del usuario. Los elementos clave, como tarjetas de información, formularios interactivos y secciones de acciones rápidas, destacaron por su adaptabilidad y por contribuir a una experiencia de usuario fluida y coherente. Esta evaluación subraya la utilidad de la biblioteca como una herramienta esencial para el desarrollo de interfaces de usuario efectivas y escalables.

Capítulo 6: Conclusiones

6.1 Resumen del trabajo realizado

Este proyecto se enfocó en el desarrollo y evaluación de una biblioteca de componentes para interfaces de usuario, diseñada específicamente para facilitar la creación de aplicaciones web en un entorno de trabajo colaborativo y ágil. A través de la implementación de esta biblioteca en un dashboard, se validaron sus capacidades de modularidad, adaptabilidad y consistencia visual en un contexto de uso real.

Con las pruebas realizadas exclusivamente en el dashboard, se pudo llevar a cabo una evaluación profunda de los componentes en un entorno complejo y dinámico, demostrando su funcionalidad y estabilidad visual. El trabajo realizado ofrece una base sólida para futuras implementaciones en otros contextos y para continuar la optimización y expansión de la biblioteca.

6.2 Impacto de la biblioteca de componentes

La biblioteca de componentes demostró ser una herramienta clave para mejorar la eficiencia en el desarrollo de interfaces de usuario dentro de TrueNorth. La prueba en el dashboard reveló que la biblioteca facilita la creación de interfaces coherentes y optimizadas, promoviendo la reutilización de elementos y asegurando la consistencia visual en todos los componentes. Además, los resultados de esta prueba destacan la utilidad de la biblioteca en la optimización del flujo de trabajo, ya que los desarrolladores pueden confiar en componentes probados y validados en un entorno de alta interacción.

El uso exclusivo del dashboard como caso de prueba permitió confirmar la robustez y adaptabilidad de los componentes en un contexto de alto impacto, proporcionando indicadores positivos sobre su aplicabilidad futura en otras interfaces de usuario. La integración de la biblioteca en un entorno de uso real contribuyó significativamente a validar su funcionalidad, optimizando tiempos y facilitando el mantenimiento de la interfaz.

6.3 Lecciones aprendidas

El proceso de desarrollo de esta librería no estuvo exento de desafíos. No obstante, estos obstáculos resultaron en valiosas lecciones que enriquecieron tanto el producto final como el proceso de desarrollo en sí. A continuación, se describen algunas de las principales lecciones aprendidas:

- **Desafíos técnicos**: uno de los mayores retos fue garantizar que los componentes fueran suficientemente modulares y escalables. Al principio del proyecto, algunos componentes tenían dependencias entre sí, lo que complicaba su uso en distintos contextos. Para resolver este problema, se introdujeron patrones de diseño más estrictos y el uso de TypeScript ayudó a definir mejor los tipos y propiedades, evitando errores y facilitando la reutilización en múltiples proyectos.
- **Accesibilidad y usabilidad**: al implementar los componentes, se encontraron áreas en las que la accesibilidad no estaba garantizada de manera óptima. Esto llevó a una mayor reflexión sobre la importancia de incluir criterios de accesibilidad desde la fase de diseño, garantizando que todos los usuarios, independientemente de sus capacidades, pudieran interactuar fácilmente con las interfaces.
- **Comunicación interdisciplinaria**: a lo largo del proyecto, fue evidente la importancia de una comunicación fluida entre los equipos de desarrollo y diseño. Esta colaboración interdisciplinaria permitió que los componentes desarrollados no solo fueran técnicamente correctos, sino también estéticamente atractivos y funcionalmente alineados con las expectativas del usuario final. Esta interacción fue clave para el éxito del proyecto.
- **Adaptabilidad y escalabilidad**: desde el inicio del proyecto, se hizo evidente la necesidad de crear componentes que pudieran escalar fácilmente en el futuro. Este enfoque a largo plazo no solo permitió que los componentes fueran reutilizables en diferentes contextos, sino que también aseguró que puedan adaptarse a las necesidades cambiantes del mercado.

6.4 Contribuciones y futuro

La creación de esta librería de componentes no solo ha aportado a la optimización interna del proceso de desarrollo en Truenorth, sino que también ha sentado las bases para futuras mejoras y extensiones. A continuación, se mencionan algunas de las principales contribuciones y posibles áreas de expansión:

- **Contribución al desarrollo web interno**: la librería ha permitido a los equipos de desarrollo de Truenorth estandarizar y agilizar la creación de interfaces, reduciendo los errores y aumentando la eficiencia. Esta estandarización ha mejorado la comunicación entre los equipos, ya que todos utilizan los mismos recursos y siguen las mismas guías de diseño.
- **Potencial de ampliación**: la estructura modular y escalable de la librería abre la puerta para la creación de nuevos componentes o la integración de nuevas

tecnologías, como inteligencia artificial, análisis de datos o incluso componentes de realidad aumentada. La librería también puede evolucionar con la adición de herramientas de personalización más avanzadas o componentes específicos para sectores verticales (por ejemplo, finanzas o salud).

- **Mejoras en la documentación y adopción**: un área clave de mejora es la **documentación**. Aunque se ha creado un sitio de documentación básica para los desarrolladores, invertir en una guía de uso más interactiva, así como en la formación de los equipos, facilitaría aún más la adopción de la librería en otros proyectos futuros.

6.5 Reflexión final

Este proyecto ha demostrado que la implementación de una librería de componentes bien estructurada puede transformar la manera en que los equipos de desarrollo abordan el diseño y la construcción de interfaces de usuario. El enfoque modular y reutilizable no solo ha reducido el tiempo y los costos de desarrollo, sino que también ha mejorado la calidad de los productos finales.

La colaboración interdisciplinaria, el enfoque en la accesibilidad y la estandarización visual han sido elementos fundamentales para garantizar el éxito del proyecto. A largo plazo, la librería desarrollada no solo ofrece un valor inmediato para Truenorth, sino que también proporciona una plataforma sólida para futuras innovaciones en el desarrollo de interfaces de usuario.

Este proyecto es una clara demostración de cómo la tecnología y la organización pueden unirse para crear soluciones eficientes, escalables y sostenibles que benefician tanto a los equipos de desarrollo como a los usuarios finales.

Capítulo 7. Referencias bibliográficas

- Atlassian. (s.f.). Jira: Project management software. Recuperado de https://www.atlassian.com/software/jira
- Babich, N. (2019). UX/UI Design Trends for 2020 and Beyond. Smashing Magazine.
- Babich, N. (2020). Collaboration Between Designers and Developers: Best Practices. Smashing Magazine.
- Babich, N. (2020). How to Design Consistent UI Components. Smashing Magazine.
- Baxter, K. (2020). Designing for Performance: Weighing Aesthetics and Speed in Web Design. A List Apart.
- Baxter, K., Courage, C., & Caine, K. (2019). Understanding Your Users: A Practical Guide to User Research Methods. Morgan Kaufmann.
- Brown, T. (2020). Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. HarperBusiness.
- BrowserStack (2024). Angular vs React vs Vue: Core Differences. Recuperado de https://www.browserstack.com/guide/angular-vs-react-vs-vue
- Cecchini, L., & Roderick, S. (2020). Designing for Accessibility: A Guide to Inclusive UX. O'Reilly Media.
- Clarkson, P. J., Coleman, R., Keates, S., & Lebbon, C. (2020). Inclusive Design: Design for the Whole Population. Springer.
- Class Variance Authority (CVA). (s.f.). CVA Documentation. Recuperado de https://github.com/joe-bell/cva
- Codemotion. (2023). Frontend Developer Trends and Frameworks for 2023. Recuperado de https://www.codemotion.com/magazine/frontend/frontend-developer-trends-frameworks-2023/
- Coyle, S. (2019). Balancing Form and Function: Practical Approaches to UX Design. UX Matters.
- Díaz-Jorge, C. (2022). Colección de componentes web mediante tecnología Web Components. Universidad Politécnica de Valencia. Recuperado de https://riunet.upv.es/bitstream/handle/10251/195622/Diaz-Jorge%20-%20Coleccion%20de%20componentes%20web%20mediante%20tecnologia%20Web%20Components.pdf?sequence=2
- Devjobsscanner (2023). Recuperado de https://www.devjobsscanner.com/blog/the-most-demanded-frontend-frameworks/
- Duckett, J. (2014). HTML & CSS: Design and Build Websites. John Wiley & Sons.

- ECMA International. (2023). ECMAScript® 2023 Language Specification. Recuperado de https://www.ecma-international.org/publications-and-standards/standards/ecma-262/.
- Facebook, Inc. (2023). React: A JavaScript library for building user interfaces. Recuperado de https://reactjs.org/
- Feldman, R. (2019). Elm in Action: Safe, Functional Programming for the Web. Manning Publications.
- Feldman, R. (2020). Design Systems: A Practical Guide to Creating Design Languages for Digital Products. Smashing Magazine.
- Figma. (2016). Figma: Collaborative interface design tool. Recuperado de https://www.figma.com/
- Figma Mate-UI (2024). Recuperado de https://www.figma.com/design/yCUtLVdoonjfiDd9eNWIDS/Mate-UI-3.0
- Fireart Studio. (2023). Frontend Trends 2023-Frontend Technologies. Recuperado de https://fireart.studio/blog/front-end-development-trends-2020-most-popular-javascript-frameworks/
- Fogg, B. J. (2019). Persuasive Technology: Using Computers to Change What We Think and Do. Morgan Kaufmann.
- FrontTribe. (2023). Front-End Development Trends: A Comprehensive Guide for 2023. Recuperado de https://www.fronttribe.com/stories/front-end-development-trends-2023-guide
- Frost, B. (2016). Atomic Design: Creating Design Systems. Capítulo 1 - Designing Systems.
- Garrett, J. J. (2020). The Elements of User Experience: User-Centered Design for the Web and Beyond. New Riders.
- GitHub. (s.f.). GitHub Documentation. Recuperado de https://docs.github.com/
- Gómez, M., López, J., & Morales, L. (2022). La accesibilidad y la semántica en HTML5: hacia una web más inclusiva. Revista de Investigación en Ingeniería y Tecnología, 12(1), 45-60.
- Google. (2023). Angular. Recuperado de https://angular.io/
- Google. (s.f.). Lighthouse. Recuperado de https://developers.google.com/web/tools/lighthouse
- Google Developers. (2023). Core Web Vitals. Recuperado de https://developers.google.com
- Gothelf, J. (2021). Lean UX: Designing Great Products with Agile Teams. O'Reilly Media.

- Hassenzahl, M., & Tractinsky, N. (2019). User Experience: A Research Agenda. Behaviour & Information Technology.
- Indianic (2023). Recuperado de https://www.indianic.com/blog/web/choosing-the-best-frontend-development-framework-for-2023.html
- Interaction Design Foundation. (2019). The Basics of User Experience (UX) Design.
- Johnson, J. (2021). Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules. Morgan Kaufmann.
- Kholmatova, A. (2016). Design Systems: A practical guide to creating design languages for digital products.
- Krug, S. (2020). Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. New Riders.
- LambdaTest. (2023). 18 Front End Development Trends to Follow in 2023. Recuperado de https://www.lambdatest.com/blog/front-end-development-trends/
- Lazar, J., Feng, J.H., & Hochheiser, H. (2017). Research Methods in Human-Computer Interaction. Elsevier.
- Lighthouse. (2023). Lighthouse Performance Metrics. Recuperado de https://web.dev/lighthouse-performance
- Lighthouse. (2023). SEO audit with Lighthouse. Recuperado de https://developers.google.com/web/tools/lighthouse
- Linkedin (2023). Recuperado de https://www.linkedin.com/pulse/top-5-frontend-frameworks-web-development-2023-inspirisys/
- Marcotte, E. (2019). Responsive Web Design: Performance and Aesthetics in Balance. A Book Apart.
- Material-UI Documentation. (2023). "Getting Started." Recuperado de Material-UI Official Documentation https://material-ui.com/
- McNulty, N. (2021). Digital Accessibility: A Primer for Web Developers and Designers. Apress.
- Mead, A. (2020). The Modern JavaScript Bootcamp. Independently published.
- Meigs, T. (2019). Professional JavaScript for Web Developers. Wiley.
- MDN Web Docs. (2023). "CSS: Cascading Style Sheets." Recuperado de MDN Web Documentation https://developer.mozilla.org/en-US/docs/Web/CSS
- MDN Web Docs. (2023). "HTML: HyperText Markup Language." Recuperado de MDN Web Documentation https://developer.mozilla.org/en-US/docs/Web/HTML
- Miller, R. (2021). UX/UI for Web: Effective User Interface Design for Modern Websites. Apress.

- Microsoft. (2023). TypeScript: JavaScript with syntax for types. Recuperado de https://www.typescriptlang.org/
- Moran, K. (2020). Usability and User Experience: Research, Design, and Evaluation. Morgan Kaufmann.
- Moran, K., & Nielsen, J. (2020). Usability and User Experience 2020: Industry Report. Nielsen Norman Group.
- Moshagen, M., & Thielsch, M. T. (2020). Facets of Visual Aesthetics for Screen-Based Interfaces. International Journal of Human-Computer Studies.
- Nielsen, J., & Budiu, R. (2020). Mobile Usability. New Riders.
- Norman, D. (1988). The Design of Everyday Things.
- Norman, D. A. (2020). The Design of Everyday Things: Revised and Expanded Edition. Basic Books.
- Norman, D., & Nielsen, J. (2013). The Definition of User Experience (UX). Nielsen Norman Group.
- Openinnova. (2024). Recuperado de https://www.openinnova.es/angular-vs-react-vs-vue-espanol-cual-elegir/
- React Documentation. (2023). "Introducing JSX." Recuperado de React Official Documentation https://react.dev/
- React: Facebook. (2023). React – A JavaScript library for building user interfaces. https://reactjs.org/
- Radix UI. (s.f.). Radix UI Documentation. Recuperado de https://www.radix-ui.com/
- Revista Social Fronteriza. (2021). Desarrollo de tecnologías web: Un enfoque en HTML5, CSS3 y JavaScript. Revista Social Fronteriza, 4(2), 85-102. Recuperado de https://www.revistasocialfronteriza.com/ojs/index.php/rev/article/view/282
- Righi, C., & James, A. (2020). User-Centered Design Stories: Real-World UCD Case Studies. Morgan Kaufmann.
- Rogers, Y., Sharp, H., & Preece, J. (2020). Interaction Design: Beyond Human-Computer Interaction. Wiley.
- Schneiderman B. (2005). Designing the User Interfaces. Addison Wesley.
- Schwaber, K., & Sutherland, J. (2017). The Scrum Guide. Scrum.org.
- Simform (2024). Recuperado de https://www.simform.com/blog/javascript-frontend-frameworks/
- Simpson, E. (2015). Advances in ECMAScript: Understanding the Evolution from ES5 to ES6 and Beyond. Journal of Web Development.
- Simpson, K. (2015). You Don't Know JS: ES6 & Beyond. O'Reilly Media.

- StackOverflow (2024).Web frameworks and Technologies survey. Recuperado de https://survey.stackoverflow.co/2024/technology#2-web-frameworks-and-technologies
- State of JavaScript (2023). Frontend Frameworks. Recuperado de https://2023.stateofjs.com/en-US/libraries/front-end-frameworks/
- Storybook. (s.f.). Storybook: Develop UI components in isolation. Recuperado de https://storybook.js.org/
- Storybook Team. (2023). Storybook: Open source tool for developing UI components in isolation. Disponible en https://storybook.js.org/
- Tailwind Labs. (2017). Tailwind CSS: A utility-first CSS framework. Recuperado de https://tailwindcss.com/
- Vercel. (2016). Next.js: The React Framework for Production. Recuperado de https://nextjs.org/
- Vercel. (s.f.). Vercel Documentation. Disponible en: https://vercel.com/docs
- Vuetify. (2020). Vuetify: Vue.js UI Library with beautifully handcrafted Material Components. Recuperado de https://vuetifyjs.com/
- Vue.js: You, E. (2014). Vue.js: The Progressive JavaScript Framework. https://vuejs.org/
- WebAIM. (2023). WAVE Web Accessibility Evaluation Tool. Recuperado de https://wave.webaim.org
- W3C. (2018). Web Content Accessibility Guidelines (WCAG) 2.1. World Wide Web Consortium (W3C).