DESARROLLO MODULAR WEB CON ATOMIC DESIGN PARA AUTENTICACIÓN DE USUARIOS Y GENERACIÓN DE REPORTES EN NEXT.JS

PRACTICA PROFESIONAL

MIGUEL ALEJANDRO RUANO QUINTERO

INSTITUCIÓN UNIVERSITARIA COLEGIO MAYOR DEL CAUCA

FACULTAD DE INGENIERÍA

INGENIERÍA INFORMÁTICA

POPAYÁN-CAUCA

2024

DESARROLLO MODULAR WEB CON ATOMIC DESIGN PARA AUTENTICACIÓN DE USUARIOS Y GENERACIÓN DE REPORTES EN NEXT.JS

PRACTICA PROFESIONAL

MIGUEL ALEJANDRO RUANO QUINTERO

ASESOR DEL PROYECTO:
MA. PEDRO ALVAREZ

COASESOR DEL PROYECTO:
MSC. FRANCISCO JAVIER OBANDO

COASESORA:

MSC ING. ALEJANDRA YICEL VARGAS MUÑOZ

INSTITUCIÓN UNIVERSITARIA COLEGIO MAYOR DEL CAUCA

FACULTAD DE INGENIERÍA

INGENIERÍA INFORMÁTICA

POPAYÁN-CAUCA

2024

Tabla de contenido

INTRODUCCIÓN **1**

ANTECEDENTES **2**

I. PLANTEAMIENTO DEL PROBLEMA **2**

II. OBJETIVOS 5

Objetivo General: 5

Objetivos Específicos 5

III. JUSTIFICACIÓN **6**

IV. METODOLOGÍA ATOMIC DESIGN **7**

- Jerarquía de Componentes en Atomic Design 7

Átomos 7

Moléculas 7

Organismos 7

Plantillas 7

Páginas 8

V. ALCANCE **9**

VI. Metodología del desarrollo. **11**

Capítulo 1- Marco de Referencia **13**

1. Marco Teórico 13

1.1. Principios de Atomic Design 14

1.2. Pattern Lab y Ejemplo de aplicación Atomic Design. 15

1.3. Trabajos relacionados con Atomic Design 17

1.4. Herramientas para el desarrollo de la aplicación web 18

1.5. Evaluación y aplicabilidad 19

1.5.1. Evaluación de la reutilización de componentes. 20

1.5.2. Consistencia y escalabilidad en el diseño atómico. 20

1.5.3. Mantenibilidad y modularidad del código. 20

Capítulo 2. Definición de Componentes Basados en la Metodología Atomic Design. 21

2. Análisis de Requerimientos Funcionales 21

2.1. Identificación de requerimientos por el centro de desarrollo tecnológico Creatic 21

2.1.1. Diagramas de flujo y de secuencia. 25

2.2. Identificación y Clasificación de Componentes. 29

2.2.1. Átomos para el módulo de Autenticación. 30

2.2.2. Átomos para el módulo de Reportes. 32

2.2.3. Moléculas para el módulo de Autenticación 33

2.2.4. Moléculas para el módulo de Reportes 34

2.2.5. Organismos para el módulo de Autenticación 34

2.2.6. Organismos para el módulo de Reportes 38

2.2.7. Exclusión en la construcción de Témplate 42

2.2.8. Paginas (Pages) 43

Capítulo 3. Desarrollo del Proyecto Aplicando Atomic Design para los Módulos Autenticación y Generación y Visualización de Reportes 44

3. Arquitectura del Proyecto 44

3.1. Estructura y desarrollo de átomos 45

3.2. Estructuras y desarrollo de moléculas 55

3.3. Estructura y desarrollo de organismos 58

Capítulo 4. Evaluación De Metodología Atomic Design 70

4. Pruebas y Evaluación de la Aplicación Web. 70

4.1. Evaluación de la Aplicabilidad de Atomic Design 70

4.1.1. Módulo de Autenticación 70
4.1.2. Módulo de Generación y Visualización de Reportes 72
4.2. Modularidad y Mantenibilidad 73
4.3. Implementación de Pruebas con React Testing Library y Jest 73
4.4. Pruebas de seguridad en el uso de Tokens. 77
4.5. Implementación de Rutas Protegidas 79
4.6. Pruebas de Autenticación 80
4.7. Uso de Claves Secretas y Algoritmos de Firma: 81
Recomendaciones 82
Conclusiones 83
Bibliografía 84

Tabla de figuras

Figura 1 .Metodología atomic design 9
Figura 2 .Ejemplo de aplicación Metodología atomic design 16
Figura 3. Diagrama de flujo de autenticación 26
Figura 4. Diagrama de flujo reportes 27
Figura 5. Diagrama de secuencia autenticación 28
Figura 6. Diagrama de secuencia reportes 29
Figura 7. Componente molécula autenticación 33
Figura 8. Componente molécula reporte 34
Figura 9. Organismo rol autenticación 35
Figura 10. Organismo inicio de sesión autenticación 36
Figura 11. Organismo registrar usuario autenticación 37

Figura 12. Organismo recuperar contraseña autenticación 37

Figura 13. Organismo reestablecer contraseña autenticación..... 38

Figura 14. Organismo resumen reportes 39

Figura 15. Organismo reporte detalle transacción. 40

Figura 16. Organismo reporte detalles pagos..... 40

Figura 17. Organismo reporte graficas 41

Figura 18. Organismo reporte grafica 42

Figura 19. Arquitectura atomic design 44

Figura 20. Archivos atómicos desarrollo 45

Figura 21. Código botón desarrollo..... 46

Figura 22. Estilos botón 47

Figura 23. Código botón de Google..... 47

Figura 24. Estilos botón de Google..... 48

Figura 25. Código mensajes de error 48

Figura 26. Estilos códigos mensaje de error..... 49

Figura 27. Código campo de entrada 49

Figura 28. Estilos campo entrada de texto 50

Figura 29. Código etiqueta 50

Figura 30. Estilos etiqueta 51

Figura 31. Código para lista de opciones 51

Figura 32. Estilos lista de opciones 52

Figura 33. Código titulo..... 53

Figura 34. Estilo titulo 53

Figura 35. Código grafica de barras 54

Figura 36. Código grafica de torta 54

Figura 37. Archivos de molécula desarrollo 55

Figura 38. Codigo para formulario de filtros para generar un reporte 56

Figura 39. Código entrada de contraseña 56

Figura 40. Estilos campo contraseña..... 57

Figura 41. Código para generar una nueva contraseña 57

Figura 42. Código entrada de correos electrónicos 58

Figura 43. Formulario de inicio de sesión 59

Figura 44. Código selección de rol 61

Figura 45. Código formulario registro 62

Figura 46. Código formulario de recuperación de contraseña 64

Figura 47. Código organismo de graficas 68

Figura 48. Código grafico de barras totales 69

Figura 49. React Testing Library Y Jest..... 74

Figura 50. Archivos con pruebas (átomos) 74

Figura 51. Resultado de pruebas(atomos) 75

Figura 52. Archivo de pruebas(moléculas) 76

Figura 53. Resultado de ejemplo pruebas (Molécula) 76

Figura 54. Código ejemplo de pruebas organismo 77

Figura 55. Resultado de pruebas (organismo)..... 77

Figura 56. Flujo de Autenticación y Validación de Tokens JWT 78

Figura 57. Código creación y expiración del token 78

Figura 58. Código middleware 78

Figura 59. Código validez de token de recuperación 79

Figura 60. Código rutas protegidas 80

Figura 61. Código captura de error token vencido 80

Figura 62. Mensaje en pantalla token expirado 81

Figura 63. Visualización de algoritmo HS256 en base de datos 81

Tabla de tablas

Tabla 1. Requerimientos autenticación. 23

Tabla 2. Requerimientos visualización y generación de reportes contables 25

Tabla 3. Componentes atómicos autenticación 31

Tabla 4. Componentes atómicos reportes 33

DEDICATORIA

Dedico este proyecto a mis padres, Marco Aurelio Ruano y Nubia Quintero, pilares fundamentales en mi vida, quienes con su amor incondicional y sacrificio me han permitido alcanzar este sueño profesional. Su ejemplo de perseverancia y determinación ha sido mi mayor inspiración durante toda mi carrera universitaria.

A mis abuelos, María Santos Zúñiga y Miguel Santos Ruano, por su sabiduría y amor incondicional, por ser ejemplo de vida y por sembrar en mí valores que han guiado mi camino hasta este momento.

A mi familia y hermanos, quienes han sido mis compañeros de vida, por su apoyo inquebrantable, por compartir mis alegrías y estar presentes en los momentos difíciles de este recorrido académico.

AGRADECIMIENTO

A mis padres, por su amor, paciencia, sacrificio y motivación que han sido el motor de mi desarrollo académico y personal. A toda mi familia, especialmente a mis hermanos y abuelos, María Santos Zúñiga y Miguel Santos Ruano, por ser parte fundamental de este logro y acompañarme en cada paso de mi formación.

Al Ingeniero Pedro Álvarez, asesor de este proyecto, por su paciencia, generosidad al compartir su conocimiento y guía durante todo el proceso de desarrollo, contribuyendo significativamente al éxito de este trabajo.

A la Ingeniera Alejandra Yicel Vargas Muñoz, mi asesora de prácticas profesionales, por su dedicación y apoyo invaluable durante esta etapa crucial de mi formación. Su comprensión ante las situaciones externas que se presentaron durante el proceso y su constante respaldo fueron fundamentales para poder culminar exitosamente esta etapa.

A la Institución Universitaria Colegio Mayor del Cauca, expreso mi más sincero agradecimiento por brindarme un ambiente propicio para el aprendizaje y por los años enriquecedores vividos en sus instalaciones. Su dedicación a proporcionar los conocimientos necesarios ha sido fundamental para mi formación profesional.

Al Centro de Desarrollo Tecnológico CREATIC, lugar donde realicé mis prácticas profesionales, por abrirme sus puertas y permitirme crecer profesionalmente.

RESUMEN

El presente documento describe el proyecto de trabajo de grado titulado "Desarrollo Modular Web con Atomic Design para Autenticación de Usuarios y Generación de Reportes en Next.js". El objetivo principal es crear una aplicación web enfocada en el sector agropecuario, utilizando la metodología Atomic Design.

Este proyecto, que forma parte del centro de desarrollo tecnológico Creatic, se enfoca en dos funcionalidades clave:

1. Autenticación de usuarios: Facilitar el acceso a usuarios previamente registrados y ofrecer la opción de registro para nuevos usuarios.
2. Visualización de reportes: Permitir a los usuarios ver información de reportes contables específicos del sector agropecuario.

La metodología Atomic Design, creada por (Frost, 2016), es fundamental para este desarrollo como práctica óptima para crear los módulos frontend de autenticación de usuarios y visualización de reportes. Esta técnica descompone la interfaz en componentes pequeños llamados "átomos", que luego se integran y reutilizan para construir páginas más complejas. Para garantizar un diseño eficiente, un desarrollo robusto y un funcionamiento óptimo, el proyecto utiliza herramientas como Balsamiq para prototipado, propuesto por el líder del proyecto, Next.js como framework frontend, Node.js, express.js para el backend, PostgreSQL como base de datos y una API RESTful para la comunicación entre frontend y backend. El objetivo es asegurar que todos los componentes, tanto en el frontend como en el backend, funcionen de manera eficaz y armoniosa, contribuyendo así al éxito global del desarrollo web en los módulos mencionados.

ABSTRACT

The document outlines a degree project titled "Modular Web Development with Atomic Design for User Authentication and Report Generation in Next.js." The primary objective is to create a web application focused on the agricultural sector, employing the Atomic Design methodology.This project, part of the technological development center Creatic, concentrates on two key functionalities:

1. User Authentication: This feature facilitates access for registered users and provides a registration option for new users.
2. Report Visualization: It allows users to view specific accounting reports related to the agricultural sector.

The Atomic Design methodology, developed by Brad Frost in 2016, is essential for this development as it serves as an optimal practice for creating frontend modules for user authentication and report visualization. This technique breaks down the interface into small components called "atoms," which are then integrated and reused to build more complex pages.To ensure efficient design, robust development, and optimal functionality, the project utilizes tools such as Balsamiq for prototyping (as proposed by the project leader), Next.js as the frontend framework, Node.js and Express.js for the backend, PostgreSQL as the database, and a RESTful API for communication between the frontend and backend. The goal is to ensure that all components, both in the frontend and backend, operate effectively and harmoniously, thus contributing to the overall success of the web development in the mentioned modules.

INTRODUCCIÓN

El Centro de Desarrollo Tecnológico Creatic se encuentra en el departamento del Cauca, Colombia, y su misión es promover el fortalecimiento del ecosistema de emprendimiento tecnológico. Creatic se enfoca principalmente en proyectos de innovación, investigación aplicada, desarrollo y transferencia de tecnología que contribuyan al desarrollo social y económico del país.

De esta manera el centro desarrollo tecnológico en sus investigaciones y con el fin de brindar apoyo e innovación, ve la necesidad de desarrollar un software contable en el sector agropecuario que brinde procesos confiables en las transacciones que realicen los usuarios del sector agropecuario, por tal motivo los módulos de autenticación y generación y visualización de reportes son esenciales para brindar en el proceso la seguridad y análisis de los datos.

La implementación de desarrollos modulares y reutilizables es fundamental para garantizar la eficiencia y las buenas prácticas en el desarrollo de software. Los diseños deben ser escalables, adaptables y flexibles. Además, la aplicación de metodologías de diseño, como el Atomic Design, es crucial para crear interfaces efectivas y brindar una experiencia de usuario óptima. Esta metodología permite la creación de componentes atómicos de interfaz, facilitando su reutilización e integración para construir interfaces más complejas y coherentes.

ANTECEDENTES

I. PLANTEAMIENTO DEL PROBLEMA

El centro de desarrollo tecnológico CREATIC tiene como necesidad desarrollar una aplicación web con el fin de llevar a cabo la contabilidad del sector agropecuario, contribuyendo a mejorar la administración de cada empresa, la protección de datos es requisito esencial para la privacidad de operaciones que se realizan dentro de la aplicación web, por tal motivo el centro de desarrollo tecnológico ve la necesidad de desarrollar la autenticación de usuarios que se ha identificado como una medida crucial para salvaguardar la seguridad en línea, ofreciendo beneficios que incluyen la protección de datos personales, la prevención de fraudes y suplantaciones de identidad, así como el acceso seguro a los recursos digitales. Sin embargo, a pesar de la importancia de esta medida, persisten desafíos en su implementación efectiva en el sector agropecuario y otros sectores económicos (Solano Romero, 2024).

Por otra parte, la necesidad de información precisa y relevante sobre el desarrollo y crecimiento económico del sector agropecuario es fundamental para la toma de decisiones informadas y el impulso de la productividad, lo que desempeña un papel fundamental en el sector agrícola, ya que permite tomar decisiones informadas y mejorar la eficiencia de las operaciones agrícolas (León Mendoza, 2018).

En consecuencia, según (Pedraza-Gutiérrez, 2023) la falta de consistencia en el diseño y la funcionalidad de los componentes a medida que el proyecto crece puede resultar en una interfaz de usuario desorganizada, con estilos y comportamientos inconsistentes en diferentes áreas del sitio web o la aplicación. Es esencial abordar esta problemática desde las etapas iniciales del proyecto para asegurar una experiencia de usuario coherente y de alta calidad.

En el desarrollo de la aplicación web, es fundamental para el centro de desarrollo garantizar una experiencia de usuario óptima. El enfoque de diferentes metodologías y la utilización de diferentes frameworks proporcionan herramientas poderosas para la construcción modular de interfaces. A pesar de estas ventajas, el proceso de desarrollo enfrenta desafíos relacionados con la complejidad en la autenticación de usuarios y la generación y visualización de reportes.

La implementación de soluciones frontend para la autenticación de usuarios y la generación de reportes se ve obstaculizada por problemas de modularidad y eficiencia en la construcción de interfaces. La carencia de una estructura modular definida y la necesidad de componentes reutilizables dificultan el desarrollo ágil y escalable de estas funcionalidades (Saldaña Cercós, 2018).

Además, es crucial resaltar que la creciente demanda de diseños estándares de alta calidad en la industria intensifica aún más los desafíos que enfrentan los desarrolladores. El desarrollo del backend que no se desarrolla adecuadamente o no se integra correctamente con el frontend, pueden surgir una serie de problemas que afectan la funcionalidad y la experiencia del usuario. (Guamanga, 2020) subraya además la necesidad de una evaluación formal de esta coherencia, ya que es esencial para el desarrollo de habilidades de pensamiento crítico. Por lo tanto, una evaluación formal de la coherencia entre la lógica backend y frontend no sólo es crucial para una experiencia de usuario fluida sino también para el desarrollo de habilidades de pensamiento crítico.

Por consiguiente, la necesidad imperante de abordar estos desafíos mediante el desarrollo de un enfoque metodológico que facilite diseñar y desarrollar la autenticación de usuarios y la generación de reportes en la plataforma. Este enfoque debe basarse en el concepto de Atomic Design y utilizar el framework Next.js. Según lo señalado por (Grajales Urquijo, 2024) “aunque los proyectos pueden operar de manera efectiva, a menudo carecen de una estructura sólida. Esta falta de organización, si bien no impide

el desarrollo ni la ejecución de tareas, dificulta la creación de un orden definido y puede tener un impacto negativo a largo plazo en el mantenimiento de la aplicación web”.

La falta de una guía metodológica específica para abordar los desafíos mencionados anteriormente dificulta aún más el proceso de desarrollo y diseño web. La ausencia de un marco de trabajo claro y definido puede llevar a decisiones subjetivas y poco fundamentadas en el diseño y la implementación de componentes, lo que a su vez afecta negativamente la coherencia y la calidad del producto final.

Lo anteriormente planteado lleva a generar la siguiente pregunta:

¿Cómo desarrollar una aplicación web modular para la autentificación de usuarios y generación de reportes?

Mejorar la modularidad con metodologías enfocadas en el diseño y construcción del frontend es lo ideal si se busca eficiencia en el desarrollo de la aplicación web, la eficiencia en el desarrollo modular comprende en la consistencia visual de cada componente, reutilización, flexibilidad, escalabilidad, colaboración y mantenimiento, como lo menciona (León Ochoa, 2024).

II. OBJETIVOS

Objetivo General:

- Desarrollar una aplicación web para la autenticación de usuarios, así como para la generación y visualización de reportes, implementando el concepto de Atomic Design de manera modular en Next.js.

Objetivos Específicos

- Definir los componentes atómicos, moleculares y organismos necesarios para los módulos de autenticación de usuarios y de reportes centrada en la experiencia del usuario.

- Desarrollar backend y frontend aplicando Atomic Design para los módulos de generación y visualización de reportes, para la construcción modular y eficiente de interfaces.

- Evaluar los módulos de autenticación y generación/visualización de reportes en la plataforma, garantizando el cumplimiento de requisitos y la aplicabilidad del concepto Atomic Design.

III. JUSTIFICACIÓN

En el mundo actual, donde la experiencia del usuario es fundamental para el éxito de cualquier software, el desarrollo de sitios web eficientes y fáciles de usar es esencial (Rivera, 2024). El diseño y desarrollo del frontend encargado de la interacción directa con los usuarios y de crear interfaces intuitivas y funcionales, juegan un papel crucial para que la autenticación de usuario sea exitosa y segura, ya que permite al usuario no preguntarse del funcionamiento del sistema al ingresar (Pedraza-Gutiérrez, 2023).

Así mismo al aplicar Atomic Design en la creación de reportes para un sitio web, se pueden seguir los principios de diseño atómico, que consisten en dividir la interfaz en componentes más pequeños y reutilizables, como átomos, moléculas, organismos, plantillas y páginas. Esto facilita la creación de reportes consistentes, eficientes y fáciles de mantener.

Al utilizar Atomic Design, se pueden diseñar componentes visuales como gráficos, tablas, tarjetas informativas, botones interactivos, entre otros, de manera modular y coherente. Esto permite una mayor flexibilidad y agilidad en la creación y personalización de reportes en una aplicación web, ya que los componentes pueden combinarse y reutilizarse de forma sencilla

IV. METODOLOGÍA ATOMIC DESIGN

La metodología utilizada en este proyecto se ha basado en la metodología Atomic Design, para el diseño de interfaces de usuario desarrollada por (Frost, 2016). Que se basa en la idea de construir sistemas de diseño modulares y escalables mediante la descomposición de las interfaces en componentes reutilizables.

- **Jerarquía de Componentes en Atomic Design**

La metodología atomic design según (Villalobos Garcia, 2023) “es caracterizada por el sentido paralelo que representa frente al campo de la química al exponer con coherencia las magnitudes entre capas y cómo estas conforman una unidad contemplada por la escala de sus componentes; átomos, moléculas, organismos, plantillas y página, ofreciendo así un marco de interpretación para las distintas fases que reúnen lo abstracto del proceso creativo para ejecutarlo lógicamente, entendiéndose como una herramienta de colaboración que forma parte de un ecosistema dinámico y está integrada en un proceso en constante evolución y repetición”.

De acuerdo al contexto mencionado anteriormente, a continuación, se describe los cinco niveles en la estructura de atomic design.

Átomos: Son los componentes más básicos, como botones, etiquetas, iconos y elementos de formulario.

Moléculas: Son grupos de átomos que forman componentes más complejos, como una barra de búsqueda o un bloque de tarjetas.

Organismos: Son combinaciones de moléculas que forman secciones completas de una interfaz, como un encabezado, un menú de navegación o un formulario.

Plantillas: Son agrupaciones de organismos que definen el esqueleto de una página o pantalla.

Páginas: Son instancias específicas de plantillas que representan la interfaz final.

Atomic Design promueve la creación de sistemas de diseño coherentes y escalables al facilitar la reutilización de componentes y la colaboración entre diseñadores y desarrolladores. Además, al descomponer las interfaces en componentes modulares, se mejora la capacidad de adaptación a diferentes contextos y dispositivos (Frost, 2016).

Las ventajas significativas que la hacen particularmente recomendable son:

En primer lugar, su enfoque modular y jerárquico facilita la creación de interfaces complejas a partir de componentes reutilizables, lo que fomenta la consistencia visual y la eficiencia en el desarrollo (Frost, 2016). Además, al descomponer las interfaces en componentes atómicos, se facilita la colaboración entre diseñadores y desarrolladores, lo que promueve una comunicación más fluida y una implementación más precisa de los diseños (Frost, 2016).

Otra ventaja clave de Atomic Design es su capacidad para adaptarse a diferentes contextos y dispositivos. Al construir interfaces con componentes modulares, es más sencillo ajustar y reutilizar estos componentes en diferentes entornos, lo que mejora la escalabilidad y la capacidad de respuesta del producto (Frost, 2016).

Finalmente, Atomic Design promueve la creación de sistemas de diseño sólidos y mantenibles. Al establecer una jerarquía clara y una biblioteca de componentes reutilizables, se reduce la deuda técnica y se facilita la actualización y el mantenimiento del producto a largo plazo (Frost, 2016).

Para concluir, Atomic Design ofrece una solución más específica y especializada para abordar los desafíos actuales del diseño de interfaces de usuario. Su enfoque modular,

su capacidad de adaptación y su capacidad para construir sistemas de diseño coherentes y escalables la convierten en una excelente opción para el anteproyecto.

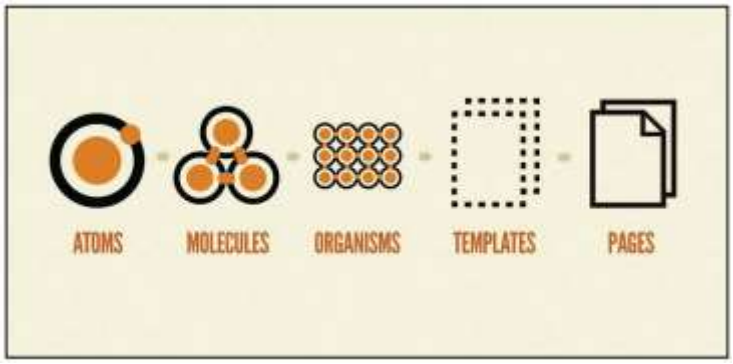


Figura 1 *.Metodología atomic design*

Nota: *Tomado de Atomic Design (p. 41), por B. Frost, 2013.*

V. ALCANCE

Para construir el alcance del proyecto "Desarrollo Modular Web con Atomic Design para Autenticación de Usuarios y Generación de Reportes", es importante detallar los objetivos y entregar un marco claro sobre los componentes que se desarrollarán, su propósito y las metodologías que se utilizarán. Teniendo en cuenta lo anterior y los objetivos del proyecto, se plantea el desarrollo en las siguientes fases:

Fase No: 1: Conceptualización de la Metodología Atomic Design: Esta fase se orienta a comprender la importancia de la metodología atomic design, su funcionamiento y estructura para el desarrollo modular de interfaces

Fase No. 2: Identificación y Definición de Componentes Atómicos: En esta fase, se identificará y definirá los componentes fundamentales como (átomos) necesarios para

los módulos de autenticación y reportes, basados en el análisis de los requerimientos funcionales de Creatic. Posteriormente, se procederá a combinar componentes para formar moléculas y organismos con el fin de obtener funciones más avanzadas (Grajales Urquijo, 2024).

Fase No. 3: Desarrollo del Backend y Frontend Modular: En esta fase, se implementará el backend utilizando express.js, Node.js y PostSQL para manejar grandes volúmenes de datos y asegurar la seguridad en el acceso a la aplicación web y sus recursos. Al mismo tiempo, se desarrolla el frontend en Next.js con TypeScript, aplicando la metodología Atomic design para construir interfaces modulares y eficientes. Además, mediante API RESTful se establece la comunicación entre backend y frontend.

Fase No. 4: Evaluación de la Metodología Atomic Design: Una vez desarrollados los módulos, se realizará una evaluación exhaustiva del frontend en términos de organización, consistencia y mantenimiento. Se verificará que los componentes como (átomos, moléculas y organismos) estén correctamente integrados y cumplan los objetivos de atomic design en términos como reutilización y modularidad. Además, se evaluará el cumplimiento de los requisitos de establecidos por Creatic. (Roedavan, 2020) adaptó el método de diseño atómico para el desarrollo rápido de juegos, mientras que (Ortiz Caballero, 2024) desarrolló un marco para construir un sistema de diseño reutilizable utilizando principios de diseño atómico.

VI. Metodología del desarrollo.

Considerando las fases establecidas en el alcance para el logro de cada uno de los objetivos específicos propuestos, se presentan las actividades correspondientes a cada una de estas etapas.

Fase de Desarrollo del Proyecto	**Actividades que comprende**
Fase 1: Conceptualización de la Metodología Atomic Design	● Revisión de artículos sobre la metodología Atomic Design y su aplicabilidad en el diseño de interfaces de usuario. ● Investigación sobre los principios de atomic design, estructura y funcionamiento.
Fase 2: Identificación y Definición de Componentes Atómicos	● Análisis de los requerimientos funcionales del Centro de Desarrollo Tecnológico Creatic. ● Definición de componentes atómicos necesarios para los módulos de autenticación y generación de reportes. ● Creación de moléculas mediante la combinación de átomos previamente establecidos ● Diseño de organismos que combinen átomos y moléculas para obtener estructuras más avanzadas.

	● Diseño de plantillas como estructuras finales de las interfaces de los módulos.
Fase 3: Desarrollo del Backend y Frontend Modular	● Implementación de express.js, Node.js, establecer la arquitectura necesaria para manejar los datos, autenticación, autorización y seguridad. ● Implementación de la estructura de la metodología atomic design para el desarrollo de componentes. ● Desarrollo y combinación de átomos, moléculas, organismos, plantillas para obtener la interfaz final.
Fase 4: Evaluación de la Metodología Atomic Design.	● Verificar que se aplicó correctamente la metodología atomic design en términos de modularidad, consistencia y reutilización. ● Verificar seguridad a través de uso de tokens. ● Verificar que la contraseña este encriptada

Capítulo 1- Marco de Referencia

1. Marco Teórico

La metodología Atomic Design es un enfoque fundamental en el diseño de interfaces de usuario, se basa en el análisis de las interfaces en componentes más pequeños y simples, que luego se combinan para formar estructuras más complejas y completas (Macías Vélez, 2023). La integración de backend y frontend es crucial para una experiencia de usuario coherente, segura y eficiente en aplicaciones web (Pierin, 2018). En el desarrollo de aplicaciones web, la elección de frameworks y lenguajes para el frontend y backend es una consideración clave (del Cerro, 2020).

El enfoque en el desarrollo modular del frontend con Atomic Design en Next.js para la autenticación de usuarios y generación de reportes busca optimizar la construcción de interfaces web mediante la implementación de componentes modulares y reutilizables para lograr una interfaz de usuario usable y eficiente, necesitando herramientas para la programación como lenguajes de programación, frameworks Next js, entorno de desarrollo Node js y base datos PostgreSQL para proporcionar datos al usuario cada vez que el usuario realice una petición, también balmasmiq y wireframes para el diseño y creación de mockups del desarrollo web.

La experiencia de usuario es importante para el desarrollo del frontend, siendo la parte de interacción visual con cliente el primer paso para ganar la satisfacción del usuario. (Fernández Ordoñez, 2021) subraya la importancia de la experiencia del usuario (UX) en el desarrollo frontend, quien enfatiza el papel del diseño emocional en la creación de interacciones únicas y satisfactorias.

Atomic Design permite la descomposición de interfaces en elementos básicos (átomos) y su combinación para formar estructuras más complejas (organismos), promoviendo la modularidad y la coherencia visual (Frost, 2016).

La aplicación del concepto de Atomic Design, propuesto por (Frost, 2016), permite descomponer la interfaz en elementos básicos (átomos), combinándolos para formar

estructuras más complejas (organismos). Esta metodología promueve modularidad y la coherencia visual en el diseño de interfaces.

Este enfoque se basa en la idea de construir interfaces mediante la combinación de componentes más pequeños y simples, llamados átomos, para crear componentes más complejos y reutilizables.

1.1. Principios de Atomic Design

Los principios de Atomic Design proviene de los principios de la química. (Frost, 2016) analiza la idea del mundo físico, donde los átomos se combinan para formar moléculas y estas se agrupan formando organismos más complejos, los principios mencionados a continuación (Frost, 2016) los aplicó al diseño web ayudando especialmente a diseñadores y desarrolladores para la construcción de interfaces.

- **Átomos:** En la química, son las unidades más pequeñas de la materia y en diseño de la interfaz representa los elementos más básicos como botones, inputs o etiquetas, etc.

- **Moléculas:** En química una molécula es una combinación de átomos, en el diseño de la interfaz son combinaciones de átomos que forman componentes funcionales e interactivos, como la combinación de una etiqueta con un input.

- **Organismos:** En química, sistemas biológicos más complejos como el corazón, compuesto por tejidos (moléculas) y células que vendrían siendo los átomos, en el diseño del interfaz conjunto de moléculas que forman secciones completas de una interfaz como un formulario o encabezado.

- **Templates:** En química sistemas de órganos que trabajan juntos para realizar funciones completas como el esqueleto del cuerpo humano, en el diseño del interfaz

conjunto de organismos que forman la estructura, su interacción y distribución en una página, pero sin datos específicos.

- **Páginas:** En la biología el cuerpo humano en su totalidad, en diseño las páginas son instancias específicas de templates que contienen contenido real y proporcionan la experiencia final al usuario. Son la representación concreta de cómo se verá y se comportará la interfaz en su contexto de uso real.

1.2. Pattern Lab y Ejemplo de aplicación Atomic Design.

Pattern Lab es una herramienta desarrollada por (Frost, 2016) que implementa los principios del Atomic Design. La herramienta mencionada funciona como un framework que permite a los desarrolladores construir, visualizar, probar y mostrar componentes de la interfaz de usuario, dentro de un sistema de diseño, Al utilizar Pattern Lab, el equipo de desarrollo puede crear interfaces a partir de componentes atómicos, facilitando así la colaboración y la eficiencia en el desarrollo.

Por otro lado, las características que incluye Pattern Lab es ser un generador estático basado en node.js que combina componentes de interfaz de usuario para crear prototipos interactivos. Esta herramienta se destaca por incluir patrones anidados, lo que significa que los usuarios pueden integrar los patrones dentro de otros patrones, facilitando la creación de prototipos utilizando datos dinámicos, lo que asegura que los componentes funcionen correctamente en un entorno de producción.

Otra característica importante es su enfoque en la documentación eficiente. La herramienta permite documentar cada patrón de la interfaz de usuario. Lo que permite la comunicación clara entre diseñadores y desarrolladores.

Ejemplo de aplicación: Una página de perfil de usuario que utiliza una plantilla de diseño de página de perfil, que muestra la información personal del usuario junto con opciones para editar la configuración.

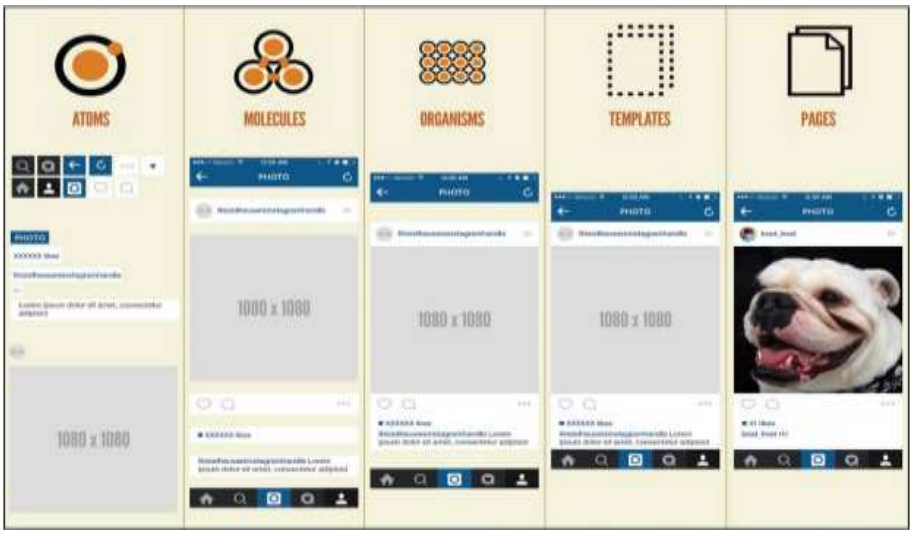


Figura 2 *.Ejemplo de aplicación Metodología atomic design*

Nota. *Tomado de Atomic Design (p. 59), por B. Frost, 2013*

"Repasemos esta interfaz atomizada de Instagram:

- Átomos: esta pantalla de la interfaz de usuario de Instagram consta de un puñado de íconos, algunos elementos a nivel de texto y dos tipos de imágenes: la imagen principal y la imagen del avatar del usuario.

- Moléculas: varios íconos forman componentes utilitarios simples como la barra de navegación inferior y la barra de acciones fotográficas donde los usuarios pueden dar me gusta o comentar una foto. Además, combinaciones simples de texto y/o imágenes forman componentes relativamente simples.

- Organismos: aquí podemos ver cómo toma forma el organismo de la foto, que consta de la información del usuario, la marca de tiempo, la foto en sí, las acciones alrededor

de esa foto y la información sobre la foto, incluido el recuento y el título. Este organismo se convierte en la piedra angular de toda la experiencia de Instagram, ya que se acumula repetidamente en un flujo interminable de fotografías generadas por los usuarios.

- Plantillas: podemos ver nuestros componentes unirse en el contexto de un diseño. Además, es aquí donde vemos el esqueleto de contenido expuesto de la experiencia de Instagram, destacando el contenido dinámico como el identificador, el avatar, la foto, el recuento de me gusta y el título del usuario.

- Pages: Y finalmente vemos el producto final, completo con contenido real, lo que ayuda a garantizar que el sistema de diseño subyacente se una para formar una interfaz de usuario hermosa y funcional." (Frost, 2016)

1.3. Trabajos relacionados con Atomic Design

A continuación, se presentan un conjunto de trabajos relacionados y asociados al desarrollo del frontend implementando la metodología Atomic Design, según (Grajales Urquijo, 2024) evidenciando mejora en la eficiencia de los proyectos trabajados con la metodología.

La exitosa transición de **IBM** hacia Atomic Design ha mejorado significativamente la eficiencia de sus proyectos frontend, con un enfoque en la reutilización y la coherencia. Este enfoque se alinea con el énfasis de la empresa en la innovación y la creatividad, como se ve en su énfasis en la innovación en el desarrollo de productos industriales. El uso de Atomic Design también ha mejorado la coherencia visual y funcionalidad.

Salesforce, un proveedor líder de soluciones CRM, ha implementado con éxito Atomic Design, lo que ha dado como resultado una mayor coherencia visual y funcional en todas sus aplicaciones. Esto no solo mejoró la experiencia del usuario, sino que también ha facilitado la escalabilidad del sistema de diseño. El uso de componentes reutilizables ha garantizado cambios consistentes en todas las aplicaciones, cruciales para las diversas necesidades de aplicaciones y clientes de Salesforce. Este enfoque se alinea con el enfoque centrado en el cliente de la empresa, que enfatiza la integración de una visión del cliente de 360 grados para mejorar los procesos de ventas, marketing y servicio.

Buffer, una empresa de gestión de redes sociales, ha implementado con éxito Atomic Design en su proceso de desarrollo, lo que ha llevado a una mayor eficiencia y coherencia de las aplicaciones. Este enfoque permite a la empresa dividir su interfaz de usuario en componentes atómicos reutilizables, acelerando el desarrollo de nuevas funciones y mejoras. La adopción de patrones de diseño reutilizables por parte de la empresa también ha dado como resultado una experiencia de usuario más consistente y atractiva en todas sus aplicaciones. Esta implementación exitosa de Atomic Design en Buffer demuestra su potencial en diversos contextos comerciales, no solo en el diseño web sino también en el desarrollo de servicios en línea.

GitHub. La comunidad de código abierto en torno a Atomic Design en GitHub ha enriquecido significativamente la metodología y ampliado su adopción en la comunidad de desarrollo. Esto se ha logrado compartiendo y colaborando en la creación de componentes atómicos y patrones de diseño reutilizables.

1.4. Herramientas para el desarrollo de la aplicación web

- **Next.js:** Es un framework de desarrollo web incorporando funciones y características necesarias construidas sobre java script para que los desarrolladores puedan desarrollar aplicaciones complejas rápidamente. Se ha

descubierto que Next.js, un marco de JavaScript, ofrece un rendimiento superior en comparación con otros marcos de front-end como React.js y Vue.js (Tong, 2023)

- **Node.js:** Es un entorno de ejecución de JavaScript, se utiliza ampliamente para desarrollar servidores web eficientes y escalables (Herron, 2020). Es particularmente útil para manejar múltiples tareas simultáneamente sin bloquear el flujo de la aplicación, debido a su naturaleza asincrónica (Herron, 2020). Esta capacidad es crucial para mejorar la experiencia del usuario, como se demuestra en la evaluación del rendimiento de una aplicación web (Hernández-Rueda, 2022).

- **PostgreSQL**: Es ampliamente utilizado como base de datos backend para aplicaciones web y móviles debido a su seguridad, escalabilidad y capacidad para manejar datos complejos. (Vintimilla, 2023). Ofrece funciones avanzadas de control de acceso y seguridad, lo que lo hace adecuado para almacenar información confidencial (Vintimilla, 2023).

- **Balsamiq:** Es una herramienta de creación de prototipos utilizada en el ámbito académico como en el industrial para crear maquetas y wareframe de interfaz de usuario (Jongwan Kim, 2021).

1.5. Evaluación y aplicabilidad

La evaluación del frontend se centrará en garantizar que los módulos de autenticación y generación/visualización de reportes cumplan con los requisitos planteados por el Centro De Desarrollo Creatic. Se evaluará la aplicabilidad del concepto Atomic Design según (León Ochoa, 2024) y (Frost, 2016) para mejorar la eficiencia en el desarrollo garantizando los siguientes aspectos:

1.5.1. Evaluación de la reutilización de componentes.

La modularidad de Atomic Design debe permitir reutilizar componentes fácilmente en diferentes partes de la aplicación, lo que ahorra tiempo y esfuerzo en el desarrollo (León Ochoa, 2024).

1.5.2. Consistencia y escalabilidad en el diseño atómico.

Al seguir una estructura jerárquica clara y consistente, Atomic Design debe facilitar mantener la coherencia visual y funcional en toda la aplicación.

Escalabilidad: Capacidad de combinar átomos y moléculas para crear componentes más complejos hace que la aplicación sea fácilmente escalable a medida que crecen las necesidades del proyecto (León Ochoa, 2024).

1.5.3. Mantenibilidad y modularidad del código.

Mantenibilidad: La organización modular del código facilita la identificación y corrección de errores, así como la realización de actualizaciones y mejoras en la interfaz de manera eficiente (Méndez Fernandez, 2022).

Capítulo 2. Definición de Componentes Basados en la Metodología Atomic Design.

El propósito de este proyecto es identificar y definir los componentes necesarios para descomponer las interfaces en elementos más pequeños y manejables. Estos elementos pueden ser combinados de manera inteligente y escalable para generar sistemas de diseño coherentes y eficientes para aplicaciones web.

2. Análisis de Requerimientos Funcionales

2.1. Identificación de requerimientos por el centro de desarrollo tecnológico Creatic

El centro de desarrollo tecnológico Creatic a través del líder de proyecto hizo la entrega de requerimientos funcionales propuestos para cada módulo. Con el fin de cumplir con los objetivos del proyecto, se realizó un análisis detallado de los requerimientos funcionales de la **tabla 1** y **tabla 2** establecidos por el centro de desarrollo Creatic. Este análisis se llevó a cabo con el fin de obtener información valiosa que se adapte a la metodología Atomic Design. El análisis detallado de los requerimientos consistió en la realización de diagramas flujo y secuencia de cada módulo para entender el funcionamiento en la aplicación web y la interacción con el usuario, para el desarrollo de catálogos como lo implementa (Méndez Fernandez, 2022). El catálogo mencionado en el proyecto software contable agropecuario se realizó en la **tabla 3** y **tabla 4**, los cuales son diseños visuales de los componentes y respectivamente aprobados por el líder del proyecto, para iniciar con la fase de desarrollo de la funcionalidad de los componentes. Identificar los componentes atómicos fué primordial, ya que son la base principal en el desarrollo del frontend utilizando la metodología Atomic Design (Frost, 2016).

A continuación, se presentan los requerimientos funcionales entregados por creatic para el desarrollo del módulo de autenticación.

Id	Requerimiento	Descripción	Prioridad	Comportamiento	Entradas	Salidas
1	Autenticación por medio de usuario y contraseña y por medio de google (priorizar)	El sistema debe permitir iniciar sesión y dar los permisos necesarios según corresponda al rol (superusuario, usuario registro información y usuario revisión informes	Alta	El usuario ingresa su usuario y contraseña o se autentica por medio de google y el sistema debe permitir o denegar el ingreso	usuario o correo y contraseña	Ingreso a la plataforma o denegación del ingreso por datos incorrectos
2	Gestión de roles	El sistema tendrá 3 tipos de roles para los usuarios y dependiendo el rol tendrá unas respectivas autorizaciones. Los roles serán: - ADMINISTRA	Alta	El sistema dependiendo del rol del usuario otorga unos permisos y autorizaciones para hacer determinadas actividades que el usuario requiera	Los datos a tener en cuenta para los usuarios son: nombre, correo, usuario, telefono, rol, dirección	El sistema debe proporcionar los permisos necesarios según el rol del usuario

		DOR: tendrá todos los permisos para crear, consultar, editar y eliminar los demás usuarios - AUXILIAR-CONTABLE: tendrá los permisos para la creación/registro de usuarios y la asignaión de roles y permisos - AUXILIAR-ADMINISTRATIVO: tendrá permisos para revisar informes			, país, departamento, ciudad, identificación, tipo de identificación	

***Tabla 1**. Requerimientos autenticación.*

***Fuente:** Elaboración propia*

A continuación, se presentan los requerimientos funcionales entregados por creatic para el desarrollo del módulo de reportes contables.

Id	Requerimiento	Descripción	Prioridad	Comportamiento	Entradas	Salidas
1	Visualización de reportes	El sistema mostrará reportes respecto a la información de facturación de un periodo determinado (mensual, semestral, anual, etc.), de acuerdo a los filtros seleccionados en la aplicación para su generación	Alta	El sistema debe generar un reporte que pueda ser visualizado dentro de la aplicación según los filtros que se hayan ingresado	Código de producto, tipo de producto, fecha, cantidad, precio unitario, cliente/proveedor.	Reporte en formato de tabla con campos como: ventas, pago de mano de obra, ingresos, egresos, costos y pagos.
2	Descarga de reportes	El sistema, tras haber	Media	El sistema debe permitir	Código de producto, tipo	Reporte en

		generado un reporte, permite descargar el resultado		descargar el reporte que se ha generado con los filtros generados	de producto, fecha, cantidad, precio unitario, cliente/proveedor.	formato .XLSX con toda la información que se visualiza en la aplicación

Tabla 2. *Requerimientos visualización y generación de reportes contables*

Fuente: *Elaboración propia*

2.1.1.Diagramas de flujo y de secuencia.

Para identificar los componentes que se necesitaban para cada módulo, se realizaron diagramas de flujo y de secuencia como ayuda para entender el comportamiento que tendrían los módulos de autenticación y de reportes en el sistema.

Los siguientes diagramas de flujo y de secuencia fueron entregas que se realizaron en la semana 1 y 2 aprobadas por el líder del proyecto. Esto brindaba seguridad de que los requerimientos fueron entendidos con éxito.

- **Diagrama de flujo para autenticación de usuarios.**

El siguiente diagrama de flujo para autenticación de usuarios en la aplicación web describe el proceso de registro e inicio de sesión, mostrando diferentes caminos o decisiones que el usuario puede tomar en su interacción con la aplicación web.

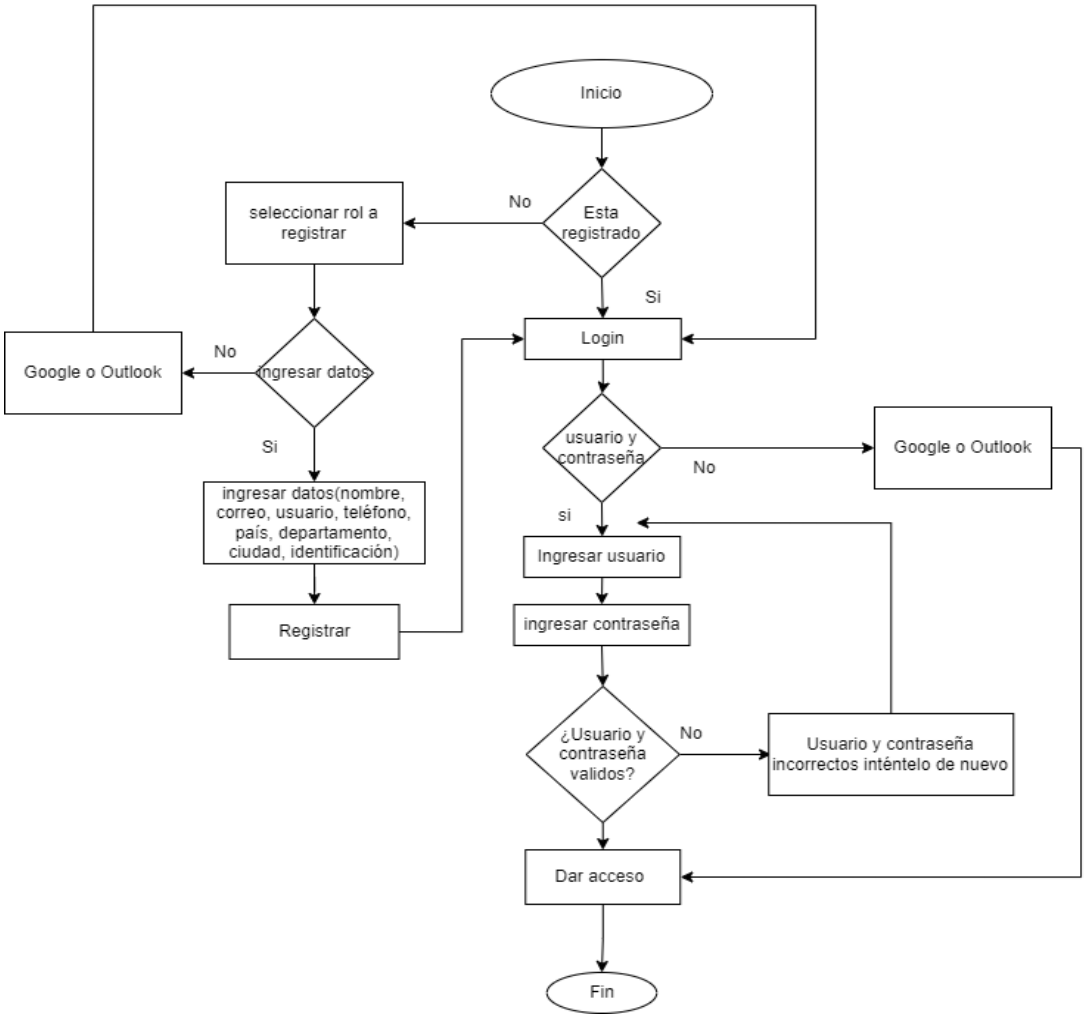


Figura 3. *Diagrama de flujo de autenticación*

Fuente: *Elaboración propia*

- **Diagrama de flujo para generación de reportes.**

El diagrama de flujo de reportes cuenta con los pasos necesarios para realizar un reporte. como lo son llenar filtros, generar reporte de acuerdo a los filtros, visualizarlos y descargarlos

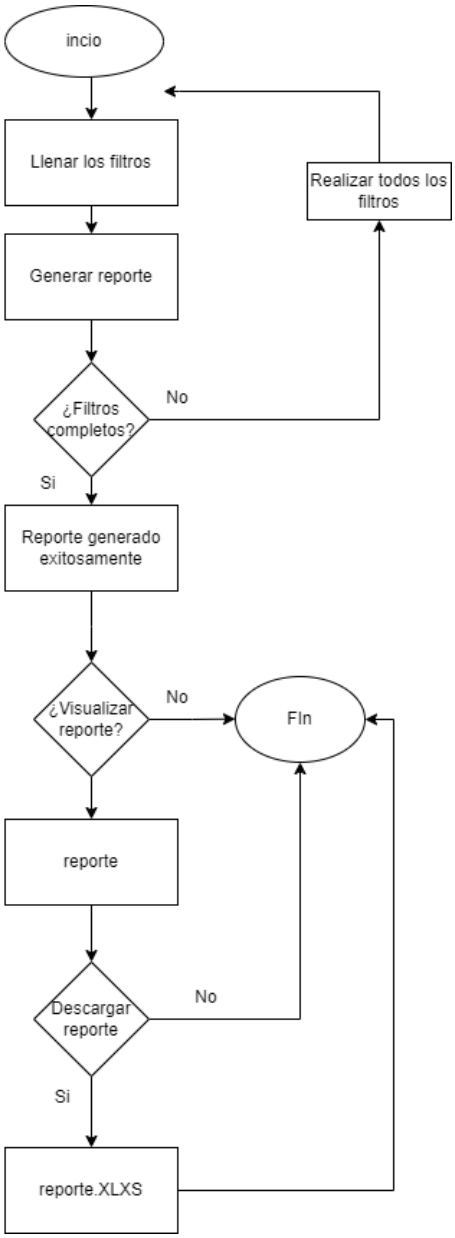


Figura 4. *Diagrama de flujo reportes*

Fuente: *Elaboración propia*

- **Diagrama de secuencia para autenticación.**

El siguiente diagrama de secuencia muestra la interacción de (usuario, sistema y servidor) en un proceso real sobre lo que se puede realizar y no. Este proceso fue ideal para el manejo de validaciones o captura de errores para admitir o denegar la confirmación de acceso a la aplicación web.

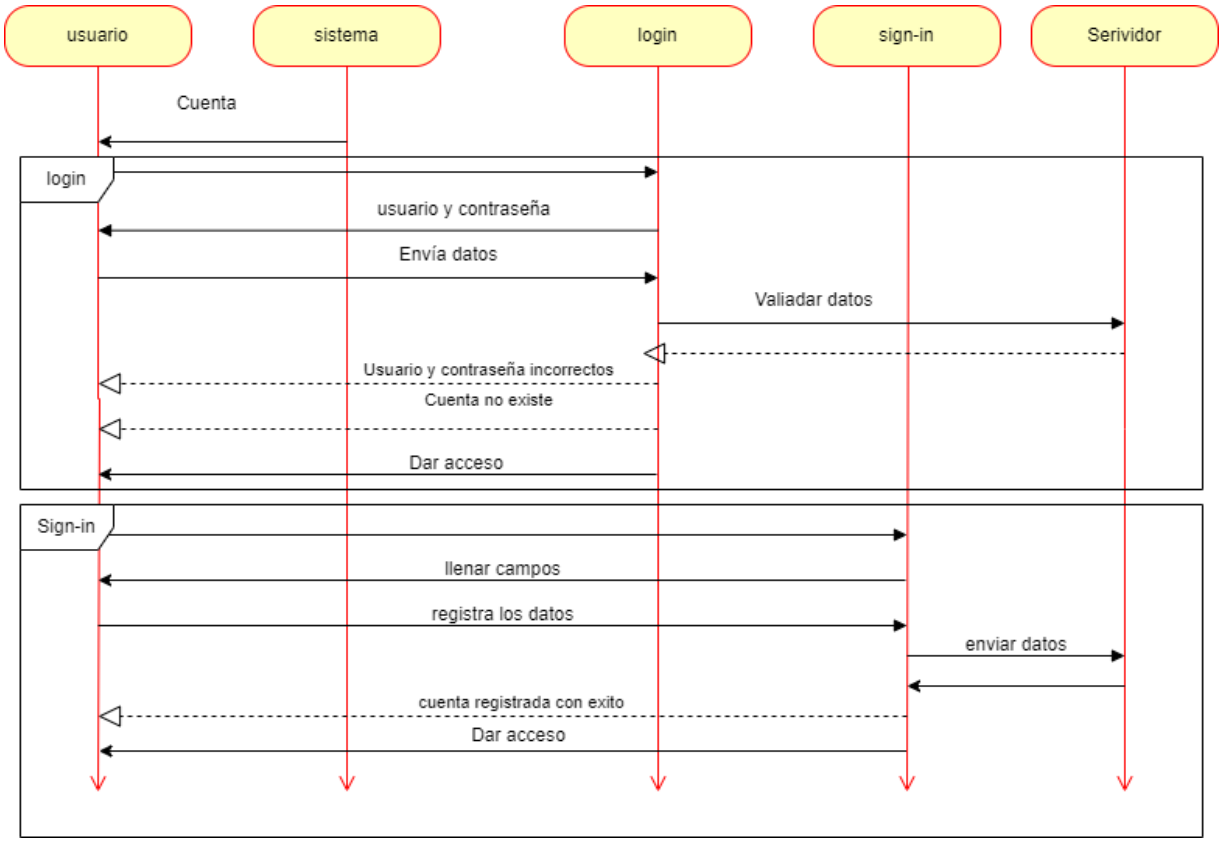


Figura 5. *Diagrama de secuencia autenticación*

Fuente: *Elaboración propia*

- **Diagrama de secuencia de reportes**

El diagrama de secuencia de reportes tiene como propósito definir los pasos claves de interacción con la aplicación web y generar un reporte adecuadamente para obtener la información necesaria.

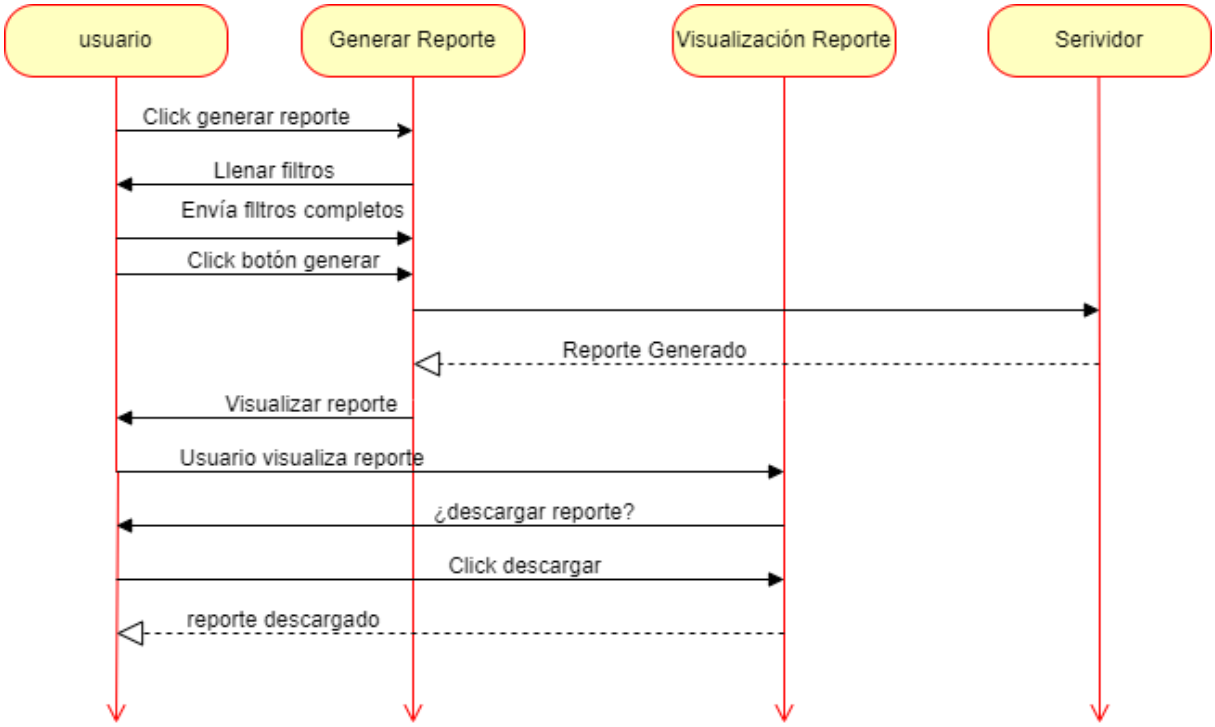


Figura 6. *Diagrama de secuencia reportes*

Fuente: *Elaboración propia*

2.2. Identificación y Clasificación de Componentes.

Para identificar los componentes necesarios luego de haber entendido el funcionamiento del sistema, en la semana 3 se realizó el diseño de mockups en balsamiq, la presentación del diseño no influyó demasiado debido que, Creatic proporcionaría diseños como guía para sus proyectos.

A continuación, se presenta la categoría de componentes establecidos para la aplicación web de software contable agropecuario basados en la metodología atomic design, específicamente en los módulos de autenticación y generación de reportes contables.

2.2.1. Átomos para el módulo de Autenticación.

Los átomos se definen como los elementos más básicos e indivisibles de la interfaz. En esta sección, se detallaron y analizaron los átomos que iban a ser utilizados en el frontend de la aplicación software contable.

Teniendo en cuenta los diagramas de la **figura 3, figura 4, figura 5, figura 6**, desarrollados en base a los requerimientos de Creatic se identificaron y diseñaron los componentes bases aprobados por el líder del proyecto.

Estos elementos fundamentales sirven como bloques de construcción esenciales para el diseño general de la interfaz. En este espacio definimos algunos componentes tomados como ejemplo de acuerdo a los diagramas previamente mencionados para autenticación:

- **Botón:** Botones como Registrar, Iniciar sesión y de inicio de sesión de Google, para envió de formulario o navegación entre páginas.
- **Input:** Campos esenciales en la interfaz para ingresar los datos de email y contraseña, etc.
- **Etiquetas:** Etiquetas que describe la información que debe ingresarse en los Input como “Email” y “Contraseña”, etc.
- **Mensajes de error:** resultado de validación para mostrar mensajes de “Usuario y contraseña incorrectos”, etc.
- **Título:** Esencial en la interfaz, para establecer la estructura y organización del contenido según el lugar donde se encuentre el usuario en la aplicación web.

Son elementos que individualmente no tienen sentido, pero que son los cimientos para la construcción de la aplicación web.

La siguiente tabla muestra algunos de los átomos que fueron empleados en la interfaz para autenticación, cabe resaltar que la tabla mostrada no abarca todos los átomos, si no que ofrece ejemplos representativos de algunos de ellos.

Entrada y etiqueta Nombre de usuario:	Botón Iniciar Sesión
Iconos	Titulo Iniciar Sesion
Mensajes de acierto Enlace de reestablecimiento de contraseña enviado a email	Imagen
Mensajes de error ⚠ Campo requerido Contraseña y/o usuario incorrectos	

Tabla 3. *Componentes atómicos autenticación*

Fuente*: Elaboración propia*

2.2.2. Átomos para el módulo de Reportes.

En este apartado, definimos los átomos para la generación de reportes, la mayoría de átomos utilizados para la interfaz de reportes son los mismo que se emplearon para el módulo de autenticación como se muestra en la **tabla 3**. Esto demostraba la eficiencia de Atomic Design como una metodología totalmente reutilizable, por ejemplo, en autenticación y generación de reportes es evidente el uso del átomo botón para el envió de datos de los formularios, la diferencia radica solo en su funcionalidad, En cuanto a su similitud se basa en que ambos están a la espera de un click para obtener un proceso diferente dentro de la aplicación web, además de tener los mismos estilos como el color de fondo y texto. En este espacio definimos algunos componentes tomados como ejemplo de acuerdo a los diagramas previamente mencionados para el módulo de reportes:

- **Input:** Campos para llenar los filtros.
- **Button:** Botones como "Generar Reporte", "Visualizar", y "Descargar".
- **Etiquetas:** Etiquetas que describe la información que debe ingresarse en los Input como “Código producto” y “Tipo de transacción”, etc.

Los componentes diferenciales que tiene el modulo de reportes son las grafica de barras y tortas, como se visualizan en la **tabla 4,** componentes que actúa como herramienta visual para el análisis de datos financiaros y toma de decisiones en la actividad contable por parte del usuario.

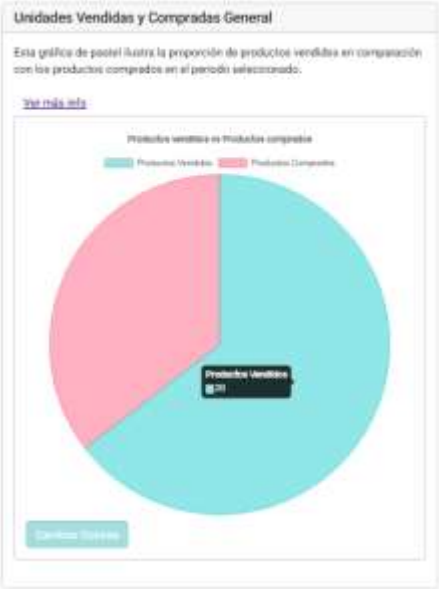
Grafica de barra	Grafica de torta
	

***Tabla 4**. Componentes atómicos reportes*

Elaboración: Fuente propia

2.2.3. Moléculas para el módulo de Autenticación

Las moléculas son uniones de átomos que trabajan juntos para cumplir una función específica. En la figura 7 se explora las moléculas integradas en el diseño de la interfaz. Cada molécula se construye mediante la composición de átomos para lograr una mayor complejidad y funcionalidad. Si unimos el átomo Label con el botón de icono de ocultar o mostrar contraseña con un input, tendremos nuestro espacio de ingresar una contraseña con la funcionalidad de ocultar o mostrar la contraseña ingresada, La molécula construida es clave para iniciar sesión, registrar usuario y restablecer contraseña.

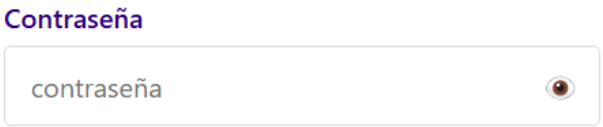


***Figura 7.** Componente molécula autenticación*

Fuente: Elaboración propia

2.2.4. Moléculas para el módulo de Reportes

Para el modo de reportes como requisito era crear el componente para filtrar y crear un reporte. Es por ello que FilterForm es un claro ejemplo de una molécula en la metodología atomic design debido a su estructura compuesta por varios átomos como input label Button y select que al integrar estos elementos básicos permite a los usuarios realizar acciones específicas dentro de la interfaz.

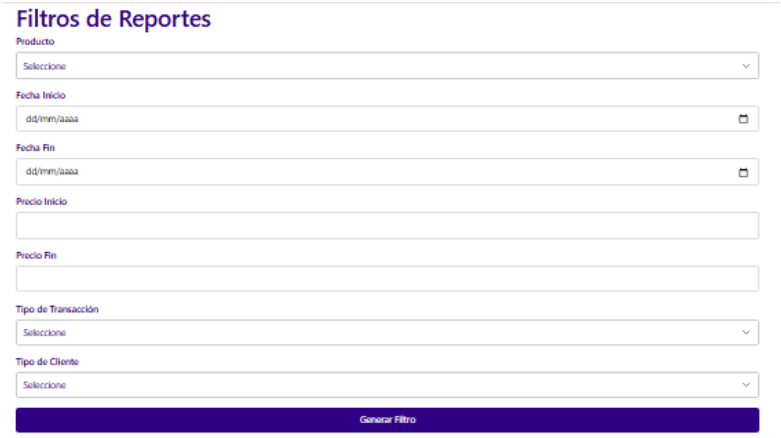


Figura 8. *Componente molécula reporte*

Fuente: *Elaboración propia*

2.2.5. Organismos para el módulo de Autenticación

Un organismo representa un componente complejo y funcional que combina átomos y moléculas para formar funciones más avanzadas y completas dentro de la interfaz. Estos bloques más coherentes y significativos permiten subdividir la página en secciones autónomas, capaces de manejar un flujo o una tarea completa dentro de la aplicación web.

En el módulo de autenticación, se utilizan átomos como título, etiqueta, entradas, selección, botón. Botón de Google, mensajes de error, mensajes de acierto, así como la molécula para ingresar y mostrar la contraseña (PasswordInput), con el objetivo de crear

componentes más complejos. Estos incluyen formularios funcionales de inicio de sesión, registro de rol de usuario y creación de cuenta, cumpliendo así con los objetivos y requerimientos del módulo.

En la **figura 9** se evidencia un sistema, que aunque esta compuesto por varios átomos, pasa hacer un organismo, ya que cumple con el objetivo de registrar un rol, integrando átomos como título y botón, esto demuestra la escalabilidad de la metodología según (Frost, 2016).

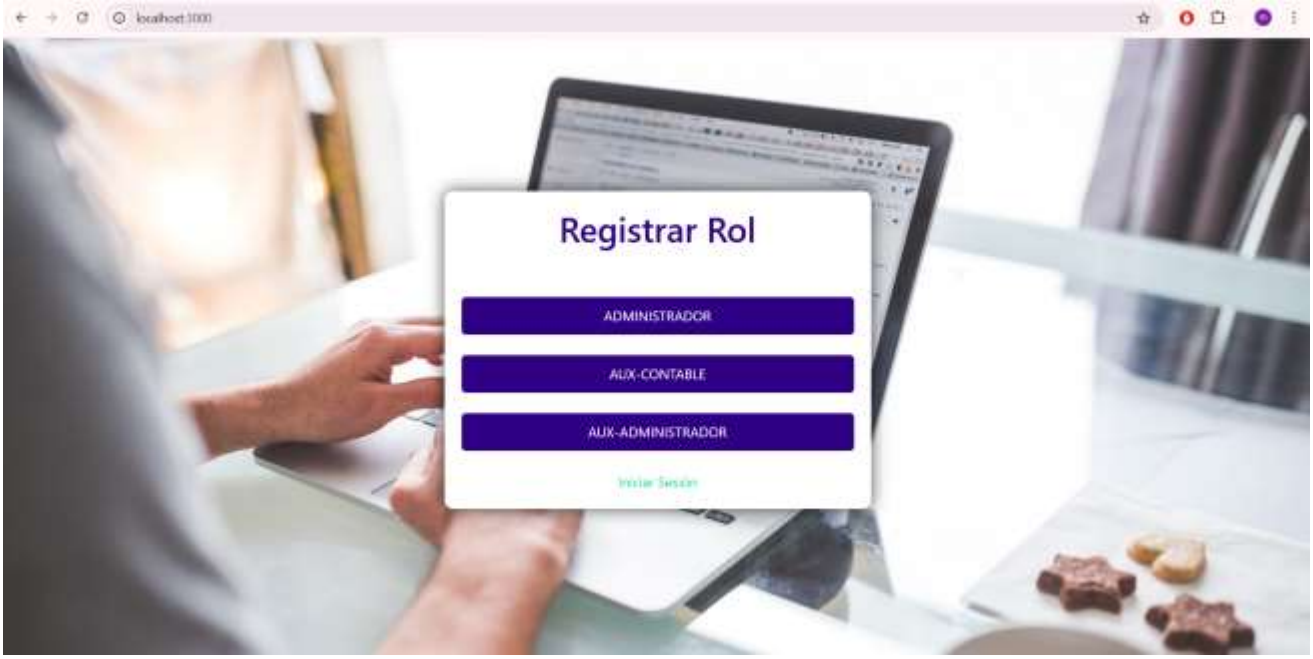


***Figura 9**. Organismo rol autenticación*

***Fuente**: Elaboración propia*

El inicio de sesión es un organismo, como se muestra en la **figura 10,** es un organismo ya que, esta integrado por átomos como título, input, etiquetas y botones y moléculas como entrada de contraseña de la **figura 7**.

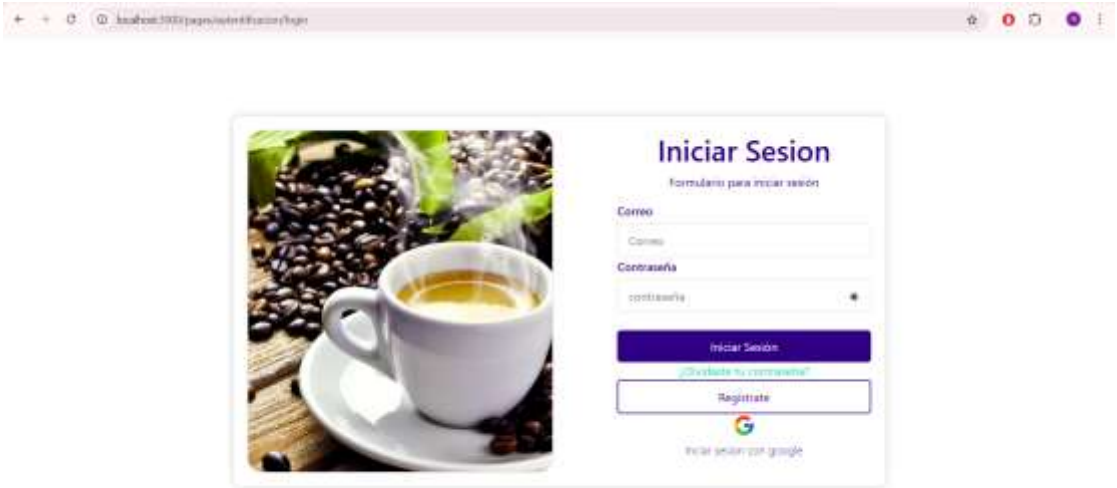


Figura 10. *Organismo inicio de sesión autenticación*

Fuente*: Elaboración propia*

El registro de usuario es un organismo, como se muestra en la **figura 11,** es un organismo ya que al igual que el inicio de sesión, está integrado por átomos como título, input, etiquetas y botones y moléculas como entrada de contraseña de la **figura 7**.

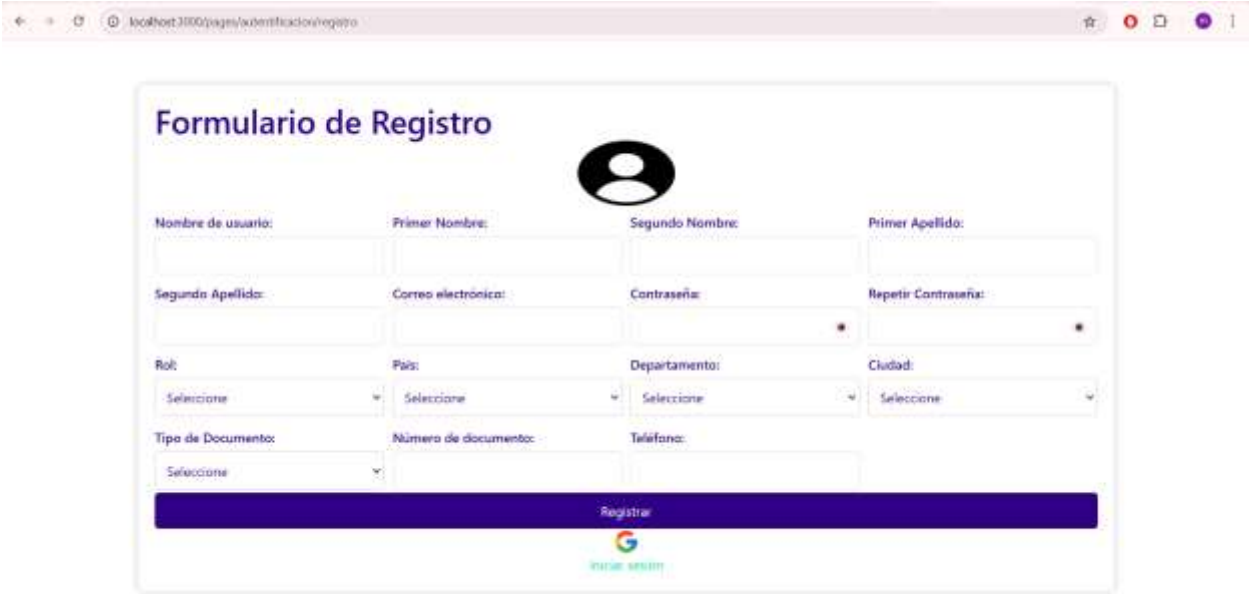


Figura 11. *Organismo registrar usuario autenticación*

Fuente: *Elaboración propia*

En la **figura 12** se evidencia un sistema, que, aunque está compuesto por átomos, pasa hacer un organismo, ya que cumple con el objetivo de registrar recuperar contraseña, integrando átomos como título y botón, esto demuestra la escalabilidad de la metodología según (Frost, 2016).

Figura 12. *Organismo recuperar contraseña autenticación*

Fuente: *Elaboración propia*

En la **figura 13** se evidencia un sistema, que, aunque está compuesto por átomos, pasa hacer un organismo, ya que cumple con el objetivo de restablecer contraseña, integrando átomos como título y botón, esto demuestra la escalabilidad de la metodología según (Frost, 2016).

***Figura 13**. Organismo reestablecer contraseña autenticación*

***Fuente:** Elaboración Propia*

2.2.6. Organismos para el módulo de Reportes

Para el módulo de reportes, se emplean átomos como PieChart, BarChart, Título, Label, Input, la molécula FilterForm, y los organismos Chart y ReporteTable. Estos componentes son esenciales para lograr la entrega final propuesta para el centro de desarrollo tecnológico Creatic.

En la **figura 14** se evidencia el organismo que es el encabezado del reporte mostrando un resumen al usuario general de su actividad financiera, donde se utilizó un átomo complejo como lo es la gráfica de torta, donde el usuario puede ver su información en porcentajes si pasa el cursor por encima de la torta.

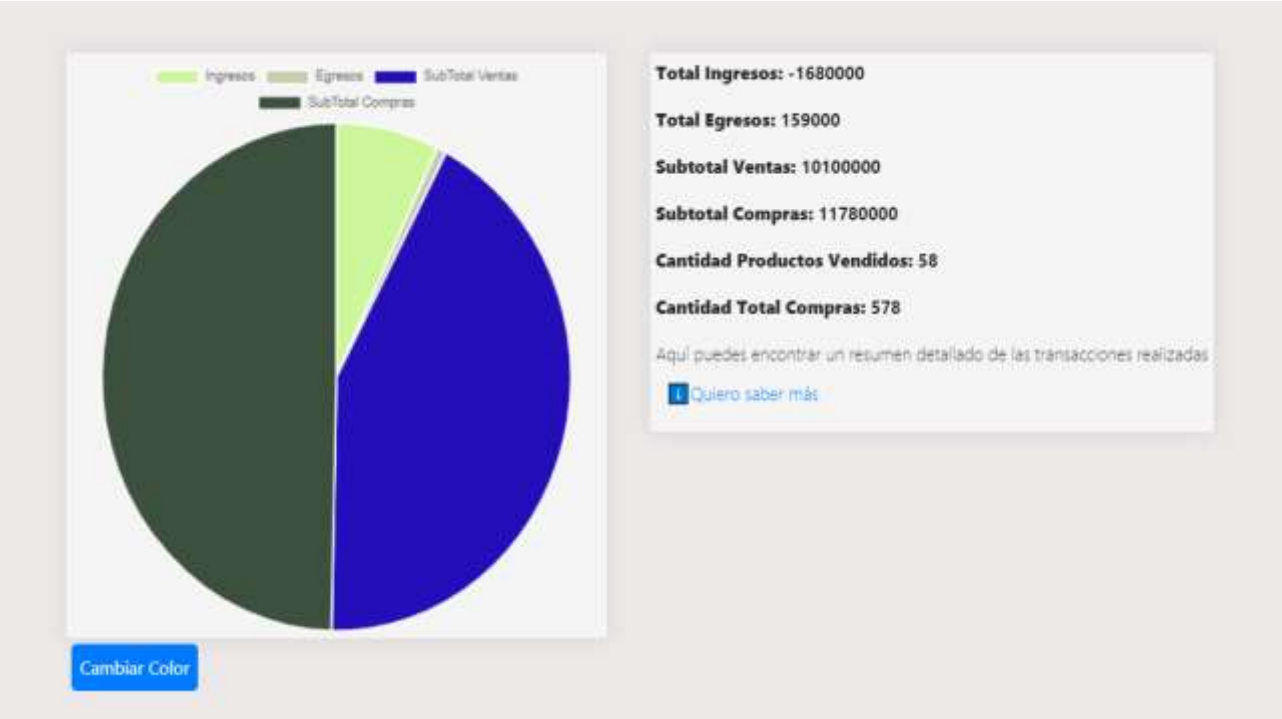


Figura 14. *Organismo resumen reportes*

Fuente: *Elaboración propia*

En la **figura 15** y **figura 16** se evidencia el organismo que es el cuerpo del reporte mostrando tablas con información transaccional, donde el usuario puede ver su información financiera, de acuerdo al tipo de transacción como venta o compra y el producto relacionado a la transacción, además de los costos o gastos que generó la transacción.

Detalle de transacciones

Esta tabla muestra un resumen detallado de las transacciones realizadas, organizadas por cada operación.

Ver menos

A continuación, te explico cada columna:

- **ID:** Es un identificador único para cada transacción. Puede haber varias filas con el mismo ID si la transacción involucra diferentes productos en un mismo evento.
- **Fecha:** Indica la fecha en la que se realizó la transacción.
- **Descripción:** Proporciona una breve descripción del tipo de productos o la naturaleza de la transacción, como la compra de fertilizantes o frutas.
- **Tipo de Transacción:** Especifica si la transacción fue una "compra" o una "venta".
- **Cliente:** Muestra el nombre del cliente o proveedor con el que se realizó la transacción.
- **Producto:** Indica el producto específico que fue comprado o vendido.
- **Cantidad:** Muestra la cantidad del producto que fue involucrado en la transacción.
- **Precio:** Indica el precio por unidad del producto para esa transacción.

ID	Fecha	Descripción	Tipo de Transacción	Cliente	Producto	Cantidad	Precio
1	16/8/2024	fertilizantes minerales	compra	tierragro	Fosfato	21	120000
1	16/8/2024	fertilizantes minerales	compra	tierragro	Urea	18	140000
2	16/8/2024	fertilizantes minerales	compra	tierragro	Fosfato	21	120000
2	16/8/2024	fertilizantes minerales	compra	tierragro	Urea	18	140000
3	16/8/2024	fertilizantes minerales	venta	tierragro	Fosfato	34	170000
3	16/8/2024	fertilizantes minerales	venta	tierragro	Urea	24	180000
4	19/8/2024	Frutas compra por kilogramo	compra	FrutasCentro	lulo	300	3000
4	19/8/2024	Frutas compra por kilogramo	compra	FrutasCentro	fresa	200	4000

Figura 15. *Organismo reporte detalle transacción.*

Fuente: *Elaboración propia*

Detalle de pagos

Esta tabla muestra un registro detallado de los pagos realizados en diferentes fechas, desglosando cada gasto asociado a una operación.

Ver menos

A continuación, te explico cada columna:

- **ID:** Este es un identificador único para cada pago registrado. Si un pago está relacionado con varios productos o conceptos, es posible que veas varias filas con el mismo ID.
- **Fecha:** Muestra la fecha exacta en la que se realizó el pago. Esto permite rastrear cuándo se llevaron a cabo las transacciones.
- **Descripción:** Proporciona una breve descripción del pago, como la compra de materiales o el pago de servicios. Esta columna ayuda a entender el propósito de cada gasto.
- **Tipo de Transacción:** Indica el tipo de transacción realizada, ya sea una compra, un servicio, u otra operación financiera.
- **Producto:** Especifica el producto o servicio por el cual se realizó el pago. Esto ayuda a relacionar los gastos de cada producto en su compra o venta.
- **Concepto:** Detalla el concepto específico del pago, como gastos de transporte, materiales, etc. Esta columna desglosa los motivos detrás de cada pago.
- **Monto:** Muestra el monto total asignado al concepto específico en esa transacción. Esta información te permite ver cuánto se pagó por cada detalle del concepto.
- **Total:** Indica el total del monto asociado con el concepto, lo que permite tener una visión global del costo total en la operación.

ID	Fecha	Descripción	Tipo de Transacción	Producto	Concepto	Monto	Total
1	16/8/2024	pago para cargar y descargar camion, comida compra en para trabajadores	compra	Fosfato	mano obra	1500	1500
1	16/8/2024	pago para cargar y descargar camion, comida compra en para trabajadores	compra	Urea	comida	500	2000
2	16/8/2024	pago para cargar y descargar camion, comida compra en para trabajadores	compra	Fosfato	mano obra	1500	1500
2	16/8/2024	pago para cargar y descargar camion, comida compra en para trabajadores	compra	Urea	comida	500	2000
3	16/8/2024	costo de envion de abono	venta	Fosfato	costo de venta	75000	75000
3	16/8/2024	costo de envion de abono	venta	Urea			
4	19/8/2024	Transporte de frutas	compra	lulo	trasnporte	80000	80000
4	19/8/2024	Transporte de frutas	compra	fresa			

Figura 16. *Organismo reporte detalles pagos*

Fuente: *Elaboración propia*

En la **figura 17** y **figura 18** se evidencia otra parte esencial del cuerpo del organismo del reporte mostrando graficas con información visual e interactiva para el usuario, como lo sigue siendo los tipos de transacción como venta o compra y el producto relacionado a la transacción, etc. Además, botón para descargar la información del reporte en pdf como se muestra en la **figura 18**.

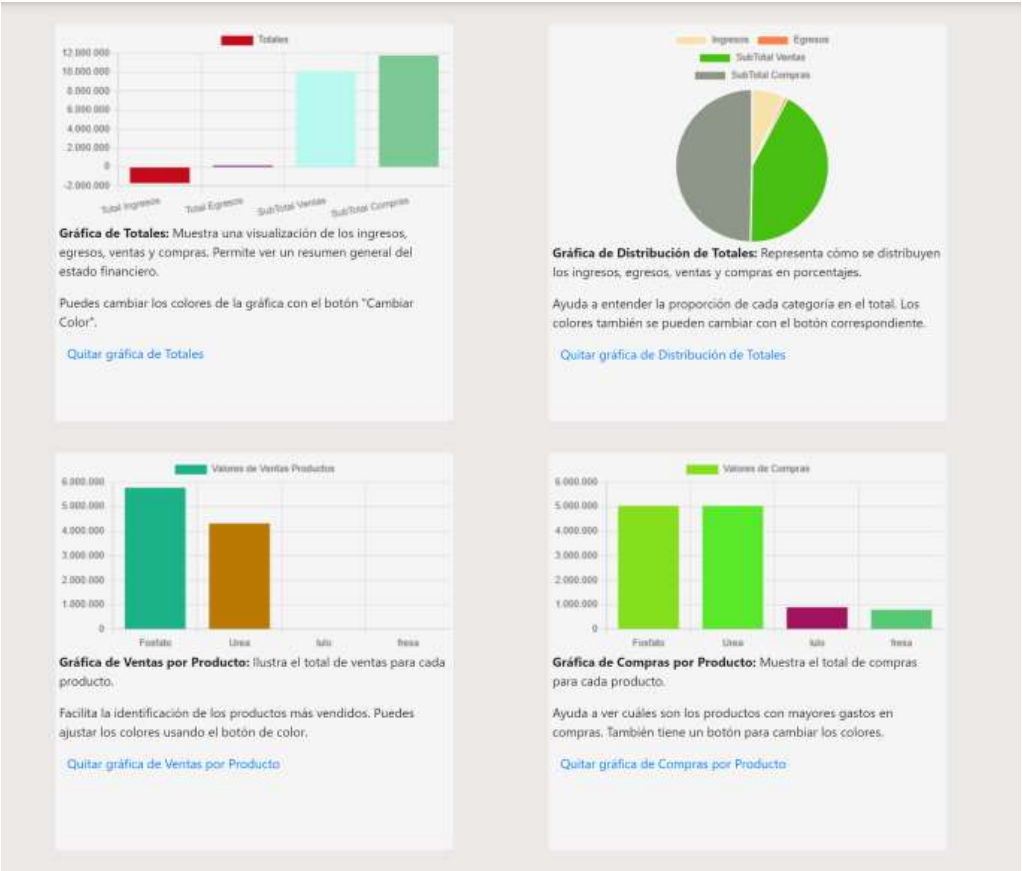


Figura 17. *Organismo reporte graficas*

Fuente: *Elaboración propia*

***Figura 18**. Organismo reporte grafica*

***Fuente:** Elaboración propia*

2.2.7. Exclusión en la construcción de Témplate

Durante la ejecución de este proyecto, se decidió saltarse la etapa de creación del template para avanzar directamente al diseño de la página web. La metodología Atomic Design no es rígida, sino que ofrece la flexibilidad necesaria para ajustarse a las características únicas de cada proyecto. En esta ocasión, se determinó que prescindir de la fase de template se alineaba mejor con los objetivos y requisitos del proyecto "Software contable agropecuario", requisitos propuestos por el Centro de Desarrollo Tecnológico Creatic.

La elección de omitir el template responde a la poca complejidad de los requerimientos brindados por Creatic y asegurando optimizar el proceso de desarrollo, al no emplear un nivel o principio de diseño atómico como el de template se lograba mayor velocidad y no crear código innecesario que afectará el rendimiento y mantenimiento. Este enfoque más directo en el diseño de la página facilita una respuesta más rápida a los cambios y permite iterar con mayor velocidad durante el desarrollo. Se priorizó la simplicidad y la eficacia en la implementación. Al eliminar la etapa del template, se simplifica el flujo de trabajo, lo que permite reducir pasos innecesarios y enfocarse directamente en la creación de las interfaces finales del módulo de autenticación y reportes.

Esta decisión demuestra la flexibilidad de la metodología Atomic Design al permitir su adaptación a las necesidades específicas de cada proyecto. En esta ocasión, elegir este enfoque más directo ha dado lugar a una implementación más ágil y eficiente.

2.2.8. Paginas (Pages)

Las páginas son el despliegue de todos los componentes visto anteriormente y donde se permite la interacción del usuario con la aplicación web. En esta jerarquía no existe diseño, ya que se centra exclusivamente en el resultado del desarrollo de la última etapa, que en este caso es el organismo.

Capítulo 3. Desarrollo del Proyecto Aplicando Atomic Design para los Módulos Autenticación y Generación y Visualización de Reportes

3. Arquitectura del Proyecto

En cuanto a la arquitectura interna del proyecto, se ha decidido seguir la convención de Next.js en su versión 14.2.1, la cual establece una estructura específica para desplegar páginas. En este enfoque, las páginas se almacenan en una carpeta designada, mientras que los elementos de Atomic Design se almacenan en la carpeta “Componentes”. Dentro de esta carpeta, se encuentran subcarpetas llamadas atoms, molecules y organisms, donde se encuentran archivos para los componentes compartidos en toda la aplicación trabajándose jerárquicamente y pueden ser reutilizados en toda la aplicación.

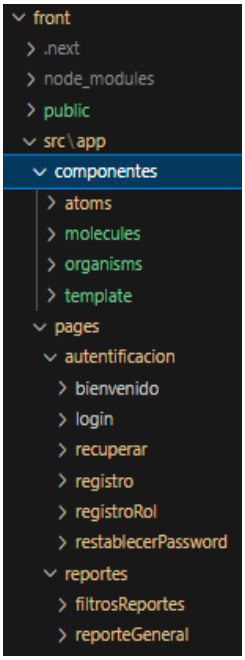


Figura 19. *Arquitectura atomic design*

Fuente: *Elaboración propia*

3.1. Estructura y desarrollo de átomos

Los átomos son los componentes más básicos del diseño y se encuentran en la parte inferior de la jerarquía como se muestra en la **figura 20**. Estos elementos representan los bloques de construcción fundamentales de la interfaz de usuario, como botones, iconos, campos de texto, etiquetas, entre otros. Al ser los componentes más simples y reutilizables, los átomos establecen una base sólida para construir componentes más complejos. Cada átomo tiene propiedades y estilos específicos que pueden ser utilizados en diferentes partes del proyecto, lo que promueve la consistencia visual y la eficiencia en el desarrollo.

En el proyecto, se desarrollaron los siguientes archivos para la construcción de átomos:

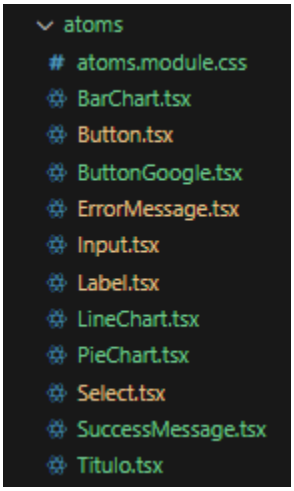


Figura 20. *Archivos atómicos desarrollo*

Fuente: *Elaboración propia*

Los Átomos que presentan y explican a continuación son componentes que se utilizan en los módulos de autentificación y reportes, ya que son fundamentales para mantener la consistencia en ambos módulos.

- **Button:** Un componente reutilizable para botones.

```
import React from 'react';

Codeium: Refactor | Explain
type ButtonProps = {
  type: 'button' | 'submit';
  onClick: () => void;
  children: React.ReactNode;
  style?: React.CSSProperties;
  className?: string;
};

Codeium: Refactor | Explain | Generate JSDoc | X
const Button: React.FC<ButtonProps> = ({ onClick, children, style, className }) => {
  return (
    <button onClick={onClick} style={style} className={className}>
      {children}
    </button>
  );
};

export default Button;
```

Figura 21. *Código botón desarrollo*

Fuente: *Elaboración propia*

El componente Button es un átomo dentro del enfoque de atomic design, lo que significa que es un elemento básico que puede ser utilizado en múltiples contextos dentro de la aplicación web. Al desarrollar un componente reutilizable se aseguraba que todos los botones tuvieran un estilo y comportamiento coherente, lo que mejoraría la experiencia de usuario y facilidad en el mantenimiento del código.

En el desarrollo del átomo como se muestra en el código anterior se puede resaltar la importancia de los props (propiedades) que permiten personalizar el componente según las necesidades específicas con las que se requerían y cumpliendo con los requisitos de Creatic, para ello se desarrollaron los estilos que se aplicarían en todas las partes de la aplicación web donde se llamará a este componente.

```
.button {
  background: #2F0084;
  width: 100%;
  padding: 10px;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
.button:hover {
  background: #1A003D ;
}
```

Figura 22. *Estilos botón*

Fuente: *Elaboración propia*

- **Button Google**: Componente reutilizable utilizado para las páginas de iniciar sesión y registrarse.

```
import React from 'react';
import styles from '../atoms.module.css';

interface ButtonGoogleProps {
  type: 'button' | 'submit';
  onClick?: () => void;
  style?: React.CSSProperties;
  className?: string;
}

const ButtonGoogle: React.FC<ButtonGoogleProps> = ({ type, onClick, className, style }) => {
  <button
    type={type}
    onClick={onClick}
    style={style}
    className={`${styles.buttonGoogle} ${className}`}
  >

  </button>
};

export default ButtonGoogle;
```

Figura 23*. Código botón de Google*

Fuente: *Elaboración propia*

Este código define un componente React llamado ButtonGoogle. Es un botón personalizable que se utiliza en la aplicación web. Sus principales características son:

Flexibilidad: Está configurado como un botón normal y de envío de formulario.
Interactividad: Permite asignar una función que se ejecutará cuando se haga clic en el botón.

Personalización visual: Acepta estilos CSS personalizados y clases adicionales para modificar su apariencia.

```
.buttonGoogle {
  width: 32px;
  height: 32px;
  padding: 0;
  border: none;
  background: url('/images/google-icon.png') no-repeat center center / cover;
  text-align: left;
}
```

Figura 24. *Estilos botón de Google*

Fuente: *Elaboración propia*

Reutilización: Al ser un componente separado, se reutiliza en diferentes partes de la aplicación web. En el módulo de autenticación, se emplea tanto en las páginas de inicio de sesión como en la de registro de usuario.

- **Mensajes de error:** Componente utilizado cuando surge una inconsistencia en la información proporcionada por el usuario y mensajes de advertencia de la aplicación web.

```
import React from 'react';
import styles from './atoms.module.css';

interface ErrorMessageProps {
  message: string;
}

const ErrorMessage: React.FC<ErrorMessageProps> = ({ message }) => {
  <p className={styles.errorMessage}>
    {message}
  </p>
};

export default ErrorMessage;
```

Figura 25. *Código mensajes de error*

Fuente: *Elaboración propia*

Este componente crea un elemento para mostrar mensajes de error en una aplicación React. Es simple, reutilizable y permite mantener un estilo consistente para los mensajes de error en toda la aplicación.

```
.errorMessage {
  color: red;
  font-size: 0.875rem;
  margin-bottom: 0.1rem; /*Espacio debajo del mensaje */
  text-align: left;
  font-weight: 600;
}
```

Figura 26. *Estilos códigos mensaje de error*

Fuente: *Elaboración propia*

- **Input:** Este componente crea un campo de entrada personalizable y reutilizable para formularios en React. Permite una fácil integración en diferentes partes de la aplicación, manteniendo una apariencia consistente y un comportamiento estandarizado.

```
import React from 'react';
import styles from './atoms.module.css';

Codeium: Refactor | Explain
interface InputProps {
  type: string;
  name?: string;
  value: string;
  placeholder?: string;
  onChange: (e: React.ChangeEvent<HTMLInputElement>) => void;
  id?: string;
  className?: string;
}

Codeium: Refactor | Explain | Generate JSDoc | X
const Input: React.FC<InputProps> = ({ type, name, value, placeholder, onChange, className, id }) => (
  <input
    type={type}
    name={name}
    value={value}
    placeholder={placeholder}
    onChange={onChange}
    id={id}
    className={`${styles.input} ${className}`}
  />
);

export default Input;
```

Figura 27. *Código campo de entrada*

Fuente: *Elaboración propia*

Define las propiedades que el componente puede recibir, incluyendo type, name, value, placeholder, onChange, id y className, además se establece la clase estilos para los estilos del componente y obtener el mismo diseño en todos los formularios de la aplicación web.

```
.input{
  width: 100%;
  height: 47px;
  padding: 15px;
  margin-bottom: 1px;
  border: 1px solid #ddd;
  border-radius: 0.25rem;
  font-size: 1rem;
}
```

Figura 28. *Estilos campo entrada de texto*

Fuente: Elaboración propia

- **Label:** El Label representa uno de los elementos más básicos y esenciales en la interfaz de usuario, especialmente en formularios de la aplicación web.Como átomo, es la unidad más pequeña y simple que mantiene su propia funcionalidad e identidad.

```
import React from 'react';
import styles from './atoms.module.css';

Codeium: Refactor | Explain
interface LabelProps {
  htmlFor: string;
  children: React.ReactNode;
  className?:string;
}

Codeium: Refactor | Explain | Generate JSDoc | X
const Label: React.FC<LabelProps> = ({ htmlFor, children, className }) => (
  <label htmlFor={htmlFor} className= {`${styles.label} ${className}`} >{children} </label>
);

export default Label;
```

Figura 29. *Código etiqueta*

Fuente: *Elaboración propia*

Reutilización: Al ser un componente atómico, la etiqueta fue reutilizada en múltiples contextos y formularios a lo largo de la aplicación web. Esto asegura consistencia en la presentación y comportamiento uniforme en todos los formularios de ambos módulos.

Composición: La etiqueta fue utilizada como bloque de construcción para crear componentes más complejos, se combinó fácilmente con otros átomos (como Input) para

formar moléculas y organismos más elaborados, ya que, proporcionaba contexto y descripción a los elementos de entrada, mejorando la usabilidad para todos los usuarios.

Personalización y flexibilidad: Aunque es un componente simple, la etiqueta ofrece opciones de personalización (estilos) que permitieron adaptarla a diferentes contextos de diseño sin perder su esencia funcional.

```
.label {
  display: block;
  font-weight: 500;
  margin-bottom: 0.5rem;
  text-align: left;
  color: #2f0084;
}
.label .labels-container-registro {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 24px;

}
```

Figura 30. *Estilos etiqueta*

Fuente: *Elaboración propia*

- **Select:** Este código define un componente React llamado Select.

```
import React from 'react';
import styles from './atoms.module.css';

Codeium: Refactor | Explain
interface SelectProps {
  name: string;
  value: string;
  onChange: (e: React.ChangeEvent<HTMLSelectElement>) => void;
  className?: string;
  options: string[];
}

Codeium: Refactor | Explain | Generate JSDoc | X
const Select: React.FC<SelectProps> = ({ name, value, onChange, className, options }) => {
  return (
    <select name={name} value={value} onChange={onChange} className={`${styles.select} ${className}`}>
      <option value="">Seleccione</option>
      {Array.isArray(options) && options.map(option => (
        <option key={option} value={option}>{option}</option>
      ))}
    </select>
  );
};

export default Select;
```

Figura 31. *Código para lista de opciones*

Fuente: *Elaboración propia*

Función: Se creó un elemento de selección desplegable reutilizable y actuó como un átomo en la arquitectura de diseño atómico.

Características clave: El componente se programó con el fin de obtener una lista de opciones, las cuales se renderizaban dinámicamente.

Es un componente básico y de suma importancia en el desarrollo del proyecto. Se combinó con otros átomos (como Label) para formar moléculas más complejas, como, por ejemplo: Listas de países, departamentos y ciudades. Promueve la consistencia y la reutilización en la interfaz de usuario en diferentes apartados de la interfaz del proyecto.

```
.select{
  width: 100%;
  height: 47px;
  padding: 10px;
  margin: 5px 0;
  box-sizing: border-box;
  border: 1px solid #ccc;
  border-radius: 0.25rem;
  color: #2f0084;
}
```

Figura 32*. Estilos lista de opciones*

Fuente: *Elaboración propia*

- **Título:** Este componente Titulo es un ejemplo, cómo incluso los elementos más básicos de una interfaz pueden beneficiarse del enfoque modular y reutilizable del diseño atómico en React. Proporciona una base sólida para la jerarquía visual y la estructura de contenido en la aplicación, contribuyendo a una interfaz de usuario coherente y fácil de mantener.

```
import React from 'react';
import styles from './atoms.module.css';

Codeium: Refactor | Explain
interface TituloProps {
  children: React.ReactNode;

}

Codeium: Refactor | Explain | Generate JSDoc | X
const Titulo: React.FC<TituloProps> = ({ children }) => (
  <h1 className={styles.titulo}>
    {children}
  </h1>
);

export default Titulo;
```

Figura 33. *Código titulo*

Fuente: *Elaboración propia*

Reutilización: Promueve la consistencia en la presentación de títulos principales de los módulos de autentificación y reportes de la aplicación web.

Facilitó cambios globales en el estilo de los títulos desde un solo lugar, generando simplicidad y mantenibilidad.

```
.titulo{
  color: #2F0084;
}
```

Figura 34*. Estilo titulo*

Fuente: *Elaboración propia*

A continuación, se presentan componentes atómicos que solamente se utilizaron en el módulo de reportes:

- **BarChart, LineChart y PieChart:** Componentes React que utilizan la biblioteca react-chartjs-2 para crear gráficos.

```
import React from 'react';
import { Bar } from 'react-chartjs-2';
import { Chart, registerables } from 'chart.js';

Chart.register(...registerables);

Codeium: Refactor | Explain
interface BarChartProps {
  data: any;
}

Codeium: Refactor | Explain | Generate JSDoc | X
const BarChart: React.FC<BarChartProps> = ({ data }) => {
  return <Bar data={data} />;
};

export default BarChart;
```

Figura 35. *Código grafica de barras*

Fuente: *Elaboración propia*

```
import React from 'react';
import { Pie } from 'react-chartjs-2';
import { Chart, registerables } from 'chart.js';

Chart.register(...registerables);

Codeium: Refactor | Explain
interface PieChartProps {
  data: any;
}

Codeium: Refactor | Explain | Generate JSDoc | X
const PieChart: React.FC<PieChartProps> = ({ data }) => {
  return <Pie data={data} />;
};

export default PieChart;
```

Figura 36. *Código grafica de torta*

Fuente: *Elaboración propia*

Función: Crear gráficos de barras, circulares y lineales reutilizables utilizando la biblioteca Chart.js. Actúan como átomos complejos o moléculas en la arquitectura de diseño atómico.

Características: Utiliza el componente Bar de react-chartjs-2 para renderizar el gráfico. Registra los componentes necesarios de Chart.js para su funcionamiento.

Propiedad data: Recibe los datos necesarios para construir el gráfico.
Diseño atómico: Aunque más complejo que un átomo simple, funciona como un bloque de construcción reutilizable para visualizaciones de datos.

Reutilización: Permite la creación consistente de gráficos de barras y circulares en toda la aplicación web. Facilita la implementación de visualizaciones de datos complejas de manera sencilla.

Flexibilidad: La propiedad 'data' permite personalizar el gráfico según las necesidades específicas de cada uso.

3.2. Estructuras y desarrollo de moléculas

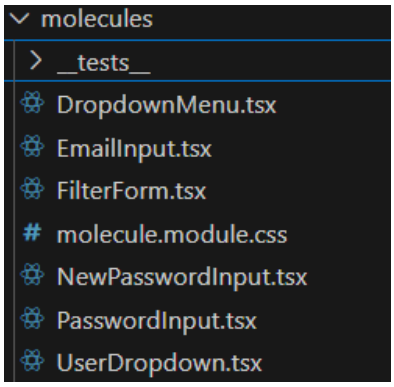


Figura 37. *Archivos de molécula desarrollo*

Fuente: *Elaboración propia*

FilterForm: El formulario para crear un reporte es un excelente ejemplo de una molécula en la metodología de diseño atómico. En este componente se combinaron varios átomos (Input, Label, y Button) para crear una unidad funcional más compleja.

Composición y estructura: Se Importo y utilizo componentes atómicos: Input, Label, y Button para crear un formulario completo con múltiples campos de entrada.

Propósito: El propósito fue desarrollar una molécula que proporcionara una interfaz de usuario para filtrar datos basados en varios criterios, demostrando cómo se pueden combinar componentes más simples para crear funcionalidades más complejas.

```
<div className="mb-3">
  <Label htmlFor="tipoCliente">Tipo de Cliente</Label>
  <Input id="tipoCliente" name="tipoCliente" type="text" value={formData.tipoCliente} onChange={handleChange} />
</div>
<Button type="submit">Generar Filtro</Button>
```

Figura 38. *Codigo para formulario de filtros para generar un reporte*

Fuente: *Elaboración propia*

PasswordInput: Este componente combina dos átomos (Input y Button) para crear una unidad funcional más compleja pero aún modular.

```
<div className={styles.passwordInput}>
  <Input
    type={showPassword ? 'text' : 'password'}
    name={name}
    value={value}
    onChange={onChange}
    placeholder={placeholder}
    className={styles.input}
  />
  <div className={styles.toggleContainer}>
  <Button
    type="button"
    onClick={toggleShowPassword}
    className={styles.toggleBoton}
  >
    {showPassword ? "🙈" : "👁"}
  </Button>
  </div>
</div>
```

Figura 39. *Código entrada de contraseña*

Fuente: *Elaboración propia*

Composición y Estructura: Se importo y utilizo componentes atómicos como Input y Button, para crear un campo de contraseña con funcionalidad de mostrar u ocultar contraseña, permitiendo alternar entre mostrar y ocultar la contraseña.

Visualización y Estilización: Se desarrollaron estilos modulares (styles from './molecule.module.css'). que aplica clases CSS específicas para estilizar el componente y sus partes como fueron necesarias, para brindar la uniformidad adecuada.

```
.passwordInput{
    display: flex;
    justify-content: flex-start;
    align-items: center;

}
```

Figura 40. *Estilos campo contraseña*

Fuente: *Elaboración propia*

Reutilización: En el caso de la aplicación web fue necesario utilizar el componente para el inicio de sesión y registro.

NewPasswordInput:

El componente NewPasswordInput fue otro claro ejemplo de la aplicación de la metodología de Atomic Design en el transcurso del proyecto.

```
return (
  <div >
    {emptyFieldError && <ErrorMessage message={emptyFieldError} />}

    <Label htmlFor="newPassword">Nueva Contraseña</Label>
    <PasswordInput
      name="newPassword"
      value={newPassword}
      showPassword={showNewPassword}
      onChange={(e: React.ChangeEvent<HTMLInputElement>) => setNewPassword(e.target.value)}
      toggleShowPassword={() => setShowNewPassword(!showNewPassword)}
    />

    <Label htmlFor="confirmPassword">Confirmar Contraseña</Label>
    <PasswordInput
      name="confirmPassword"
      value={confirmPassword}
      showPassword={showConfirmPassword}
      onChange={(e: React.ChangeEvent<HTMLInputElement>) => setConfirmPassword(e.target.value)}
      toggleShowPassword={() => setShowConfirmPassword(!showConfirmPassword)}
    />
  </div>
);
```

Figura 41. *Código para generar una nueva contraseña*

Fuente: *Elaboración propia*

NewPasswordInput en sí es una molécula más compleja, donde se combinaron átomos como: (Label, ErrorMessage) y otras moléculas como (PasswordInput) para crear una unidad funcional más elaborada.

Reutilización: El componente PasswordInput se reutilizo dos veces dentro de NewPasswordInput, demostrando la eficiencia del diseño atómico.

EmailInput: El componente EmailInput es una molécula, en la cual se combinaron algunos átomos comó (Label, Input y ErrorMessage) para crear una unidad funcional, más avanzada y enfocada en la captura de correos electrónicos.

```
const EmailInput: React.FC<EmailInputProps> = ({ email, setEmail, emptyFieldError }) => (
  <div className="container-labels-login">
    {emptyFieldError && <ErrorMessage message={emptyFieldError} />}
    <Label htmlFor="email">Correo</Label>
    <Input
      id="email"
      type="email"
      value={email}
      placeholder="Correo"
      onChange={(e) => setEmail(e.target.value)}
    />
  </div>
);
```

Figura 42. *Código entrada de correos electrónicos*

Fuente: *Elaboración propia*

Este componente demuestra cómo los átomos simples se pueden combinar para crear funcionalidades más avanzadas, como la validación de entradas y la gestión de errores en tiempo real.

3.3. Estructura y desarrollo de organismos

Para el módulo de Autentificación se desarrollan los siguientes organismos finales para cumplir con los objetivos del proyecto software contable agropecuario en el módulo autenticación.

LoginForm: El LoginForm es un organismo que ejemplifica la composición avanzada de componentes en la arquitectura de diseño atómico. Este componente integra múltiples átomos y moléculas para crear una interfaz de usuario compleja y funcional, demostrando cómo los principios del diseño atómico se aplican a niveles superiores de composición.

Figura 43. *Formulario de inicio de sesión*

Fuente: *Elaboración propia*

Características principales: Se Integraron múltiples átomos como (Label, Input, Button, ErrorMessage, Título, ButtonGoogle) y molécula (PasswordInput y EmailInput), cabe destacar que los componentes mencionados deben ser importados para su utilización, como se muestra en la **figura 43**.

Para la navegación en la aplicación, se utiliza el componente Link de Next.js, lo que permite una transición fluida entre las distintas secciones. Esta implementación demuestra cómo la combinación de elementos simples puede dar lugar a una interfaz coherente y compleja. En el desarrollo del formulario, se emplearon múltiples estados, como email, password, error, emptyFieldsError y showPassword, para gestionar la lógica de manera eficiente. Además, se implementó el hook useEffect de React, que resultó ser útil para limpiar los errores en pantalla después de un tiempo determinado, mejorando así la experiencia del usuario. En cuanto a la interacción con la API, se maneja la lógica de autenticación a través de una llamada a una API externa, gestionando diferentes escenarios de respuesta, tanto de éxito como de error, y actualizando la interfaz en consecuencia para reflejar el estado actual del proceso de autenticación.

La navegación y enrutamiento en la aplicación se realiza mediante el uso del hook useRouter de Next.js, lo que permite una navegación programática eficiente, especialmente en el login, donde se gestionan rutas como registro y recuperación de contraseña. En cuanto a la validación de formularios, se implementó un sistema que verifica campos vacíos, proporcionando mensajes de error específicos y utilizando los estilos del átomo ErrorMessage previamente mencionado para mejorar la claridad visual. Además, se incluyó la opción de iniciar sesión con Google a través del átomo BottonGoogle, lo que demuestra una efectiva integración con servicios externos. Para asegurar una estilización modular, se emplean CSS modules, específicamente login.module.css, lo que permite encapsular estilos y evitar conflictos entre diferentes componentes de la interfaz.

RoleSelection: El componente RoleSelection está compuesto por varios átomos, como Título, Button y Link. Cada uno de estos átomos tiene su propia función simple y específica, pero cuando se combinan, forman una unidad funcional más compleja. Un organismo en Atomic Design es precisamente eso: una agrupación de átomos (y en algunos casos, moléculas) que trabajan juntos para cumplir un propósito más significativo en la interfaz de usuario.

```
const RoleSelection: FC<RoleSelectionProps> = ({ onRoleSelect }) => {
  <div className={styles.rolBox}>
    <Titulo> Registrar Rol</Titulo>
    <br />
    <br />
    <Button type='submit' onClick={() => onRoleSelect('/pages/autentificacion/registro', 'ADMINISTRADOR')}>
      ADMINISTRADOR
    </Button>
    <br />
    <br />
    <Button type='submit' onClick={() => onRoleSelect('/pages/autentificacion/registro', 'AUX-CONTABLE')}>
      AUX-CONTABLE
    </Button>
    <br />
    <br />
    <Button type='submit' onClick={() => onRoleSelect('/pages/autentificacion/registro', 'AUX-ADMINISTRADOR')}>
      AUX-ADMINISTRADOR
    </Button>
    <br />
    <br />
    <div className={styles.loginLinks}>
      <Link href="/pages/autentificacion/login">Iniciar Sesión</Link>
    </div>
  </div>
};
```

Figura 44. *Código selección de rol*

Fuente: *Elaboración propia*

RoleSelection es un organismo, aunque es el componente más sencillo, encapsula una funcionalidad específica y significativa que combina múltiples átomos en una unidad funcional independiente. Su capacidad para manejar la interacción del usuario, tomar decisiones y realizar acciones dentro de la interfaz, lo clasificó claramente como un organismo en la metodología Atomic Design.

RegisterForm: Es un organismo que refleja la composición avanzada de componentes dentro de la metodología de diseño atómico. Este componente combina varios átomos y moléculas para construir una interfaz de usuario robusta y funcional, demostrando cómo los principios del diseño atómico pueden escalar para manejar casos más complejos de interacción.

```
import React, { useState, useEffect } from 'react';
import { useRouter } from 'next/navigation';
import Link from 'next/link';
import Modal from 'react-modal';
import axios from 'axios';
import Input from '../atoms/Input';
import Select from '../atoms/Select';
import Label from '../atoms/Label';
import Button from '../atoms/Button';
import ErrorMessage from '../atoms/ErrorMessage';
import PasswordInput from '../molecules/PasswordInput';
import Titulo from '../atoms/Titulo'
import '@/app/globals.css';
import styles from './estilos-organismos/registro.module.css';
```

```
<div className={styles.registroBox}>
  <Titulo>Formulario de Registro <img src="/images/FondoRegister.png" alt="Login Image" className={styles.IconoRegistro}/>
  </Titulo>
  <div>
    <div className={styles.labelsRegistro}>
      <div className="labels-box-registro">

        {emptyFieldsError.username && <ErrorMessage message={emptyFieldsError.username} />}
        <Label htmlFor="username">Nombre de usuario:</Label>
        <Input type="text" name="username" value={formData.username} onChange={handleChange} className="labels-input-listas" />
      </div>
```

(property) InputProps.className?: string | undefined

```
      <div className="labels-box-registro">
        {emptyFieldsError.telefono && <ErrorMessage message={emptyFieldsError.telefono} />}
        {inputErrors.telefono && <ErrorMessage message={inputErrors.telefono} />}
        <Label htmlFor="telefono">Teléfono:</Label>
        <Input type="text" name="telefono" value={formData.telefono} onChange={handleChange} className="labels-input-listas" />
      </div>

    </div>
    <div className="button-container">
      {error && <ErrorMessage message={error} />}
      <Button type='submit'  onClick={handleRegistration}>Registrar</Button>
    </div>
```

***Figura 45**. Código formulario registro*

***Fuente:** Elaboración propia*

Características principales: En la realización del organismo se integró múltiples átomos (Label, Input, Button, ErrorMessage, Titulo) y moléculas (PasswordInput, Select), lo que permitió la creación de un formulario de registro completo, donde cada componente está diseñado para ser altamente reutilizable, garantizando coherencia y eficiencia en la construcción de la interfaz. El formulario utiliza varios estados (email, password, confirmPassword, role, error, emptyFieldsError, showPassword) además se realizó llamadas a una API externa para gestionar el registro y actualizando la interfaz según los diversos escenarios de respuesta, ya sean exitosos o errores. Adicionalmente se incorporó la funcionalidad de selección de roles que se comunica con el backend para asignar permisos específicos que serían desarrollados por otro miembro del proyecto. Todo esto se complementa con un sistema de validación riguroso que asegura que los

campos obligatorios sean completados correctamente, utilizando mensajes de error específicos a través del átomo ErrorMessage.

El organismo fue desarrollado para utilizar estilos CSS modules (register.module.css) para encapsular los estilos, garantizando que estos no entren en conflicto con otros estilos en la aplicación y manteniendo una apariencia consistente.

Aunque se diseñado específicamente para el registro de usuarios, su estructura modular permite su reutilización y adaptación en diferentes contextos donde se necesite la creación de cuentas o la gestión de perfiles.

RecoverForm: Es un organismo clave dentro de la arquitectura de diseño atómico, enfocado en la funcionalidad de recuperación de contraseñas en la aplicación web. Este componente combina átomos y moléculas de manera estratégica para ofrecer una interfaz sencilla pero eficaz, centrada en la experiencia del usuario y en la seguridad del proceso de recuperación.

```
'use client';
import React, { useState, useEffect } from 'react';
import Link from 'next/link';
import { useRouter } from 'next/navigation';
import EmailInput from '../molecules/EmailInput';
import Button from '../atoms/Button';
import ErrorMessage from '../atoms/ErrorMessage';
import SuccessMessage from '../atoms/SuccessMessage';
import '@/app/globals.css';
import styles from './estilos-organismos/recuperar.module.css';
```

```
<div className={styles.containerRecuperar}>
  <div className={styles.RecuperarBox}>
    <h2>Recuperar Contraseña</h2>
    <p>Introduce tu correo electrónico para recibir un enlace de recuperación</p>
    <EmailInput email={email} setEmail={setEmail} emptyFieldError={emptyFieldError} />
    <Button type='submit' onClick={handleRecover}>Enviar Enlace de Recuperación</Button>
    {error && <ErrorMessage message={error} />}
    {successMessage && <SuccessMessage message={successMessage} />}
    <div className={styles.recuperarLinks}>
    <Link  href="/pages/autentificacion/login">Volver</Link>
    </div>
  </div>
</div>
```

Figura 46. *Código formulario de recuperación de contraseña*

Fuente: *Elaboración propia*

Características principales: Se integró varios componentes atómicos como Button, ErrorMessage, y SuccessMessage, además de la molécula EmailInput. Esta combinación permite la creación de un formulario que guía al usuario a través del proceso de recuperación de contraseña de manera intuitiva.

La utilización de estilos modulares con CSS modules (en recuperar.module.css) garantiza que el diseño sea consistente, evitando conflictos con otros estilos en la aplicación.

Al igual que otros organismos ya menciondos RecoverForm maneja múltiples estados (email, error, successMessage, emptyFieldError) para controlar la lógica del formulario, asegurando que se responda de manera adecuada a las acciones del usuario.

La validación de campos vacíos se gestiona de forma dinámica. Si el campo de correo electrónico está vacío, se muestra un mensaje de error específico (Campo requerido), mejorando la usabilidad y la claridad del formulario.

La lógica del componente incluye una interacción con una API externa para enviar la solicitud de recuperación de contraseña. Si la solicitud es exitosa, el componente muestra un mensaje de éxito y redirige al usuario a la página de inicio de sesión tras un breve retraso.

En caso de un error en la respuesta de la API, el componente captura el mensaje de error proporcionado y lo muestra al usuario, garantizando que se ofrezca una retroalimentación clara y útil en todo momento.

Así mismo se utilizó useEffect para gestionar la eliminación automática de mensajes de error relacionados con campos vacíos, configurando un temporizador que limpia el error después de dos segundos. Esto evita que los errores persistan innecesariamente en la interfaz, mejorando la experiencia del usuario.

También incluye un enlace (Link) que permite al usuario regresar a la página de inicio de sesión de forma manual, mejorando la accesibilidad y la facilidad de uso.

El proceso de recuperación de contraseña implementado en este formulario incluye la comunicación segura con un servidor a través de fetch, asegurando que los datos sensibles, como el correo electrónico, se manejen de manera segura y eficiente.

ResetPasswordForm: Es un organismo que se encarga de gestionar el proceso de restablecimiento de contraseñas en la aplicación web. Este componente utiliza una combinación de átomos y moléculas para ofrecer una experiencia de usuario segura y eficiente, asegurando que el proceso de restablecimiento de contraseñas sea claro y fácil de entender para el usuario.

Características principales: El ResetPasswordForm integra componentes atómicos como Button, ErrorMessage, y SuccessMessage, junto con la molécula NewPasswordInput. Esta combinación permitió crear un formulario de ingreso de contraseña y confirmación de una nueva contraseña.

Los estilos globales importados (globals.css) y las propiedades de estilo en línea aseguran que la interfaz sea consistente y visualmente atractiva.

Se manejan múltiples estados (newPassword, confirmPassword, error, successMessage, emptyFieldError) para controlar la lógica del formulario, garantizando que el usuario reciba retroalimentación inmediata sobre las acciones realizadas.

Se implementa una validación robusta para asegurar que ambos campos de contraseña sean completados y que las contraseñas coincidan, mostrando mensajes de error específicos en caso de que estas condiciones no se cumplan.

El componente maneja la interacción con una API externa para realizar la solicitud de restablecimiento de contraseña. Si la operación es exitosa, el usuario es notificado y redirigido automáticamente a la página de inicio de sesión después de un breve período.

En caso de error en la solicitud, el componente captura y muestra el mensaje de error, garantizando una experiencia de usuario informada y transparente.

Validación de formularios: La validación de campos vacíos y la coincidencia de contraseñas se realiza en el cliente, proporcionando mensajes de error inmediatos cuando el usuario no cumple con los requisitos establecidos, como contraseñas no coincidentes o campos incompletos.

En el desarrollo de este organismo de recuperación se utilizando useRouter y useSearchParams de Next.js, el ResetPasswordForm maneja la lógica de navegación y la extracción de parámetros de búsqueda de la URL, como el token de recuperación y el correo electrónico del usuario, asegurando que los datos necesarios para el restablecimiento estén disponibles.

El formulario de restablecimiento de contraseña asegura que las nuevas contraseñas ingresadas sean manejadas de manera segura, enviándolas a través de una solicitud POST a la API con los encabezados y el cuerpo de la solicitud configurados correctamente, el uso de un token de seguridad extraído de los parámetros de la URL asegura que solo los usuarios autorizados puedan realizar el restablecimiento de su contraseña.

Chart: Es un organismo dentro de la arquitectura de diseño atómico que se encarga de visualizar datos financieros y de productos en la aplicación web. Este componente combina átomos y funciones de utilidad para ofrecer una representación gráfica clara y efectiva de diferentes métricas, como ingresos, egresos, ventas, y compras.

```
import React, { useState } from 'react';
import BarChart from '../atoms/BarChart';
import PieChart from '../atoms/PieChart';
import styles from './estilos-organismos/reporte.module.css';
```

```
        <div>
          <BarChart data={barData} />
          <p>
            <strong>Gráfica de Totales:</strong> Muestra una visualización de los ingresos, egresos, ventas y compras. Permite ver un resumen general del estado financiero.
          </p>
          {showBarChart && (
            <p>Puedes cambiar los colores de la gráfica con el botón "Cambiar Color".</p>
          )}
        </div>
      )}
      <span
        className={styles.more}
        onClick={() => setShowBarChart(!showBarChart)}
      >
        {showBarChart ? 'Quitar gráfica de Totales' : 'Ver gráfica de Totales'}
      </span>
      {showBarChart && (
        <button className={styles.colorButton} onClick={handleBarColorChange}>
          Cambiar Color
        </button>
      )}
    </div>

    <div className={styles.containerCharts}>
      {showPieChart && (
        <div>
          <PieChart data={pieData} />
          <p>
            <strong>Gráfica de Distribución de Totales:</strong> Representa cómo se distribuyen los ingresos, egresos, ventas y compras en porcentajes.
          </p>
```

***Figura 47**. Código organismo de graficas*

***Fuente:** Elaboración propia*

Características principales: Átomos Utilizados: El componente Charts se compone de átomos como BarChart y PieChart. Estos átomos permiten la representación de datos en forma de gráficos de barras y gráficos circulares respectivamente.

Estilos Personalizados: Se importan estilos específicos (reporte.module.css) que aseguran que la interfaz gráfica sea consistente con el resto de la aplicación, manteniendo una apariencia profesional y visualmente atractiva.

El componente recibe un conjunto de datos (reportData) a través de sus props (propiedades), los cuales incluyen totales de ingresos, egresos, ventas, compras, y una lista detallada de productos con sus respectivas ventas y compras. Estos datos son procesados y visualizados dinámicamente en las gráficas.

BarChartTotals: Este componente se creó con la finalidad de utilizarlo individualmente para mostrar gráfica conocida como torta en la página principal del reporte, como se puede apreciar contiene la misma estructura del componente **Chart**, solamente se utiliza dos importaciones.

```
import PieChart from '../atoms/PieChart';
import styles from './estilos-organismos/reporte.module.css';
```

```
<div>
  <div className={styles.containerCharts}>
    {showPieChart && <PieChart data={pieData} />}
    {showPieChart && (
      <button className={styles.colorButton} onClick={handlePieColorChange}>
        Cambiar Color
      </button>
    )}
  </div>
</div>
```

Figura 48. *Código grafico de barras totales*

Fuente: *Elaboración propia*

Capítulo 4. Evaluación De Metodología Atomic Design

El objetivo principal de este apartado es presentar los resultados obtenidos en el desarrollo e implementación de la aplicación web, evaluando específicamente la estructura y funcionalidad de los módulos de autenticación y generación de reportes siguiendo la metodología Atomic design. Además, se examina cómo esta metodología permitió el desarrollo de componentes reutilizables, tiempo de renderizado de los componentes, funcionalidad de cada componente y una jerarquía bien definida para el cumplimiento de los requisitos propuestos para los módulos.

4. Pruebas y Evaluación de la Aplicación Web.

4.1. Evaluación de la Aplicabilidad de Atomic Design

La implementación de Atomic Design en este proyecto no sólo alcanzó los objetivos que se plantearon en los requisitos presentados por Creatic, sino que demostró ser una metodología robusta y eficiente para el desarrollo de aplicaciones modulares en Next.js, la reutilización de componentes fue clave en la construcción de una interfaz coherente y escalable.

4.1.1. Módulo de Autenticación

Componentes atómicos: Los componentes básicos o átomos como (Button, Input, Label y ErrorMessage) fueron desarrollados y configurados para ser reutilizables en múltiples secciones de la aplicación cada uno de estos componentes cumple una función elemental, como recibir entradas del usuario o mostrar mensajes de error.

Evaluación: la reutilización de estos átomos se confirmó en el módulo de autenticación, donde tanto el formulario de inicio de sesión como el de registro comparten las mismas

entradas de texto y botones. Esto no solo ahorró tiempo en el desarrollo, sino que aseguró consistencia en el estilo y funcionalidad a través de toda la interfaz de usuario.

Resultado: Los átomos funcionaron correctamente en distintos contextos y cualquier cambio en el átomo fue inmediatamente reflejado en todas las instancias del sistema facilitando el mantenimiento y garantizando una estética uniforme en la interfaz.

Componentes Moleculares: En el módulo de autenticación, Los átomos se combinan para crear moléculas más complejas, como el campo de contraseña (que incluye un campo de entrada y un botón para mostrar u ocultar el texto de la contraseña) y así mismo la combinación de entradas de textos permitieron realizar formularios uniformes.

Evaluación: Las moléculas demostraron un alto nivel de reutilización y coherencia. PasswordInput (contraseña) fue usado en el formulario de inicio de sesión y en el registro, reduciendo el tiempo de desarrollo y permitiendo cambios centralizados en la funcionalidad y estilo.

Resultado: La funcionalidad de las moléculas cumplió con los requisitos de usabilidad y respuesta visual mostrando consistencia una experiencia intuitiva y eficaz para el usuario.

Componentes Organismos: Los organismos, como formulario de inicio de sesión y registro, integraron moléculas y átomos en una estructura completa que facilitó la captura de credenciales del usuario y la validación de los datos.

Evaluación: Los organismos lograron una experiencia de usuario óptima, fueron fácilmente configurables y adaptables, y se ajustaron con éxito a los requerimientos de autenticación.

Resultado: La funcionalidad del módulo de autenticación web fue completamente satisfactoria; el sistema gestionó tanto el inicio de sesión como el registro sin problemas y respondió adecuadamente errores de entrada.

4.1.2. Módulo de Generación y Visualización de Reportes

Componentes Atómicos y Moleculares en Reportes: Los átomos y moléculas que se han mencionado en el desarrollo de la aplicación web contable fueron creados para permitir al usuario seleccionar filtros y opciones para visualizar reportes específicos según la información o datos que necesite el usuario de las actividades contables que ha realizado.

Evaluación: La reutilización de átomos fue evidente en el formulario de filtros y visualización de reportes, lo cual facilitó la configuración de una interfaz consistente y modular. Esto se pudo ver reflejado en la combinación de selectores para formar listas desplegables asegurando la homogeneidad en los datos entrantes.

Resultado: Estos componentes operaron de manera óptima al permitir a los usuarios personalizar sus reportes, filtrando datos específicos sin redundancias en el código o la interfaz.

Componentes Organismos en Reportes: Los organismos como resultado de reportes y graficas presentaron una estructura de manera organizada y visualmente atractiva y fácil de comprender para el usuario como la creación de tablas y la implementación 2 tipos de gráficos (barra y pastel) para representar los ingresos egresos, ventas y compras de forma visual y atractiva para los usuarios.

Evaluación: La renderización y reutilización de gráficas y reporte de tablas Fueron satisfactorias en distintos contextos de visualización punto. Al definir una plantilla de página específica para la visualización de reportes, los organismos se integraron de forma coherente y optimizada. Los gráficos y tablas responden adecuadamente a las

selecciones de los filtros y muestran la información de forma clara, accesible y adaptativa.

Resultado: Los reportes se generaron correctamente y la visualización de datos se adaptó bien a los distintos dispositivos y resoluciones cumpliendo con los requisitos de usabilidad y accesibilidad.

4.2. Modularidad y Mantenibilidad

La metodología atomic design permitió crear componentes reutilizables, lo cual resultó en una estructura modular que facilita tanto el mantenimiento como la escalabilidad de la aplicación web, es decir cualquier cambio en un átomo o molécula se reflejó automáticamente en todas las estancias lo que optimizó el proceso de ajustes y mejoras, un ejemplo claro de esto son los cambios que pueden suceder en los requisitos, como modificar el color o el tamaño de algún componente, donde para lograrlo solo se debió ir a la sección de átomos y estilos del átomo correspondiente y modificar el estilos con los nuevos requerimientos, a partir de ahí los cambios se aplicarán en todos los lugares donde se utiliza el componente.

4.3. Implementación de Pruebas con React Testing Library y Jest

Para validar las funcionalidades de cada componente descrito anteriormente, se implementaron pruebas utilizando React Testing Library que es una biblioteca que facilita las pruebas de componentes en react y proporciona cierta certeza de seguridad en las pruebas al interactuar con los componentes de manera similar como lo haría un usuario en la aplicación web, por otro lado Jest es un complemento recomendado para trabajar con la biblioteca anteriormente mencionada que ofrece capacidades de integradas para simular funciones y contar con afirmaciones en los resultados de las pruebas que facilita la verificación del comportamiento del código (Tran, 2023). A continuación, se presentan

ejemplos básicos de cómo se estructuraron las pruebas y sus resultados en cada uno de las jerarquías de atomic design, utilizadas para los módulos de la aplicación.

***Figura 49**. React Testing Library Y Jest*

***Fuente:** Tomada de Cuarzo.log*

- **Pruebas realizadas a componentes atómicos.**

  Las pruebas realizadas para los componentes son:

  1. Verificar que el los componentes se muestre correctamente
  2. Comprobar el funcionamiento y el tiempo que este tarda en responder a un click por parte un usuario.
  3. Comprobar que los estilos y clases personalizadas funciones correctamente.

Para realizar las pruebas se crearon archivos de extensión .test que es una extensión de typeScript que ayuda a identificar de manera rápida que archivos contienen pruebas.

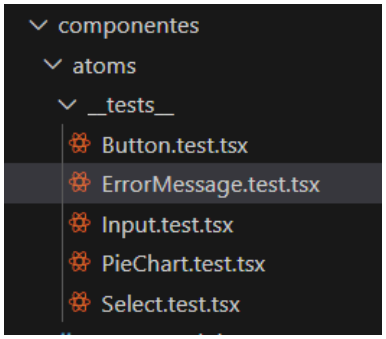


***Figura 50.** Archivos con pruebas (átomos)*

Fuente: Elaboración propia

Al ejecutar el comando **npm test** para las pruebas se obtuvieron los resultados del funcionamiento optimo de cada componente.

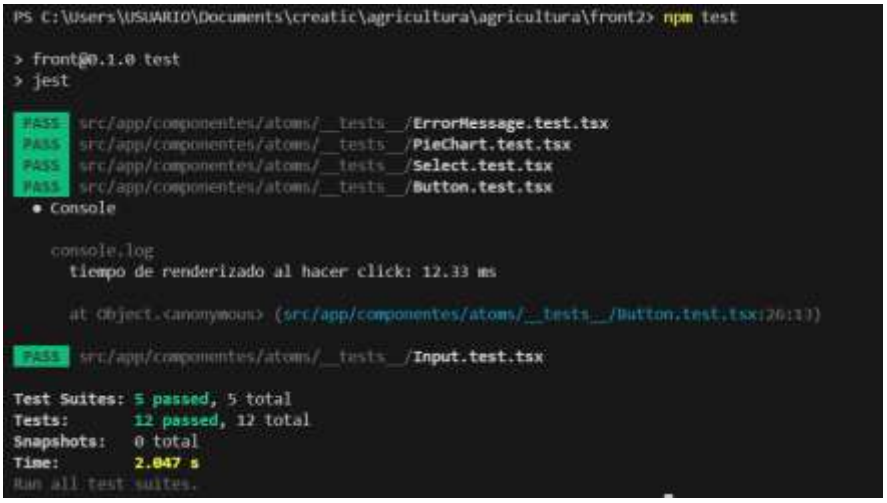


Figura 51. *Resultado de pruebas(atomos)*

Fuente: *Elaboración propia*

Cada línea que muestra “**PASS”** indica que las pruebas pasaron con éxito, así mismo los “**test suites”** son la cantidad de archivos que se utilizaron para las pruebas y los “**test”** son las pruebas individuales totales de todos los archivos en conjunto, por otro lado, se realizó prueba del tiempo que tarda en renderizar el botón al hacer click, como resultado se obtuvo un tiempo de 12.33 ms, el cual es un excelente indicador de rendimiento del componente ante un posible click del usuario.

- **Pruebas realizadas a moléculas:**

Las pruebas realizadas en las moléculas incluyeron un factor importante, como lo fue verificar que los componentes atómicos mencionados anteriormente se encontraran dentro de la molécula, Para este ejemplo se realizó la prueba a la molécula creada para el formulario de filtros, mencionada anteriormente en el desarrollo como (FilterForm), donde se obtuvieron los siguientes resultados.

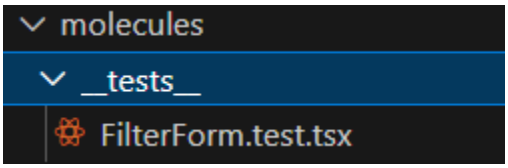


Figura 52. *Archivo de pruebas(moléculas)*

Fuente: *Elaboración propia*

```
console.log
  Componente encontrado: Label

  at Object.<anonymous> (src/app/componentes/molecules/__tests__/FilterForm.test.tsx:25:15)

console.log
  Componente encontrado: Input

  at Object.<anonymous> (src/app/componentes/molecules/__tests__/FilterForm.test.tsx:28:15)

console.log
  Componente encontrado: Select

  at Object.<anonymous> (src/app/componentes/molecules/__tests__/FilterForm.test.tsx:31:15)

console.log
  Componente encontrado: Button

  at Object.<anonymous> (src/app/componentes/molecules/__tests__/FilterForm.test.tsx:34:15)
```

Figura 53. *Resultado de ejemplo pruebas (Molécula)*

Fuente: *Elaboración propia*

Organismo

En las pruebas realizadas para la jerarquía del proyecto se verifico que los componentes utilizados en los organismos se renderizaran correctamente, es decir que se mostraran en pantalla, un ejemplo básico de ello fue el renderizado del título del reporte.

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import '@testing-library/jest-dom';
import Titulo from '../../atoms/Titulo';

test('renders Titulo component with correct text', () => {
  render(<Titulo>Software Contable Reporte General</Titulo>);

  // Verifica que el texto dentro de `Titulo` se renderiza correctamente
  const tituloElement = screen.getByText(/Software Contable Reporte General/i);
  expect(tituloElement).toBeInTheDocument();
});
```

Figura 54. *Código ejemplo de pruebas organismo*

Fuente: *Elaboración propia*

La implementación del código evaluó el renderizado que se definió para el título de la página de reportes, verificando su correcta visualización en pantalla. **Título** es un componente ubicado en el organismo reporte para la creación de títulos con diseño ya defino en la jerarquía atómica.

```
RUNS  src/app/componentes/atoms/__tests__/Input.test.tsx
RUNS  src/app/componentes/atoms/__tests__/PieChart.test.tsx
RUNS  src/app/componentes/organisms/__tests__/ReporteTable.test.tsx
PASS  src/app/componentes/organisms/__tests__/ReporteTable.test.tsx
```

Figura 55*. Resultado de pruebas (organismo)*

Fuente: *Elaboración propia*

4.4. Pruebas de seguridad en el uso de Tokens.

Para brindar seguridad al usuario y por recomendación del ingeniero (programador senior) encargado de la evaluación técnica del proyecto software contable sugirió trabajar con el uso de Json Web tokens que se generan tras el registro y la autenticación exitosa de un usuario. Estos tokens contienen una firma que verifica en cada solicitud a rutas protegidas asegurando que el usuario esté autorizado para acceder a los recursos.

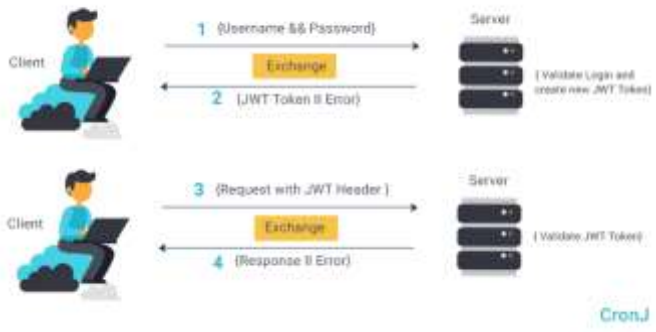


Figura 56. *Flujo de Autenticación y Validación de Tokens JWT*

Fuente: *Tomada de CronJ*

En el proyecto los JWT se utilizan en métodos de inicio de sesión con Google, registro e inicio de sesión. En cada de estos métodos, tras la autenticación o el registro de un usuario se crea un token de 1 hora, lo cual limitaba la duración de las sesiones y reducía la vulnerabilidad ante ataques externos (Microsoft, 2024).

```
const accessToken = jwt.sign(
  { id: document.id, email: document.email },
  secretKey,
  { expiresIn: "1h" }
);
```

Figura 57. *Código creación y expiración del token*

Fuente: *Elaboración propia*

Por otro lado, se trabajó con middleware. El cual extrae el token en el encabezado de autorización y utiliza jwt.verify para validar la autenticidad de la información del usuario.

```
const authenticateToken = (req, res, next) => {
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1];
  if (!token) {
    console.log("No se proporcionó token");
    return res.sendStatus(401);
  }

  jwt.verify(token, secretKey, { algorithms: ["HS256"] }, (err, usuario) => {
    if (err) {
```

Figura 58. *Código middleware*

Fuente: *Elaboración propia*

Durante las pruebas de seguridad, se simularon diferentes escenarios para evaluar la robustez del sistema ante intentos de acceso no autorizados. Estos son:

- **Token Ausente o Inválido:** Al intentar acceder a rutas protegidas sin token, el sistema responde con un código de estado 401 Unauthorized, indicando que el token es requerido.

- **Token Expirado o Manipulado:** Al enviar un token manipulado o expirado, el middleware rechaza la solicitud y devuelve un código 403 Forbidden. Esto asegura que solo los tokens válidos y firmados por el servidor puedan acceder a los recursos protegidos.

- **Recuperación de Contraseña:** En el método recoverPassword, se genera un token de un solo uso para la recuperación de contraseña. Este token expira en una hora, lo que limita el tiempo en que puede ser utilizado y reduce el riesgo de ataques (Microsoft, 2024).

```
const token = crypto.randomBytes(20).toString('hex');
user.resetToken = token;
await user.save();

const expirationTime = Date.now() + 3600000; // Token válido por 1 hora

await usuarioService.updateResetToken(user.id, token, expirationTime);
```

***Figura 59**. Código validez de token de recuperación*

***Fuente:** Elaboración propia*

4.5. Implementación de Rutas Protegidas

Las rutas protegidas del sistema incluyen endpoints que requieren autenticación, los cuales son establecidos en el archivo de rutas (model.index.js). Estas rutas, como la asociada a reporte, están encapsuladas en un subgrupo de rutas que utiliza

authenticateToken para la verificación previa de cada solicitud y brindar una mayor seguridad en el acceso a la información. Esto aseguraba que todas las rutas solo se puedan acceder tras la autenticación y validación del token.

Por ejemplo, la ruta /protected/initialLoad/all está diseñada para cargar datos protegidos. La solicitud es bloqueada en caso de no contar con un token válido, lo cual protege los datos sensibles de accesos no autorizados.

```
const protectedRouter = express.Router();
protectedRouter.use(authenticateToken);

router.use("/protected", protectedRouter);

// Ruta para cargar todos los datos
protectedRouter.get("/initialLoad/all", initialLoadController.findAll);
```

***Figura 60**. Código rutas protegidas*

***Fuente:** Elaboración propia*

4.6. Pruebas de Autenticación

Las pruebas de autenticación se realizaron mediante:

Tokens de Sesión Vencidos: En el desarrollo de autenticación una de las prioridades fue el manejo de errores. Un error importante a capturar fue el error que se produce tras la expiración del token, de este modo al intentar acceder con un token vencido, se obtuvo que el sistema redirija al usuario a la página de autenticación, exigiendo un inicio de sesión.

```
.catch((error) => {
  console.error("Error en la verificación de Firebase:", error);
  return res.status(403).json({ success: false, message: "El token ha expirado. Por favor, inicia sesión nuevamente." });
});
```

***Figura 61**. Código captura de error token vencido*

***Fuente:** Elaboración propia*

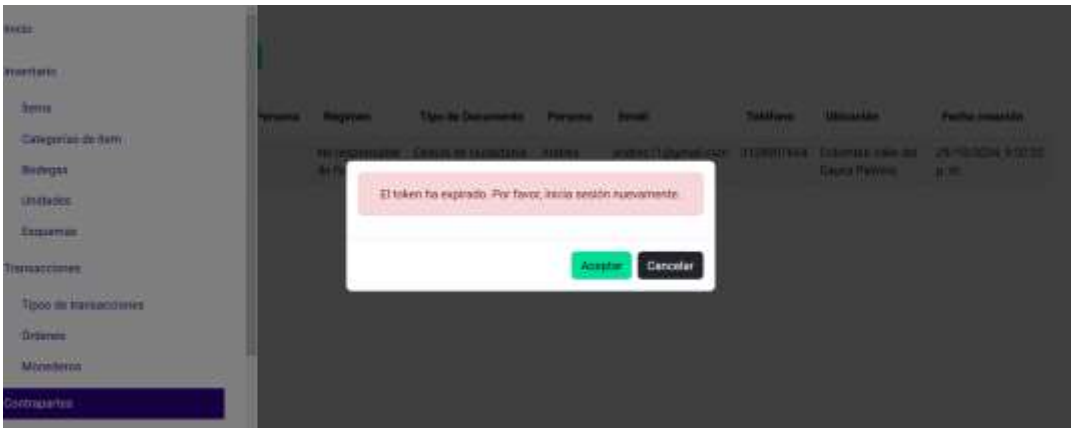


Figura 62. *Mensaje en pantalla token expirado*

Fuente: *Elaboración propia*

4.7. Uso de Claves Secretas y Algoritmos de Firma:

El sistema se configuró para usar el algoritmo HS256 en la firma de los JWT. Las pruebas confirmaron que el sistema rechaza tokens firmados con algoritmos alternativos o vencidos como se muestra en la figura 62, incrementando la seguridad.

email character varying (255)	contrasena character varying (255)
aaa@gmail.com	$2a$10$0gM0e61kB5TllIz2Afrkfeg7Pn03E06EgXw6xN7PLxwUPHe..f592

Figura 63. *Visualización de algoritmo HS256 en base de datos*

Fuente: *Elaboración Propia*

Como se muestra en la **figura 63**, el sistema almacena de forma segura el email y contraseña del usuario en la base de datos, además la contraseña esta encriptada y la firma de los tokens no se almacenó en la base de datos, si no que como se mencionado anteriormente esta genera en distintos inicios de sesión, minimizando el riesgo de exposición de los datos.

Recomendaciones

Durante el desarrollo de los módulos de autenticación y generación de reportes, utilizando la metodología Atomic Design se garantizó un diseño escalable y reutilizable, sin embargo, para que el proyecto sea exitoso, es recomendable continuar con el uso de la metodología Atomic Design en los módulos faltantes de la aplicación web “Software Contable Agropecuario”, asegurando que los componentes sean reutilizables, reducir tiempo de desarrollo y facilitar el mantenimiento del código a futuro, Además se recomienda utilizar o modificar la misma estructura atómica de los módulos de autenticación y generación de reportes para llevar consistencia en el diseño.

Si bien se siguieron buenas prácticas, en algunos casos, la documentación del código no fue lo suficientemente detallada, lo que puede dificultar el trabajo en equipo, por tal motivo debe implementar una documentación estricta como guía de los componentes globales y sus estilos finales para mantener la consistencia total de la interfaz.

Durante el desarrollo de módulos con Atomic Design, se observó que, aunque la metodología fomenta la reutilización y organización, no garantiza por sí sola la mejor experiencia de usuario. Sin un enfoque en normas de usabilidad, los componentes fueron visualmente consistentes, pero no necesariamente funcionales o intuitivos. De este modo se ve la necesidad de Incorporar un conjunto de **normas de usabilidad** como estándar en el desarrollo con Atomic Design. Estas normas deben guiar el diseño y la implementación de componentes en todos los niveles (átomos, moléculas, organismos, plantillas y páginas). Como incluir los estilos adecuados como tamaño y color, etc. Según las normas establecidas por la empresa:

Conclusiones

En conclusión, a través del desarrollo de este proyecto, se identificó la importancia fundamental de realizar un análisis exhaustivo de requerimientos funcionales antes de comenzar cualquier desarrollo. Esta fase inicial fue crucial para establecer las bases de los módulos de autenticación y generación de reportes que debían ser implementados. El proceso de identificación y revisión de requerimientos permitió entender las necesidades del cliente, y así poder asegurar que los componentes desarrollados se alinearan correctamente con la metodología Atomic Design que elegí para este proyecto.

Durante la fase de desarrollo, encontré que la implementación de la metodología Atomic Design fue una decisión acertada, ya que me permitió descomponer las interfaces en componentes más pequeños y reutilizables. Este enfoque no solo me ayudó a optimizar el tiempo de desarrollo, sino que también logré mejorar significativamente la escalabilidad y modularidad de la aplicación. La creación de mockups y diagramas de flujo resultó ser una herramienta invaluable para visualizar la estructura y el comportamiento de los módulos, asegurándome de que cada elemento cumpliera con los requisitos establecidos.

En la fase de evaluación y pruebas de la aplicación web, me enfoqué principalmente en verificar la funcionalidad y estructura de los módulos implementados. Las pruebas que realicé fueron especialmente rigurosas en los aspectos de autenticación y generación de reportes. A través de estas pruebas, pude comprobar la correcta integración entre el frontend y el backend, confirmando que la metodología Atomic Design facilitó efectivamente la creación de componentes eficientes y una experiencia de usuario fluida.

En resumen, La combinación de un análisis minucioso de requerimientos, un desarrollo sistemático y pruebas exhaustivas permitió crear módulos de autenticación y reportes que cumplieron con las expectativas del cliente. Esta experiencia demostró el valor práctico de la metodología Atomic Design en la creación de sistemas modulares y reutilizables.

Bibliografía

del Cerro, P. V. (2020). Frameworks y lenguajes de desarrollo para Frontend. (36), pág. 6.

Fernández Ordoñez, J. M. (2021). *Evaluación de experiencia afectiva de usuario (UAX) con IA conversacional y recursos didácticos digitales diseñados para alumnos con síndrome de Down.* Tesis de licenciatura.

Frost, B. (2016). *Atomic design.* Pittsburgh: Brad Frost.

Grajales Urquijo, J. M. (2024). *Aplicación del Atomic Design para mejorar la eficiencia de proyectos Frontend: un enfoque metodológico para optimizar el desarrollo de interfaces de usuario.* Trabajo de grado, Universidad de Antiquia. Obtenido de https://bibliotecadigital.udea.edu.co/handle/10495/38582

Guamanga, M. H. (2020). *La lógica formal, el pensamiento crítico y la cognición situada.* Revista. doi:https://doi.org/10.26620/uniminuto.polisemia.16.30.2020.117-132

Hernández-Rueda, K. V. (2022). *Evaluación del rendimiento de una aplicación web: Evaluating the performance of a web application.* Revista. doi:https://doi.org/10.46932/sfjdv3n1-034

Herron, D. (2020). *Node.js Web Development: Server-side web development made easy with Node 14 using practical examples.* Packt Publishing Ltd.

Jongwan Kim, J.-w. J.-y. (2021). *Improving Student's Design Prototyping Skills using Interactive Prototyping Tool.* Revista. doi:https://www.jmis.org/archive/view_article?pid=jmis-8-1-75

León Mendoza, M. (2018). *El crecimiento de la información y la toma de decisiones.* Revista. doi:https://doi.org/10.21855/ecociencia.44.46

León Ochoa, D. S. (2024). *Un gestor de tareas basado en kanban para la planificación operativa: desarrollo del front-end y back-end de una aplicación web bajo el*

enfoque ágil: desarrollo del front-end. ESCUELA POLITÉCNICA NACIONAL. Obtenido de http://bibdigital.epn.edu.ec/handle/15000/25967

Macías Vélez, E. M. (2023). *Propuesta de diseño de una metodología de desarrollo de software basado en el enfoque UI/UX.* Revista. doi:https://doi.org/10.55813/gaea/ccri/v4/nE2/167

Méndez Fernandez, M. J. (2022). *Design system: estandarización de un lenguaje de diseño compartido para el sistema de facturación Billo.* Obtenido de https://hdl.handle.net/2238/14382

Microsoft. (2024). *Documentación de la plataforma de identidad de Microsoft.* Obtenido de https://learn.microsoft.com/es-es/entra/identity-platform/access-tokens

Ortiz Caballero, E. (2024). *Diseño de interfaz gráfica para la aplicación Barrio Universitario.* Revista. doi:https://doi.org/10.22201/dgtic.ctud.2024.2.1.29

Pedraza-Gutiérrez, S. I.-G.-R.-H. (2023). *Diseño centrado en el usuario y experiencia de usuario en el sistema de control de acceso de la Universidad Libre.* Revista. doi:https://doi.org/10.51252/rcsi.v3i1.426

Pierin, F. L. (2018). Integraweb: uma arquitetura baseada em mapeamentos semânticos. *In Proceedings of the XI Seminar on Ontology Research in Brazil and II Doctoral and Masters Consortium on Ontologies*, (págs. pp. 116-127).

Rivera, L. A. (2024). *Tendencia en el diseño web sobre los marcos de trabajo para esquemas adaptativos: Revisión sistemática de la literatura.* Revista. doi:https://doi.org/10.59814/resofro.2024.4(3)282

Roedavan, R. P. (2020). *Adaptation Atomic Design Method for Rapid Game Development Model.* Revista.

Saldaña Cercós, D. (2018). *Implementación de la autenticación y autorización de usuarios en un sistema multitenant.* Trabajo final de grado, UPC, Facultat d'Informàtica de Barcelona. Obtenido de http://hdl.handle.net/2117/121156

Solano Romero, J. D. (2024). *Propuesta para mejorar la seguridad de la comunicación de datos de las organizaciones de Costa Rica a través de internet.* Tesis doctoral, Universidad Cenfotec.

Tong, J. J. (2023). Comparative Performance Analysis of Javascript Frontend Web Frameworks. En IEEE (Ed.), *In 2023 3rd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, (págs. pp. 81-86).

Tran, M. (2023). *Testing React Applications Using React Testing Library.* Obtenido de https://urn.fi/URN:NBN:fi:amk-202304145317

Villalobos Garcia, A. F. (2023). *Diseño de una interfaz que haga eficiente el proceso del practicante al interior de la empresa NTT DATA como Diseñador Industrial.* Obtenido de http://hdl.handle.net/20.500.12010/31747

Vintimilla, M. I. (2023). *Estudio comparativo para la toma de requerimientos de software para el desarrollo de una aplicación web en investigación psicológica del bullying educativo: Comparative study for software requirements gathering for the development of a web application in.* Revista.