

Retail Company  
Software Architecture Document (SAD)

**CONTENT OWNER: Mauricio Alfonso Donado  
Canedo, Paula Andrea Márquez Orlando**

DOCUMENT NUMBER:	RELEASE/REVISION:	RELEASE/REVISION DATE:
• 1	• V0.1	• 18/09/2024
• 2	• V0.2	• 21/09/2024
• 3	• V0.3	• 24/09/2024
• 4	• V0.4	• 1/10/2024
• 5	• V0.5	• 20/10/2024
• 6	• V0.6	• 24/11/2024

---

# Tabla de contenido

<b>List of Figures.....</b>	<b>1</b>
<b>List of Tables .....</b>	<b>3</b>
<b>1     Documentation Roadmap.....</b>	<b>4</b>
<b>Section 1.1 Document Management and Configuration Control Information .....</b>	<b>4</b>
<b>Section 1.2 Purpose and Scope of the SAD.....</b>	<b>5</b>
<b>Section 1.3 Stakeholder Representation.....</b>	<b>6</b>
<b>Section 1.4.0 Viewpoint Definitions .....</b>	<b>7</b>
<b>    1.4.1.0 Bussines Logic Viewpoint Definition .....</b>	<b>7</b>
<b>    1.4.2.0 Web App Viewpoint Definition.....</b>	<b>8</b>
<b>2     Architecture Background.....</b>	<b>11</b>
<b>    2.1              Problem Background.....</b>	<b>11</b>
2.1.0 System Overview .....	11
2.1.1 Goals and Context.....	11
3.4.0 Significant Driving Requirements.....	14
<b>    3.5              Solution Background .....</b>	<b>17</b>
3.5.0 Requirements Coverage .....	17
<b>    3.0              Views .....</b>	<b>20</b>
<b>    3.1              Bussiness Logic Model View .....</b>	<b>20</b>
<b>3.1.1 View Description .....</b>	<b>20</b>

---

<b>3.2</b>	<b>Web App View.....</b>	<b>26</b>
3.2.0	View Description.....	26
3.2.1	View Packet Overview.....	27
<b>3.3</b>	<b>API View.....</b>	<b>39</b>
3.3.0	View Description.....	39
3.3.1	View Packet Overview.....	39
<b>4</b>	<b>Referenced Materials.....</b>	<b>42</b>
<b>5</b>	<b>Directory .....</b>	<b>43</b>
<b>5.4</b>	<b>Index.....</b>	<b>43</b>
<b>5.5</b>	<b>Glossary .....</b>	<b>43</b>
<b>5.6</b>	<b>Acronym List.....</b>	<b>44</b>

---

## List of Figures

<b>Imagen 3.1. Inicio de sesión .....</b>	<b>21</b>
<b>Imagen 3.2. Gestión de inventario .....</b>	<b>22</b>
<b>Imagen 3.3. Compra de Productos .....</b>	<b>23</b>
<b>Imagen 3.4. Diagrama de clases .....</b>	<b>24</b>
<b>Imagen 3.5. Diagrama de despliegue .....</b>	<b>24</b>
<b>Imagen 3.6 Diagrama de Secuencia .....</b>	<b>26</b>
<b>Imagen 3.7 DashBoard de Administrador.....</b>	<b>27</b>
<b>Imagen 3.8 Vista de Carrito .....</b>	<b>28</b>
<b>Imagen 3.9 vista de Producto .....</b>	<b>29</b>
<b>Imagen 3.10 Vista de Iniciar sesión .....</b>	<b>30</b>
<b>Imagen 3.11 Vista Crear cuenta.....</b>	<b>31</b>
<b>Imagen 3.12 Mockup Vista de Inicio .....</b>	<b>32</b>
<b>Imagen 3.13 Vista Featured Categories .....</b>	<b>33</b>
<b>Imagen 3.14 Vista Innovación.....</b>	<b>33</b>
<b>Imagen 3.15 Vista Admin en gestión de Usuario .....</b>	<b>34</b>
<b>Imagen 3.16 Vista Admin Gestión de Clientes.....</b>	<b>34</b>
<b>Imagen 3.17 Vista Admin en gestión de inventario .....</b>	<b>35</b>
<b>Imagen 3.18 Vista Admin Ordenes y promociones .....</b>	<b>36</b>
<b>Imagen 3.19 Vista Admin para gestión de la cadena de suministros .....</b>	<b>36</b>
<b>Imagen 3.20 Vista de perfil de Usuario.....</b>	<b>37</b>
<b>Imagen 3.21 Vista de stay update.....</b>	<b>37</b>

---

**Imagen 3.22 Vista de pie de pag informativo ..... 38**

**Imagen 3.23 Vista de pie de pag con formulario de suscripcion..... 38**

**Imagen 3.24.1 Vista de Nuestra Vision ..... 39**

---

## **List of Tables**

Tabla 1 Stakeholder Representation .....	6
Tabla 2 Viewpoint Definitions .....	7
Tabla 3 Referenced Material .....	42
Tabla 4 Glosario.....	44
Tabla 5 Lista de Acronimos.....	44

---

# 1 Documentation Roadmap

## Section

1. Revision Number: 1
  - Revision Release Date: 18/09/2024
  - Purpose of Revision: First paper
  - Scope of Revision: Purpose and Scope of the SAD, Architecture Background, Problem Background, Solution Background, System Overview, Goals and Context, Significant Driving Requirements.
2. Revision Number: 2
  - Revision Release Date: 21/09/2024
  - Purpose of Revision: Second paper
  - Scope of Revision: Context Diagram Add
3. Revision Number: 3
  - Revision Release Date: 24/09/2024
  - Purpose of Revision: Third paper
  - Scope of Revision: Sequence diagrams add, viewpoints definition.
4. Revision Number: 4
  - Revision Release Date: 1/10/2024
  - Purpose of Revision: Fourth paper
  - Scope of Revision: Views definition
5. Revision Number: 5
  - Revision Release Date: 20/10/2024
  - Purpose of Revision: Fifth paper
  - Scope of Revision: Update
6. Revision Number: 6
  - Revision Release Date: 24/11/2024
  - Purpose of Revision: Fifth paper
  - Scope of Revision: Update of view

---

## Section 1.2

Una gran empresa minorista tiene múltiples sistemas heredados para comercio electrónico, compras en tienda, cadena de suministro y gestión de inventarios. Estos sistemas fueron construidos a lo largo de muchos años como silos separados con sus propias bases de datos. Con adquisiciones y crecimiento, la compañía ahora tiene una compleja red de sistemas interconectados que no comparten datos de manera eficiente. Los datos de clientes y productos están fragmentados entre los sistemas, lo que obstaculiza el análisis.

La falta de una vista unificada de los datos de clientes y productos dificulta la entrega de experiencias omnicanal personalizadas. Por ejemplo, las promociones en línea y en tienda no están coordinadas, lo que genera mensajes inconsistentes. Los procesos de inventario y cadena de suministro también son subóptimos debido a la incapacidad de tomar decisiones basadas en una vista consolidada a través de todos los sistemas y ubicaciones. La empresa necesita una arquitectura mejorada que permita compartir datos sin problemas entre las capacidades comerciales para satisfacer las expectativas cambiantes de los clientes.

La solución debe descomponer los monolitos existentes en microservicios por capacidad de negocio. Estos servicios deben exponer sus datos a través de API bien definidas que puedan ser consumidas por otros servicios. Un gateway de API debe manejar preocupaciones transversales como seguridad, monitoreo y limitación de velocidad. El enfoque debe centrarse en construir una arquitectura de datos flexible y basada en API, centrada en dominios clave. Algunas especificaciones son:

- Implementar APIs unificadas para el perfil del cliente, catálogo de productos e inventario, e historial de pedidos agregados de todos los sistemas.
  - Exponer APIs que puedan ser consumidas por varias aplicaciones frontend y servicios.
  - Utilizar un patrón de gateway de API para la seguridad, control de tráfico, enrutamiento, monitoreo y limitación de velocidad.
-

---

## Section 1.3

Access To	Administrator	User	Customer	Developer
Create Products	X			X
Update Products	X			X
Delete Products	X			X
Read Products	X	X	X	X
Create Customers	X			X
Update Customers	X			X
Get Customers	X			X
Delete Customers	X		X	X
Get User	X			
Create User	X			X
Update User	X			X
Delete User	X			X

*Tabla 1 Stakeholder Representation*

---

## Section 1.4.0

Stakeholders	Viewpoint(s) that apply to that class of
	stakeholder's concerns
Gestores de Proyectos, Cliente, Desarrollador	Bussines Logic Viewpoint
Desarrollador, Cliente, Usuarios	Web App ViewPoint
Desarrollador	API ViewPoint
Gestores de Proyectos, Desarrollador	Data Viewpoint

*Tabla 2 Viewpoint Definitions*

### 1.4.1.0 Bussines Logic Viewpoint Definition

#### Abstract

La perspectiva de la lógica del negocio describe las principales entidades del modelo de negocio y sus interacciones, tanto internas como con el sistema en cuestión. Esta visión permite comprender cómo se relacionan los elementos clave del negocio y cómo interactúan con el sistema en estudio.

#### Stakeholders and Their Concerns Addressed

- Gerente: Establece los requisitos necesarios para el negocio.
- Desarrollador: Utiliza la lógica del negocio para desarrollar interfaces que cumplan con los requisitos del negocio.

#### Elements, Relations, Properties, and Constraints

#### Elements

- **Diagrama de modelo de procesos de negocio:** Representa los flujos de trabajo y procesos del negocio, proporcionando una visión general de cómo se llevan a cabo las actividades y sus interacciones.
- **Diagrama de clases:** Muestra la estructura estática del sistema, con clases, atributos y

---

relaciones, lo cual es clave para modelar las entidades del sistema y entender la lógica del negocio.

- **Diagramas de secuencia:** Detallan el orden de las interacciones entre las interfaces del sistema para cada microservicio, lo que ayuda a comprender la secuencia de eventos y la comunicación entre componentes, siendo fundamental para el diseño y la implementación efectiva del sistema.
- **Diagrama de componentes:** Describe la estructura de los componentes del sistema y sus relaciones, lo cual es esencial para entender cómo se organiza el software y cómo interactúan sus partes para ofrecer funcionalidades específicas.

## Relations

- Los diagramas de procesos de negocio y de contexto muestran cómo las interfaces del sistema se alinean con los procesos del negocio.
- El diagrama de clases muestra las relaciones entre las diferentes entidades dentro del sistema.
- El diagrama de componentes ilustra cómo se relacionan las diferentes partes del sistema.

## Language(s) to Model/Represent Conforming Views

Se utilizó UML para diseñar los elementos de esta perspectiva, permitiendo a los interesados visualizar rápidamente el funcionamiento del sistema a través de diagramas.

---

## Abstract

### 1.4.2.0 Web App Viewpoint Definition

#### 1.4.2.1

La perspectiva de la aplicación web permite visualizar cómo interactúan los usuarios con los diversos microservicios que ofrece el sistema y especifica qué accesos tiene cada tipo de usuario dentro del mismo.

#### Stakeholders and Their Concerns Addressed

- **Desarrollador:** Establece cómo deben interactuar las interfaces del sistema y el flujo de comunicación entre frontend, backend y base de datos.
- **Cliente:** Define las funcionalidades clave que debe ofrecer el sistema, como gestión de productos, clientes y pedidos, asegurando que se cumplan los requerimientos de negocio.
- **Usuario:** Interactúa directamente con la aplicación web y necesita comprender su funcionamiento, capacidades y limitaciones para realizar tareas como actualizar inventarios o gestionar promociones.

#### Elements, Relations, Properties, and Constraints

##### Elements

• **Mockups:** Diseñados inicialmente con Wix, proporcionando una visión preliminar del diseño y flujo de la aplicación web antes de implementar las funcionalidades finales.

• **Frontend:** Desarrollado con Next.js, un framework avanzado basado en React, enfocado en optimizar la experiencia del usuario y la eficiencia del sitio web.

• **Backend:** Construido utilizando Express.js sobre Node.js, proporcionando una arquitectura sólida y flexible para gestionar la lógica del negocio y coordinar microservicios.

• **Base de Datos:** Implementada con MongoDB, una base de datos NoSQL que permite gestionar eficientemente la persistencia y consulta de datos del sistema.

- **API:** Conecta el frontend con el backend para realizar solicitudes, procesar datos y responder a las acciones del usuario.
- **Usuarios:** Interactúan con la interfaz para acceder a las funcionalidades del sistema, como agregar o consultar productos.
- **Base de Datos:** El backend accede y modifica los datos a través de la API para garantizar la integridad de la información.

---

## Constraints

- **Acceso Basado en Roles:** Define los permisos de los usuarios, restringiendo el acceso a ciertas funcionalidades según el rol asignado (administrador, cliente, etc.).

## API Viewpoint Definition Abstract

Este punto de vista describe la estructura de los microservicios y los métodos para acceder a las funcionalidades del sistema, incluyendo la interacción entre los usuarios, la API y la base de datos.

## Stakeholders and Their Concerns Addressed

- **Desarrollador:** Requiere comprender la estructura del sistema API para su mantenimiento, la integración con nuevos servicios y la resolución de problemas.

## Elements, Relations, Properties, and Constraints

### Elements

- **Vistas del API:** Describe los endpoints y sus funcionalidades, como crear, leer, actualizar y eliminar datos.

### Relations

- **Web App:** La API sirve como intermediario, gestionando solicitudes del usuario y asegurando la integridad de las operaciones.
- **Base de Datos:** Procesa solicitudes de datos a través de MongoDB, asegurando el acceso y la persistencia de la información.

### Constraints

- **Estructura de Datos:** Los datos deben cumplir con las validaciones definidas en el backend para garantizar consistencia y seguridad.

## Language(s) to Model/Represent Conforming Views

El desarrollo del API se realizó con Node.js, utilizando Express.js como framework para estructurar y manejar las solicitudes del sistema. La API incluye validaciones robustas para garantizar la consistencia de los datos y emplea MongoDB como base de datos principal para el almacenamiento y la gestión de la información.

---

# Architecture Background

## 1.1 Problem Background

Una gran empresa minorista ha crecido mediante adquisiciones y expansiones, lo que ha generado una red compleja de sistemas heredados que funcionan como silos independientes. Estos sistemas gestionan diferentes áreas del negocio, como el comercio electrónico, las ventas en tienda, la logística y el control de inventarios, pero no comparten datos de manera eficiente.

Como resultado, la información clave sobre clientes y productos está fragmentada, lo que dificulta los análisis, limita la capacidad de personalizar experiencias omnicanal y afecta la optimización de los procesos de inventario y cadena de suministro. Esta falta de integración provoca inconsistencias en las promociones, decisiones subóptimas en la logística y una experiencia de cliente incoherente entre canales físicos y digitales.

### 1.1.0 System Overview

Para resolver estos problemas, se propone una arquitectura basada en microservicios que sustituya los sistemas monolíticos heredados. La solución se centra en:

- Implementar **APIs unificadas** para proporcionar un acceso centralizado y consistente a datos clave.
- Integrar un **API Gateway** para gestionar la seguridad, el monitoreo y el control del tráfico.
- Diseñar una arquitectura modular y flexible que permita escalar componentes según las necesidades del negocio.

### 1.1.1 Goals and Context

#### 1. Unificación de Datos y Accesibilidad:

- Desarrollar APIs que integren información de perfiles de clientes, catálogos de productos, inventarios y órdenes de compra.
- Consolidar datos fragmentados en una única fuente de verdad para garantizar la consistencia en todos los sistemas.

#### 2. Mejora de la Eficiencia Operativa:

- Descomponer los sistemas monolíticos en microservicios autónomos que puedan ser desarrollados y gestionados independientemente.
- Utilizar un **API Gateway** para manejar aspectos clave como autenticación, monitoreo y limitación de solicitudes.

#### 3. Experiencia del Cliente y Personalización:

- Ofrecer experiencias unificadas y personalizadas en todos los puntos de contacto del cliente, aprovechando datos centralizados y consistentes.

#### 4. Escalabilidad y Flexibilidad:

- 
- Escalar componentes individuales del sistema de acuerdo con las necesidades de la carga de trabajo.
  - Facilitar la implementación de nuevas funcionalidades mediante un diseño modular y ágil.

## 2. Contexto de la Arquitectura de Software

### 1. Ciclo de Vida de la Arquitectura de Software:

#### 1. Fase de Diseño:

- Identificación de microservicios y definición de APIs unificadas.
- Configuración inicial del API Gateway y elaboración de especificaciones técnicas detalladas.

#### 2. Desarrollo e Implementación:

- Desarrollo de microservicios en **Node.js** y **Express**, con pruebas iterativas y despliegues mediante metodologías ágiles y **DevOps**.
- Configuración del API Gateway para asegurar accesos y monitorear rendimiento.
- Integración de MongoDB como base de datos operativa.

#### 3. Operación y Mantenimiento:

- Monitorización continua del sistema mediante herramientas conectadas al API Gateway.
- Actualización y escalabilidad de microservicios sin afectar la operación general.

### 2. Relación con Resultados y Artefactos de la Ingeniería de Sistemas:

- **Análisis de Requisitos:** La solución se alinea con las necesidades del negocio, mejorando la experiencia del cliente y la eficiencia operativa.
- **Modelos de Dominio:** Se establecen modelos claros para clientes, productos, inventarios y órdenes que guían la estructura de los microservicios.
- **Gestión de Proyectos:** La arquitectura modular facilita la alineación entre hitos técnicos y objetivos de negocio.

### 3. Factores Relevantes:

- **Integración de Sistemas Heredados:** Los datos de los sistemas existentes se migran gradualmente hacia el nuevo ecosistema, reduciendo interrupciones.
- **Tecnologías y Herramientas:** Se utilizan frameworks como **Next.js** para el frontend, **Express** para el backend y **MongoDB** como base de datos, además de un API Gateway robusto.
- **Seguridad y Cumplimiento:** Se implementan medidas avanzadas de seguridad y políticas de acceso basadas en roles para proteger la información sensible.

---

### **3.4.0 Significant Driving Requirements**

Los requisitos de comportamiento especifican cómo debe operar el sistema en cuanto a las interacciones entre sus componentes y con los usuarios. Estos requisitos han sido esenciales para definir la arquitectura de software de la empresa minorista.

#### **1. Interoperabilidad entre Sistemas:**

- **Requisito:** Los microservicios deben poder comunicarse de manera eficiente y segura entre sí, así como con los sistemas heredados.
- **Justificación:** Esto es crucial para asegurar una integración fluida de datos y procesos, eliminando la fragmentación y mejorando la coherencia operativa.

#### **2. Acceso en Tiempo Real:**

- **Requisito:** Los datos de clientes, productos, inventarios y pedidos deben ser accesibles en tiempo real a través de las APIs.
- **Justificación:** La disponibilidad en tiempo real es vital para proporcionar una experiencia de usuario coherente y actualizada, así como para optimizar las operaciones de inventario y cumplimiento de pedidos.

#### **3. Escalabilidad Horizontal:**

- **Requisito:** La arquitectura debe permitir el escalado horizontal de microservicios para manejar aumentos en la carga de trabajo sin degradación del rendimiento.
- **Justificación:** La capacidad de escalar según la demanda es esencial para mantener el rendimiento y la disponibilidad del sistema durante picos de tráfico, como en temporadas de ventas.

### **1. Requisitos de Atributos de Calidad**

Los atributos de calidad se refieren a las propiedades del sistema que no son funcionales, pero son críticas para su éxito. Estos atributos han influido significativamente en el diseño de la arquitectura.

---

---

## 1. Seguridad:

- **Requisito:** El sistema debe implementar mecanismos robustos de autenticación y autorización, y garantizar la protección de datos sensibles.
- **Justificación:** La seguridad es fundamental para proteger la información del cliente y cumplir con las normativas de privacidad de datos.

## 2. Disponibilidad:

- **Requisito:** El sistema debe garantizar una alta disponibilidad, con medidas para la recuperación rápida ante fallos.
- **Justificación:** Una alta disponibilidad es crucial para evitar interrupciones en las operaciones comerciales y garantizar una experiencia de cliente confiable.

## 3. Rendimiento:

- **Requisito:** Las APIs deben tener tiempos de respuesta rápidos y ser capaces de manejar un alto volumen de solicitudes concurrentes.
- **Justificación:** Un rendimiento óptimo es esencial para proporcionar una experiencia de usuario fluida y eficiente, especialmente durante transacciones críticas.

## 4. Mantenibilidad:

- **Requisito:** La arquitectura debe facilitar la actualización y el mantenimiento de microservicios individuales sin afectar al sistema en su conjunto.
  - **Justificación:** La mantenibilidad es importante para la rápida implementación de mejoras y corrección de errores, asegurando la evolución continua del sistema.
-

---

## 2. Escenario de Seguridad:

- **Descripción:** Un usuario intenta acceder a la API de gestión de inventarios desde una ubicación no autorizada.
- **Respuesta Esperada:** El sistema debe rechazar el acceso y registrar el intento de intrusión.
- **Impacto:** Esto asegura que solo usuarios autorizados puedan acceder a datos sensibles, protegiendo la integridad del sistema.

## 3. Escenario de Disponibilidad:

- **Descripción:** Durante un evento de alta demanda (por ejemplo, Black Friday), el sistema experimenta un aumento significativo en el tráfico.
- **Respuesta Esperada:** El sistema debe escalar horizontalmente los microservicios relevantes para manejar la carga adicional sin degradar el rendimiento.
- **Impacto:** Garantiza que el sistema permanezca disponible y funcional durante picos de tráfico intensos.

## 4. Escenario de Rendimiento:

- **Descripción:** Un cliente realiza una búsqueda de productos en la plataforma de comercio electrónico.
- **Respuesta Esperada:** La API de catálogo de productos debe devolver los resultados en menos de 200 ms.
- **Impacto:** Proporciona una experiencia de usuario rápida y eficiente, mejorando la satisfacción del cliente.

## 5. Escenario de Mantenibilidad:

- **Descripción:** Se descubre un error en el microservicio de procesamiento de pedidos.
-

- 
- **Respuesta Esperada:** El equipo de desarrollo debe poder desplegar una corrección sin afectar la disponibilidad de otros microservicios.
  - **Impacto:** Minimiza el tiempo de inactividad y permite una rápida respuesta a problemas, mejorando la fiabilidad del sistema.

## 3.5 Solution Background

La arquitectura propuesta surge como respuesta a los desafíos inherentes a la infraestructura heredada fragmentada de la empresa minorista. La adopción de microservicios, con APIs unificadas y un gateway de API, se presenta como la solución ideal para mejorar la integración de datos y satisfacer los objetivos de comportamiento y calidad. Este enfoque modular y flexible permite una evolución independiente de componentes y una mayor adaptabilidad a cambios futuros. En conjunto, esta arquitectura garantiza una experiencia de cliente mejorada y una operación comercial más eficiente.

### 3.5.0 Requirements Coverage

#### 1. Requisito: Autenticación de Usuarios

##### **Descripción:**

El sistema debe proporcionar funcionalidades para registrar nuevos usuarios y permitir el inicio de sesión.

##### **Cobertura en la Arquitectura:**

Implementado en el archivo authRoutes.ts utilizando el framework Express.js. Las rutas RESTful gestionan las siguientes operaciones:

##### **Registro de usuarios:**

Ruta POST /register, vinculada al controlador registerUser.

##### **Inicio de sesión:**

Ruta POST /login, vinculada al controlador loginUser.

Ambas rutas interactúan con el controlador authController.ts, que contiene la lógica para manejar solicitudes de autenticación.

#### 2. Requisito: Gestión de Clientes

---

**Descripción:** El sistema debe permitir la creación, consulta, actualización y eliminación de registros de clientes.

**Cobertura en la Arquitectura:** Implementado en el archivo customerRoutes.ts utilizando el framework Express.js. Las rutas RESTful gestionan las siguientes operaciones:

- **Consulta de clientes:** Ruta GET /, vinculada al controlador getCustomers.
- **Creación de un cliente:** Ruta POST /, vinculada al controlador createCustomer.
- **Actualización de un cliente:** Ruta PUT /:id, actualmente implementada con una respuesta básica (res.send), en espera de lógica en el controlador correspondiente.
- **Eliminación de un cliente:** Ruta DELETE /:id, vinculada al controlador deleteCustomer.

### 3. Requisito: Gestión de Inventario

**Descripción:** El sistema debe permitir realizar operaciones CRUD (crear, consultar, actualizar y eliminar) en los registros del inventario.

**Cobertura en la Arquitectura:** Implementado en el archivo inventoryRoutes.ts utilizando el framework Express.js. Las rutas RESTful gestionan las siguientes operaciones:

- **Creación de inventario:** Ruta POST /, vinculada al controlador createInventory.
- **Consulta del inventario:** Ruta GET /, vinculada al controlador getInventory.
- **Actualización del inventario:** Ruta PUT /:id, vinculada al controlador updateInventory.
- **Eliminación de inventario:** Ruta DELETE /:id, vinculada al controlador deleteInventory.

Estas rutas interactúan con el controlador inventoryController.ts, donde se define la lógica para manejar las solicitudes relacionadas con el inventario.

---

#### 4. Requisito: Gestión de Órdenes

**Descripción:** El sistema debe permitir la creación, consulta, actualización y eliminación de órdenes, así como la obtención de detalles específicos de una orden por su identificador único.

**Cobertura en la Arquitectura:** Implementado en el archivo orderRoutes.ts utilizando el framework Express.js. Las rutas RESTful gestionan las siguientes operaciones:

- **Creación de órdenes:** Ruta POST /, vinculada al controlador createOrder.
- **Consulta de todas las órdenes:** Ruta GET /, vinculada al controlador getOrders.
- **Consulta de una orden específica:** Ruta GET /:id, vinculada al controlador getOrderById.
- **Actualización de órdenes:** Ruta PUT /:id, vinculada al controlador updateOrder.
- **Eliminación de órdenes:** Ruta DELETE /:id, vinculada al controlador deleteOrder.

#### 5. Requisito: Gestión de productos

**Descripción:** El sistema debe permitir la creación, consulta, actualización y eliminación de productos en la base de datos.

**Cobertura en la Arquitectura:** Implementado en el archivo productRoutes.ts utilizando el framework Express.js. Las rutas RESTful permiten las siguientes operaciones:

- **Consulta de productos:** Ruta GET /, vinculada al controlador getProducts.
- **Creación de un producto:** Ruta POST /, vinculada al controlador createProduct.
- **Actualización de un producto:** Ruta PUT /:id, vinculada al controlador updateProduct.
- **Eliminación de un producto:** Ruta DELETE /:id, vinculada al controlador deleteProduct.

Estas rutas interactúan con el controlador productController.ts, que contiene la lógica para realizar operaciones CRUD en la colección de productos.

---

## 6. Requisito: gestión de promociones

**Descripción:** El sistema debe permitir la gestión de promociones, incluyendo su creación, consulta, actualización y eliminación.

**Cobertura en la Arquitectura:** Implementado en el archivo promotionsRoutes.ts utilizando el framework Express.js. Las rutas RESTful proporcionan las siguientes operaciones:

- **Consulta de promociones:** Ruta GET /, vinculada al controlador getPromotions.
- **Creación de promociones:** Ruta POST /, vinculada al controlador addPromotion.
- **Actualización de promociones:** Ruta PUT /:id, vinculada al controlador updatePromotion.
- **Eliminación de promociones:** Ruta DELETE /:id, vinculada al controlador deletePromotion.

Estas rutas interactúan con el controlador promotionsController.ts, que contiene la lógica de negocio necesaria para manejar las operaciones relacionadas con las promociones.

## 7. Requisito: Gestión de cadena de suministro

**Descripción:** El sistema debe proporcionar funcionalidades para gestionar la cadena de suministro, incluyendo la creación, consulta, actualización y eliminación de registros.

**Cobertura en la Arquitectura:** Implementado en el archivo supplyChainRoutes.ts utilizando el framework Express.js. Las rutas RESTful permiten las siguientes operaciones:

- **Consulta de cadenas de suministro:** Ruta GET /, vinculada al controlador getSupplyChains.
- **Creación de una cadena de suministro:** Ruta POST /, vinculada al controlador addSupplyChain.
- **Actualización de una cadena de suministro:** Ruta PUT /:id, vinculada al controlador updateSupplyChain.
- **Eliminación de una cadena de suministro:** Ruta DELETE /:id, vinculada al controlador deleteSupplyChain.

Estas rutas están asociadas al controlador supplyChainController.ts, encargado de implementar la lógica de negocio necesaria para las operaciones en la base de datos relacionadas con la cadena de suministro.

---

## 8. Requisito: Gestión de usuarios y autenticación

**Descripción:** El sistema debe permitir a los usuarios registrarse, iniciar sesión y realizar operaciones administrativas sobre otros usuarios, como consulta, actualización y eliminación.

**Cobertura en la Arquitectura:** Implementado en el archivo userRoutes.ts utilizando el framework Express.js. Las rutas RESTful gestionan las siguientes operaciones:

- **Registro de usuarios:** Ruta POST /register, vinculada al controlador registerUser.
- **Inicio de sesión:** Ruta POST /login, vinculada al controlador loginUser.
- **Consulta de usuarios:** Ruta GET /, protegida mediante los middlewares protect y admin, vinculada al controlador getUsers.
- **Actualización de usuarios:** Ruta PUT('/:id'), protegida mediante los middlewares protect y admin, vinculada al controlador updateUser.
- **Eliminación de usuarios:** Ruta DELETE '/:id', protegida mediante los middlewares protect y admin, vinculada al controlador deleteUser.

## 9. Requisito: Gestión de Archivos y Subida de Imágenes

**Descripción:** El sistema debe permitir la carga de archivos, específicamente imágenes, a un servicio de almacenamiento en la nube (en este caso, Cloudinary). Una vez cargada la imagen, el sistema debe devolver la URL segura de la imagen para su uso posterior, como en perfiles de usuario, productos, etc.

**Cobertura en la Arquitectura:** Este requisito se implementa en el archivo imageRoutes.ts, utilizando Express.js, Multer y Cloudinary. A continuación, se describen las operaciones relacionadas con la carga de archivos:

- **Carga de Imagen:** Ruta: POST /upload
- **Controlador:** Función anónima en la ruta que maneja la subida de la imagen a Cloudinary.
- **Middleware:** multer (para manejar la carga del archivo de forma temporal en el servidor local)
- **Proceso:**
  1. Multer recibe el archivo de imagen y lo guarda temporalmente en el servidor.
  2. Cloudinary sube la imagen desde el servidor local a su plataforma de almacenamiento en la nube.
  3. Si la carga es exitosa, Cloudinary devuelve una URL segura que se envía como respuesta.
  4. En caso de error, se devuelve un mensaje de error con un estado 500.

Estas rutas interactúan con el controlador userController.ts, donde se implementa la lógica para las operaciones sobre usuarios, y utilizan los middlewares authMiddleware.js para garantizar que solo usuarios autenticados y con permisos administrativos puedan acceder a las rutas protegidas.

---

## 3.0 Views

### 3.1 Bussiness Logic Model View

#### 3.1.1 View Description

Esta vista describe, a un alto nivel, la lógica del negocio, los microservicios disponibles para cada actor dentro del sistema y cómo ocurren las interacciones entre los componentes del sistema.

En el diagrama se representan los actores clave y la manera en que interactúan con los microservicios del sistema:

- **Clientes:** Interactúan principalmente a través de la interfaz de usuario (web o móvil) para realizar compras, ver el historial de pedidos y actualizar su perfil.
- **Administradores de Inventario:** Gestionan el stock, agregan nuevos productos y actualizan el inventario en tiempo real mediante los microservicios de gestión de inventario.

Cada microservicio opera de manera independiente y expone una API que facilita la interacción de los usuarios con los datos en tiempo real. Las interacciones clave incluyen:

- **Gestión de Usuarios:** Proporciona autenticación y autorización de usuarios, garantizando que cada usuario acceda solo a las funcionalidades adecuadas.
- **Gestión de Pedidos:** Permite la creación, actualización y consulta de pedidos, sincronizando en tiempo real el estado de cada pedido.
- **Sincronización de Inventario:** Actualiza y consulta el inventario en tiempo real, asegurando que la información de stock esté disponible para todos los actores.

### 3.1.2.0 View Packet Overview

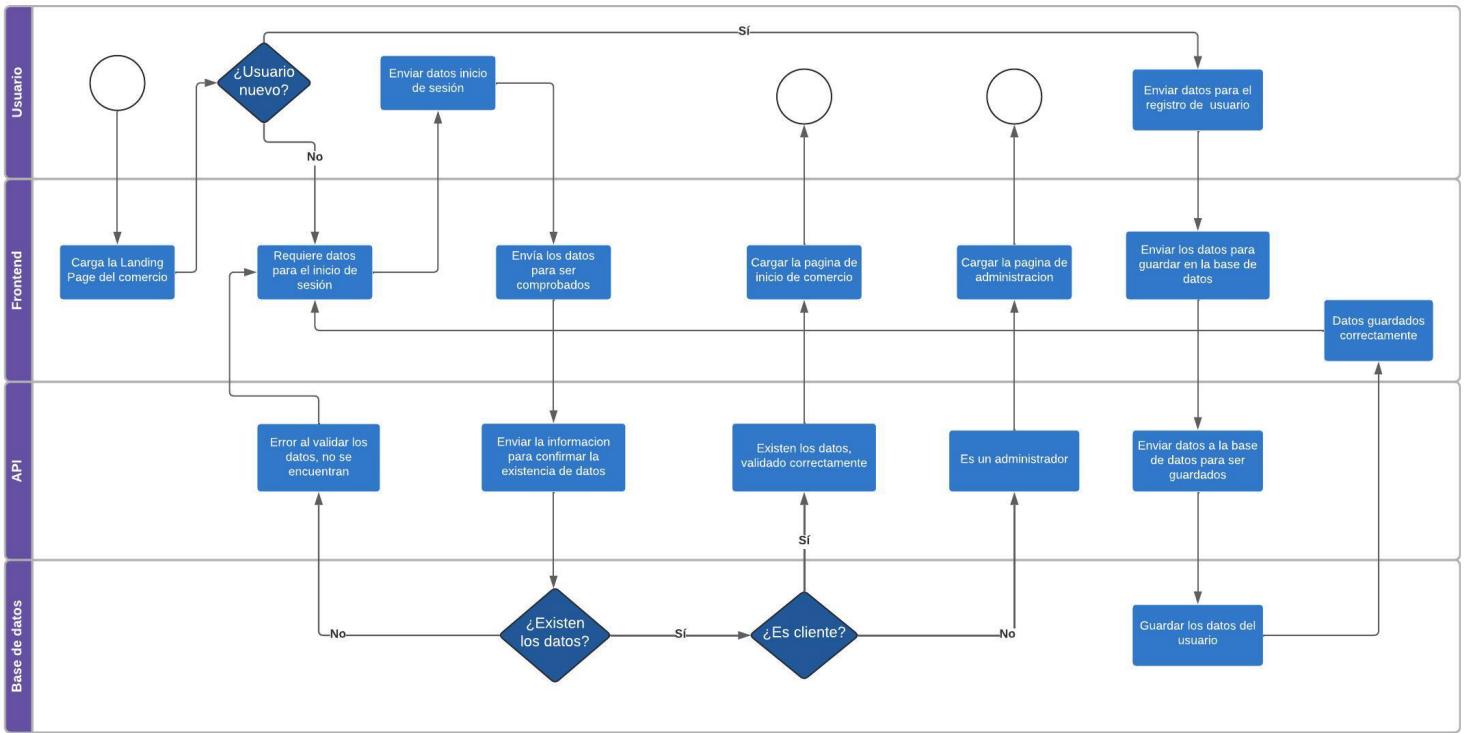


Imagen 3.1. Inicio de sesión

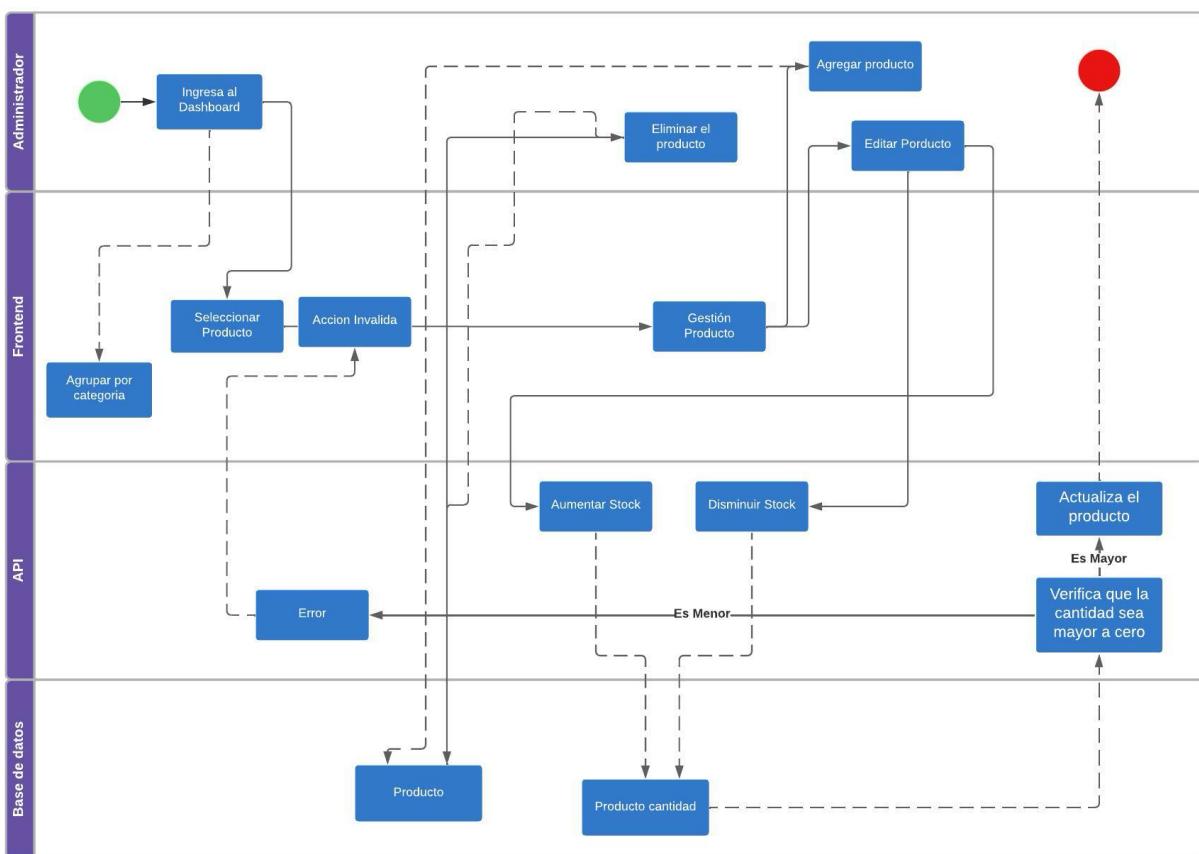


Imagen 3.2. Gestion de Inventario

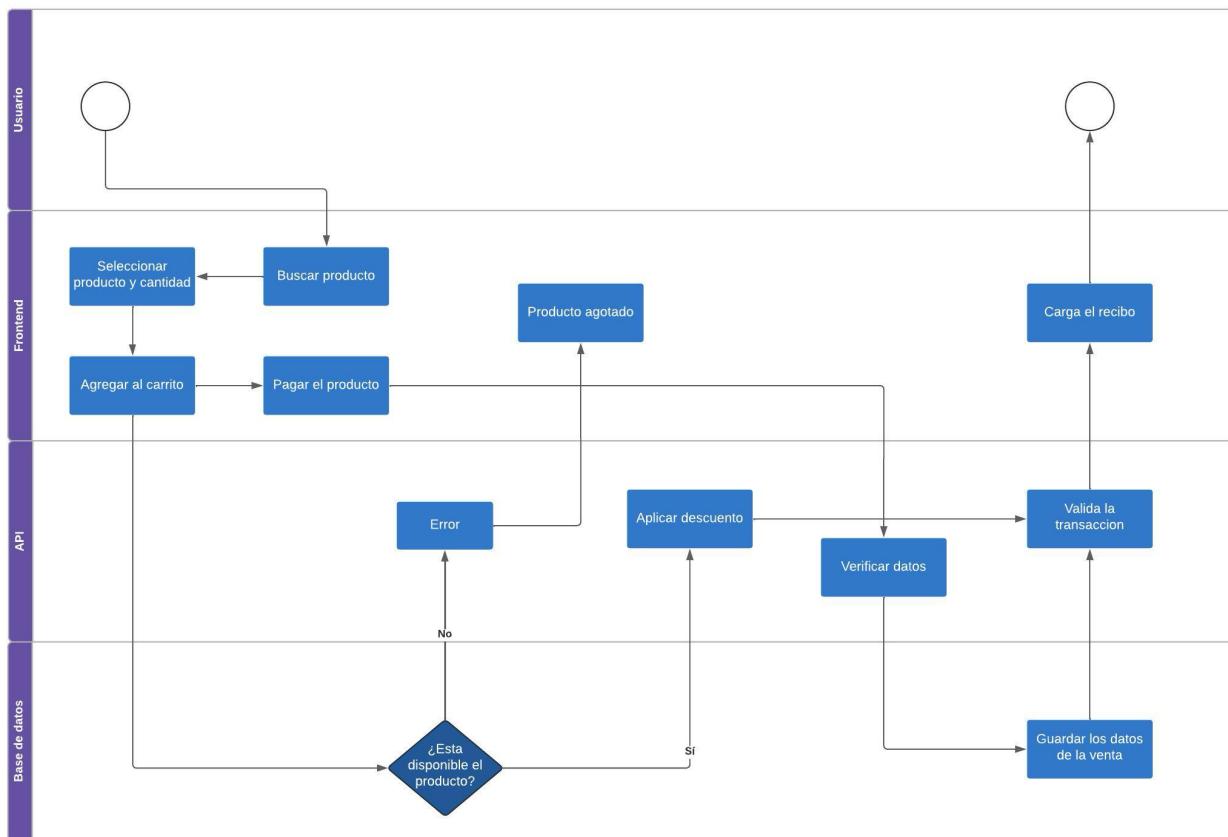


Imagen 3.3. Compra de Productos

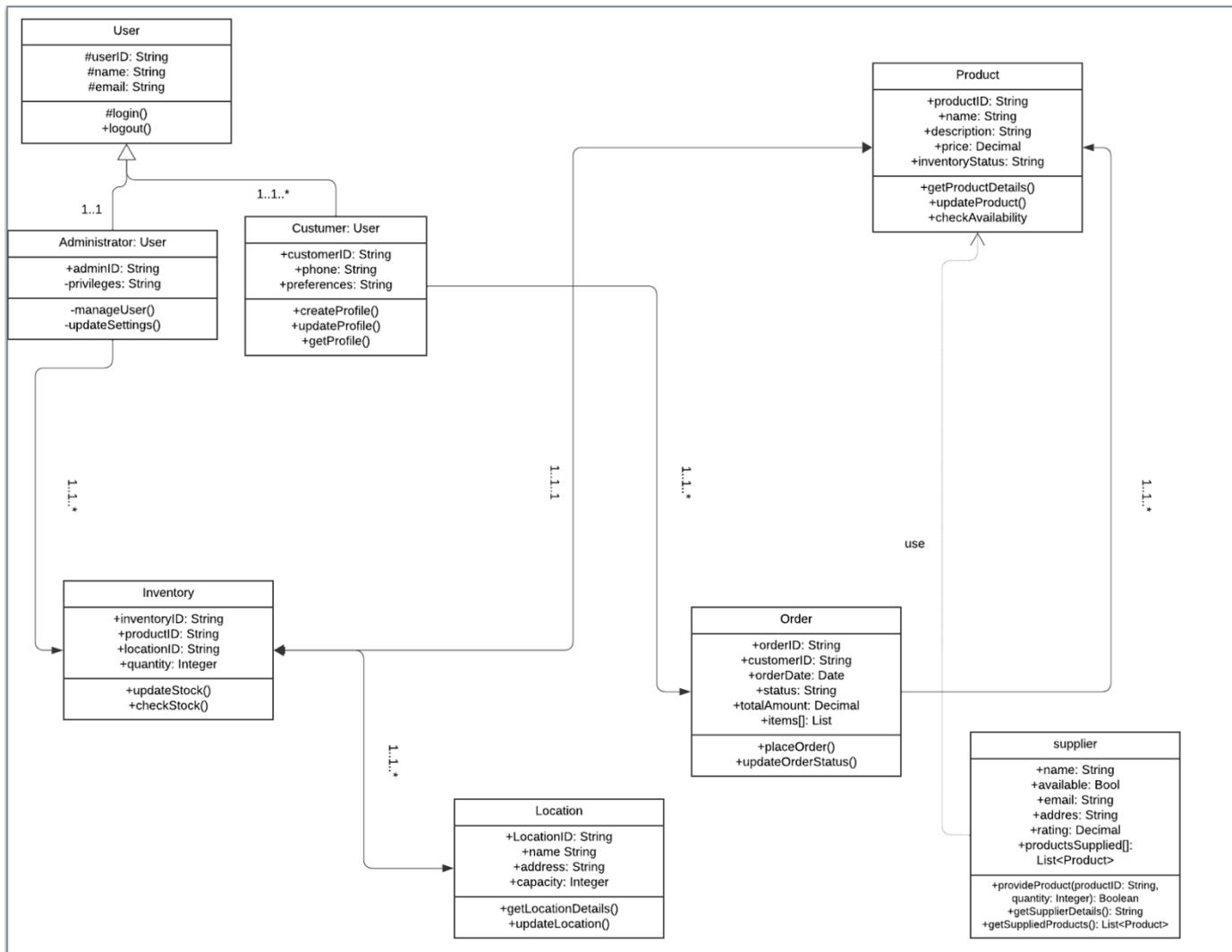


Imagen 3.4. Diagrama de clases

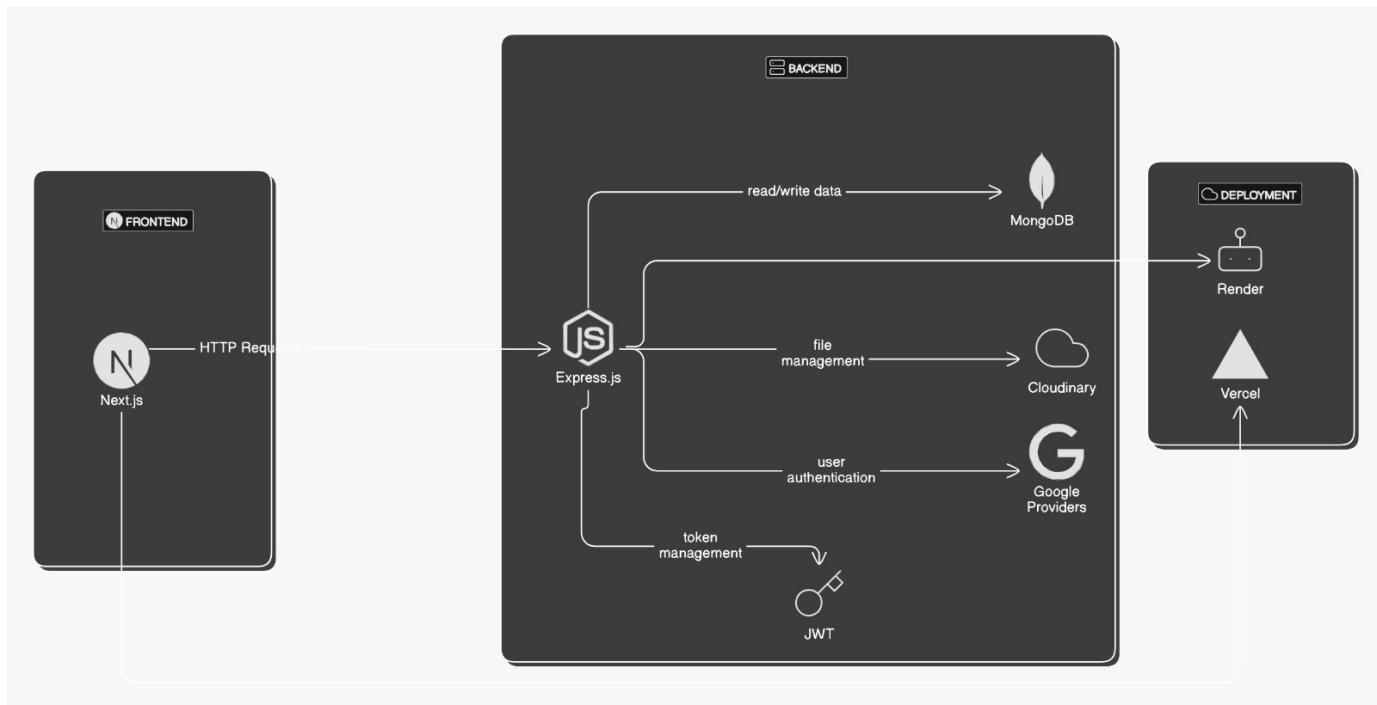
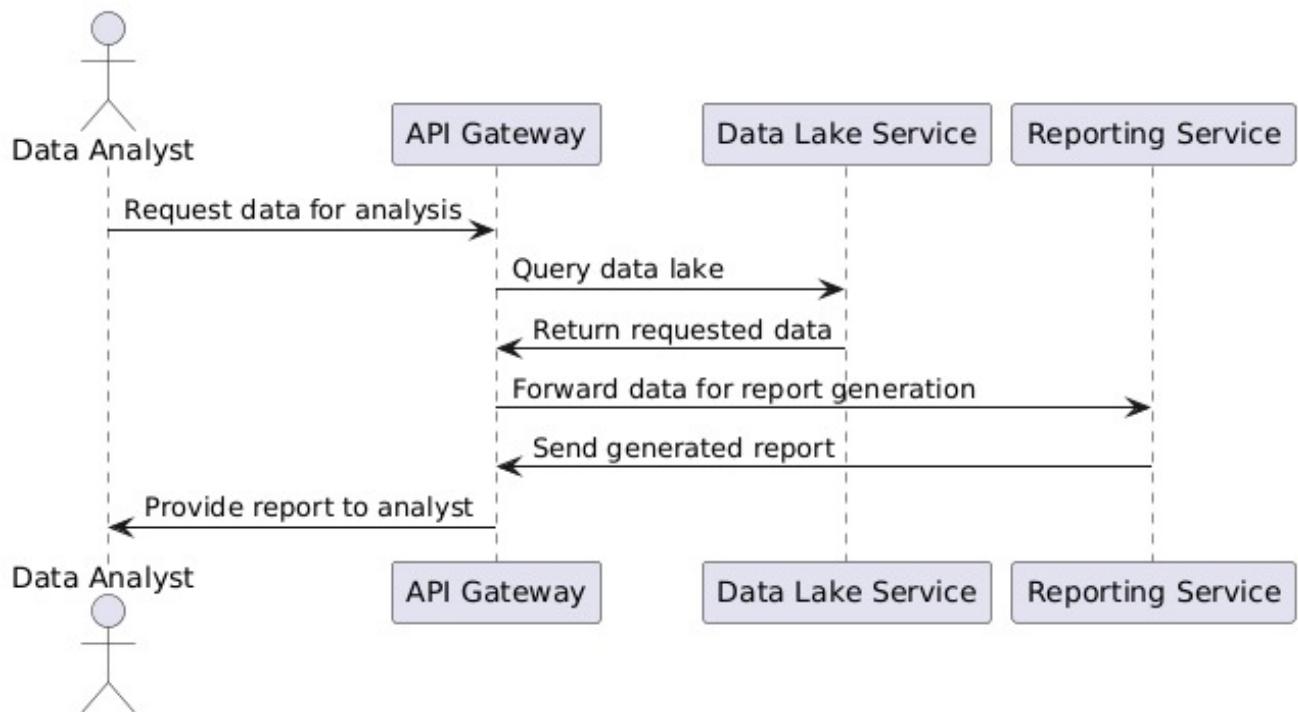
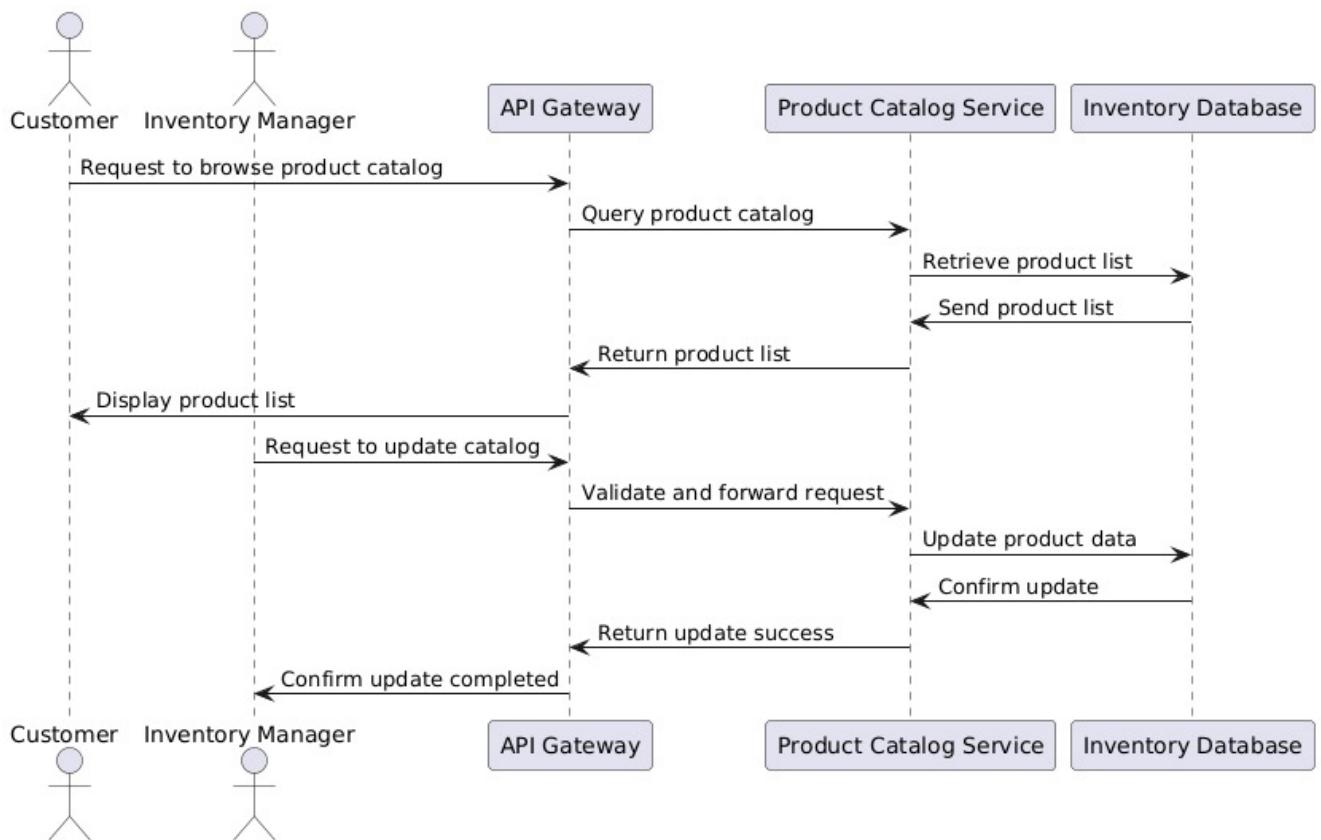
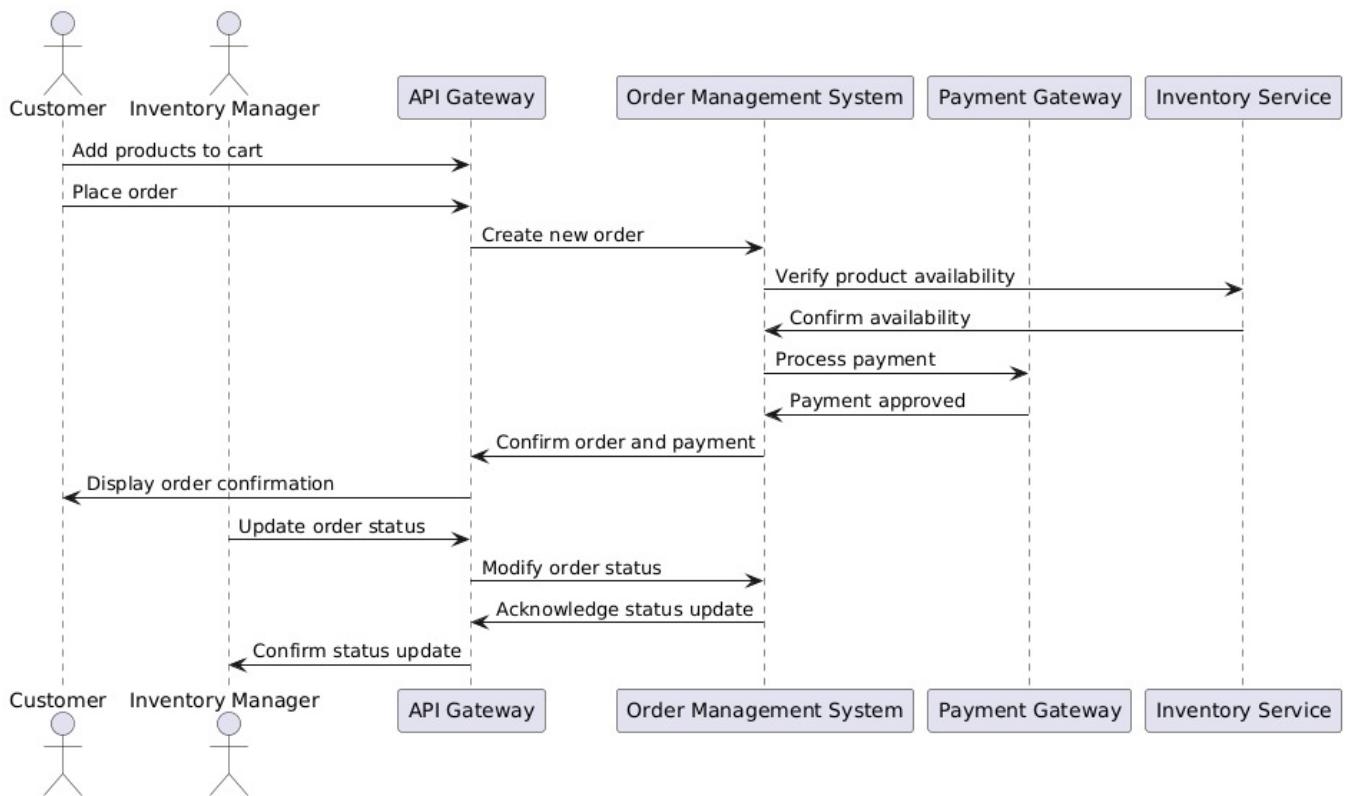


Imagen 3.5. Diagrama de despliegue





*Imágenes 3.6. Diagramas de Secuencia (Inv; Dt y Gest.Ped.)*

---

## 3.2 Web App View

### 3.2.0 View Description

Esta vista proporciona una representación detallada que permite a los usuarios visualizar todos los componentes de la interfaz con los que interactúan. Además, ofrece una comprensión clara de las diversas funcionalidades disponibles, adaptadas específicamente para cada tipo de usuario. Esto no solo facilita la navegación y el uso del sistema, sino que también optimiza la experiencia del usuario al asegurarse de que cada persona pueda acceder a las herramientas y opciones que mejor se ajustan a sus necesidades particulares.

- **Visibilidad y Claridad:** La página principal, con el mensaje de bienvenida “Welcome to 4F Wears” y el botón destacado de “Shop New Arrivals”, crea un enfoque claro para que los usuarios comiencen su navegación. Esto orienta rápidamente a los usuarios hacia las novedades, generando un punto de entrada atractivo.
- **Categorías Destacadas:** La sección de “Featured Categories” permite a los usuarios explorar categorías específicas (como “Shop Women”, “Men’s Wear” y “Jewelry”) de manera sencilla. Esto está diseñado para diferentes tipos de usuarios con preferencias variadas, facilitando la personalización de la experiencia de compra.
- **Productos Nuevos:** La sección de “New Arrivals” exhibe productos recientes de manera visualmente accesible y funcional. Con precios y botones de “Add to Cart” debajo de cada artículo, esta estructura facilita la acción de compra, mejorando la eficiencia para aquellos interesados en productos específicos.
- **Navegación Intuitiva y Accesibilidad:** Las opciones de visualización y navegación en el sitio están alineadas para asegurar que todos los usuarios, independientemente de sus preferencias, puedan interactuar con los elementos principales de la tienda. Las herramientas de navegación y las categorías visibles hacen que sea fácil para los usuarios acceder a las opciones que mejor se adaptan a sus necesidades.
- **Optimización de la Experiencia del Usuario:** Al proporcionar vistas específicas y secciones bien definidas, el sitio permite que cada usuario tenga una experiencia personalizada y enfocada en su interés principal. La estructura de la interfaz es limpia, con opciones de acción directa, optimizando así la navegación y el flujo de compra.

### 3.2.1 View Packet Overview

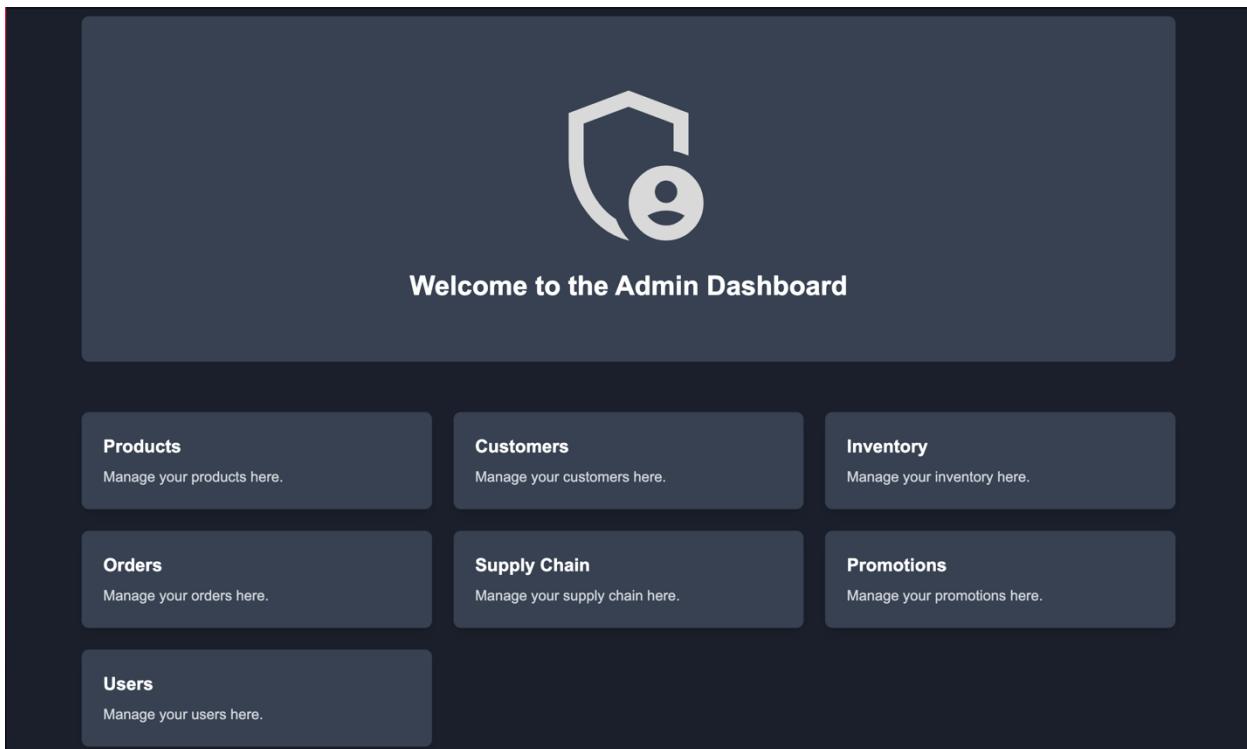


Imagen 3.7. DashBoard de Administrador

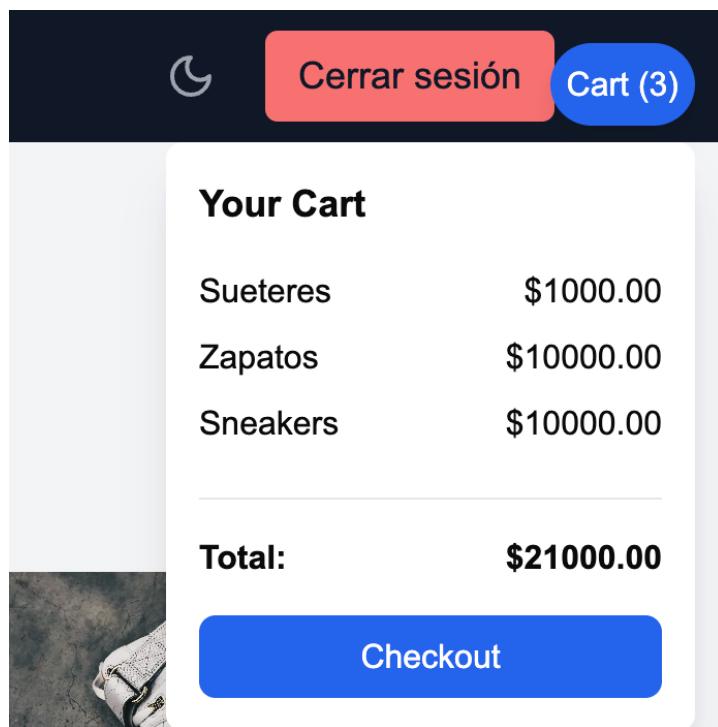


Imagen 3.8. Vista de Carrito

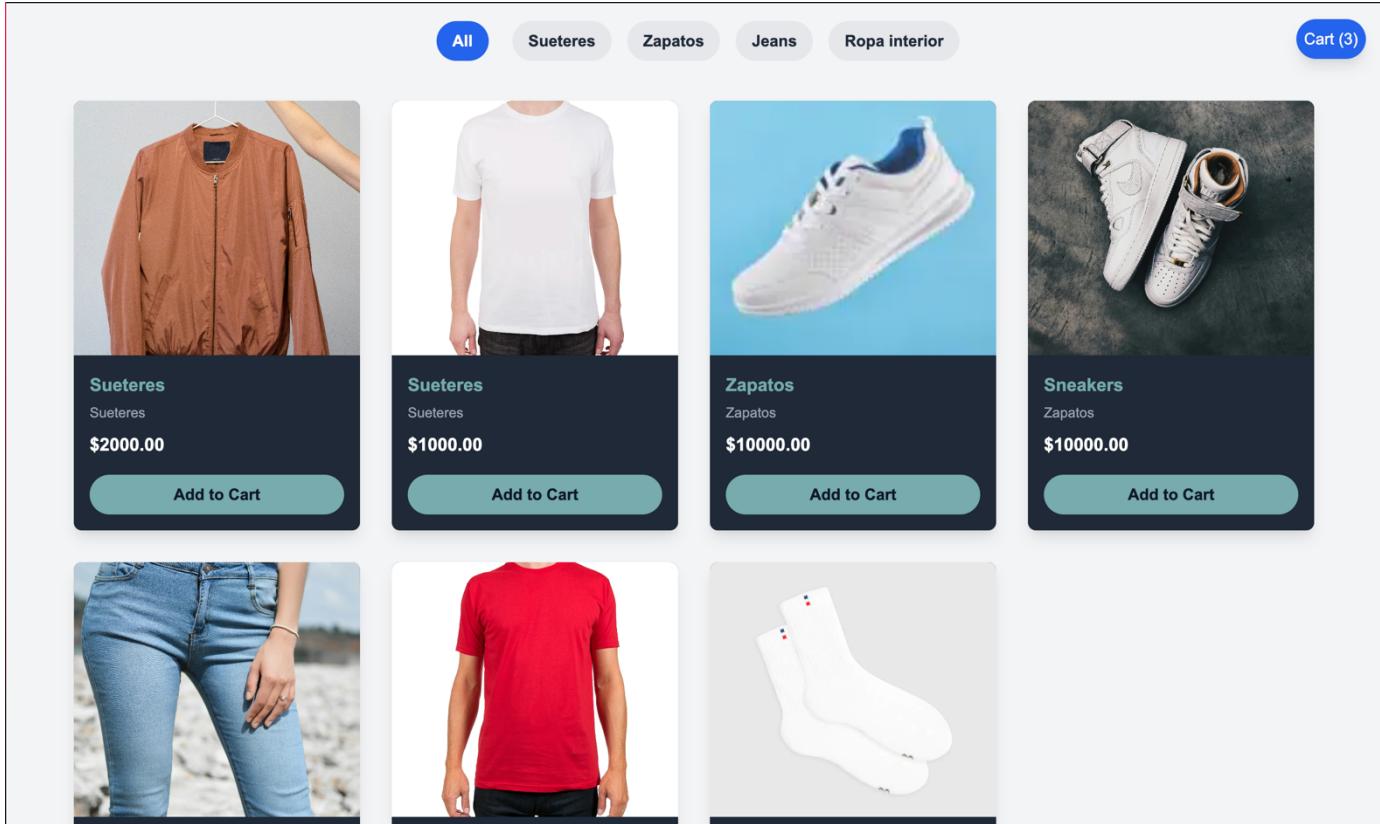


Imagen 3.9. vista de Producto

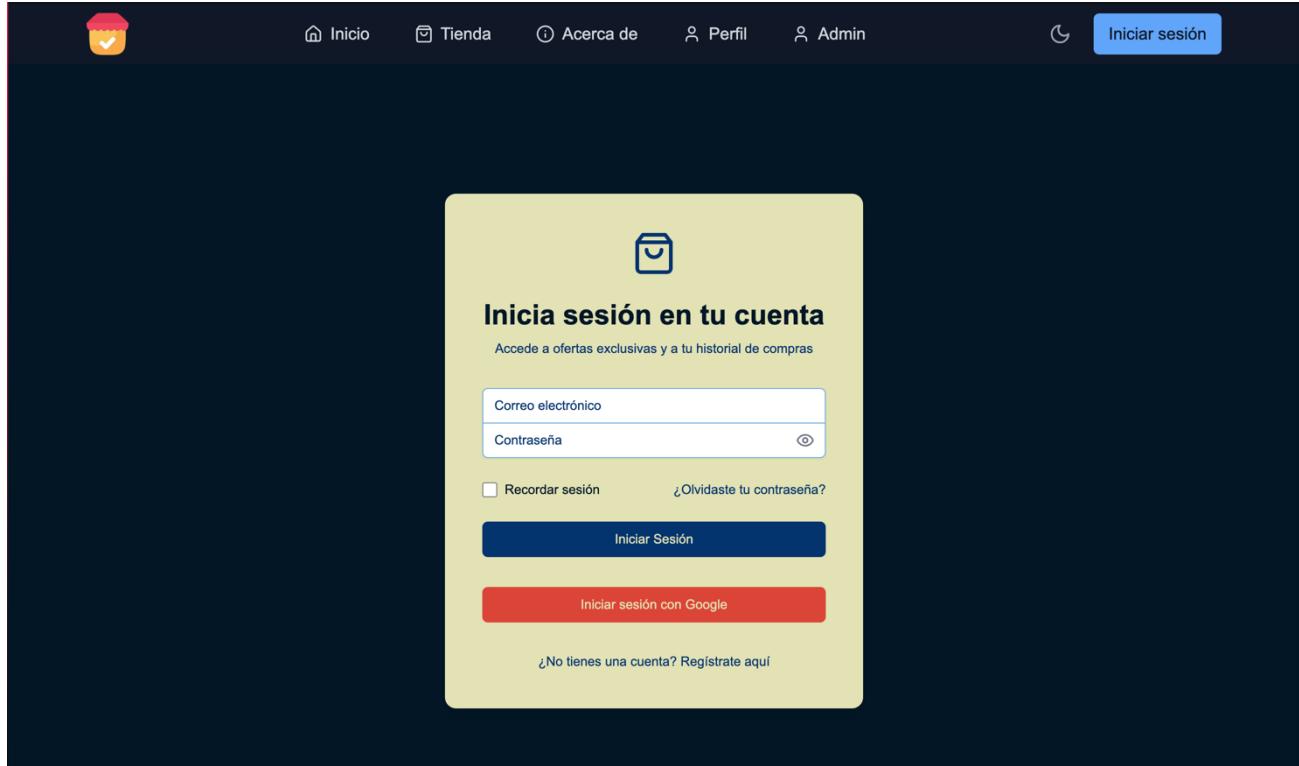


Imagen 3.10. Vista de iniciar sesión

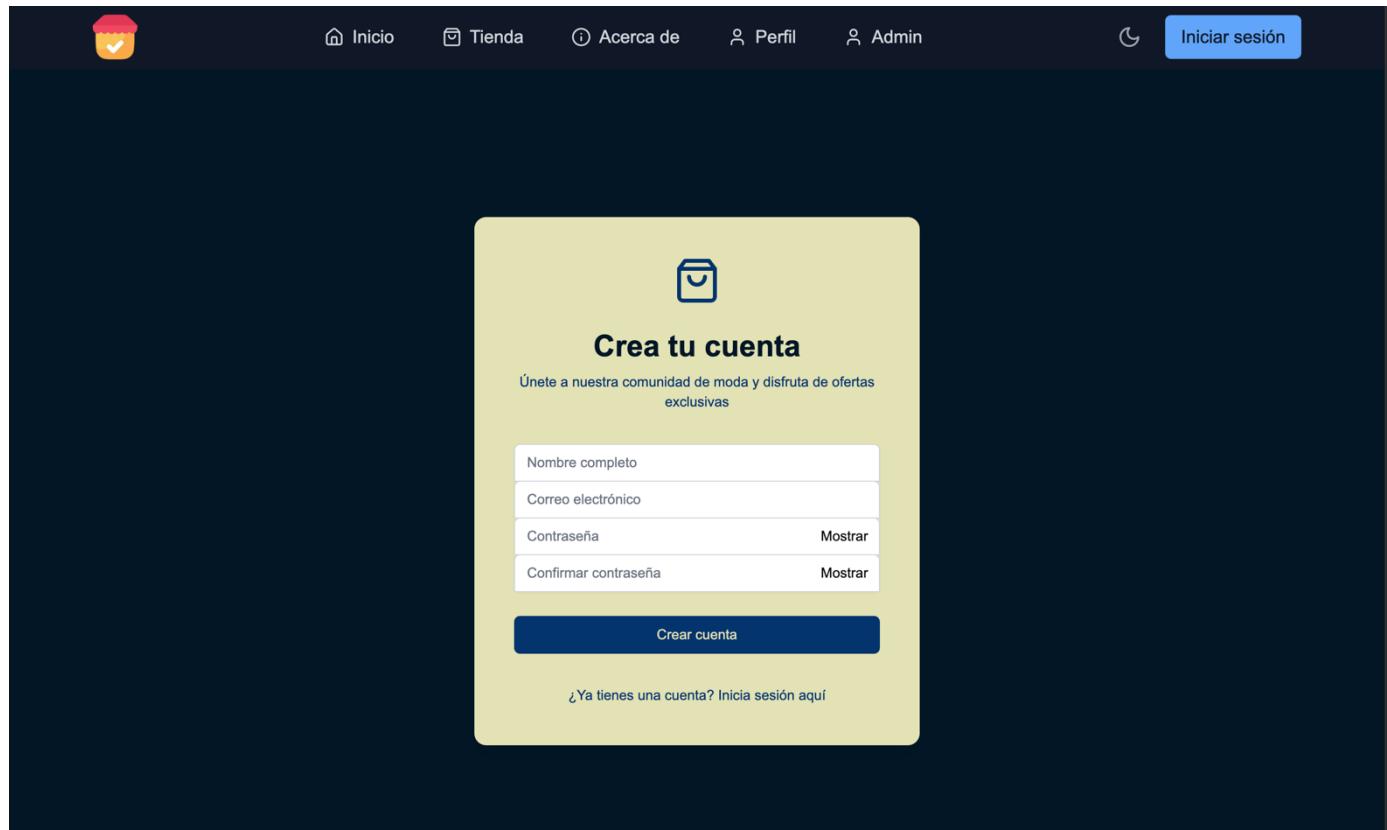


Imagen 3.11. Vista crear tu cuenta

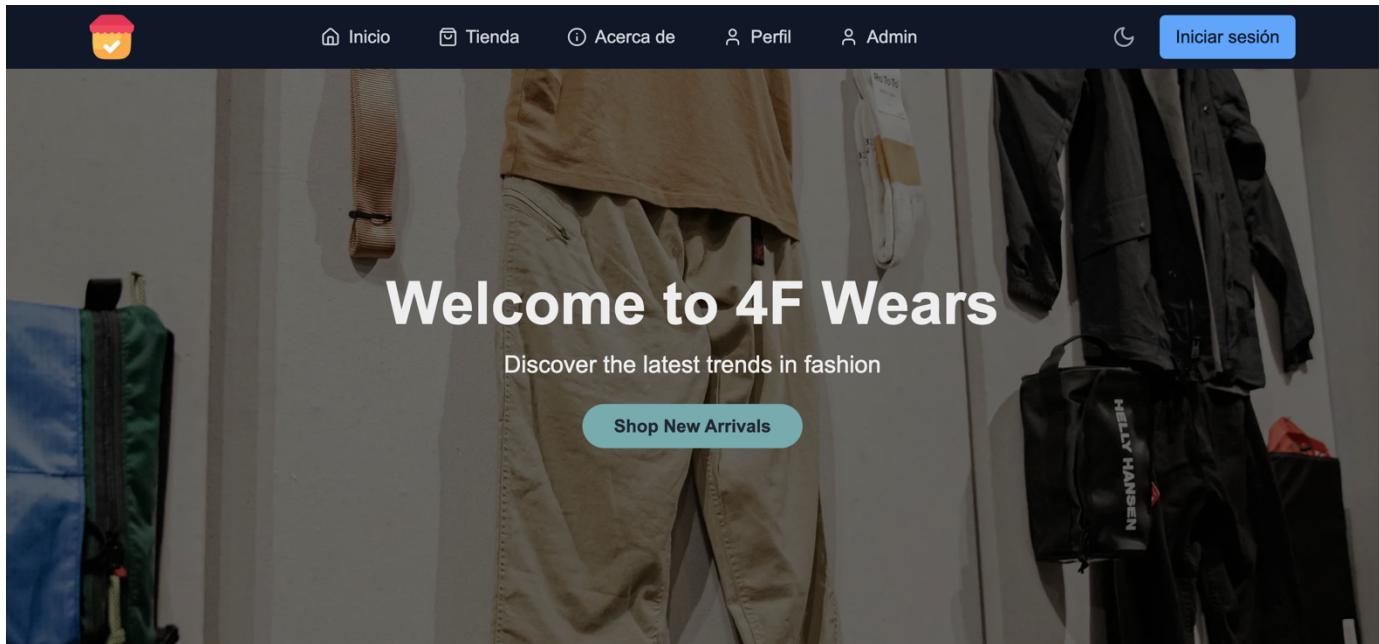


Imagen 3.12. Mockup Vista de Inicio

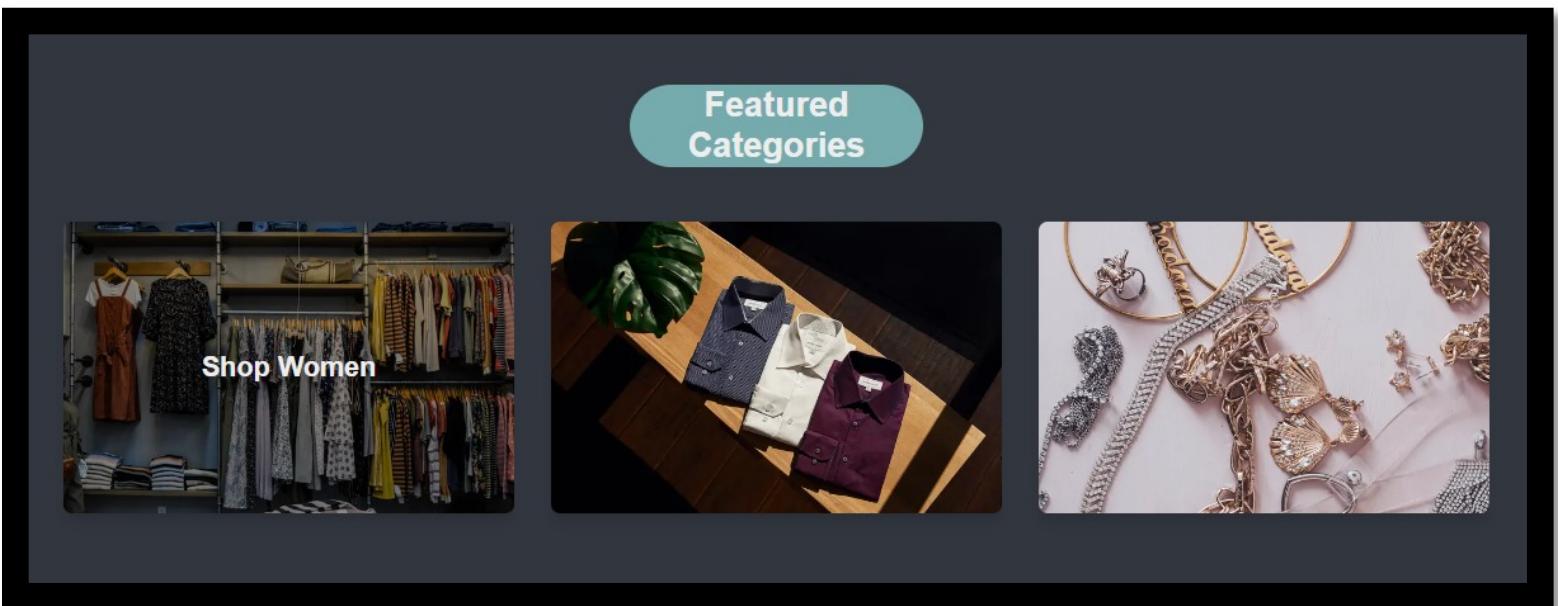


Imagen 3.13. Vista Featured Categories

A screenshot of a mobile application interface titled "4F WEARS". At the top left, it says "New Spring 2023 Collections". In the center, there is a large text block: "Reflect Who You Are with Our Style". Below this text is a paragraph: "The power of a great outfit is impossible to overstate. At its best, fashion has the ability to transform your mood, identity, and of course, your look. It can be refreshing and purposeful." To the right of the text is a collage of four images featuring a woman in a red coat and sunglasses holding shopping bags. At the bottom, there are two teal-colored rounded rectangles with the text "Shop Now" and "Learn More".

Imagen 3.14. Vista Home

## Our Features



Sustainable Fashion



Personalized Style



Quality Craftsmanship

### Featured Collection



**Spring Awakening**

Embrace the season of renewal with our latest collection. Featuring light fabrics, pastel hues, and nature-inspired patterns, this line embodies the freshness and vitality of spring.

[Explore Collection](#)

Imagen 3.14.1 Vista Our features

### Innovación en Cada Detalle

Desde nuestro proceso de diseño asistido por IA hasta nuestras tiendas interactivas, cada aspecto de MODA Innovadora está diseñado para ofrecerte una experiencia de compra única y personalizada.



**Diseño Inteligente**



**Calidad Superior**

Materiales sostenibles y técnicas de fabricación avanzadas.



**Estilo a Medida**

Imagen 3.14.2 Vista innovación

**Vogue Verse Dashboard**

**Products**

**Customers**

**Orders**

**Supply Chain**

**Promotions**

**Customers**

**Customers**

_ID	NAME	EMAIL	ADDRESS	PHONE	_V	ACCIONES
671c7dde07e42a93d19bd508	Paula Marquez	Marquez123@gmail.com	Turbaco	3018723567	0	<a href="#">Editar</a> <a href="#">Eliminar</a>
672269245dd2e55e74aaa893	Jose pepito	Josepepito@gmail.com	Manga	3027656382	0	<a href="#">Editar</a> <a href="#">Eliminar</a>
6723ea0647ff134ba8889e73	Adrian Vizcaino	Adrian@gmail.com	13 de Junio	3018725183	0	<a href="#">Editar</a> <a href="#">Eliminar</a>

Imagen 3.15. Vista Admin en gestión de Usuario

**Vogue Verse Dashboard**

**Products**

**Customers**

**Orders**

**Supply Chain**

**Promotions**

**Customers**

**Customers**

_ID	NAME	EMAIL	ADDRESS	PHONE	_V	ACCIONES
671c7dde07e42a93d19bd508	Paula Marquez	Marquez123@gmail.com	Turbaco	3018723567	0	<a href="#">Editar</a> <a href="#">Eliminar</a>
672269245dd2e55e74aaa893	Jose pepito	Josepepito@gmail.com	Manga	3027656382	0	<a href="#">Editar</a> <a href="#">Eliminar</a>
6723ea0647ff134ba8889e73	Adrian Vizcaino	Adrian@gmail.com	13 de Junio	3018725183	0	<a href="#">Editar</a> <a href="#">Eliminar</a>

Imagen 3.16. Vista Admin Gestión de Clientes

## Inventory Management

### Add New Inventory Item

Add Item

---

**Manilla**

SKU: 121  
Quantity: 10  
\$ 10000.00

Imagen 3.17. Vista Admin en gestión de Inventario

## Order Management

### Add New Order

Add Order

---

**Order #**

👤 Mauricio Donado  
📅 23/11/2024  
฿ N/A

Status: pending

## Promotions Management

Add New Promotion

Code	Description	Discount	Start Date	End Date	Status	Actions
3213	Promoción para sueteres	40%	21/11/2024	22/11/2024	Active	<button>Edit</button> <button>Delete</button>

Imagen 3.18. Vista Admin para Ordenes y Promociones

## Supply Chain Management

Add New Item

Item Name	SKU	Quantity	Supplier	Status	Expected Delivery	Actual Delivery	Actions
Bolso	13432	10	Vogue	Ordered	21/11/2024	N/A	<button>Edit</button> <button>Delete</button>
Tela negra	4231	100	Velez	Ordered	27/11/2024	N/A	<button>Edit</button> <button>Delete</button>

Imagen 3.19. Vista Admin para gestión de la cadena de suministros

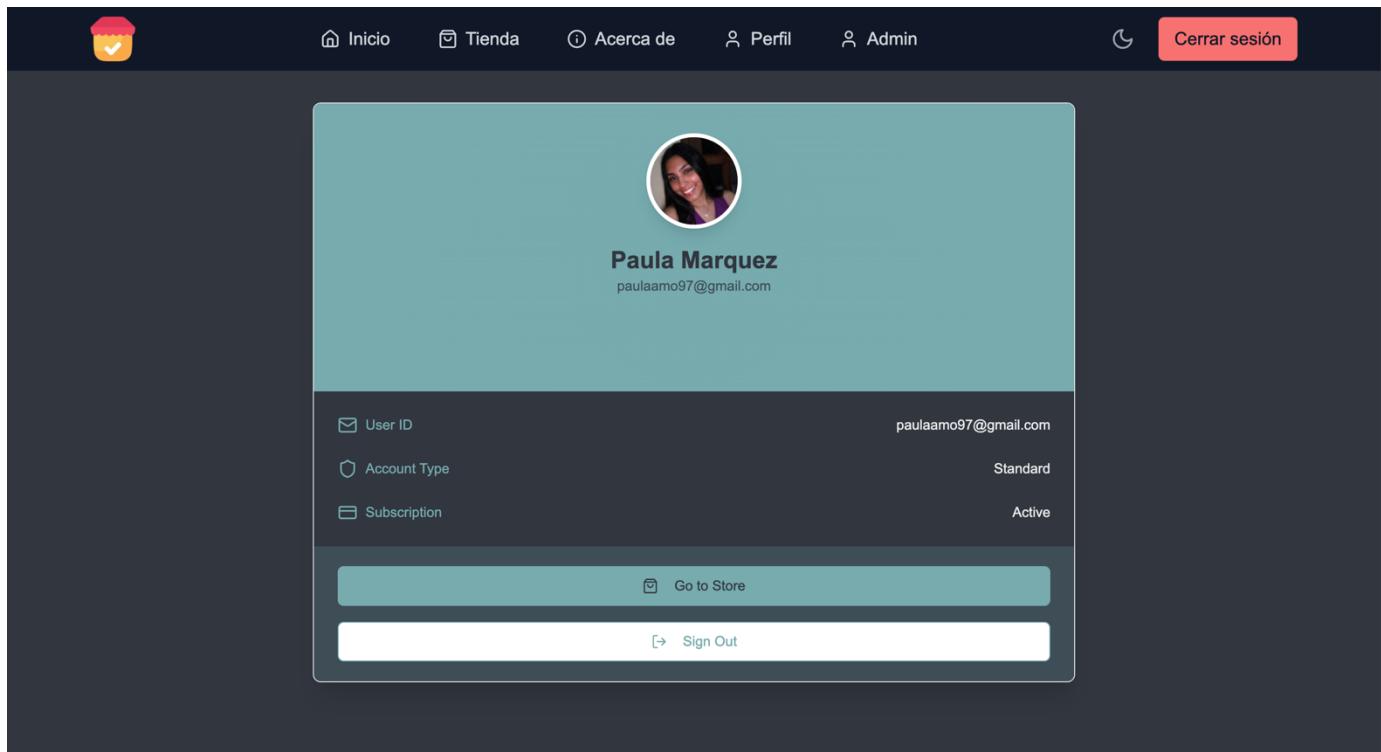


Imagen 3.20.20 Vista del perfil de Usuario

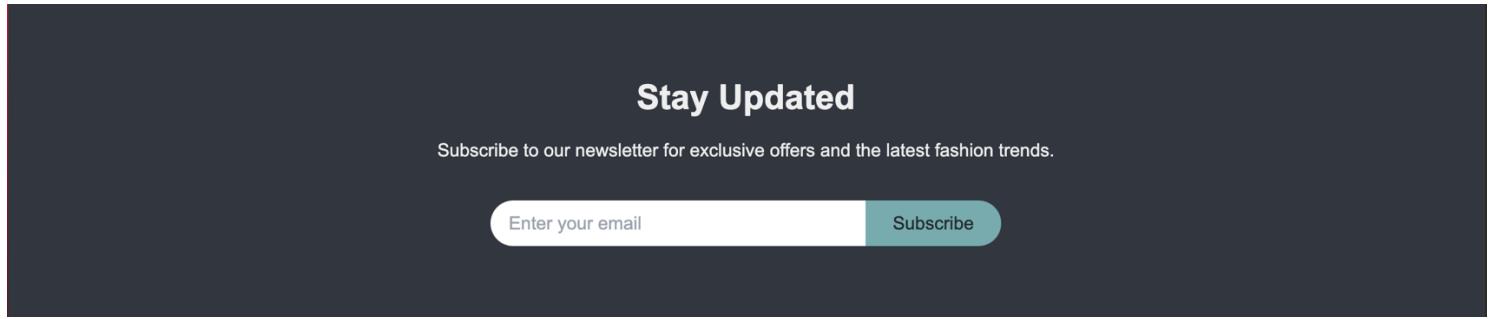


Imagen 3.21.21 Vista de Stay Update

## About 4F Wears

Discover the latest fashion trends and express your unique style with 4F Wears.

## Quick Links

- [About Us](#)
- [Contact](#)
- [FAQ](#)
- [Shipping & Returns](#)

## Follow Us



© 2024 4F Wears. All rights reserved.

*Imagen 3.22. Vista de pie de pag informativo*

## Quick Links

- [About Us](#)
- [Collections](#)
- [Sustainability](#)
- [Contact](#)

## Stay Connected

Subscribe to our newsletter for the latest updates and exclusive offers.

Your email

Subscribe

© 2023 4F WEARS. All rights reserved.

*Imagen 3.23. Vista de Pie de página con formulario de suscripción*

---

## Somos 4F WEARS

### Nuestra Visión

En 4f Wears, combinamos alta costura con tecnología de vanguardia para crear prendas que van más allá de la moda. Cada pieza es una obra maestra de diseño y funcionalidad, concebida para el consumidor moderno que valora la calidad, el estilo único y la sostenibilidad.



*Imagen 3.24. Vista de Nuestra Visión*

---

## 4 Referenced Materials

Barbacci 2003	Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. <i>Quality Attribute Workshops (QAWS)</i> , Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. < <a href="http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html">http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html</a> >.
Bass 2003	Bass, Clements, Kazman, <i>Software Architecture in Practice</i> , second edition, Addison Wesley Longman, 2003.
Clements 2001	Clements, Kazman, Klein, <i>Evaluating Software Architectures: Methods and Case Studies</i> , Addison Wesley Longman, 2001.
Clements 2002	Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Staf- ford, <i>Documenting Software Architectures: Views and Be- yond</i> , Addison Wesley Longman, 2002.
IEEE 1471	ANSI/IEEE-1471-2000, IEEE Recommended Practice for Ar- chitectural Description of Software-Intensive Systems, 21 September 2000.

Tabla 3 Referenced Material

---

## 5 Directory

### 5.4 Index

### 5.5 Glossary

Term	Definition
Software architecture	<p>The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.</p>
view	<p>A representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint.</p>
view packet	<p>The smallest package of architectural documentation that could usefully be given to a stakeholder. The documentation of a view is composed of one or more view packets.</p>
viewpoint	<p>A specification of conventions for creating and using a view, providing a template to develop individual views by defining their purpose, audience, and techniques for creation and analysis [IEEE 1471]. It outlines the concerns to address, and the modeling, evaluation, and consistency-checking techniques used in each view.</p>

Tabla 4 Glosario

---

## 5.6 Acronym List

<b>API</b>	<b>Application Programming Interface; Application Program Interface; Application Programmer Interface</b>
<b>CMM</b>	<b>Capability Maturity Model</b>
<b>CMMI</b>	<b>Capability Maturity Model Integration</b>
<b>CORBA</b>	<b>Common object request broker architecture</b>
<b>COTS</b>	<b>Commercial-Off-The-Shelf</b>
<b>EPIC</b>	<b>Evolutionary Process for Integrating COTS-Based Systems</b>
<b>IEEE</b>	<b>Institute of Electrical and Electronics Engineers</b>
<b>KPA</b>	<b>Key Process Area</b>
<b>OO</b>	<b>Object Oriented</b>
<b>ORB</b>	<b>Object Request Broker</b>
<b>OS</b>	<b>Operating System</b>
<b>QAW</b>	<b>Quality Attribute Workshop</b>
<b>RUP</b>	<b>Rational Unified Process</b>
<b>SAD</b>	<b>Software Architecture Document</b>
<b>SDE</b>	<b>Software Development Environment</b>
<b>SEE</b>	<b>Software Engineering Environment</b>
<b>SEI</b>	<b>Software Engineering Institute</b> <b>Systems Engineering &amp; Integration</b> <b>Software End Item</b>
<b>SEPG</b>	<b>Software Engineering Process Group</b>
<b>SLOC</b>	<b>Source Lines of Code</b>
<b>SW-CMM</b>	<b>Capability Maturity Model for Software</b>
<b>CMMI-SW</b>	<b>Capability Maturity Model Integrated - includes Software Engineering</b>
<b>UML</b>	<b>Unified Modeling Language</b>

*Tabla 5 Lista de Acronimos*