**Proyecto: A Hui Hou**

**Alumnas: Laura Gabriela Balcazar Cardozo**

**Docente: Rodrigo Castro Caicedo**

**Universidad Libre**  
**Facultad de Ingeniería**  
**Ingeniería de Sistemas**  
**Ingeniería de Software II**  
**5 semestre**  
**Corte 2**  
**2025-2**

**Tabla de contenido**  
 **Portada ............................................................................................................................. 1**  
**Tabla de Contenido ......................................................................................................... 2**  
 **Introducción ..................................................................................................................... 3**  
 **Idea principal ................................................................................................................... 3**  
**Objetivo General ............................................................................................................. 3**  
**Objetivos Específicos ....................................................................................................... 3**  
**Requisitos Funcionales .................................................................................................... 4**  
 **Casos de Uso ............................................................................................................... 5 y 6**  
 **Tablas de Requerimientos Funcionales ............................................................... 7, 8 y 9**  
 **Tablas de Requerimientos No Funcionales ................................................... 10, 11 y 12**  
 **Historias de Usuario .............................................................................................. 13 y 14**  
 **Diagrama de Clase ........................................................................................................ 15**   
 **Diagrama de Despliegue ............................................................................................... 16**  
 **Diagrama de Actividades ............................................................................................. 17**  
 **Diagramas de Secuencias (Organizados, Reestructurados y Redefinidos ....... 18 y 19**  
 **Figma del proyecto ........................................................................................................ 20**  
 **Patrones Arquitectónicos .............................................................................................. 21**  
 **Patrones de Diseño GoF ............................................................................ 22, 23, 24 y 25**  
 **GitHub ............................................................................................................................ 26**

**Introducción**

El sector minorista de alimentos, en particular el de frutos secos y productos saludables, se enfrenta a un reto común: la gestión manual de inventarios, ventas y pedidos. Este proceso suele generar errores, pérdida de tiempo y dificultades en el control del negocio. En el caso de la tienda A Hui Hou, actualmente se utilizan cuadernos para el registro de inventario y WhatsApp Business para los pedidos, lo que limita la eficiencia y el crecimiento.  
El presente proyecto busca transformar este modelo de operación mediante el diseño y desarrollo de una solución digital que permita centralizar y automatizar los procesos, mejorando la productividad y ofreciendo un servicio más ágil a los clientes.

**Idea Principal**

Diseñar y desarrollar una aplicación multiplataforma que gestioné de forma digital el inventario, las ventas y los pedidos a domicilio de A Hui Hou, facilitando tanto la administración interna del negocio como la experiencia del cliente.

**Objetivo General**

Desarrollar una solución digital integral que permita a A Hui Hou gestionar de forma eficiente sus procesos de inventario y ventas, reduciendo errores manuales, optimizando recursos y mejorando la atención al cliente.

**Objetivos Específicos**

1. Analizar los procesos actuales de gestión en A Hui Hou para identificar oportunidades de mejora.
2. Diseñar una base de datos estructurada que permita el control eficiente de productos y ventas.
3. Implementar un módulo de inventario con actualización automática.
4. Diseñar una interfaz sencilla, accesible y multiplataforma.

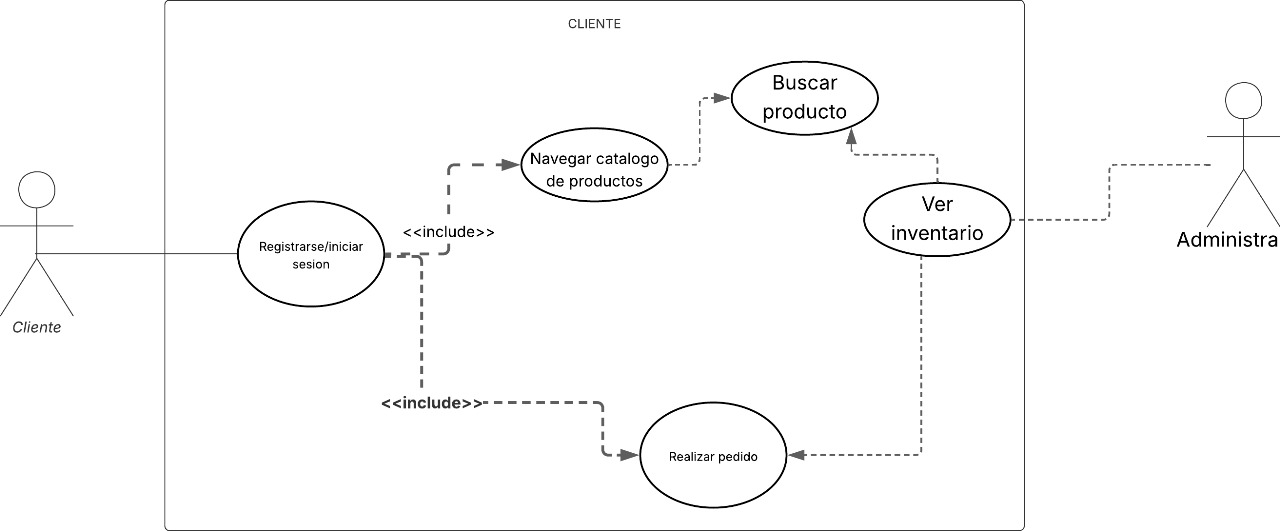
**Requisitos funcionales**

**Tenemos los principales requisitos los cuales son:**

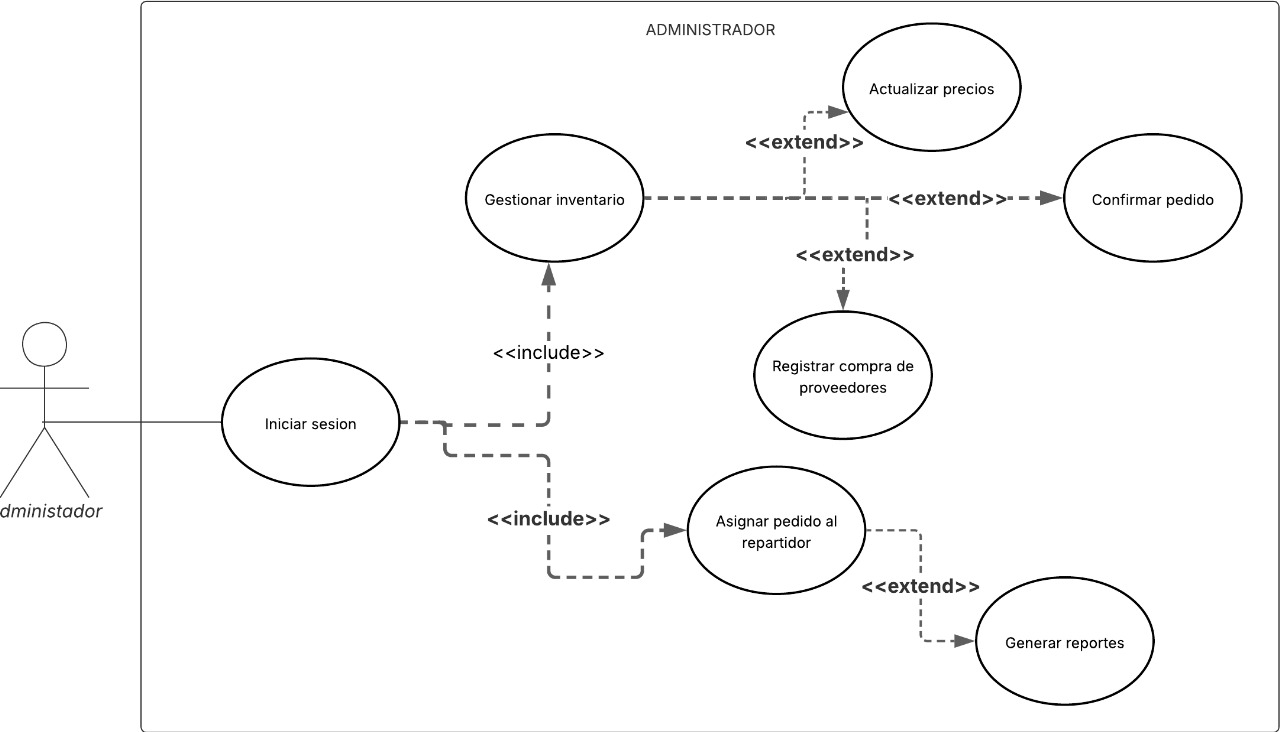
|  |  |
| --- | --- |
| **Usuarios y roles** | * El sistema debe permitir el registro y autenticación de usuarios (administrador, empleados). * El administrador debe poder gestionar los permisos y roles de cada usuario. * El cliente debe poder registrarse con correo, número de teléfono o redes sociales. |
| **Inventario** | * El sistema debe permitir registrar productos con nombre, categoría, precio, stock y fecha de caducidad. * El administrador debe poder cargar y modificar productos de forma masiva (ejemplo: archivo Excel/CSV) |
| **Ventas** | * El sistema debe registrar cada venta realizada en la tienda física y en línea. * El sistema debe permitir la consulta de ventas por rango de fechas, producto o cliente. |
| **Clientes** | * El sistema debe registrar la información básica de cada cliente (nombre, teléfono, correo, historial de compras) * El cliente debe poder acceder a su historial de pedidos y otro tipo de información |
| **Reportes y estadísticas** | * El sistema debe generar reportes de ventas, inventarios y pedidos en formatos descargables (PDF/Excel) |
| **Administración general** | * El sistema debe permitir configurar parámetros básicos (horarios de entrega, métodos de pago, impuestos). * El sistema debe registrar logs de actividades de los usuarios (ventas, modificaciones de inventario, creación de pedidos). |

**Casos de uso:**

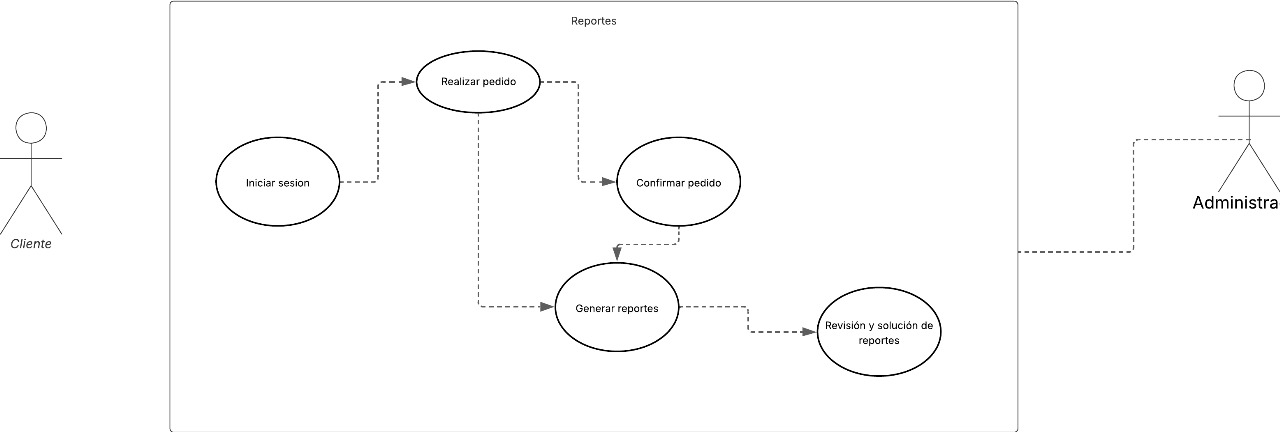
**Clientes:**



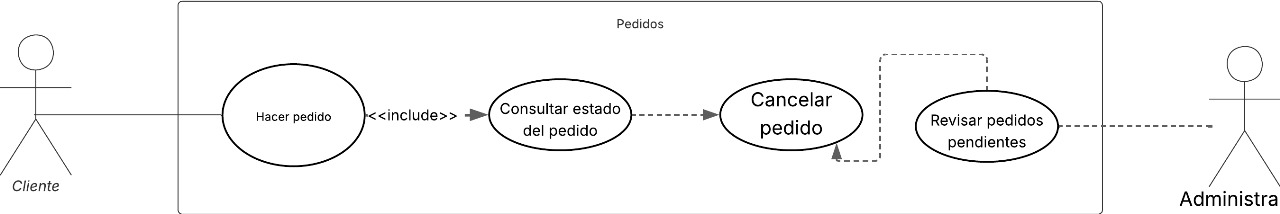
**Administrador**



**Reportes**



**Pedidos**



**Tablas de requerimientos:**

**Requerimientos Funcionales:**

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RF01 |
| **Nombre del Requerimiento:** | Registro y autenticación de usuarios |
| **Características:** | El sistema debe permitir la creación de cuentas para administrador, empleados y clientes |
| **Descripción del Requerimiento:** | Los usuarios podrán registrarse con correo, número de teléfono o redes sociales. Los administradores podrán asignar roles y gestionar permisos. |
| **Observaciones:** | Se debe asegurar que la autenticación sea segura. |
| **Prioridad del Requerimiento** | Alta |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RF02 |
| **Nombre del Requerimiento:** | Gestión de roles y permisos |
| **Características:** | Control diferenciado según tipo de usuario. |
| **Descripción del Requerimiento:** | El administrador debe poder definir los permisos de empleados y usuarios en el sistema. |
| **Observaciones:** | Facilita el control y la seguridad. |
| **Prioridad del Requerimiento** | Alta |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RF03 |
| **Nombre del Requerimiento:** | Registro de productos (Inventario) |
| **Características:** | Ingreso de productos con nombre, categoría, precio y stock |
| **Descripción del Requerimiento:** | El administrador podrá agregar, editar y eliminar productos. |
| **Observaciones:** | Los productos deben actualizarse en tiempo real. |
| **Prioridad del Requerimiento** | Alta |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RF04 |
| **Nombre del Requerimiento:** | Registro de ventas |
| **Características:** | Almacenamiento de cada venta en tienda física y en línea. |
| **Descripción del Requerimiento:** | El sistema debe guardar información de fecha, cliente y productos vendidos. |
| **Observaciones:** | Permitir integración futura con métodos de pago. |
| **Prioridad del Requerimiento** | Alta |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RF05 |
| **Nombre del Requerimiento:** | Consulta de ventas |
| **Características:** | Filtros por rango de fechas, cliente y producto. |
| **Descripción del Requerimiento:** | Los administradores podrán visualizar las ventas para análisis. |
| **Observaciones:** | Compatible con reportes en PDF/Excel. |
| **Prioridad del Requerimiento** | Media |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RF06 |
| **Nombre del Requerimiento:** | Registro de clientes |
| **Características:** | Datos básicos (nombre, teléfono, correo, historial de compras). |
| **Descripción del Requerimiento:** | El sistema debe almacenar y permitir consulta del historial de cada cliente. |
| **Observaciones:** | Los clientes deben acceder a su historial de pedidos. |
| **Prioridad del Requerimiento** | Alta |

**Requerimientos No Funcionales:**

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RNF01 |
| **Nombre del Requerimiento:** | Multiplataforma |
| **Origen Análisis:** | Necesidad de acceso desde web y móvil. |
| **Características:** | El sistema debe ser accesible tanto desde navegadores web como desde dispositivos móviles (Android/iOS). |
| **Descripción del Requerimiento:** | Garantizar que los usuarios puedan acceder al sistema en cualquier dispositivo con conexión a internet, manteniendo la misma funcionalidad. |
| **Prioridad del Requerimiento** | Alta |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RNF02 |
| **Nombre del Requerimiento:** | Usabilidad |
| **Origen Análisis:** | Experiencia de usuario en la tienda. |
| **Características:** | La interfaz debe ser sencilla, clara e intuitiva. |
| **Descripción del Requerimiento:** | Permitir que administradores, empleados y clientes utilicen el sistema sin necesidad de capacitación avanzada, reduciendo errores humanos. |
| **Prioridad del Requerimiento** | Alta |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RNF03 |
| **Nombre del Requerimiento:** | Rendimiento |
| **Origen Análisis:** | Procesamiento de inventario, ventas y pedidos en tiempo real. |
| **Características:** | Procesamiento ágil de consultas, registros y actualizaciones. |
| **Descripción del Requerimiento:** | El sistema debe dar respuesta inmediata a las acciones realizadas (menos de 3 segundos en promedio) para garantizar una experiencia fluida. |
| **Prioridad del Requerimiento** | Alta |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RNF04 |
| **Nombre del Requerimiento:** | Seguridad |
| **Origen Análisis:** | Protección de datos sensibles de clientes y empleados. |
| **Características:** | Autenticación segura y cifrado de datos. |
| **Descripción del Requerimiento:** | El sistema debe garantizar la confidencialidad e integridad de la información mediante protocolos HTTPS, cifrado de contraseñas y control de accesos. |
| **Prioridad del Requerimiento** | Alta |

|  |  |
| --- | --- |
| **Identificación del Requerimiento:** | RNF05 |
| **Nombre del Requerimiento:** | Escalabilidad |
| **Origen Análisis:** | Crecimiento del negocio A Hui Hou. |
| **Características:** | Soporte a mayor volumen de datos y usuarios sin pérdida de rendimiento. |
| **Descripción del Requerimiento:** | El sistema debe estar preparado para permitir el crecimiento en cantidad de clientes, productos y funcionalidades sin comprometer la estabilidad. |
| **Prioridad del Requerimiento** | Media |

**Historias de usuario**

|  |  |
| --- | --- |
| Historia de usuario | |
| Número: 1 | Usuario: Administrador de la tienda |
| Nombre Historia: Registro y control de inventario | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 8 | Puntos reales: |
| Programador Responsable: | |
| Descripción: Como administrador, quiero registrar y actualizar el inventario de productos de manera digital, para reducir los errores manuales y tener un mejor control de los insumos. | |
| Observaciones: Validar que los productos tengan código único | |

|  |  |
| --- | --- |
| Historia de usuario | |
| Número: 2 | Usuario: Empleado |
| Nombre Historia: Actualización automática del inventario | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Alto |
| Puntos estimados: 13 | Puntos reales: |
| Programador Responsable: | |
| Descripción: Como administrador, quiero que el sistema descuente automáticamente el inventario cuando se registre una venta, para evitar inconsistencias en el stock | |
| Observaciones: Debe generar alertas cuando el inventario esté bajo | |

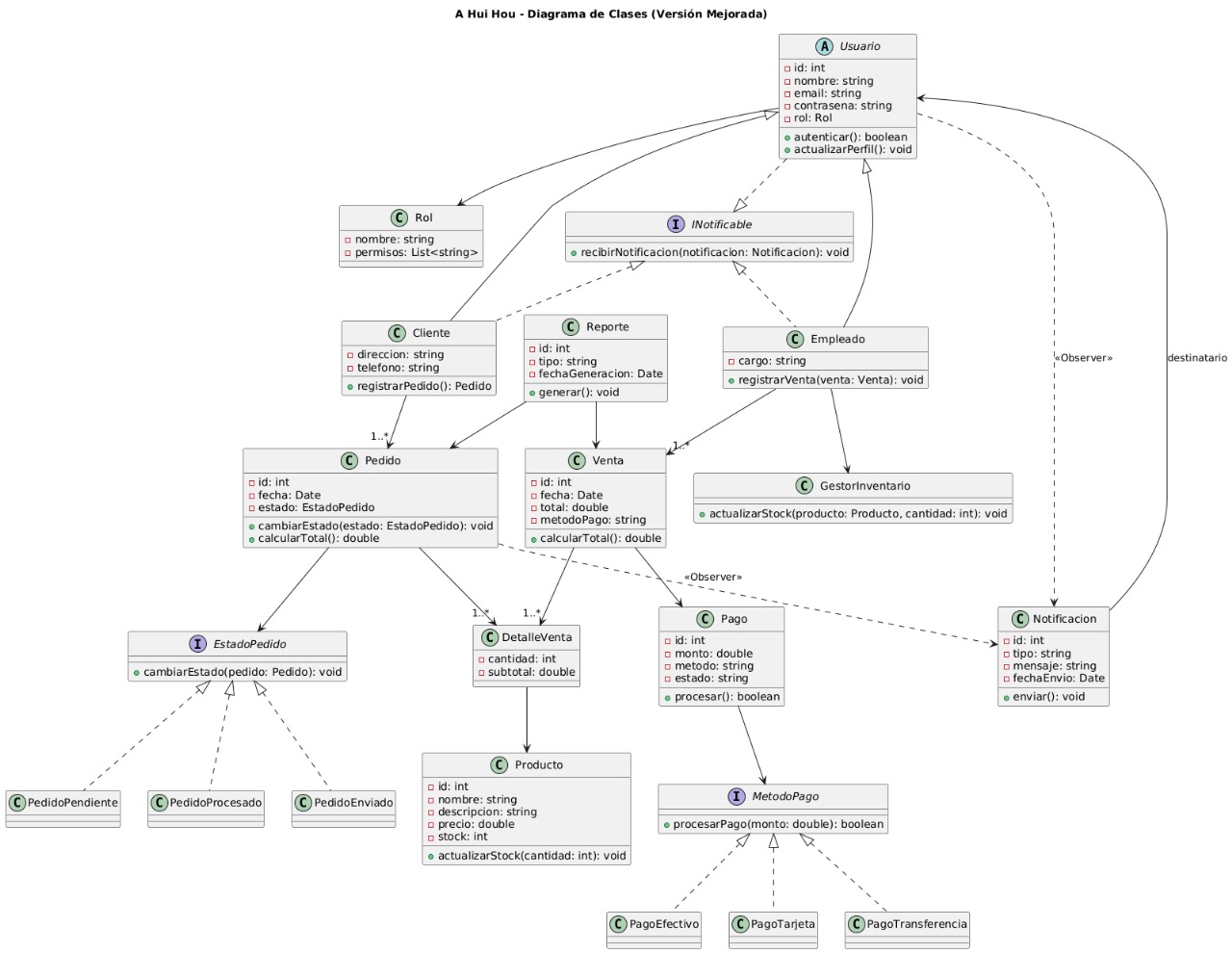
|  |  |
| --- | --- |
| Historia de usuario | |
| Número: 3 | Usuario: Cliente |
| Nombre Historia: Pedido a domicilio desde la aplicación | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: Medio |
| Puntos estimados: 8 | Puntos reales: |
| Programador Responsable: | |
| Descripción: Como cliente, quiero realizar pedidos a domicilio desde la aplicación, para recibir mis productos sin necesidad de escribir por WhatsApp | |
| Observaciones: Permitir seleccionar dirección de entrega y método de pago. | |

|  |  |
| --- | --- |
| Historia de usuario | |
| Número: 4 | Usuario: Administrador de la tienda |
| Nombre Historia: Reportes de ventas e inventario | |
| Prioridad en negocio: Media | Riesgo en desarrollo: Medio |
| Puntos estimados: 8 | Puntos reales: |
| Programador Responsable: | |
| Descripción: Como administrador, quiero generar reportes de ventas e inventario, para analizar el rendimiento del negocio y tomar decisiones. | |
| Observaciones: Reportes descargables en PDF y Excel. | |

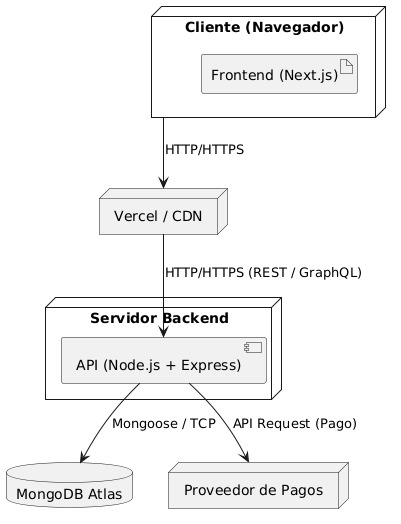
|  |  |
| --- | --- |
| Historia de usuario | |
| Número: 5 | Usuario: Administrador de la tienda |
| Nombre Historia: Gestión de ventas | |
| Prioridad en negocio: Alta | Riesgo en desarrollo: baja |
| Puntos estimados: 5 | Puntos reales: |
| Programador Responsable: | |
| Descripción: Como administrador, quiero un módulo de ventas donde se registren las transacciones diarias, para llevar un control exacto de ingresos. | |
| Observaciones: Debe permitir registrar forma de pago (efectivo, transferencia, etc.). | |

|  |  |
| --- | --- |
| Historia de usuario | |
| Número: 6 | Usuario: Sistema/plataforma |
| Nombre Historia: Acceso multiplataforma | |
| Prioridad en negocio: Media | Riesgo en desarrollo: Alta |
| Puntos estimados: 13 | Puntos reales: |
| Programador Responsable: | |
| Descripción: Como usuario del sistema, quiero acceder a la aplicación desde diferentes dispositivos (móvil, tableta y computador), para gestionar pedidos y ventas de forma flexible y cómoda | |
| Observaciones: Compatible con Android, iOS y versión web. | |

**Diagrama de Clase**



**Diagrama de Despliegue**

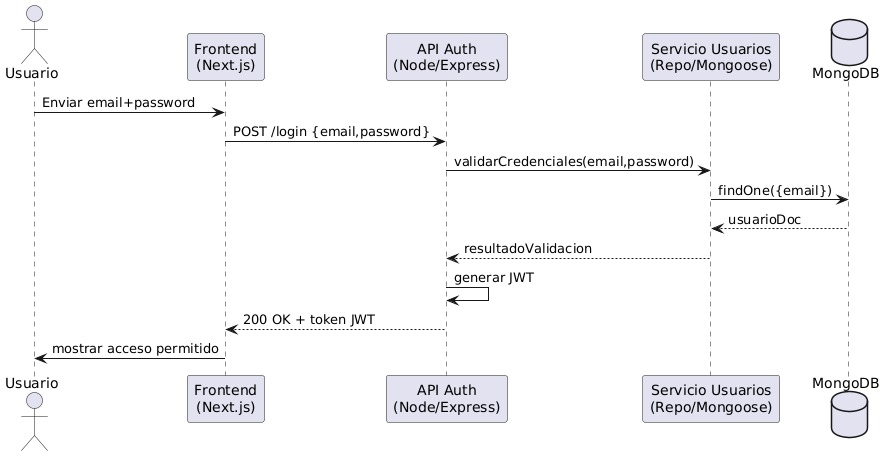


**Diagrama de Actividades**

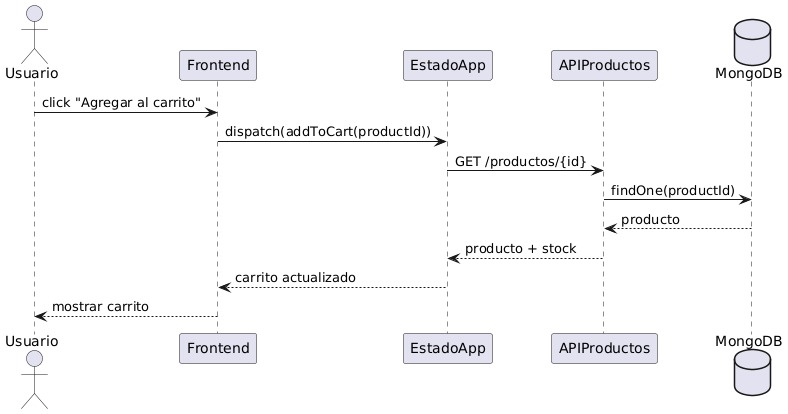


**Diagramas de secuencia Organizados reestructurados y redefinidos**

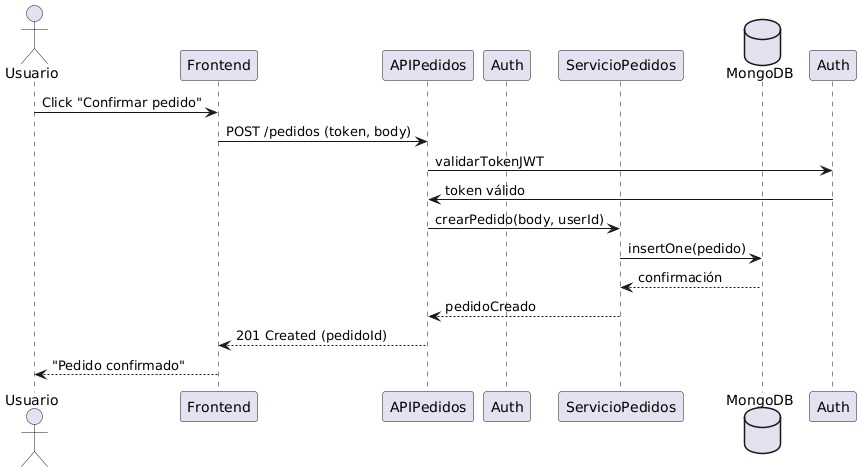
1. **Login**



1. **Agregar producto al carrito**

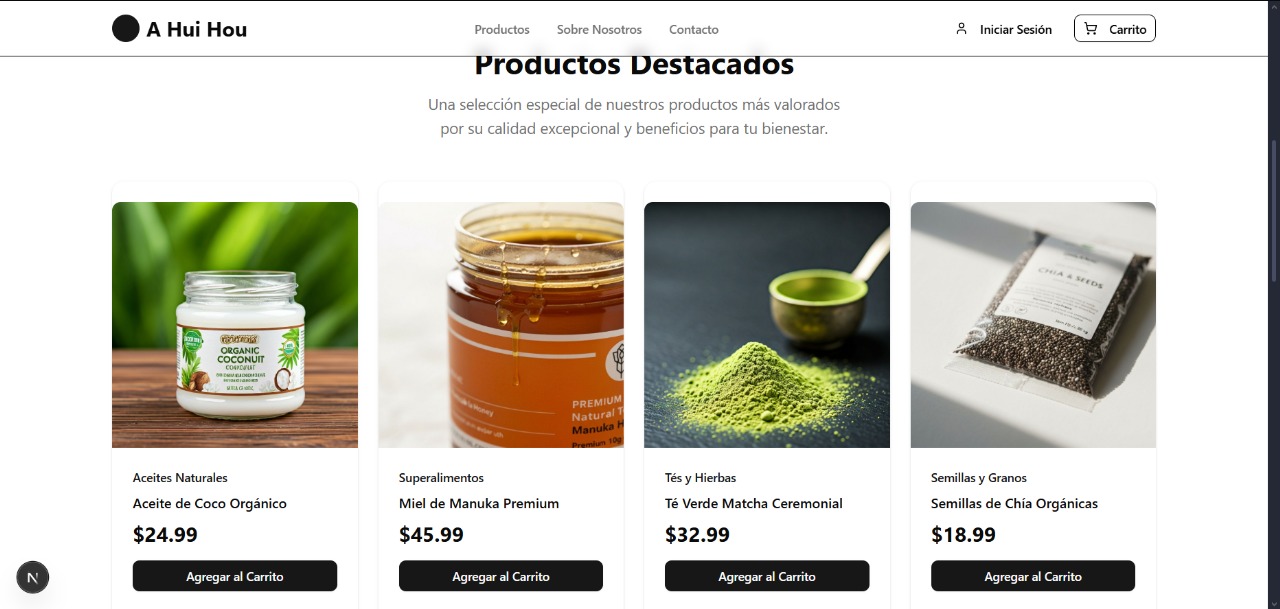


1. **Registrar pedido**



**Figma del proyecto**





**Definir cuáles son los patrones de diseño, creacionales, estructurales y de comportamiento que se han utilizado en el proyecto**

**Patrones Arquitectónicos Usados**

**1. Arquitectura en Capas**

* **Aplicación directa**:
  + **Capa de presentación:** Páginas y componentes de React/Next.js.
  + **Capa de negocio:** Controladores y servicios (por ejemplo, authService, productService).
  + **Capa de persistencia:** Prisma ORM con la base de datos.
* **Ventajas aplicadas:** Separación de responsabilidades, facilidad de pruebas y mantenimiento

**2. Arquitectura Cliente-Servidor**

* **Aplicación directa:** Comunicación entre el **frontend (cliente)** y el **backend (servidor)** mediante peticiones HTTP.
* **Rol del servidor:** Centraliza la lógica y la gestión de datos; el cliente ejecuta la presentación.
* **Soporte documental:** Patrón clásico de distribución asimétrica

**3. Arquitectura de Microservicios (parcial)**

* Aunque el proyecto no es un microservicio completo, **sigue principios inspirados en él**:
  + Servicios independientes para usuarios, productos y autenticación.
  + Comunicación a través de **API REST**, actuando como mini-servicios especializados
  + Implementación de “API Gateway” (ruteo centralizado de peticiones).

**4. Arquitectura Basada en Eventos (inspiración)**

* **Presente parcialmente** en el manejo de eventos del frontend (por ejemplo, acciones que disparan re-render o validaciones).
* Equivale a una **arquitectura reactiva local**, donde los cambios de estado generan reacciones automáticas (patrón Observer aplicado en React)

**Patrones de Diseño GoF Utilizados**

Basado en la clasificación GoF y en cómo se implementan en proyectos de tipo Next.js + Node.js:

1. **Patrones Creacionales**

Dirigen **cómo se crean los objetos** dentro del sistema.

1. **Singleton**

* **Dónde:** Conexión a la base de datos mediante Prisma.
* **Ejemplo:** Se crea una única instancia de PrismaClient compartida en todo el backend.
* **Ventaja:** Evita múltiples conexiones y mantiene un punto de acceso global a la BD.
* **Referencia teórica:** GoF – “Garantiza una única instancia y un punto de acceso global”

1. **Factory Method**

* **Dónde:** En la lógica de negocio cuando se crean entidades como usuarios, productos o pedidos.
* **Ejemplo:** Funciones como createUser() o createProduct() abstraen la creación de instancias.
* **Ventaja:** Desacopla la lógica de creación de la implementación concreta.
* **Referencia teórica:** “Define una interfaz para crear objetos, delegando la instanciación”

1. **Builder (implícito)**

* **Dónde:** En la construcción de objetos complejos (por ejemplo, respuestas JSON o tokens JWT).
* **Ventaja:** Permite construir estructuras paso a paso de forma flexible.
* **Referencia teórica:** “Separa la construcción de un objeto de su representación”

1. **Patrones Estructurales**

Definen **cómo se relacionan las clases y objetos** para formar estructuras más grandes.

**a) MVC (Model-View-Controller)**

* **Dónde:** Estructura global del sistema.
  + **Modelos:** Prisma (base de datos).
  + **Vistas:** Páginas Next.js.
  + **Controladores:** Rutas API o servicios.
* **Ventaja:** Separación clara de responsabilidades.
* **Relación teórica:** Mencionado como patrón arquitectónico derivado en tus documentos.

**b) Adapter**

* **Dónde:** En la integración con servicios externos (p. ej., autenticación con Google o manejo de librerías externas).
* **Función:** Adaptar interfaces externas al formato interno del sistema.
* **Referencia:** “Permite la colaboración entre interfaces incompatibles”

**c) Facade**

* **Dónde:** En los servicios del backend.
* **Ejemplo:** UserService o AuthService actúan como fachada simplificada frente a la complejidad de varias operaciones internas.
* **Ventaja:** Reduce el acoplamiento entre capas.
* **Referencia:** “Proporciona una interfaz unificada a un conjunto de interfaces de un subsistema”

**d) Component (Frontend / React)**

* **Dónde:** En todos los elementos visuales del sistema (botones, formularios, tarjetas).
* **Ventaja:** Reutilización, cohesión y mantenibilidad.
* **Relación teórica:** Basado en el principio GoF de composición de objetos

1. **Patrones de Comportamiento**

Definen **cómo se comunican y colaboran los objetos**.

**a) Observer**

* **Dónde:** En el frontend con React Hooks (useState, useEffect).
* **Ejemplo:** Cuando el estado cambia, los componentes observadores se actualizan automáticamente.
* **Ventaja:** Sincronización automática entre datos y UI.
* **Referencia:** “Un sujeto notifica a múltiples observadores registrados”

**b) Strategy**

* **Dónde:** En el módulo de autenticación (local, Google, OAuth).
* **Ventaja:** Permite intercambiar fácilmente estrategias de login sin cambiar la estructura del código.
* **Referencia:** “Define una familia de algoritmos intercambiables en tiempo de ejecución”

**c) Command**

* **Dónde:** En acciones del frontend (botones, formularios, CRUD).
* **Función:** Cada acción se encapsula como un comando que ejecuta una tarea concreta.
* **Ventaja:** Centraliza la gestión de acciones y facilita el testing.
* **Referencia:** “Encapsula una solicitud como un objeto, permitiendo parametrización y deshacer”

**d) Chain of Responsibility**

* **Dónde:** En los middlewares de autenticación y validación (Next.js / API routes).
* **Ejemplo:** Cada middleware procesa una parte de la solicitud y la pasa al siguiente.
* **Ventaja:** Permite agregar responsabilidades sin modificar el flujo principal.
* **Referencia:** “Evita el acoplamiento entre el emisor y el receptor, pasando la solicitud por una cadena”

|  |  |  |  |
| --- | --- | --- | --- |
| **Módulo o Funcionalidad del Sistema** | **Patrón de Diseño Aplicado** | **Tipo de Patrón** | **Propósito / Beneficio en el Proyecto** |
| **Gestión de usuarios y roles (Administrador, Empleado, Cliente)** | **Factory Method** | Creacional | Permite crear diferentes tipos de usuarios de forma dinámica según su rol, sin modificar la lógica central del sistema. |
| **Conexión y acceso a la base de datos** | **Singleton** | Creacional | Garantiza una única instancia de conexión a la base de datos, evitando sobrecarga y errores de concurrencia. |
| **Registro y actualización de inventario** | **Observer** | Comportamiento | Cada vez que se realiza una venta, el inventario se actualiza automáticamente y los reportes reflejan los nuevos datos en tiempo real. |
| **Gestión de pedidos (pendiente, en proceso, entregado)** | **State** | Comportamiento | Define comportamientos diferentes para cada estado del pedido, facilitando su seguimiento y control. |
| **Módulo de reportes (PDF/Excel)** | **Builder + Template Method** | Creacional / Comportamiento | Permiten construir reportes paso a paso y reutilizar estructuras comunes (encabezado, cuerpo, pie de página). |
| **Interfaz de comunicación entre capas (Modelo, Vista, Controlador)** | **MVC** | Estructural | Separa la lógica del negocio, la presentación y el control, facilitando el mantenimiento y las pruebas del sistema. |
| **Carga masiva de productos (archivos Excel o CSV)** | **Adapter** | Estructural | Transforma los datos de archivos externos a objetos del sistema, integrando fuentes de información heterogéneas. |
| **Gestión de ventas y registros de acciones** | **Command** | Comportamiento | Cada acción del usuario (venta, modificación, eliminación) se encapsula como un comando, lo que permite registrar y auditar todas las operaciones. |
| **Procesamiento de pagos (efectivo, transferencia, tarjeta)** | **Strategy** | Comportamiento | Permite definir y cambiar dinámicamente la estrategia de pago según el método seleccionado por el cliente. |
| **Acceso unificado a funcionalidades (Inventario, Ventas, Reportes)** | **Facade** | Estructural | Proporciona una interfaz simple para operaciones complejas, reduciendo el acoplamiento entre los módulos del sistema. |

La correcta aplicación de los patrones de diseño en el sistema A Hui Hou permitió estructurar un software más ordenado, mantenible y escalable. Los patrones creacionales facilitaron la creación de objetos complejos sin acoplamiento; los estructurales contribuyeron a una arquitectura limpia bajo el modelo MVC; y los de comportamiento garantizaron la comunicación eficiente entre los módulos, optimizando la respuesta del sistema ante las acciones del usuario.

**Github**  
[**https://github.com/Brayan3-4/Proyecto-A-Hui-Hou.git**](https://github.com/Brayan3-4/Proyecto-A-Hui-Hou.git)