DOCUMENTO DE DISEÑO Y ARQUITECTURA DE LA SOLUCIÓN

Artesanias Bogotá Ltda.

Integrantes:

Serrano Rodríguez Juan Manuel – 20211020091

Docente:

Helio Henry Ramirez Arevalo

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Diseño arquitectural de software y patrones
Bogotá, 2024

Contents

List of Figures

List of Tables

1 Descripción del Documento

1.1 Propósito y Audiencia

Este documento tiene como propósito describir la arquitectura de software definida para la plataforma de comercio electrónico Artesanías Bogotá Ltda., orientada a desarrolladores, arquitectos de software y demás interesados en entender la estructura de la aplicación.

1.2 Organización del Documento

El documento está dividido en secciones, abordando desde los requerimientos y motivadores arquitecturales hasta los modelos y vistas arquitectónicas de la solución.

1.3 Convenciones

Las convenciones en este documento están alineadas con las mejores prácticas de arquitectura de software, siguiendo una organización en capas y basada en eventos.

1.4 Terminología y Definiciones

- **Frontend**: Parte de la aplicación accesible y visible para los usuarios. Implementado en Next.js y TypeScript, se encarga de presentar la interfaz gráfica, recibir entradas del usuario y mostrar resultados.
- **Backend**: Capa que contiene la lógica de negocio y las reglas del sistema, desarrollada en Django. Se ocupa de gestionar las operaciones internas, como el procesamiento de pedidos, la gestión de usuarios, y la comunicación con la base de datos y servicios externos.
- **Bull**: Biblioteca de gestión de colas para Node.js que facilita la creación y manejo eficiente de colas de tareas. Permite el procesamiento de trabajos en segundo plano, mejorando la escalabilidad y el rendimiento de la aplicación.
- **Redis**: Sistema de almacenamiento en memoria utilizado como almacenamiento rápido para datos temporales y como backend para Bull. Proporciona alta velocidad en la gestión de colas de tareas y asegura la persistencia necesaria para operaciones asíncronas.

- **PostgreSQL**: Sistema de gestión de bases de datos relacional, utilizado para almacenar los datos estructurados de la plataforma (como productos, usuarios, y pedidos).
- **API REST**: Conjunto de protocolos que define las interacciones entre el frontend y el backend, donde el backend expone funciones para operaciones como el inicio de sesión, la gestión de carritos, y la visualización de productos.
- **E-commerce**: Plataforma digital que permite la compra y venta de productos en línea. En este proyecto, el enfoque de la plataforma es la venta de artesanías de Bogotá, integrando inventarios de puntos de venta físicos.

2 Generalidades del Proyecto

2.1 Problema por Resolver

El problema que se desea resolver es facilitar la compra y gestión de productos artesanales a través de una plataforma en línea, integrando la gestión de inventarios de puntos de venta físicos y el análisis de comportamiento del cliente.

2.2 Descripción General del Sistema a Desarrollar

El sistema permite a los usuarios comprar productos artesanales en línea y a los administradores gestionar inventarios en tiempo real y obtener métricas del comportamiento de los usuarios.

2.3 Objetivos de la Solución

- Desarrollar una plataforma e-commerce para la venta de artesanías.
- Integrar la gestión de inventario en tiempo real.
- Generar reportes sobre ventas y comportamiento del usuario.

2.4 Stakeholders

Stakeholder	**Descripción**
Administradores de la plataforma	Son responsables de la gestión y configuración general del sitio, así como de la gestión de usuarios y productos.
Vendedores de Artesanías	Ofrecen sus productos a través de la plataforma. Buscan una plataforma estable y sencilla para publicar, gestionar inventarios y vender productos.
Clientes	Usuarios que compran productos a través de la plataforma. Buscan una interfaz fácil de usar y confiable para realizar sus compras.
Equipo de Desarrollo	Desarrolla y mantiene la plataforma. Responsable de implementar y gestionar los componentes técnicos para cumplir con los requisitos del negocio.
Gerencia General	Define la estrategia y toma decisiones clave de inversión y dirección del proyecto.

Table 1: Stakeholders

Stakeholder	**Expectativas**
Administradores de la plataforma	Esperan contar con una plataforma confiable que permita realizar configuraciones y gestionar la base de datos de productos y usuarios sin interrupciones.
Vendedores de Artesanías	Esperan facilidad para gestionar sus productos y una plataforma que maximice la visibilidad y venta de sus artesanías.
Clientes	Esperan un sitio seguro, intuitivo y de rápida respuesta para realizar sus compras de manera confiable.
Equipo de Desarrollo	Espera que la arquitectura sea escalable y mantenible, con una infraestructura que permita desarrollo ágil y la integración de mejoras.
Gerencia General	Espera que la plataforma apoye el crecimiento del negocio y cumpla con los objetivos de rentabilidad y expansión del mercado.

Table 2: Expectativas de los Stakeholders

3 Especificación y Recabación de Requerimientos Funcionales

Los requerimientos funcionales son las capacidades específicas que debe cumplir el sistema. Los principales requerimientos son:

- **RF1: Registro e inicio de sesión de usuarios.**
- **RF2: Visualización y búsqueda de productos.**
- **RF3: Gestión de carrito de compras.**

- **RF4: Generación y exportación de reportes de ventas y comportamiento.**
- **RF5: Procesamiento de pagos y confirmación de pedidos.**

3.1 Motivadores de Negocio

El negocio busca expandir la venta de artesanías a través de una plataforma que permita llegar a más clientes de manera digital y optimizar el manejo de inventario.

Nombre del Motivador de Negocio	**Descripción del Motivador de Negocio**	
Expansión de Ventas	Ampliar el alcance de mercado para las artesanías, permitiendo a la organización ofrecer productos a nivel nacional e internacional a través de una plataforma digital.	
Medida del Impacto		
Incremento en las ventas online.		
Rangos	**Cota Mínima**	**Cota Máxima**
Ninguno		
Bajo		
Moderado		
Fuerte		
Muy Fuerte		
Asociación del motivador con el negocio	**Definido por:**	
	Ejecutado por:	Equipo de Desarrollo

Table 3: Motivadores de Negocio 1

Nombre del Motivador de Negocio	**Descripción del Motivador de Negocio**	
Reducción de Costos Operativos	Minimizar costos asociados a la gestión de inventarios y ventas físicas al integrar inventarios y reducir la necesidad de puntos de venta físicos.	
Medida del Impacto		
Ahorro en costos logísticos y operativos.		
Rangos	**Cota Mínima**	**Cota Máxima**
Ninguno		
Bajo		
Moderado		
Fuerte		
Muy Fuerte		
Asociación del motivador con el negocio	**Definido por:**	
	Ejecutado por:	Administración

Table 4: Motivadores de Negocio 2

Nombre del Motivador de Negocio	Descripción del Motivador de Negocio	
Mejora de la Experiencia del Cliente	Ofrecer una experiencia de usuario intuitiva y amigable que facilite la compra de artesanías, aumente la satisfacción del cliente y genere reseñas positivas.	
Medida del Impacto		
Incremento en la tasa de conversión y retención de clientes.		
Rangos	**Cota Mínima**	**Cota Máxima**
Ninguno		
Bajo		
Moderado		
Fuerte		
Muy Fuerte		
Asociación del motivador con el negocio	**Definido por:**	
	Ejecutado por:	Desarrollo (UX/UI)

Table 5: Motivadores de Negocio 3

3.2 Restricciones de Tecnología

ID Restricción:	RT-01	Tipo: (X) Tecnología ( ) Negocio	Nombre Restricción:	Uso de PostgreSQL como Base de Datos
Descripción:	La base de datos debe ser relacional y normalizada, y PostgreSQL es la elegida por compatibilidad y características avanzadas.			
Establecido por:	Equipo de Desarrollo			
Alternativas:	MySQL, MariaDB			
Observaciones:	PostgreSQL garantiza mayor consistencia de datos para las transacciones críticas de la plataforma.			

Table 6: Restricción de tecnología 1.

ID Restricción:	RT-02	Tipo: (X) Tecnología ( ) Negocio	Nombre Restricción:	Herramientas de Mensajería Asíncrona (Bull y Redis)
Descripción:	La solución debe ser capaz de manejar eventos de forma asíncrona usando un sistema de colas. Se utilizarán Bull como gestor de colas y Redis como sistema de almacenamiento en memoria.			
Establecido por:	Arquitectura de Software			
Alternativas:	BullMQ, Bee-Queue			
Observaciones:	Bull y Redis se seleccionan por su integración nativa, rendimiento y facilidad de implementación en aplicaciones basadas en eventos.			

Table 7: Restricción de tecnología 2.

ID Restricción:	RB-01	Tipo: ( ) Tecnología (X) Negocio	Nombre Restricción:	Cumplimiento de Normativa de Protección de Datos
Descripción:	La plataforma debe cumplir con regulaciones de privacidad como la GDPR y normas locales para proteger los datos de los usuarios.			
Establecido por:	Departamento Legal y Compliance			
Alternativas:	No aplicable			
Observaciones:	Esto afecta la implementación de medidas de seguridad en la arquitectura, asegurando la protección de datos sensibles y la confidencialidad de las transacciones.			

Table 8: Restricción de negocio 1.

ID Restricción:	RB-02	Tipo: ( ) Tecnología (X) Negocio	Nombre Restricción:	Operatividad en Horario 24/7
Descripción:	La plataforma debe estar disponible 24/7 para todos los usuarios, lo que exige alta disponibilidad y resistencia ante caídas del sistema.			
Establecido por:	Gerencia de Operaciones			
Alternativas:	Tiempo de mantenimiento nocturno planificado			
Observaciones:	La disponibilidad continua de la plataforma es crítica para satisfacer la demanda y asegurar la retención de clientes.			

Table 9: Restricción de negocio 2.

3.3 Atributos de Calidad

Escalabilidad: La solución debe permitir un crecimiento de usuarios sin afectar el rendimiento.
Mantenibilidad: La arquitectura en capas debe permitir modificaciones sin impactar al sistema global.
Disponibilidad: Se debe asegurar que el sistema esté operativo al menos el 99% del tiempo.

3.4 Recabación y Justificación de Requerimientos No Funcionales

La solución debe cumplir con los siguientes requerimientos no funcionales, cada uno justificado por la naturaleza y los objetivos del sistema:

- **Escalabilidad**: El sistema debe ser capaz de manejar un aumento en el número de usuarios y productos sin comprometer el rendimiento. Esto se justifica por la posible expansión del negocio, que requiere soportar un tráfico creciente.
- **Disponibilidad**: Se debe asegurar una disponibilidad de al menos el 99%, garantizando que los usuarios puedan acceder a la plataforma de manera constante. La estabilidad es crítica en plataformas de e-commerce.
- **Seguridad**: Dado que el sistema maneja información sensible, como datos de usuarios y

transacciones de pago, se deben implementar mecanismos robustos de autenticación y autorización, además de cumplir con la normativa de protección de datos.

- **Mantenibilidad**: La arquitectura modular y en capas facilita que los desarrolladores puedan hacer cambios o añadir funcionalidades sin afectar otras áreas del sistema.

3.5 Escenarios de Calidad

Para evaluar el rendimiento y robustez del sistema, se plantean los siguientes escenarios de calidad:

Escenario de Calidad 1:	Alta Disponibilidad	**Stakeholder:**	Administradores de la plataforma
Atributo de Calidad	Disponibilidad		
Justificación	Los administradores requieren alta disponibilidad para mantener la operación continua y confiable de la plataforma.		
Fuente	Interacción directa con la plataforma		
Estímulo	Alta demanda de usuarios concurrentes		
Artefacto	Plataforma web completa		
Ambiente	Producción		
Respuesta	Garantizar la disponibilidad continua con redundancia de servidores y balanceo de carga.		
Medida de la Respuesta	Tasa de disponibilidad de 99.9% anual		

Table 10: Escenario de Calidad 1: Alta Disponibilidad

Escenario de Calidad 2:	Facilidad de Uso	**Stakeholder:**	Vendedores de Artesanías
Atributo de Calidad	Usabilidad		
Justificación	La facilidad de uso permite a los vendedores gestionar sus productos e inventario sin problemas.		
Fuente	Panel de vendedor de la plataforma		
Estímulo	Interacción frecuente con la interfaz de gestión		
Artefacto	Interfaz de usuario		
Ambiente	Producción		
Respuesta	Interfaz intuitiva y accesible que permita una gestión eficiente.		
Medida de la Respuesta	Reducción del tiempo de gestión a menos de 2 minutos por operación		

Table 11: Escenario de Calidad 2: Facilidad de Uso

Escenario de Calidad 3:	Interfaz Intuitiva	**Stakeholder:**	Clientes
Atributo de Calidad	Usabilidad		
Justificación	Los clientes requieren una interfaz amigable y fácil de navegar para mejorar la experiencia de usuario.		
Fuente	Interacción con la plataforma de compra		
Estímulo	Acceso a la plataforma de e-commerce		
Artefacto	Interfaz web de usuario		
Ambiente	Producción		
Respuesta	Ofrecer una interfaz intuitiva y responsive.		
Medida de la Respuesta	Tiempo de respuesta promedio de 2 segundos		

Table 12: Escenario de Calidad 3: Interfaz Intuitiva

Escenario de Calidad 4:	Mantenibilidad del Sistema	**Stakeholder:**	Equipo de Desarrollo
Atributo de Calidad	Mantenibilidad		
Justificación	La mantenibilidad es crucial para que el equipo de desarrollo realice mejoras y soluciones sin comprometer la estabilidad del sistema.		
Fuente	Código fuente y documentación		
Estímulo	Cambios en requerimientos o mejoras		
Artefacto	Arquitectura del sistema y código fuente		
Ambiente	Desarrollo y producción		
Respuesta	Documentación clara y modularidad en el código para facilitar cambios.		
Medida de la Respuesta	Tiempo de implementación de cambios dentro del 10% de desviación		

Table 13: Escenario de Calidad 4: Mantenibilidad del Sistema

Escenario de Calidad 5:	Escalabilidad de la Plataforma	**Stakeholder:**	Gerencia General
Atributo de Calidad	Escalabilidad		
Justificación	La plataforma debe poder crecer y soportar un aumento en la carga de usuarios y transacciones.		
Fuente	Crecimiento del negocio y aumento de la demanda		
Estímulo	Incremento en el número de usuarios concurrentes		
Artefacto	Infraestructura del sistema		
Ambiente	Producción		
Respuesta	Aumentar la capacidad de la plataforma mediante infraestructura flexible y escalable.		
Medida de la Respuesta	Capacidad de soporte de un 200% de incremento en carga sin afectación en desempeño		

Table 14: Escenario de Calidad 5: Escalabilidad de la Plataforma

4 Contexto

4.1 Escenarios Operacionales

El sistema debe permitir la interacción de clientes en la plataforma para navegar, seleccionar productos, y realizar compras, mientras los administradores gestionan productos y reportes.

Título del Escenario Operacional: Gestión de la Plataforma			
Los administradores deben tener acceso a herramientas para gestionar usuarios, productos, y configurar la plataforma.			
Stakeholder	Administradores de la Plataforma	ID	EO-01
Descripción general de la funcionalidad	Permite a los administradores gestionar las configuraciones, productos y usuarios desde un panel de control.		
Describa lo que el stakeholder hace ahora o le gustaría poder hacer	Acceder al panel de administración para gestionar datos y operaciones en tiempo real.		
Describa cualquier entrada provista o disponible al momento del inicio	Información de usuarios, productos y configuración de la plataforma.		
Describa el contexto de la operación	La operación se realiza en un entorno de producción, con conexión a la base de datos y al sistema de mensajería.		
Describa cómo el sistema debe responder	El sistema debe responder de manera rápida y precisa a los comandos de administración, actualizando los datos en tiempo real.		
Describa las salidas que el sistema produce como resultado de la acción	Reportes de actividad y estado actualizado de configuraciones, usuarios y productos.		
Describa quién o qué usa la salida para qué es utilizada	El equipo de administración para monitorear y ajustar operaciones.		

Table 15: Escenario Operacional 1: Gestión de la Plataforma

Título del Escenario Operacional: Gestión de Productos			
Los vendedores pueden gestionar sus productos, inventario y precios en la plataforma para maximizar sus ventas.			
Stakeholder	Vendedores de Artesanías	ID	EO-02
Descripción general de la funcionalidad	Permitir a los vendedores agregar, editar o eliminar productos desde su panel de gestión.		
Describa lo que el stakeholder hace ahora o le gustaría poder hacer	Actualizar información de productos y revisar el inventario en tiempo real.		
Describa cualquier entrada provista o disponible al momento del inicio	Datos del producto, como nombre, descripción, precio y cantidad en inventario.		
Describa el contexto de la operación	Se realiza en el entorno de producción, interactuando con la base de datos y la capa de negocio.		
Describa cómo el sistema debe responder	De manera rápida y precisa para reflejar cambios instantáneamente en la interfaz de usuario.		
Describa las salidas que el sistema produce como resultado de la acción	Confirmación de la actualización de productos y registro de cambios en el inventario.		
Describa quién o qué usa la salida para qué es utilizada	Los vendedores para monitorear sus productos y ajustar la oferta de acuerdo con la demanda.		

Table 16: Escenario Operacional 2: Gestión de Productos

Título del Escenario Operacional: Proceso de Compra			
Los clientes pueden seleccionar productos y realizar compras de forma segura y eficiente en la plataforma.			
Stakeholder	Clientes	ID	EO-03
Descripción general de la funcionalidad	Permitir a los clientes buscar productos, agregarlos al carrito y completar la compra.		
Describa lo que el stakeholder hace ahora o le gustaría poder hacer	Realizar la compra de manera rápida y sin problemas en cualquier momento.		
Describa cualquier entrada provista o disponible al momento del inicio	Productos seleccionados, información de pago y dirección de envío.		
Describa el contexto de la operación	Operación en producción con conectividad a la pasarela de pagos y la base de datos.		
Describa cómo el sistema debe responder	Respuesta rápida, procesando pagos y generando confirmación de compra.		
Describa las salidas que el sistema produce como resultado de la acción	Confirmación de compra y actualización de inventario.		
Describa quién o qué usa la salida para qué es utilizada	Clientes y sistema de inventario para gestionar disponibilidad de productos.		

Table 17: Escenario Operacional 3: Proceso de Compra

Título del Escenario Operacional: Implementación de Actualizaciones			
El equipo de desarrollo despliega mejoras y actualizaciones al sistema sin interrumpir la operación.			
Stakeholder	Equipo de Desarrollo	ID	EO-04
Descripción general de la funcionalidad	Habilitar a los desarrolladores para realizar actualizaciones del sistema y correcciones de errores.		
Describa lo que el stakeholder hace ahora o le gustaría poder hacer	Ejecutar despliegues sin afectar la disponibilidad para los usuarios finales.		
Describa cualquier entrada provista o disponible al momento del inicio	Código fuente actualizado y documentación de cambios.		
Describa el contexto de la operación	Entorno de producción, con control de versiones y pruebas automatizadas.		
Describa cómo el sistema debe responder	Mantener estabilidad mientras se aplican las actualizaciones sin interrupciones.		
Describa las salidas que el sistema produce como resultado de la acción	Actualización exitosa y registro de cambios en la versión.		
Describa quién o qué usa la salida para qué es utilizada	Equipo de desarrollo para futuras referencias y mejora continua.		

Table 18: Escenario Operacional 4: Implementación de Actualizaciones

Título del Escenario Operacional: Monitoreo de Rendimiento			
La gerencia puede acceder a informes y métricas de rendimiento para la toma de decisiones estratégicas.			
Stakeholder	Gerencia General	ID	EO-05
Descripción general de la funcionalidad	Acceso a datos sobre transacciones, visitas y rendimiento del sistema en un tablero de control.		
Describa lo que el stakeholder hace ahora o le gustaría poder hacer	Revisar los informes y métricas de rendimiento para evaluar el estado del negocio.		
Describa cualquier entrada provista o disponible al momento del inicio	Datos de ventas, tráfico y métricas de la plataforma.		
Describa el contexto de la operación	Entorno de producción con acceso seguro a la base de datos y sistema de reportes.		
Describa cómo el sistema debe responder	Proveer reportes y estadísticas en tiempo real de forma confiable.		
Describa las salidas que el sistema produce como resultado de la acción	Reportes y gráficos sobre rendimiento y métricas clave.		
Describa quién o qué usa la salida para qué es utilizada	Gerencia para toma de decisiones estratégicas.		

Table 19: Escenario Operacional 5: Monitoreo de Rendimiento

4.2 Casos de Uso: Descripción y Modelo

A continuación se describen los principales casos de uso:

- **UC01: Registro de Usuarios** - El usuario crea una cuenta en la plataforma proporcionando sus datos. La cuenta será validada y almacenada en la base de datos.
- **UC02: Iniciar Sesión** - Permite a los usuarios registrados acceder a sus cuentas usando credenciales. En caso de autenticación exitosa, el usuario obtiene acceso a la plataforma.
- **UC03: Buscar Producto** - El usuario realiza búsquedas de productos en la plataforma mediante un campo de búsqueda que devuelve resultados relevantes.
- **UC04: Agregar al Carrito** - Los usuarios pueden añadir productos al carrito de compras para continuar con el proceso de compra.
- **UC05: Eliminar del Carrito** - Permite a los usuarios quitar productos del carrito antes de confirmar la compra.
- **UC06: Realizar Compra** - Los usuarios finalizan la compra de los productos en el carrito, seleccionando el método de pago y proporcionando la información necesaria.
- **UC07: Calificar Producto** - Luego de la compra, el usuario puede calificar y comentar sobre los productos adquiridos.
- **UC08: Crear Producto (Administrador)** - Los administradores pueden agregar nuevos productos al inventario disponible en la plataforma.
- **UC09: Editar Producto (Administrador)** - Permite que los administradores editen la información de productos ya registrados.
- **UC10: Eliminar Producto (Administrador)** - Los administradores pueden eliminar productos del catálogo de la plataforma.
- **UC11: Generar Reportes** - Permite al administrador crear y exportar reportes de ventas, inventario, y comportamiento de usuarios.

5 Descripción y Justificación de la Arquitectura de Software Definida

La arquitectura del sistema se basa en un estilo de arquitectura en capas, complementado con una orientación hacia eventos distribuidos para soportar tareas asíncronas.

- **Arquitectura en Capas** - La organización en capas permite separar la lógica de presentación, negocio y datos. Esto facilita la mantenibilidad y la escalabilidad del sistema al permitir que cada capa pueda ser desarrollada y mantenida de manera independiente.
- **Arquitectura Basada en Eventos** - Se emplean Bull y Redis para la gestión de colas y el almacenamiento en memoria. Este estilo arquitectural permite que el sistema responda eficientemente a acciones que no requieren procesamiento inmediato, como actualizaciones de inventario y confirmaciones de pedidos.

5.1 Descripción específica de la arquitectura

5.1.1 Capa de Presentación (Frontend - Next.js con TypeScript)

Descripción: El frontend, implementado en Next.js con TypeScript, maneja la interacción directa con el usuario, proporcionando una interfaz visual donde se pueden navegar y visualizar productos, agregar elementos al carrito, realizar pagos y calificar productos.

Componentes:

- **Páginas de usuario:** Registro, inicio de sesión, perfil de usuario.
- **Catálogo de productos:** Vista de productos, búsqueda, detalles de productos.
- **Carrito de compras:** Gestión de productos añadidos y proceso de checkout.
- **Interacción con el backend:** Las peticiones HTTP se realizan a través de API REST hacia el backend, utilizando Axios u otra librería para la comunicación HTTP.
- **Interacción con Bull y Redis:** Next.js no se comunica directamente con Bull y Redis, pero visualiza el estado actualizado tras la confirmación de los eventos procesados en el backend.

5.1.2 Capa de Negocio (Backend - Strapi)

Descripción: La capa de negocio está compuesta por un backend en Strapi que gestiona la lógica de negocio y maneja la mayor parte de la funcionalidad crítica del sistema.

Componentes:

- **API RESTful:** Strapi expone APIs que permiten el acceso a datos y funcionalidades del sistema para el frontend. Las APIs incluyen:
  - **Autenticación:** Manejo de usuarios y autenticación.
  - **Gestión de productos:** Endpoints CRUD para visualizar, agregar, editar y eliminar productos.
  - **Carrito de compras:** Endpoints para gestionar los productos en el carrito.
  - **Pedidos y Pagos:** Gestión del procesamiento de pagos y confirmación de pedidos.
  - **Reportes:** Generación y exportación de reportes en formatos como PDF o Excel.
- **Eventos y Comunicación Asíncrona:** Strapi utiliza Bull para gestionar tareas asíncronas en función de acciones del usuario. Por ejemplo:
  - Actualización de inventario al realizar una compra.
  - Envío de correos de confirmación después del pago.
- **Integración con Bull y Redis:** Strapi se integra con Bull para la gestión de colas de tareas asíncronas, y Redis actúa como sistema de almacenamiento en memoria para estas colas. Esto permite manejar tareas que requieren procesamiento en segundo plano sin afectar la experiencia del usuario.

5.1.3 Capa de Persistencia (Base de Datos - PostgreSQL)

Descripción: PostgreSQL es la base de datos relacional que almacena toda la información estructurada del sistema.

Componentes:

- **Tablas de usuarios, productos, pedidos, inventario, reportes y reseñas:** Estas tablas están normalizadas para mejorar la eficiencia y asegurar la consistencia de los datos.
- **Conexión mediante ORM de Strapi:** Strapi utiliza su propio ORM para interactuar con PostgreSQL, lo que permite manipular datos a través de modelos definidos en el CMS.
- **Integridad de los datos:** La base de datos asegura que los datos de pedidos, inventarios y usuarios permanezcan consistentes, incluso si fallan los procesos de tareas asíncronas.

5.1.4 Capa de Integración de Mensajería (Bull y Redis)

Descripción: Bull y Redis forman la infraestructura de comunicación basada en eventos y el procesamiento de tareas asíncronas.

Componentes:

- **Redis (Almacenamiento en Memoria):** Actúa como almacenamiento rápido para datos temporales y como backend para Bull. Proporciona alta velocidad en la gestión de colas de tareas y asegura la persistencia necesaria para operaciones asíncronas.
- **Bull (Gestor de Colas de Tareas):** Bull facilita la creación y manejo eficiente de colas de tareas. Permite el procesamiento de trabajos en segundo plano, mejorando la escalabilidad y el rendimiento de la aplicación.
- **Gestión de Errores y Reintentos:** Bull maneja los errores en la ejecución de tareas, permitiendo reintentos automáticos en caso de fallos de procesamiento.

5.1.5 Diagrama de Arquitectura

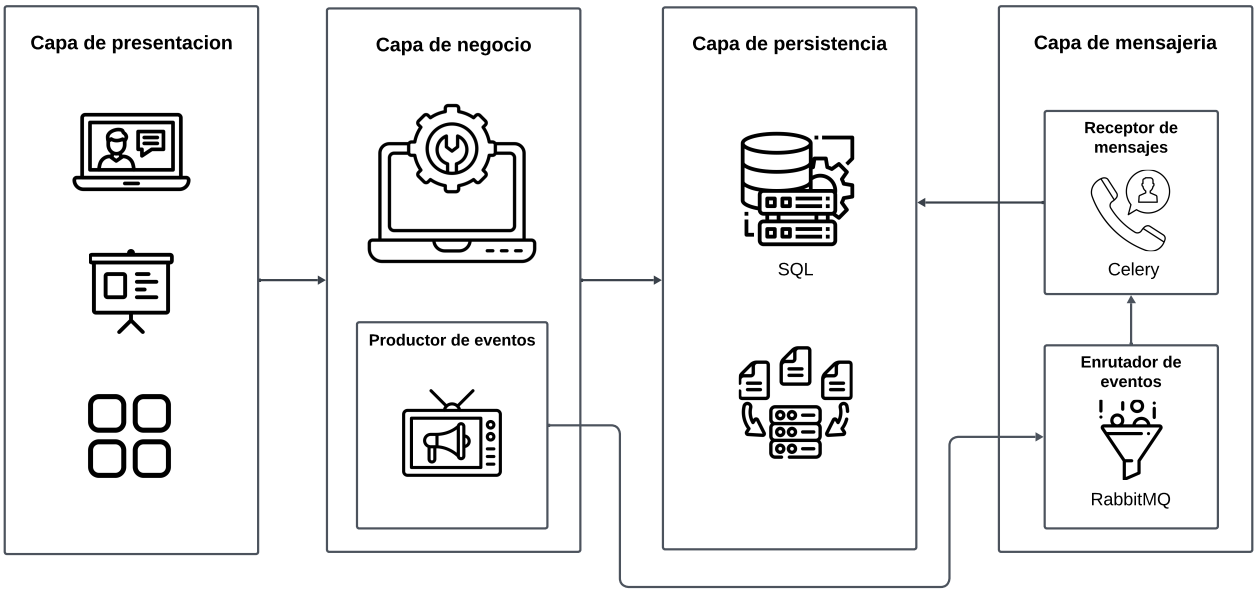


Figure 1: Diagrama de Arquitectura

6 Descripción y Justificación del (los) Patrón(es) de Diseño Adoptado(s)

6.1 Patrón Observador (Observer)

Descripción y Justificación: El patrón Observador permite que uno o varios objetos (observadores) se suscriban a un objeto principal (sujeto) y reciban notificaciones cuando ocurren cambios. En una arquitectura de eventos, este patrón es útil para permitir la comunicación asíncrona entre componentes.

Uso en el Proyecto: En este caso, el backend (Strapi) actúa como el sujeto, y Bull actúa como el observador que procesa los eventos generados.

Implementación:

- Cuando un usuario realiza una compra, el backend genera un evento “pedido confirmado” y lo envía a Bull.

- Bull, como observador, está a la escucha y procesa el evento, actualizando el inventario y enviando una confirmación al usuario.

Beneficio: Permite un flujo de trabajo sin bloqueo, en el cual los consumidores procesan tareas en segundo plano sin interrumpir la experiencia del usuario.

6.2 Patrón Mediador (Mediator)

Descripción y Justificación: El Mediador coordina la comunicación entre diferentes componentes para reducir dependencias directas. En este proyecto, Bull actúa como el mediador que distribuye eventos entre los componentes sin que interactúen directamente entre sí.

Uso en el Proyecto: Bull actúa como mediador entre Strapi y los servicios de procesamiento de tareas, manejando la comunicación de eventos y asegurando que las tareas se enruten correctamente.

Implementación:

- Strapi envía un mensaje a Bull cuando se realiza un pedido.
- Bull distribuye el mensaje a los consumidores correspondientes, que procesan las tareas necesarias.

Beneficio: Elimina dependencias directas entre componentes, lo cual hace que el sistema sea más flexible y escalable al permitir la adición de nuevos consumidores sin modificar el backend.

6.3 Patrón Fábrica (Factory)

Descripción y Justificación: El patrón Fábrica permite la creación de objetos complejos sin especificar la clase exacta del objeto que se creará. Este patrón es útil cuando tienes diferentes tipos de productos o usuarios que comparten algunas propiedades comunes.

Uso en el Proyecto: La fábrica puede encargarse de crear diferentes tipos de productos con atributos específicos según sus categorías.

Implementación:

- Un "ProductoFactory" puede recibir un tipo de producto (por ejemplo, "artesanía textil" o "cerámica") y devolver un objeto con los atributos específicos de cada tipo.

Beneficio: Centraliza la creación de objetos, lo que permite un manejo más estructurado y facilita el mantenimiento si se necesita cambiar la lógica de creación de productos en el futuro.

6.4 Patrón Fachada (Facade)

Descripción y Justificación: La Fachada proporciona una interfaz simplificada para un sistema complejo, lo que permite que otros componentes interactúen con él sin conocer su estructura interna.

Uso en el Proyecto: Strapi puede utilizar controladores de rutas que actúen como una fachada, exponiendo solo los endpoints necesarios para el frontend, simplificando así la interacción.

Implementación:

- Un controlador de rutas en Strapi que ofrece los endpoints para la gestión de usuarios, el carrito y los pedidos actúa como una fachada, ocultando la lógica compleja de negocio y acceso a datos.

Beneficio: Facilita el mantenimiento del backend al exponer solo lo necesario y ocultar los detalles complejos de la lógica de negocio.

6.5 Patrón Repositorio (Repository)

Descripción y Justificación: El patrón de Repositorio proporciona una capa de abstracción entre la lógica de negocio y la capa de acceso a datos. Esto ayuda a organizar y reutilizar el acceso a datos de forma eficiente.

Uso en el Proyecto: En Strapi, el patrón de repositorio se puede implementar encapsulando consultas complejas en servicios que gestionan la interacción con PostgreSQL.

Implementación:

- Un servicio de productos puede ofrecer métodos específicos, como obtenerProductosPorCategoría o actualizarInventario, que interactúan con el ORM de Strapi y encapsulan la lógica de consulta de datos.

Beneficio: Simplifica el acceso a datos y mantiene la lógica de negocio desacoplada del acceso directo a la base de datos, lo que facilita la adaptación del código a cambios en la estructura de datos.

6.6 Patrón Singleton

Descripción y Justificación: El patrón Singleton garantiza que una clase tenga una única instancia y proporciona un punto de acceso global a ella. Es ideal para recursos que deben ser únicos en la aplicación.

Uso en el Proyecto: Este patrón se puede aplicar a la configuración de conexión de la base de datos o a la configuración de Bull en el backend.

Implementación:

- La instancia de conexión a la base de datos puede ser un Singleton, asegurando que Strapi utilice siempre la misma conexión, lo que reduce el uso de recursos.

Beneficio: Evita la duplicación de instancias para recursos críticos, ahorra memoria y simplifica el acceso a configuraciones compartidas en diferentes partes del sistema.

7 Puntos de Vista y Modelos Arquitecturales

7.1 Punto de Vista: Descripción del Problema

7.1.1 Punto de Vista: Contexto de la Solución (Diagrama de Contexto)

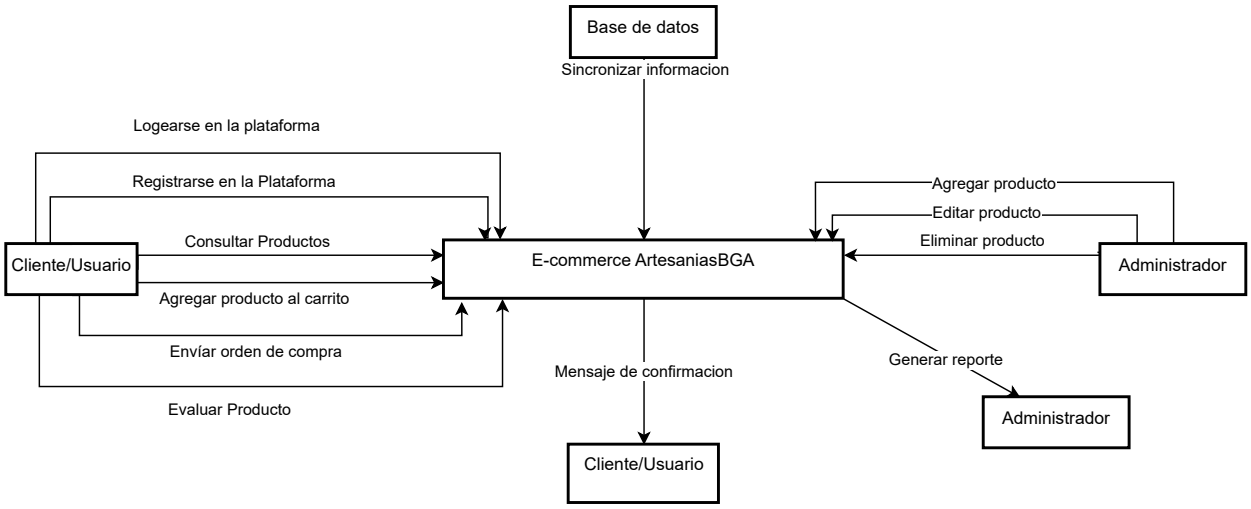
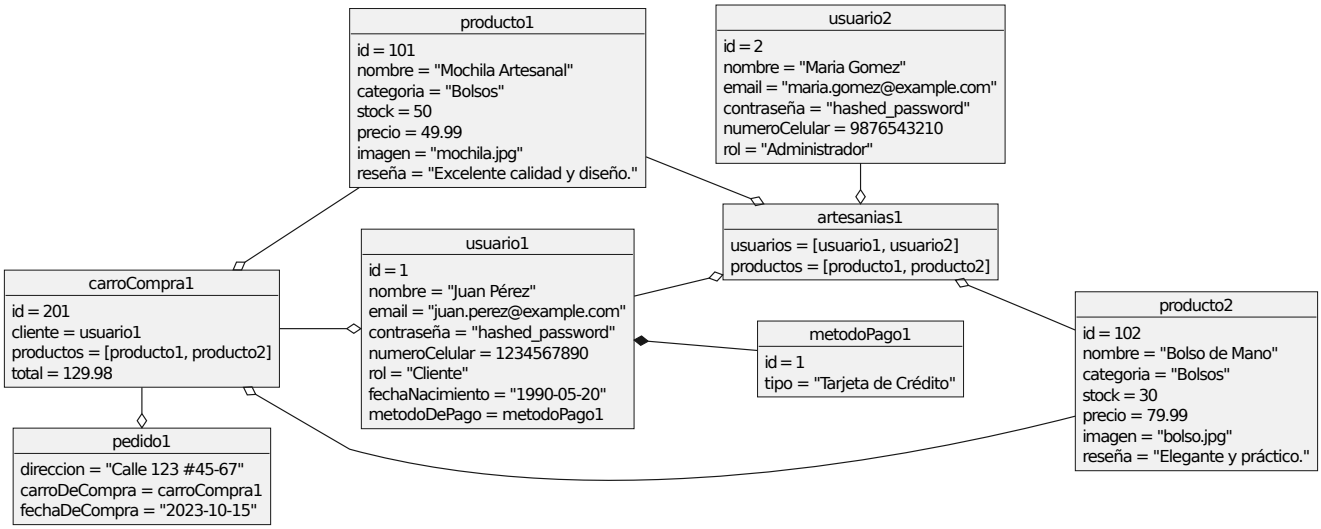


Figure 2: Diagrama de Contexto

7.1.2 Punto de Vista: Objetos (Diagrama de Objetos)

Figure 3: Diagrama de Objetos



7.2 Punto de Vista: Interacción

7.2.1 Diagrama de Casos de Uso

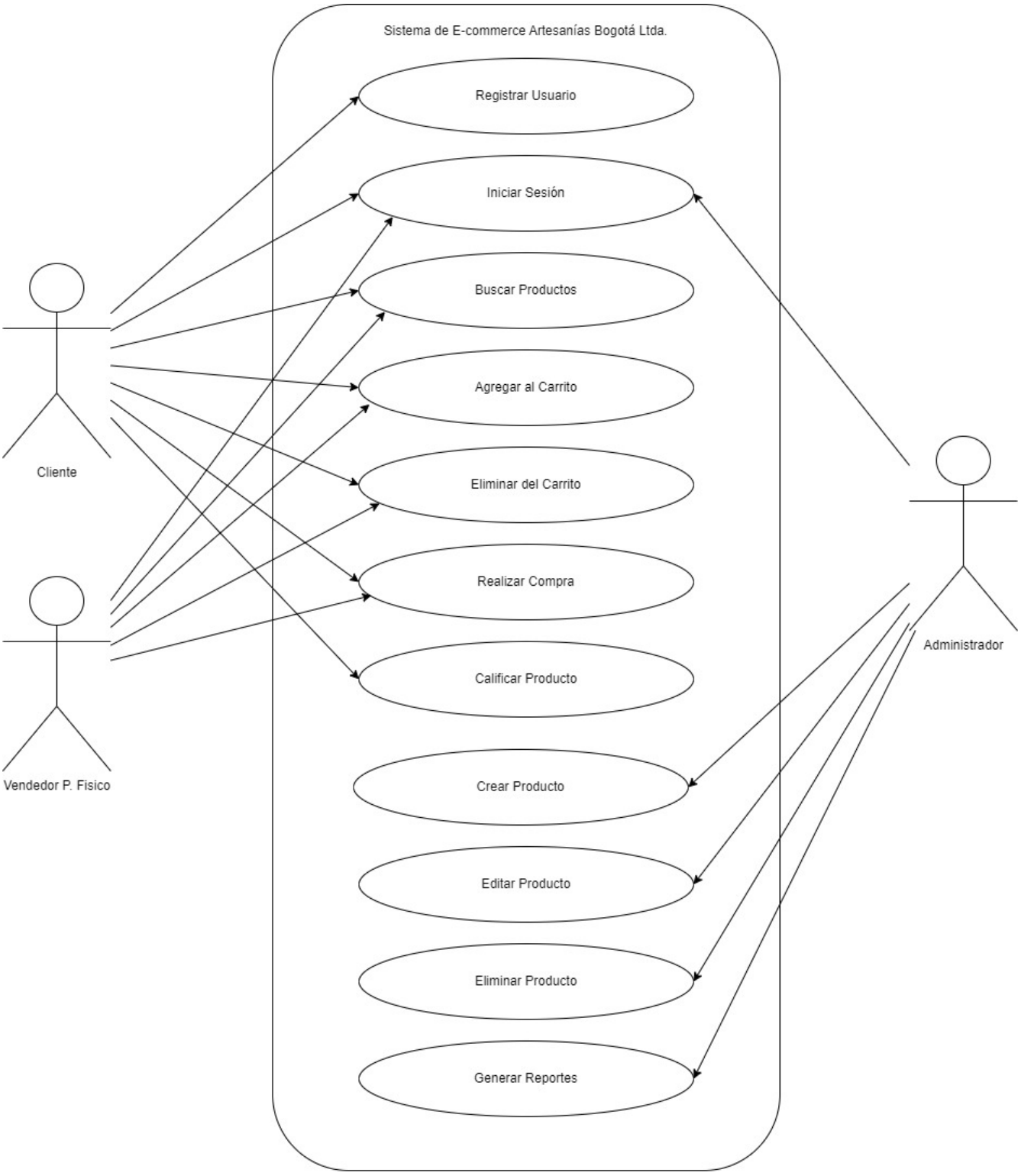


Figure 4: Diagrama de Casos de Uso

7.2.2 Diagrama de Secuencia

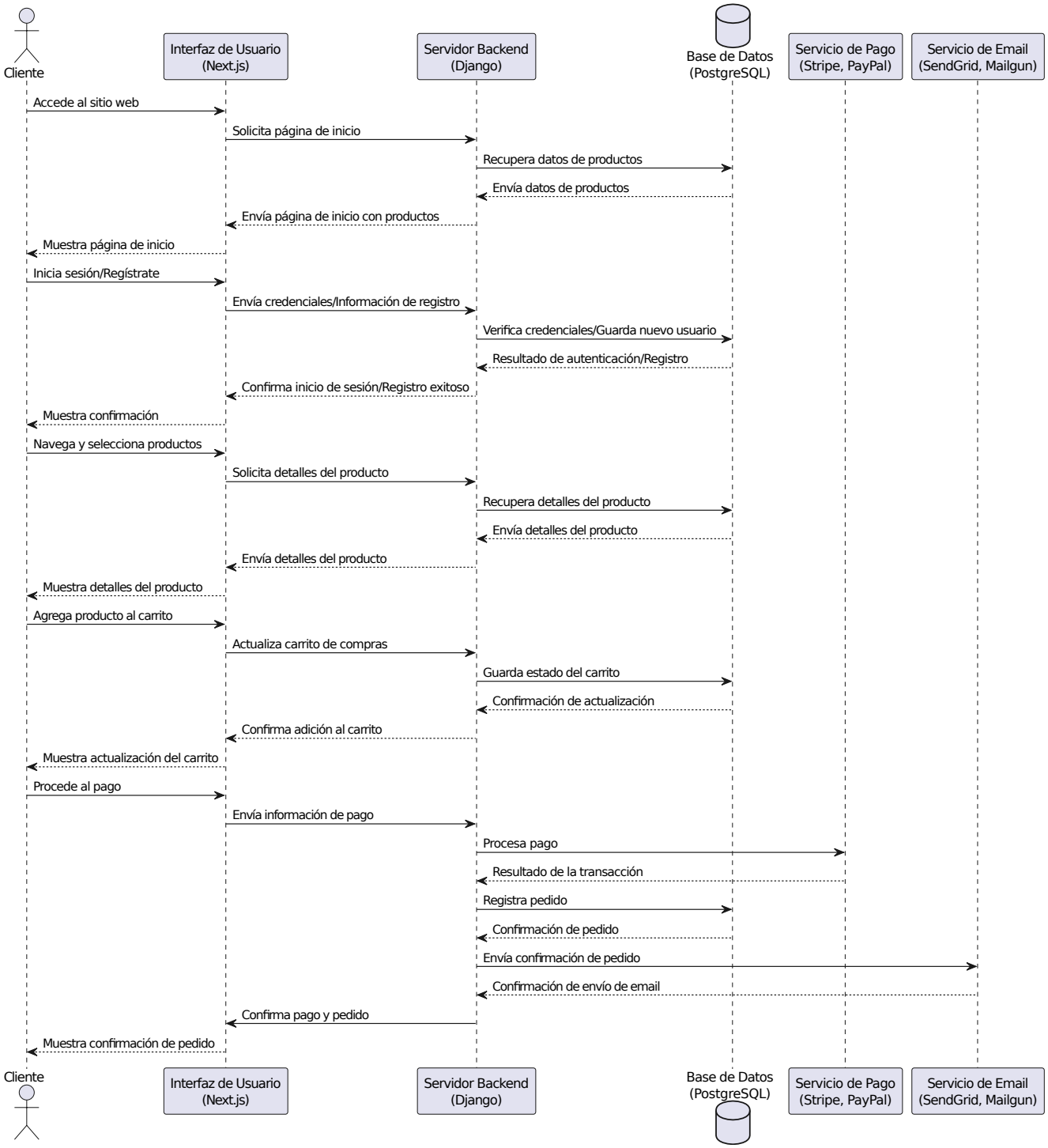


Figure 5: Diagrama de Secuencia

7.3 Punto de Vista: Funcional (Diagrama de Componentes)

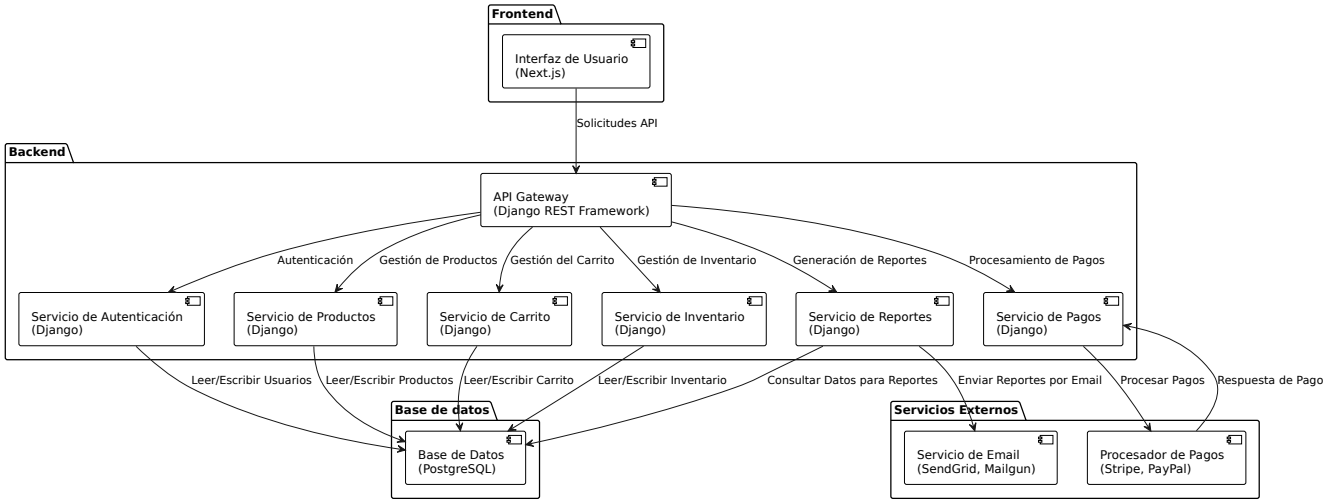


Figure 6: Diagrama de Componentes

7.4 Punto de Vista: Despliegue (Diagrama de Despliegue)

Figure 7: Diagrama de Despliegue

7.5 Punto de Vista: Información (Diagrama(s) de Actividades)

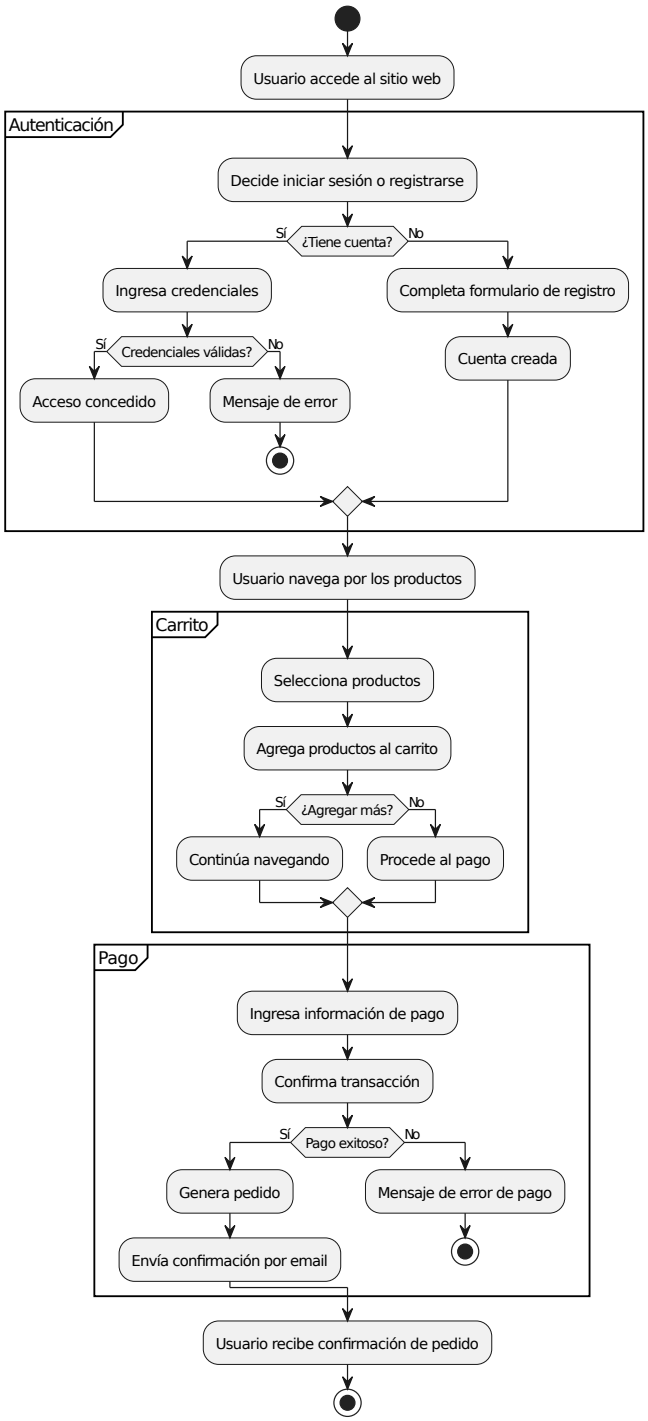


Figure 8: Diagrama de Actividades

7.6 Punto de Vista: Desarrollo (Diagrama de Clases)

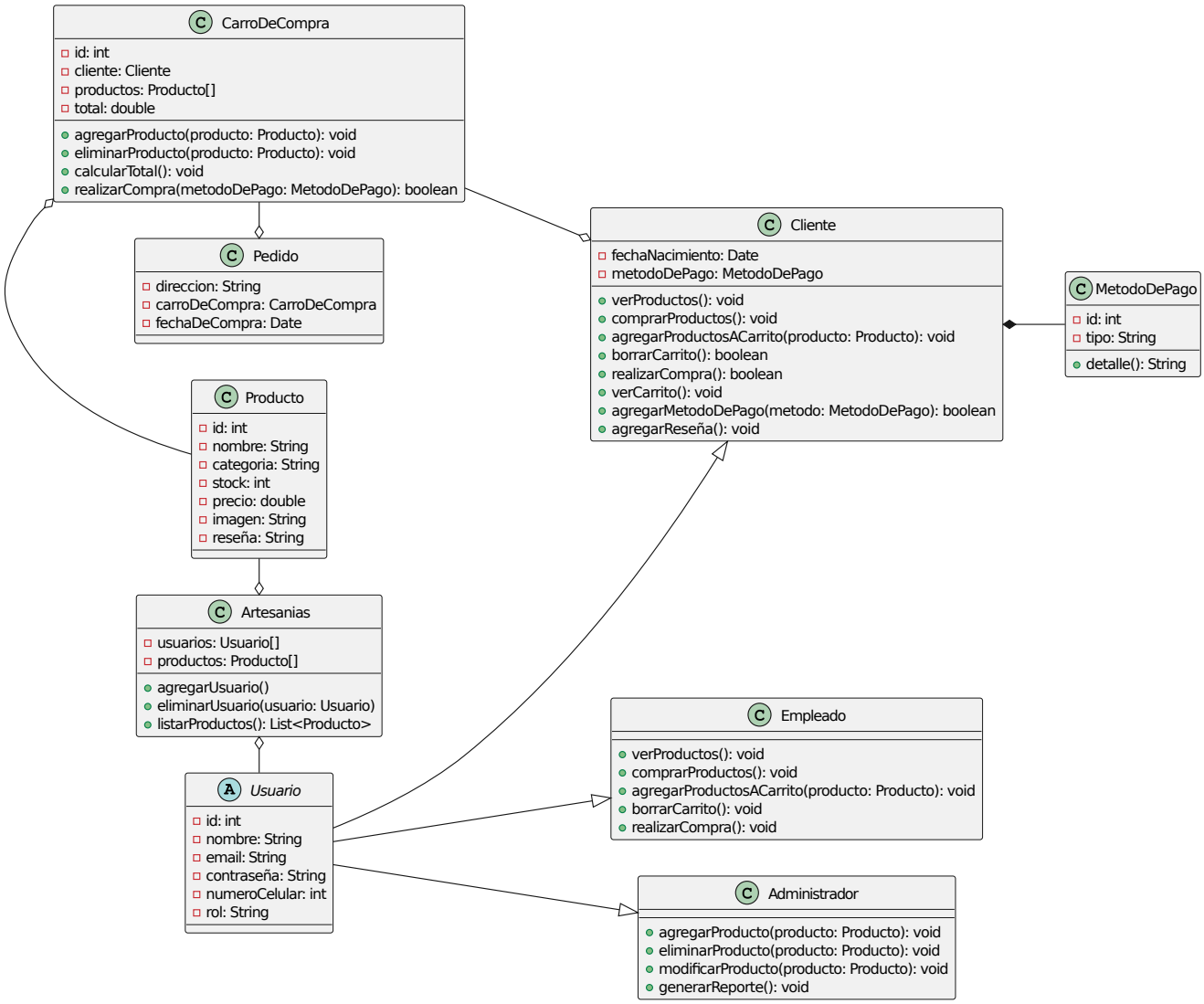


Figure 9: Diagrama de Clases

7.7 Punto de Vista: Modelo Relacional de la BBDD

Figure 10: Modelo Relacional de la BBDD

8 Justificación y Relaciones entre los Puntos de Vista

asd

9 Firmas de Aceptación

- Firma 1: _____ Fecha: DD/MM/AAAA
- Firma 2: _____ Fecha: DD/MM/AAAA
- Firma 3: _____ Fecha: DD/MM/AAAA
- Firma 4: _____ Fecha: DD/MM/AAAA
- Firma 5: _____ Fecha: DD/MM/AAAA
- Firma 6: _____ Fecha: DD/MM/AAAA