

Capstone

SIGLAS Y SECCIÓN: 005D

NOMBRE DE LA SEDE: Antonio Varas

Documentación Técnica (Modelo 4+1 y Arquitectura)

Integrantes:

Tushar Mirwani / 21.149.877-3

Diego Infantas / 21.532.547-4

Jean Pier Huansha / 26.841.686-2

Profesor:

CINDY BETZABE CONTADOR CISTERNA

04/10/2025

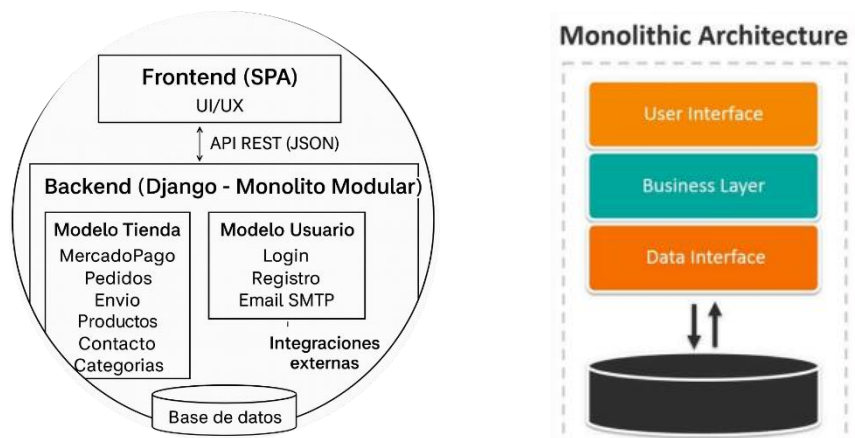
Contenido

Modelo de Vistas de Arquitectura 4+1	2
Vista de Escenarios	3
➤ Casos de Uso.....	3
Vista Lógica	4
➤ Diagramas de Secuencia (5)	4
➤ Diagramas de Comunicación (3).....	5
➤ Diagrama de Clases	5
Vista de Despliegue.....	6
➤ Diagrama de Componentes	6
➤ Diagrama de Paquetes	7
Vista de Procesos	7
➤ Diagramas de Actividad (1).....	8
Vista Física.....	9
➤ Diagrama de Despliegue	9

Estilo Arquitectónico

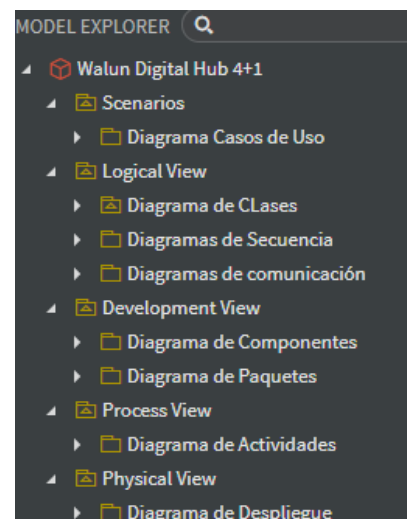
La arquitectura de este sistema se basa en un **backend monolítico modular desarrollado en Django**, organizado en diferentes apps internas según el dominio funcional, como tienda, usuarios, pedidos, pagos, envíos y contacto. Aunque cada módulo mantiene una separación lógica, todos comparten el mismo proyecto, la misma base de datos y el mismo despliegue, lo que permite centralizar la seguridad, las dependencias y la administración del sistema. Además, el backend expone una **API REST en formato JSON**, a la cual también se conectan integraciones externas como MercadoPago y BlueExpress para la gestión de pagos y envíos.

El frontend está implementado como una **Single Page Application (SPA) independiente en React**, completamente desacoplada del backend y encargada de la experiencia de usuario (UI/UX). La comunicación entre el frontend y el backend se realiza exclusivamente mediante llamadas HTTP/JSON, lo que permite una separación clara de responsabilidades, facilita la escalabilidad y abre la posibilidad de incorporar futuras aplicaciones móviles o nuevos clientes sin modificar el núcleo del sistema. Esta combinación de monolito modular con frontend desacoplado ofrece un equilibrio ideal entre orden interno, mantenibilidad y flexibilidad de integración.



Modelo de Vistas de Arquitectura 4+1

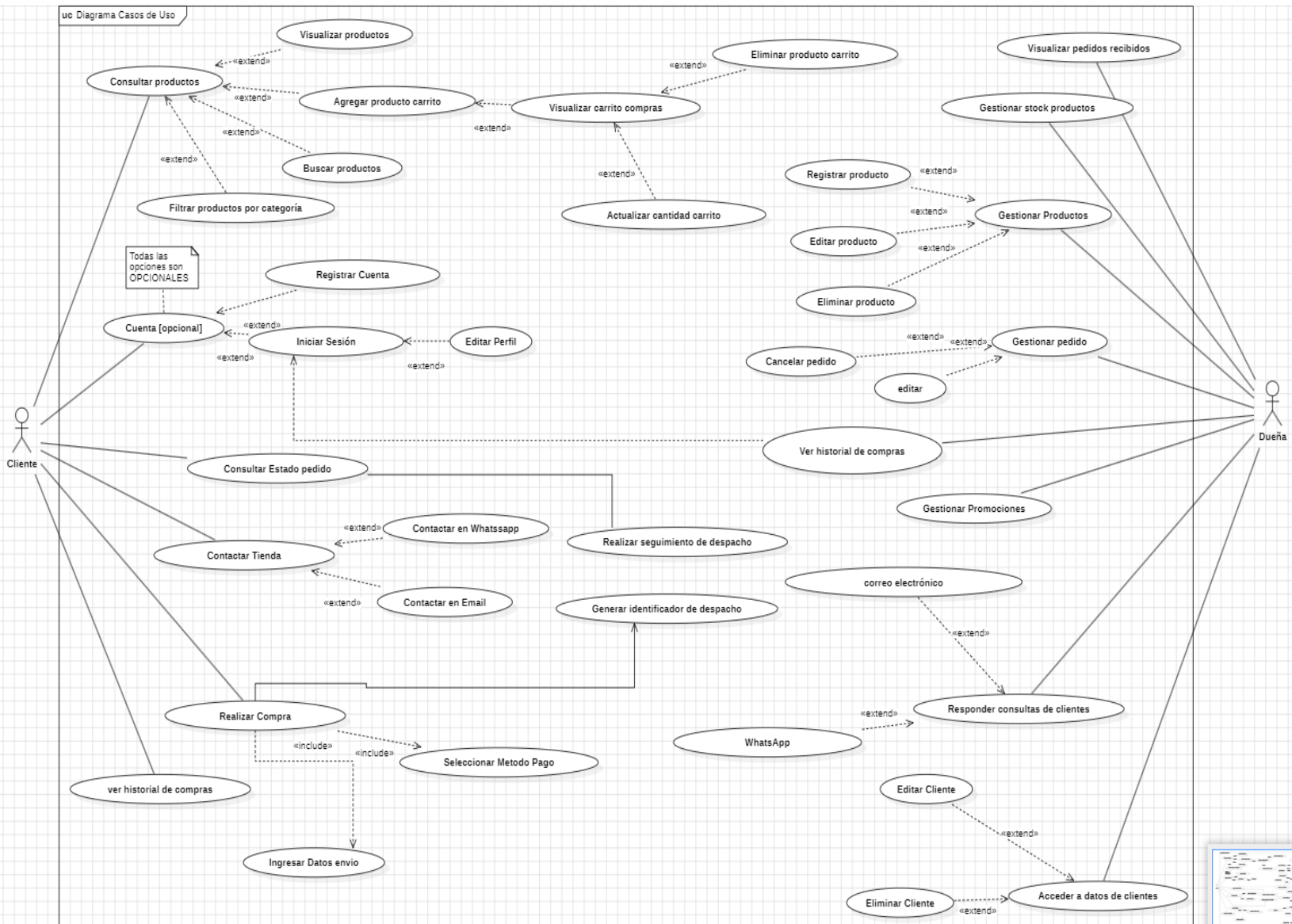
El modelo 4+1 organiza las vistas de nuestro sistema de sw para que distintos actores entiendan el proyecto desde sus intereses.



Vista de Escenarios

Mostramos cómo los usuarios interactúan con nuestro sistema y qué funcionalidades esperan, ya que con esto podemos capturar los requisitos funcionales y validar que la arquitectura (**microservicios**) que propusimos sea capaz de soportar los casos clave definidos para su funcionamiento.

➤ Casos de Uso

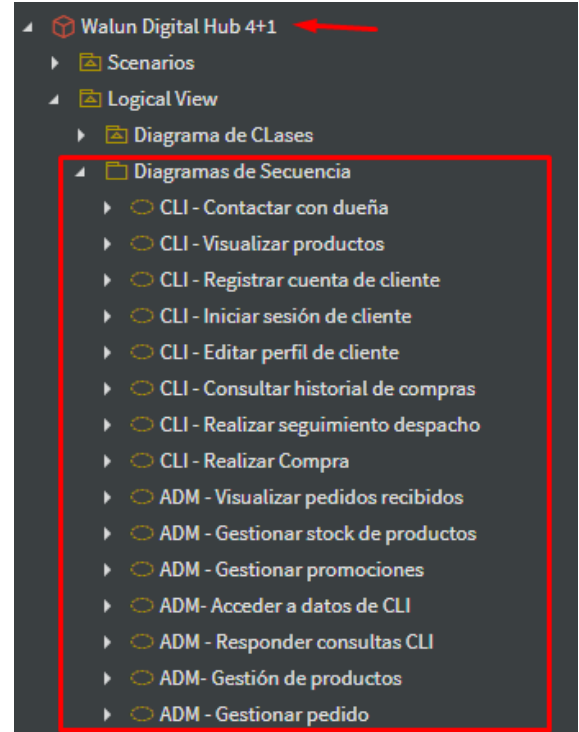


Vista Lógica

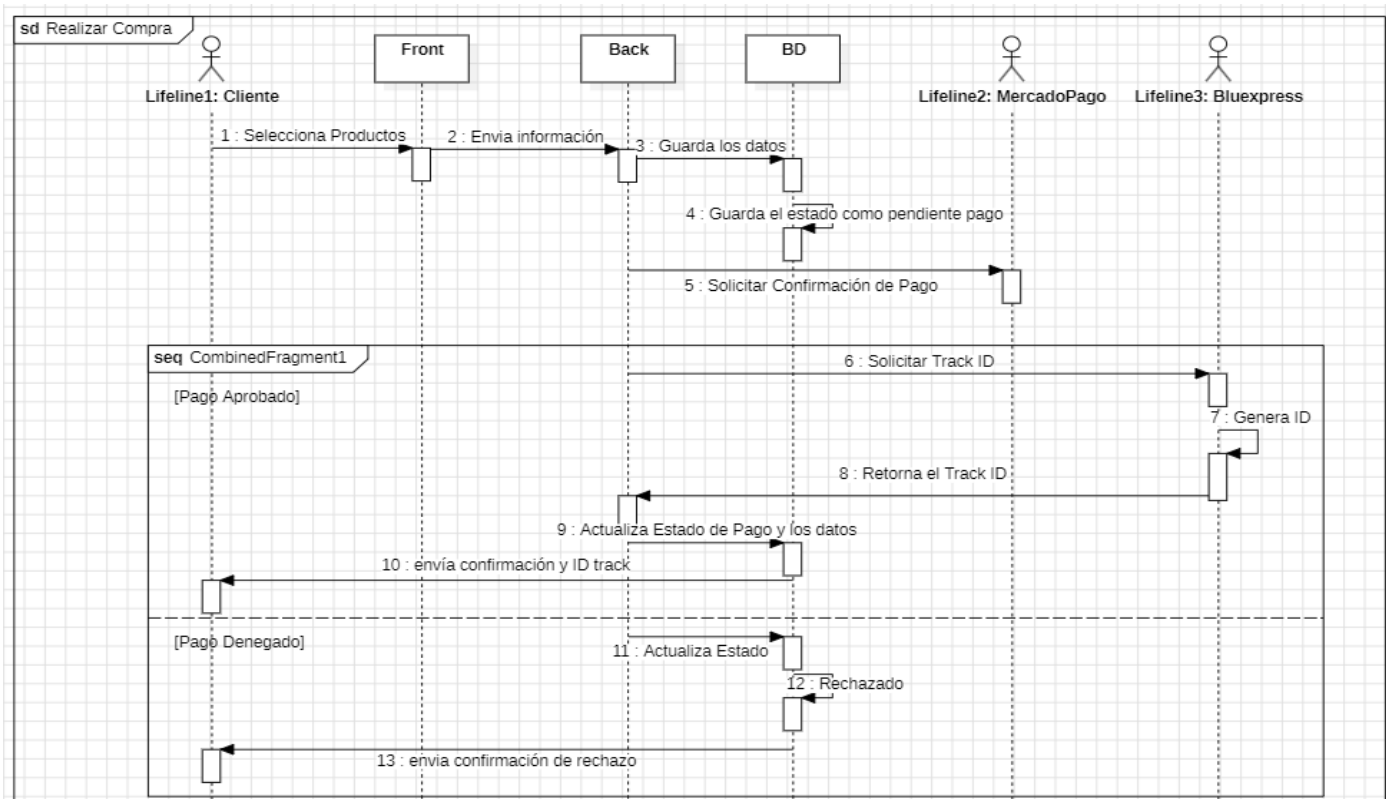
En la vista lógica representamos la estructura estática de nuestro sistema, donde mostramos las **clases, entidades, relaciones y componentes principales**, ya que con esto podemos asegurar que los desarrolladores comprendan de manera clara la organización interna del proyecto.

➤ Diagramas de Secuencia

Se realizaron **15 diagramas** de secuencia en total.
(uno por c/u de los casos de uso)

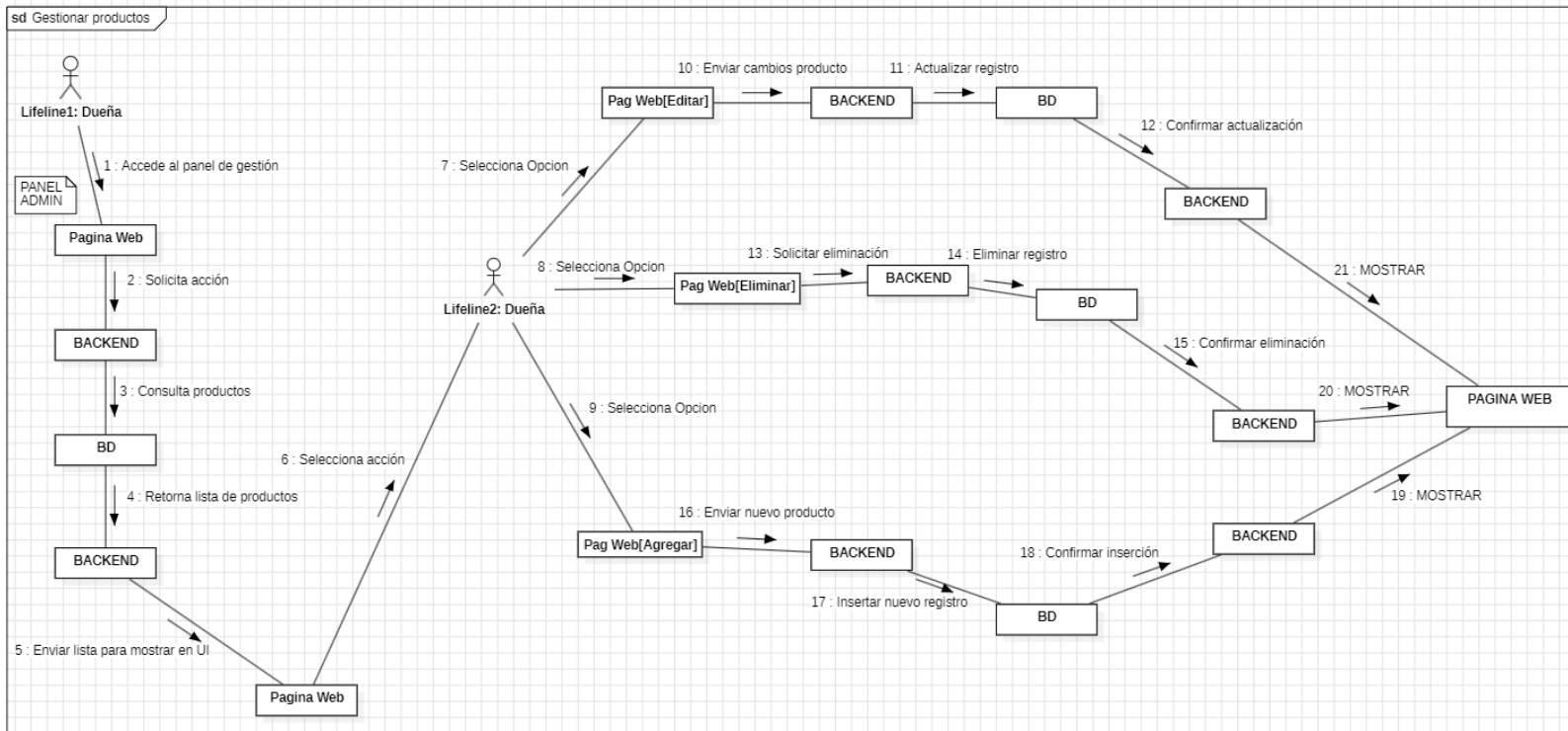


EJ: CLI - Realizar Compra



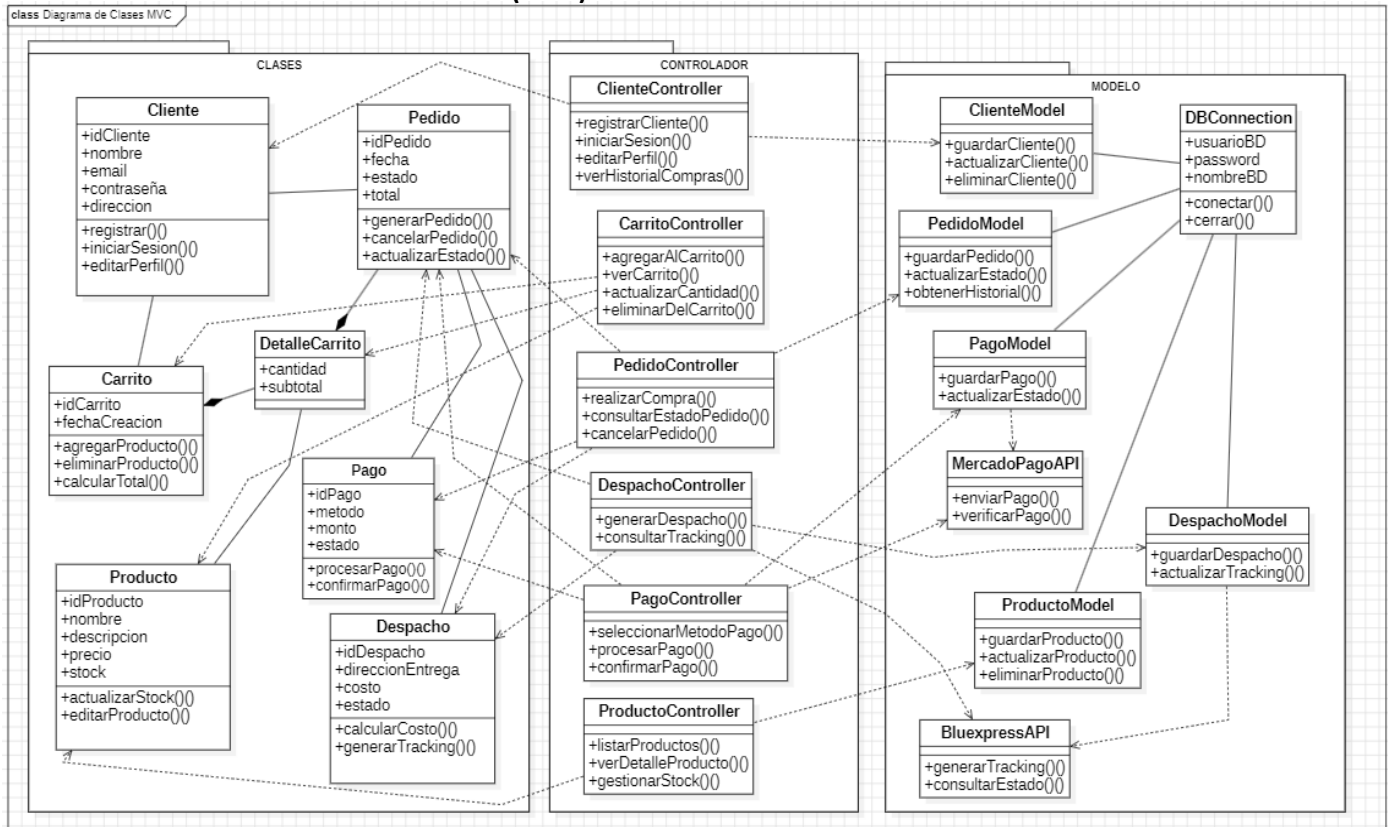
➤ Diagramas de Comunicación (3)

EJ: DG COM. Gestionar Productos 4+1



➤ Diagrama de Clases

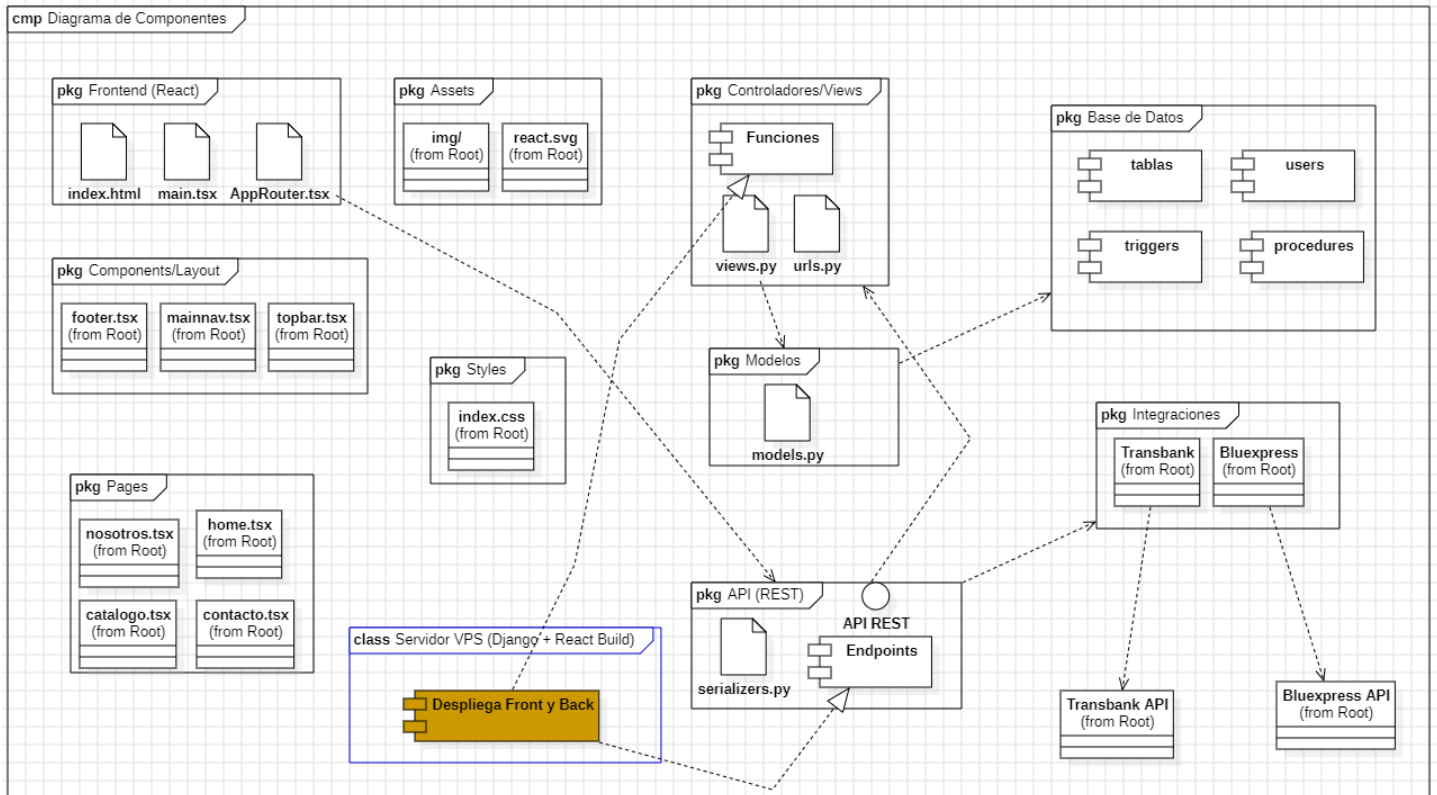
EJ: Modelo Vista Controlador (MVC)



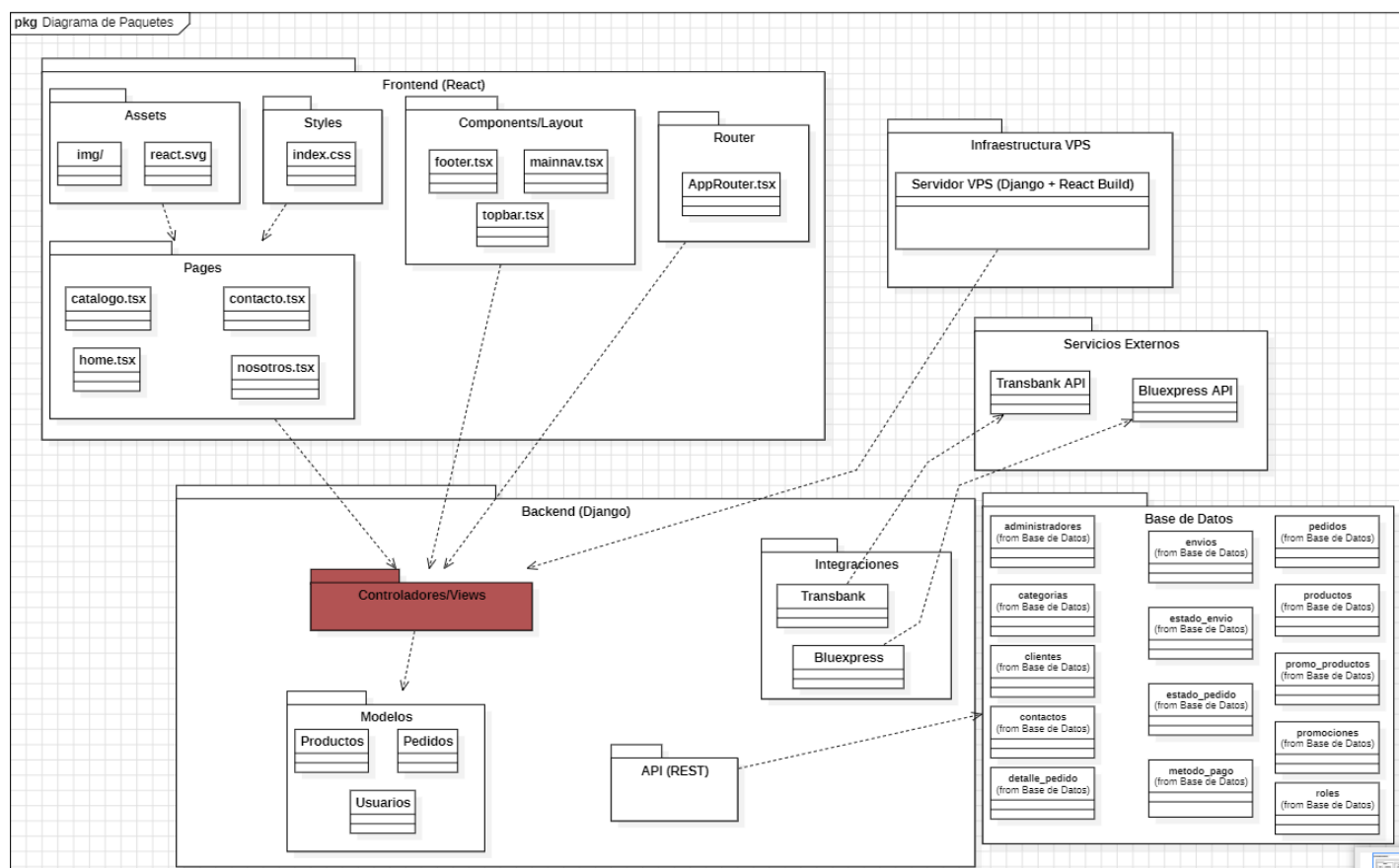
Vista de Despliegue

En la vista de despliegue describimos cómo se desplegará físicamente nuestro sistema en el hardware y la red, lo que nos permite identificar servidores, conexiones, balanceadores, bases de datos y otros componentes de la infraestructura.

➤ Diagrama de Componentes



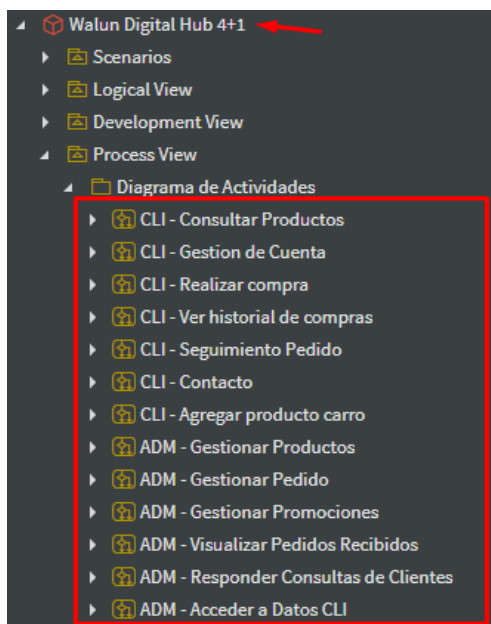
➤ Diagrama de Paquetes



Vista de Procesos

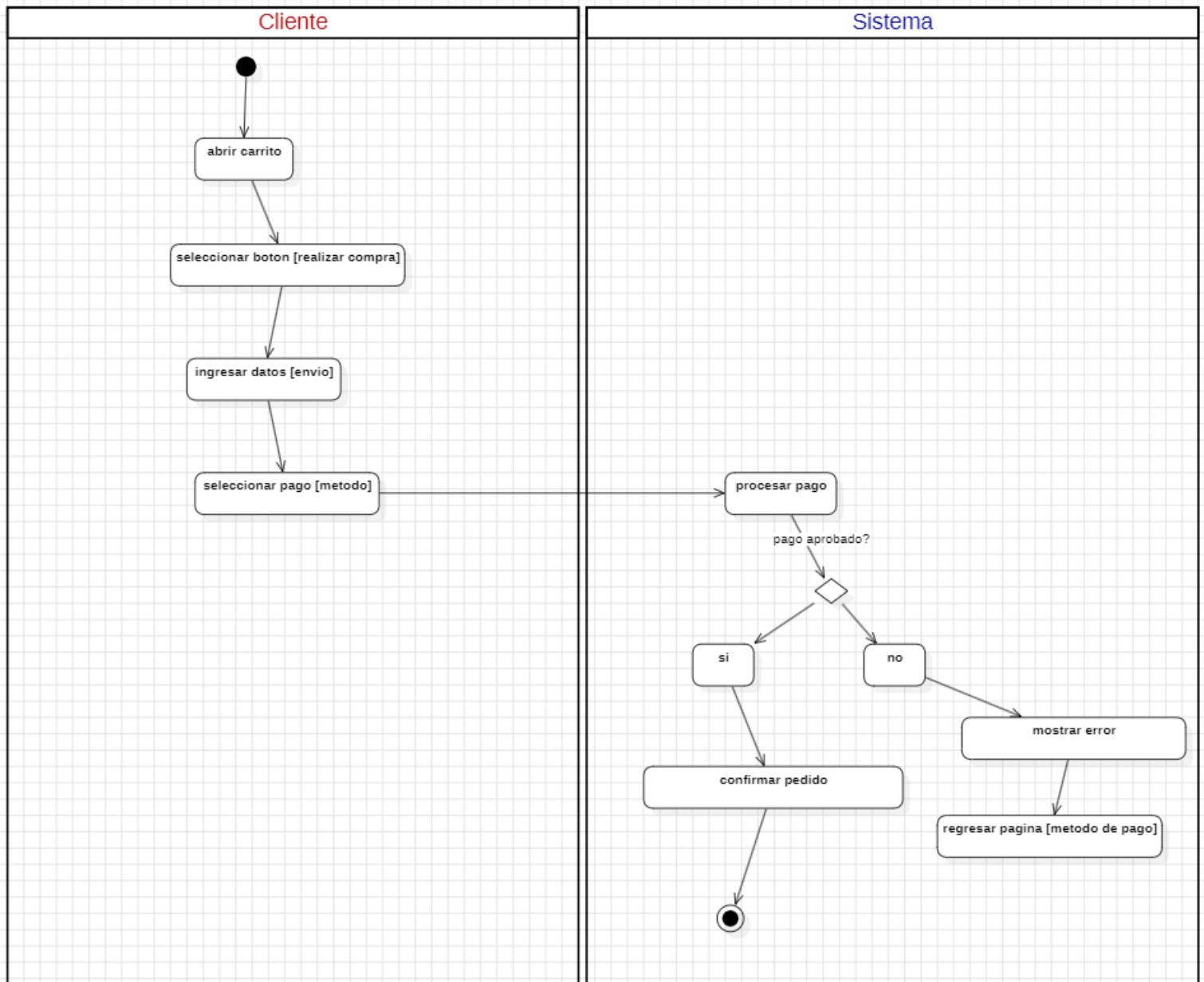
En la vista de procesos mostramos el comportamiento en ejecución de nuestro sistema, representando los procesos concurrentes, los hilos, la comunicación entre ellos y la sincronización, lo que resulta clave para analizar el rendimiento, la concurrencia y la escalabilidad de la solución que estamos desarrollando.

Se realizaron **13 diagramas** de actividad en total.
(uno por c/u de los casos de uso)



➤ Diagramas de Actividad

EJ: CLI – Realizar Compra (pagar)



Vista Física

En la vista física nos enfocamos en cómo se organiza el código fuente, las librerías, los módulos y los repositorios, ya que esto es fundamental para la gestión de la configuración y el control de versiones dentro de nuestro proyecto.

➤ Diagrama de Despliegue

