Arquitectura de software II

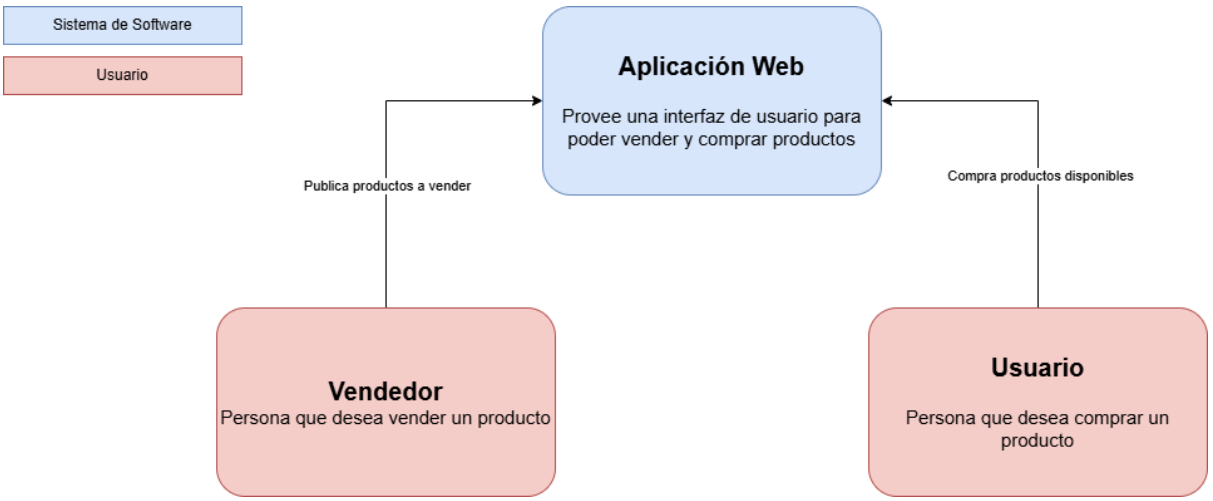
Trabajo Práctico N1- 2025s1

Jeremias Fuentes

Diagrama de arquitectura

Usaré el modelo c4 para describir el sistema, yendo desde una explicación más general a los componentes más específicos.

- Contexto

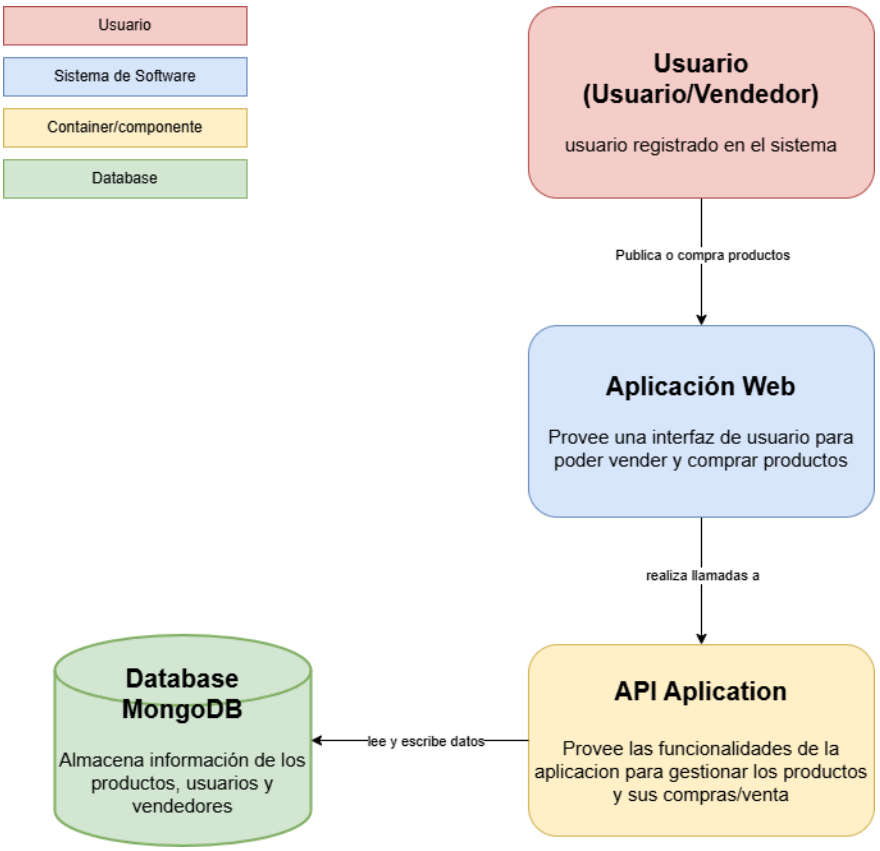


Usuarios principales:

- **Vendedor**: Usa la aplicación para publicar y vender productos.
- **Usuario**: Usa la aplicación para buscar y comprar productos.

Aplicación Web: Actúa como intermediario entre vendedores y usuarios para realizar la compra/venta de productos.

- Containers



Aplicación Web:

- Para Usuarios y Vendedores. Permite publicar productos a la venta, buscarlos y comprarlos.

API Application:

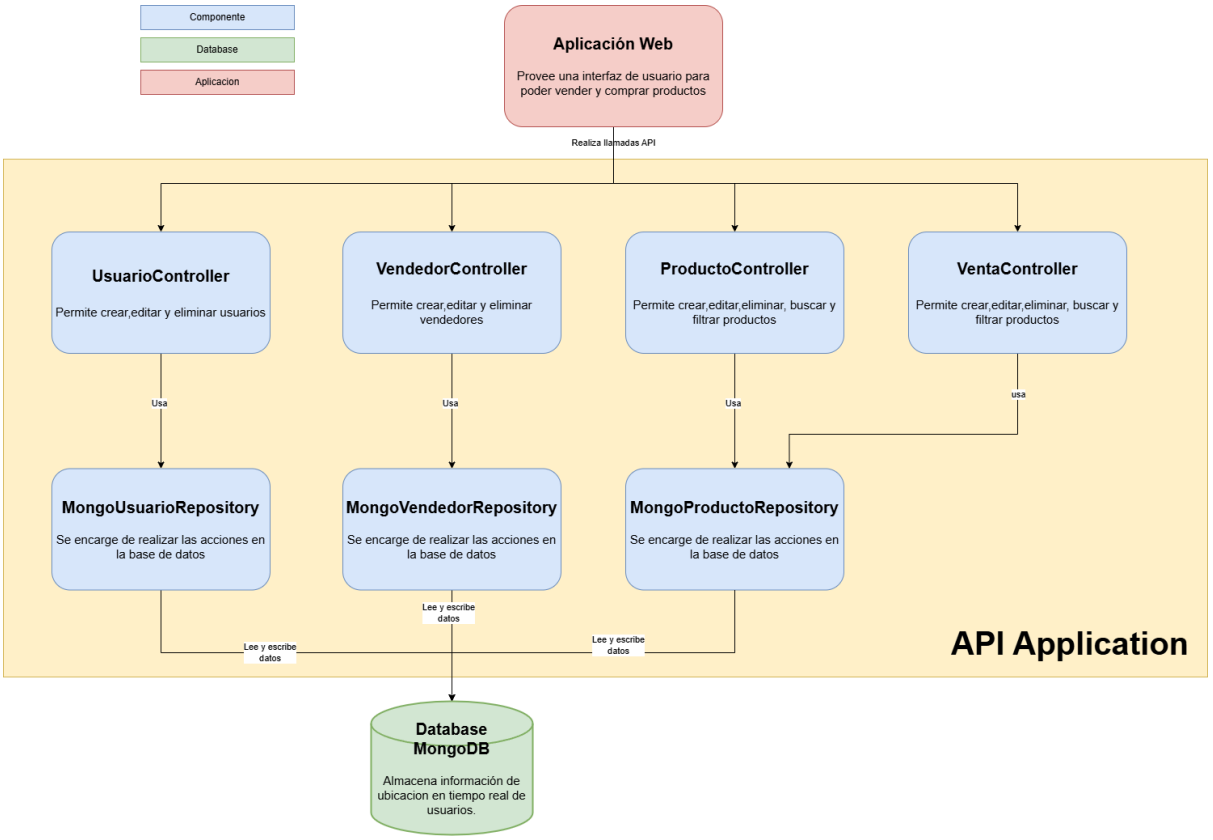
- Gestiona los vendedores, usuarios y productos,permitiendo realizar una venta.

Base de Datos (MongoDB):

- Almacena información de Usuarios, Vendedores y Productos.

- Componentes

(En el sistema C4 el diagrama de componentes funciona como diagrama de módulos.)



Explicación general de la arquitectura:

La plataforma de compra-venta de productos está diseñada como un **único componente de software**, siguiendo una arquitectura modular inspirada en principios de **Arquitectura Hexagonal**.
Esta organización busca separar las responsabilidades en **Adaptadores**, **Puertos** y **Dominio**, garantizando **escalabilidad**, **mantenibilidad** y **flexibilidad** para futuras integraciones o cambios tecnológicos.

Toda la interacción de los usuarios con la plataforma se realiza a través de una **Aplicación Web**, que se comunica mediante llamadas API al componente principal.

Componentes Principales:

Aplicación Web

Responsabilidad:

- Proveer una interfaz de usuario para permitir la creación, edición, eliminación, búsqueda y compra de productos.

Dependencias:

- Se comunica exclusivamente con los Controllers expuestos por la API de la plataforma mediante **llamadas API REST (HTTP/JSON)**.

Supuestos:

- Los usuarios tienen acceso a Internet y un navegador compatible.

Justificación:

- Centralizar la interacción en una única aplicación facilita el control de la experiencia de usuario, asegura la escalabilidad del sistema y simplifica el mantenimiento.

UsuarioController

Responsabilidad:

- Permite crear, editar y eliminar usuarios.

Dependencias:

- Usa `MongoUsuarioRepository` para leer y escribir datos de usuarios en la base de datos.

Supuestos:

- Los datos de los usuarios deben ser consistentes y únicos (especialmente el email).

Justificación:

- Mantener un controlador específico para usuarios permite modularizar la lógica y facilita futuras extensiones como autenticación o validación avanzada.

VendedorController

Responsabilidad:

- Permite crear, editar y eliminar vendedores.

Dependencias:

- Usa `MongoVendedorRepository` para manejar la persistencia de vendedores en MongoDB.

Supuestos:

- Los vendedores pueden publicar múltiples productos en la plataforma.

Justificación:

- Separar los vendedores de los usuarios permite modelar correctamente las diferencias de roles y responsabilidades en el sistema.

ProductoController

Responsabilidad:

- Permite crear, editar, eliminar, buscar y filtrar productos.

Dependencias:

- Usa `MongoProductoRepository` para gestionar productos en la base de datos.

Supuestos:

- Los productos tienen stock controlado y un vendedor asociado.

Justificación:

- Un controlador especializado permite aplicar reglas específicas sobre productos, como validaciones de stock o rangos de precios en búsquedas.

VentaController

Responsabilidad:

- Permite procesar ventas, actualizar stock de productos, y manejar la lógica de compra.

Dependencias:

- Usa `MongoProductoRepository` para actualizar el stock de productos vendidos.

Supuestos:

- No se permite vender productos si el stock disponible es insuficiente.

Justificación:

- Centralizar la lógica de venta facilita la validación de procesos críticos y asegura la consistencia en el stock de productos.

MongoUsuarioRepository

Responsabilidad:

- Realizar las operaciones CRUD sobre los usuarios en MongoDB.

Dependencias:

- Base de datos MongoDB Atlas.

Supuestos:

- La conexión a MongoDB es persistente y segura.

Justificación:

- Aislar el acceso a datos asegura que el dominio no dependa de la tecnología de persistencia.

MongoVendedorRepository

Responsabilidad:

- Manejar la persistencia de los datos de los vendedores.

Dependencias:

- Base de datos MongoDB Atlas.

Supuestos:

- Cada vendedor tiene un email único.

Justificación:

- Abstracción del acceso a datos para permitir flexibilidad y cambios tecnológicos futuros.

MongoProductoRepository

Responsabilidad:

- Gestionar la lectura, escritura y actualización de productos en la base de datos.

Dependencias:

- Base de datos MongoDB Atlas.

Supuestos:

- El stock debe ser consistente en todas las operaciones de venta.

Justificación:

- Permite controlar la integridad de los productos publicados en la plataforma.

Database MongoDB

Responsabilidad:

- Almacenar usuarios, vendedores y productos.

Dependencias:

- MongoRepositories para todas las operaciones de lectura y escritura.

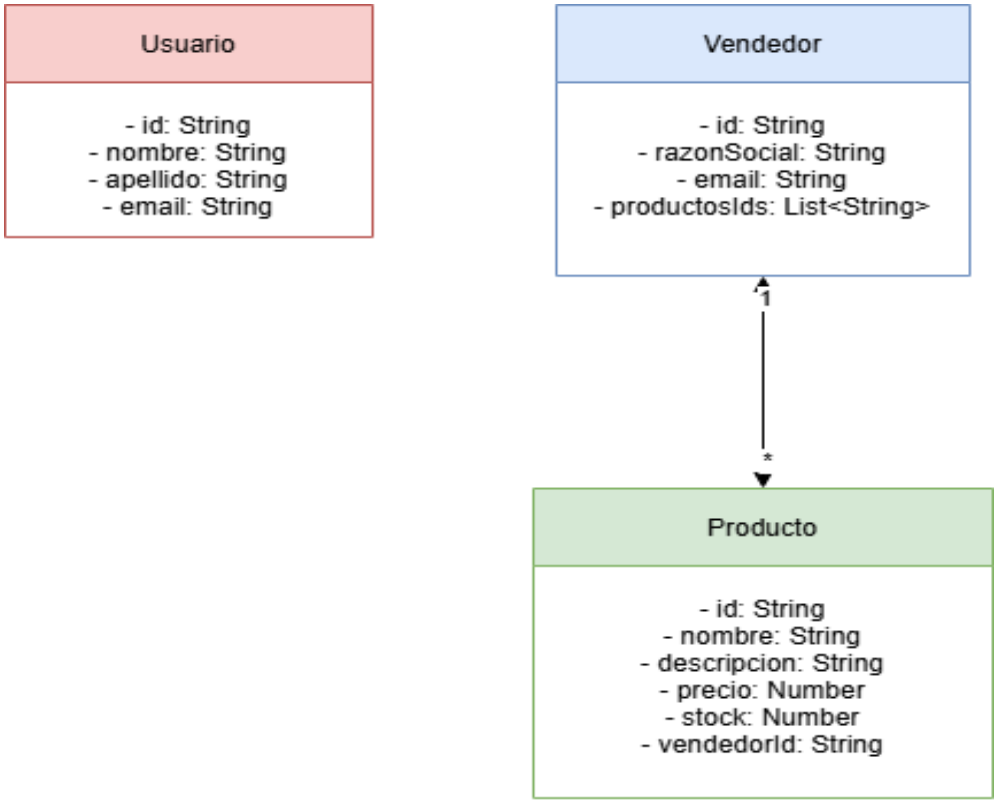
Supuestos:

- MongoDB maneja la persistencia de forma segura, rápida y escalable.

Justificación:

- MongoDB es adecuado para esquemas flexibles y escalables, ideal para una plataforma de e-commerce en crecimiento.

Diagrama de Entidades:



Entidades Principales

1. **Usuario**: Representa a todos los usuarios registrados en la plataforma.
    - **Atributos**:
        - `id` (String, PK): Identificador único del usuario.
        - `nombre` (String): Nombre del usuario.
        - `apellido` (String): Apellido del usuario.
        - `email` (String, único): Correo electrónico del usuario.

2. **Vendedor**: Contiene la información específica de los usuarios que publican productos a la venta..
    - **Atributos**:
        - `id` (String, PK): Identificador único del vendedor.
        - `razonSocial` (String): Nombre comercial o razón social del vendedor.

        - `email` (String, único): Correo electrónico de contacto del vendedor.
        - `productosIds` (List<String>): Identificadores de los productos publicados por el vendedor.

3. **Producto**: Representa un artículo disponible para la compra dentro de la plataforma.
    - **Atributos**:
        - `id` (String, PK): Identificador único del producto.
        - `nombre` (String): Nombre del producto.
        - `descripcion` (String): Descripción detallada del producto.
        - `precio` (Number): Precio de venta del producto.
        - `stock` (Number): Cantidad disponible en inventario.
        - `vendedorId` (String, FK a Vendedor): Referencia al vendedor que publicó el producto.

Relaciones Clave

- **Vendedor - Producto**:
  Relación **uno a muchos**
  ➔ Un vendedor puede publicar múltiples productos.
  ➔ Cada producto pertenece a un único vendedor.

- **Usuario - (compra Producto)**:
  Relación indirecta a través del procesamiento de ventas (no hay FK directo en este modelo básico, pero se implementa a través del proceso de "procesar venta").

Agregados

- **Vendedor** podría ser considerado un Agregado, ya que agrupa:
    - Información propia (`razonSocial`, `email`)
    - Y una relación lógica con sus productos publicados (`productosIds`).
- **Producto** también puede ser tratado como un Agregado independiente para simplificar la gestión de inventario.

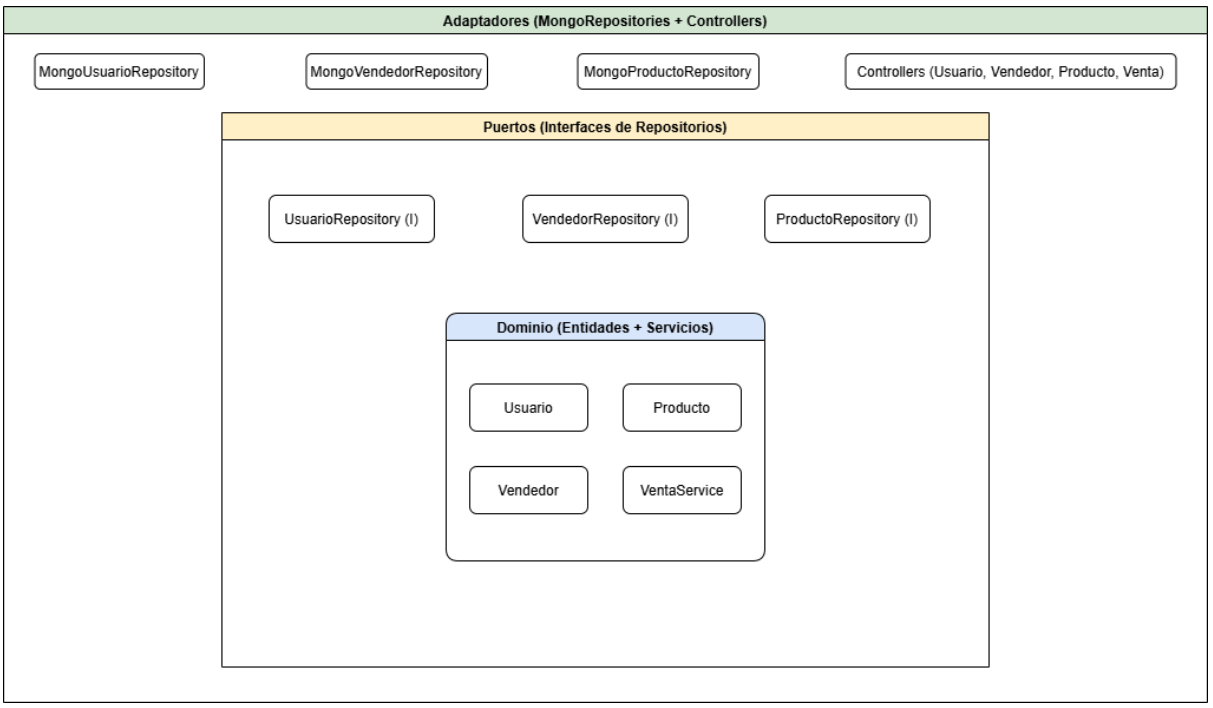
Agregados Raiz

- **Vendedor** actúa como Agregado Raíz para su colección de productos.
- **Producto** es su propio Agregado Raíz (cuando se modela independiente de la entidad Vendedor).

Objetos de Valor

- **Email**:
    - Asociado tanto a Usuario como a Vendedor.
    - Reglas como "debe ser válido", "debe ser único", etc.
- **Precio**:
    - Valor asociado a un Producto.
    - Se podría validar que el precio siempre sea positivo.

Diagrama de Clases



Dominio

El **Dominio** se encuentra en el núcleo del sistema y contiene:

- **Entidades**:
  - `Usuario`
  - `Vendedor`
  - `Producto`
- **Servicios de Dominio**:
  - `VentaService`: contiene reglas de negocio relacionadas al stock y procesamiento de ventas.

El Dominio no depende de infraestructura externa ni de frameworks. Solo define las reglas puras del negocio.

Puertos

Los **Puertos** son las interfaces que conectan el Dominio hacia el exterior:

- `UsuarioRepository`
- `VendedorRepository`
- `ProductoRepository`

Estas interfaces definen las operaciones requeridas para persistir o recuperar información, sin depender de tecnologías concretas.

Adaptadores

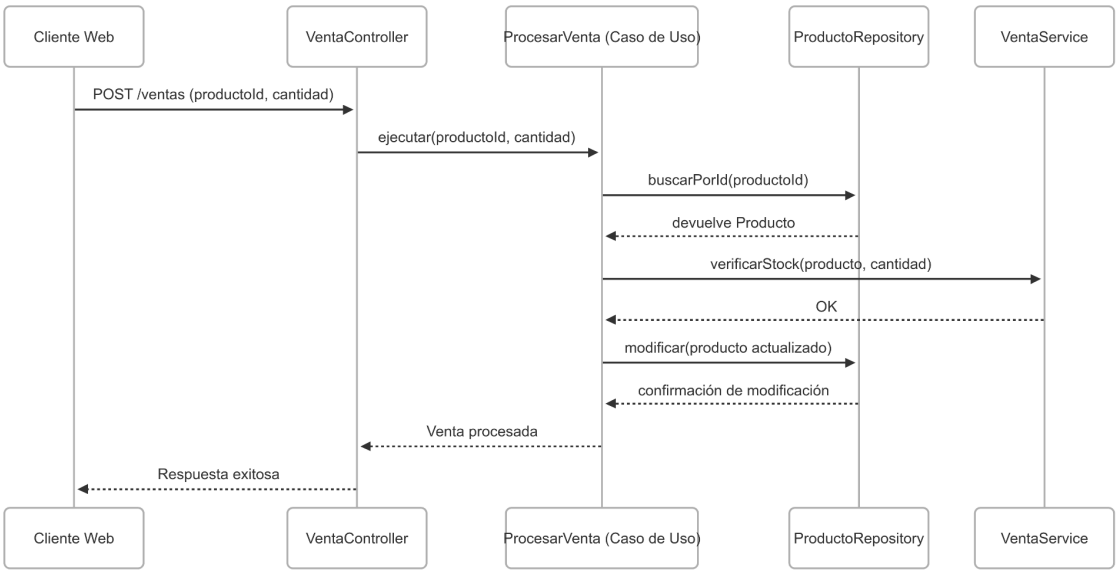
Los **Adaptadores** implementan los puertos para integrar el Dominio con tecnologías externas:

- **Adaptadores de Salida** (persistencia):
  - `MongoUsuarioRepository`
  - `MongoVendedorRepository`
  - `MongoProductoRepository`
- **Adaptadores de Entrada** (API REST):
  - `UsuarioController`
  - `VendedorController`
  - `ProductoController`
  - `VentaController`

Los Adaptadores permiten que el Dominio se mantenga limpio y separado de detalles técnicos como MongoDB o HTTP.

Diagrama de Secuencia

Diagrama de Secuencia 1: Procesar Venta



- Cliente Web llama al Controller.
- Controller invoca el Caso de Uso.
- Caso de Uso consulta Repositorio.
- Caso de Uso usa Servicio de Dominio.
- Caso de Uso actualiza Repositorio.
- Controller responde.

Diagrama de Secuencia 2: Crear Producto

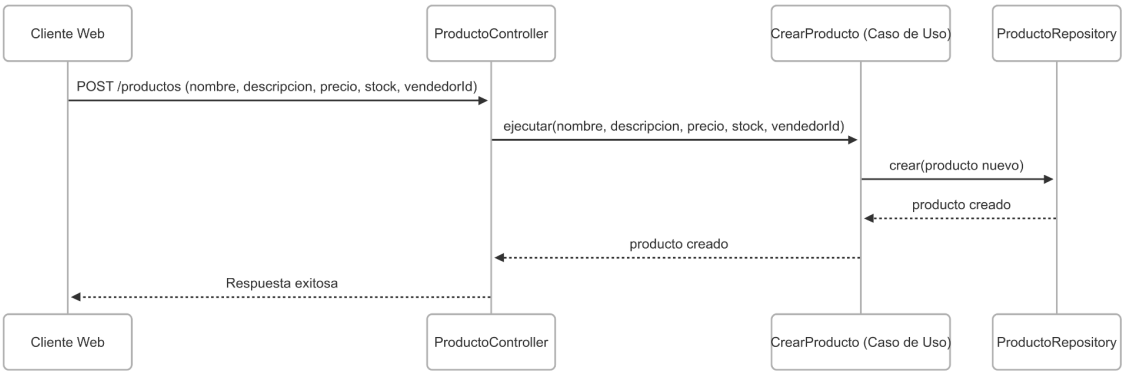


Diagrama de Secuencia 3: Modificar Producto

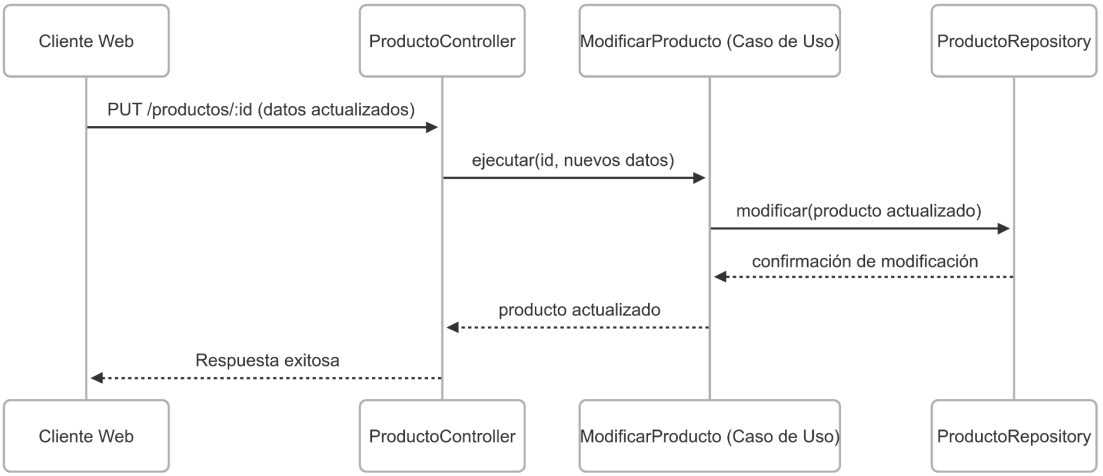


Diagrama de Secuencia 4: Eliminar Producto

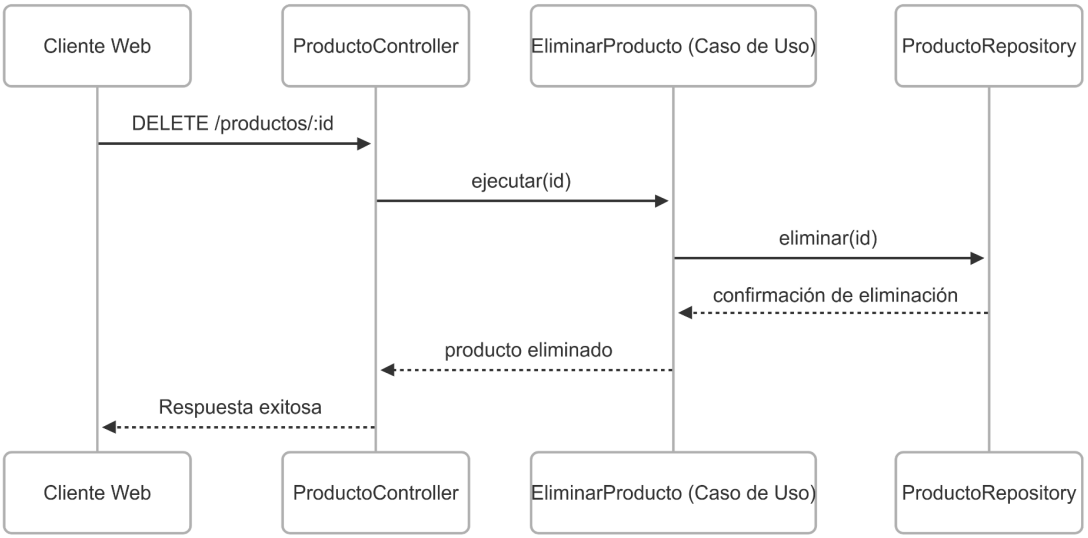
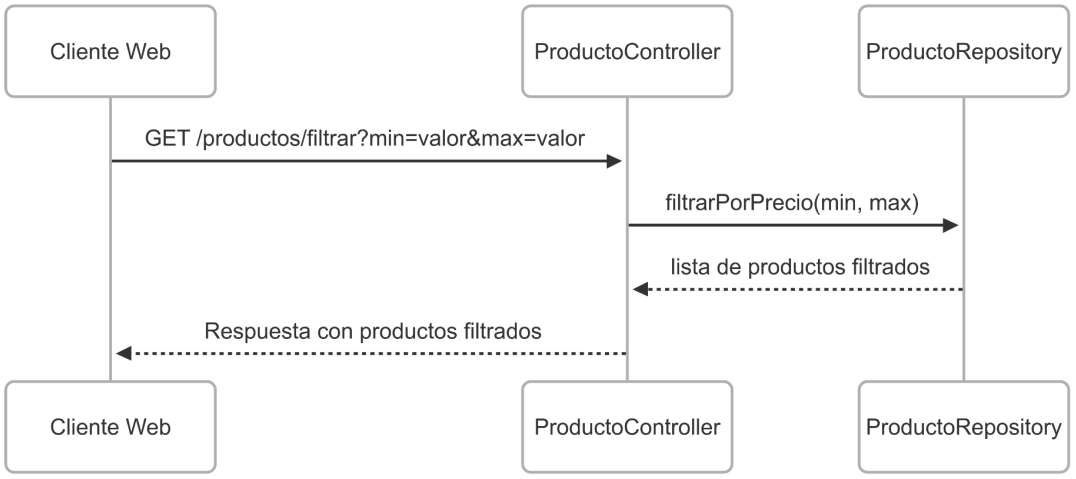


Diagrama de Secuencia 5: Filtrar producto por precio



Referencias:

Casos de uso:

- Plantilla de Descripción de Caso de Uso
- https://www.ibm.com/docs/es/product-master/12.0.0?topic=processes-defining-use-cases
- https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416

Atributos de calidad:

- https://cs.uns.edu.ar/~ece/ads/downloads/Clases/2019%2006%20AyDS%20-%20Atributos%20de%20Calidad%20Parte%201.pdf

Diagrama de arquitectura (módulos):

- https://c4model.com/
- https://adictosaltrabajo.com/2023/05/06/diagramas-de-arquitectura-con-c4-model/

Diagrama de secuencia:

- https://www.mermaidchart.com/
- https://support.microsoft.com/es-es/topic/crear-un-diagrama-de-secuencia-de-uml-c61c371b-b150-4958-b128-902000133b26

Modelo de datos:

- https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos
- https://www.uv.mx/personal/lizhernandez/files/2013/03/4.-ModeloRelacional.pdf