Tecnológico Nacional de México

Instituto Tecnológico de Culiacán

Ingeniería en Sistemas Computacionales

Proyecto:

Te Acerco Salud

Materias:

Proyecto Integrador de Ingeniería de Software

Ingeniería Web

Docentes:

MC. Martha Estela Valenzuela Tirado

DR. Clemente García Gerardo

Alumnos:

Jorge Eduardo Escalante Guadarrama

Carlos Iván Carrasco Medina

Jatniel Alejandro Peña Vizcarra

Grupo:

9:00 AM – 11:00 AM

Fecha de entrega: 2 de diciembre de 2025

Índice

Descripción del Problema

Suponga que usted, como Ingeniero en Sistemas Computacionales, ha sido contactado por la empresa con razón social **Te Acerco Salud (TAS)** para solicitar sus servicios profesionales para el desarrollo de aplicaciones de software tanto para la web como para dispositivos móviles.

En el sector salud, uno de los principales retos que enfrentan los pacientes es el surtido completo de sus recetas médicas. Al acudir a una consulta, los pacientes reciben prescripciones que, en muchas ocasiones, no logran satisfacer en una sola farmacia, lo que los obliga a recorrer diferentes sucursales o cadenas farmacéuticas. Esta situación genera pérdida de tiempo, incremento en los costos de traslado, retrasos en el inicio de los tratamientos y, en casos críticos, riesgos para la salud del paciente al no contar de manera oportuna con los medicamentos prescritos.

La fragmentación del inventario entre las distintas cadenas de farmacias, sumada a la falta de un sistema de colaboración, provoca una experiencia poco eficiente y frustrante para los usuarios. Actualmente, **no existe** un mecanismo que permita al paciente enviar su receta a una sucursal de su preferencia y asegurarse de que ahí podrá obtener todos los medicamentos prescritos; ya sea porque esa farmacia no cuente con ellos o porque o no existe un acuerdo de trabajo entre diferentes cadenas.

Por lo tanto, surge la necesidad de desarrollar una **aplicación web** que facilite la conexión entre pacientes y farmacias, permitiendo a los usuarios seleccionar la sucursal más conveniente, cargar su receta y garantizar la disponibilidad de los medicamentos requeridos en la sucursal seleccionada (todos los productos que no tenga la sucursal seleccionada se surtirán de la(s) farmacia(s) más cercana para disminuir tiempo de recolección).

Con esta solución tecnológica, se busca optimizar la experiencia del paciente, mejorar la coordinación entre las farmacias y reducir las barreras que actualmente dificultan el acceso oportuno a los tratamientos médicos.

Algunas políticas de TAS:

- Los pacientes deben registrarse como usuario en la aplicación web.
- Las cadenas farmacéuticas deben registrar las sucursales que forman parte de este mecanismo de cooperación.
- Cada sucursal debe mantener actualizado su inventario de productos para garantizar la disponibilidad de medicamentos y el cumplimiento del mecanismo de cooperación.

De esta manera, la aplicación propuesta busca optimizar la experiencia del paciente, mejorar la coordinación entre las farmacias y reducir las barreras que actualmente dificultan el acceso oportuno a los tratamientos médicos.

La empresa no tiene presencia en la web y desea lanzar un sitio web (www.teacercosalud.com) junto con una aplicación móvil, para que los empleados de las distintas cadenas farmacéuticas y los pacientes puedan realizar sus actividades de manera eficiente, tales como: cotizaciones, registros, rastreos, disponibilidad de espacio.

Tecnologías a utilizar:

1. Metodología de desarrollo: PU (Proceso Unificado), SCRUM.
2. CASE: Enterprise Architect.
3. Notación de modelado: UML y UWE.
4. Marco de desarrollo: LARAVEL (MVC -Modelo Vista Controlador-).
5. Paradigma de desarrollo: Orientado a objetos.
6. Android Studio.
7. Sistema Gestor de Base de Datos (Orientado a Objetos, Relacional).
8. Servicio web.

Fase de Inicio

Visión y Análisis del Negocio

- *Historia de revisiones*

Versión	Fecha	Descripción	Autores
Borrador de Inicio	15/09/25	Primer Borrador: Para refinarse durante la elaboración	- Equipo 3
Corrección del 1er avance	20/09/25	Corrección de la introducción y especificación de las consecuencias en las farmacias	- Equipo 3
Corrección del 2do avance	08/10/25	Corrección en la perspectiva del producto	- Equipo 3

- *Introducción*

Desarrollamos una aplicación de sistema de pedidos (OMS) tolerante a fallos de próxima generación, Te Acerco Salud (TAS), con interoperabilidad con otras cadenas, múltiples mecanismos e interfaz de usuario y la integración de servicios externos.

- *Enunciado del Problema*

El proceso actual para el surtido de recetas médicas es difícil, carece de coordinación entre farmacias y no cuenta con un mecanismo que permita a los pacientes enviar su receta a una sucursal y garantizar así la disponibilidad completa de los medicamentos prescritos, esto provoca a las farmacias la pérdida de clientes por no encontrar los medicamentos en un solo lugar, dificultad para ofrecer un servicio integral. Además, esta situación afecta directamente a los pacientes, que los obliga a recorrer distintas sucursales o cadenas farmacéuticas que les genera una pérdida de tiempo, incremento en los costos de traslado, retrasos en el inicio de los tratamientos y, en casos críticos, riesgos para la salud del paciente al no contar de manera oportuna con los medicamentos prescritos.

- *Enunciado de la Posición en el Mercado del Producto*

El producto está dirigido principalmente a pacientes que necesitan surtir de manera

completa y rápida sus recetas médicas, y a cadenas y sucursales farmacéuticas que buscan una clientela persistente, optimizar su inventario y ofrecer un servicio integral mediante la colaboración entre distintas farmacias.

- *Alternativas y Competencia*

Actualmente, los pacientes que desean surtir completamente sus recetas médicas cuentan solo con opciones parciales:

  - Farmacias físicas y en línea por separado: Cada una maneja su propio inventario sin coordinación entre farmacias de distintas cadenas.
  - Apps de entrega a domicilio: Facilitan el pedido en una sola tienda, pero no garantizan surtido completo en la misma farmacia.
  - Sistemas internos de cada cadena: Orientados a venta e inventario dentro de la misma empresa, sin interoperabilidad con otras farmacias de la misma cadena.

Estas alternativas no ofrecen un mecanismo integrado para que un paciente cargue su receta, seleccione una sucursal preferida y tenga la seguridad de recibir todos sus medicamentos en una sola entrega.

- *Descripción del Personal Involucrado*
  - Pacientes: Personas que registran su cuenta y suben sus recetas para un surtido completo.
  - Farmacéuticos: Actualizan inventarios, reciben recetas, preparan el surtido y confirman la disponibilidad de la farmacia.
  - Administrador de Farmacia: Encargado de dar de alta las distintas farmacias de la cadena y la gestión de los farmacéuticos.
  - Administrador General: Es el administrador del sistema TAS, encargado de gestionar los administradores de farmacias.
- *Objetivos de Alto Nivel y Problemas Claves del Personal Involucrado*

Objetivos de Alto Nivel	Prioridad	Problemas e Inquietudes	Soluciones Actuales
Garantizar al paciente el surtido completo de su receta	Alta	Falta de inventario, pérdida de tiempo y dinero al recorrer varias farmacias, retrasos en su tratamiento y riesgos para la salud	El paciente llama o visita varias farmacias, o usa aplicaciones de entrega de una sola farmacia, sin garantía de un surtido completo

Mantener inventarios actualizados y disponibles	Alta-Media	Inventarios desactualizados, dificultad para confirmar el surtido completo de la receta	Sistemas PDV internos, comunicación telefónica entre farmacias
Mejorar la coordinación entre cadenas farmacéuticas	Alta	Falta de colaboración, pérdida de clientes por no encontrar los medicamentos en un solo lugar, dificultad para ofrecer un servicio integral	Los sistemas internos de cada cadena solamente gestionan inventarios propios, sin la comunicación con terceros

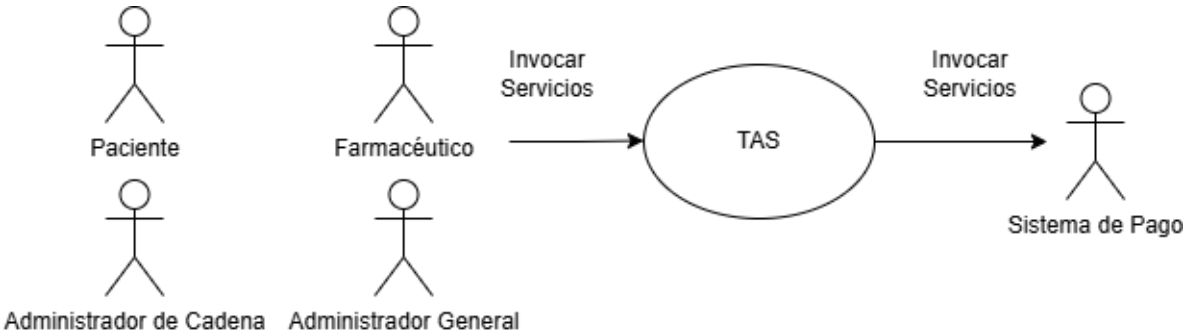
- *Objetivos a Nivel de Usuario*

Los usuarios (y los sistemas externos) necesitan un sistema para satisfacer sus objetivos:

- Paciente: Registrar su cuenta, ingresar al sistema, subir su receta, consultar el estado de sus recetas, recoger su receta.
- Farmacéutico: Actualizar inventarios en tiempo real, recibir recetas, preparar surtidos, confirmar disponibilidad y gestionar entregas.
- Administrador de Cadena: Registrar sus farmacias, gestionar el personal de farmacias.
- Administrador General: Gestionar los administradores de farmacias y coordinar la cooperación entre las distintas farmacias.

- *Perspectiva del Producto*

La plataforma Te Acerco Salud residirá principalmente como un servicio web y aplicación móvil. Proporcionará servicios directos a pacientes y empleados de farmacias, y colaborará con otros sistemas (por ejemplo, sistema de pago).



- *Resumen de Beneficios*

Característica Soportada	Beneficio al Personal Involucrado
Registro y autenticación de pacientes	Los pacientes pueden crear cuentas seguras y acceder desde cualquier dispositivo.
Carga de recetas médicas	Los pacientes suben su receta sin acudir físicamente, reduciendo traslados y tiempo.
Consulta de disponibilidad de medicamentos en tiempo real	Pacientes saben de inmediato que medicamentos hay en el sistema, farmacéuticos evitan llamadas y consultas manuales.
Coordinación automática entre farmacias para surtido completo	Los pacientes reciben todos los medicamentos en una sola recolección, las farmacias aumentan sus ventas colaborando con otras.
Actualización de inventarios en tiempo real	Administradores de sucursal mantienen datos precisos, se minimizan errores y faltantes.
Gestión centralizada de sucursales	Las farmacias administran inventarios desde un solo panel.
Notificaciones del estado del surtido	Pacientes reciben notificaciones sobre la preparación y disponibilidad de entrega de su receta surtida

- *Resumen de las Características del Sistema*
  - Registro y autenticación de pacientes.
  - Carga de recetas médicas por parte de los pacientes.
  - Coordinación automática para surtir medicamentos faltantes desde farmacias cercanas.
  - Actualización de inventarios en tiempo real por parte de administradores de sucursal.
  - Gestión centralizada de sucursales por cada cadena farmacéutica.
  - Notificaciones al paciente sobre el estado del surtido de su receta.
  - Integración con sistemas externos de pago, cuando sea necesario.
- *Otros Requisitos y Restricciones*
  - Disponibilidad y Fiabilidad: El sistema debe ser tolerante a fallos para garantizar la consulta y confirmación de medicamentos aun con interrupciones temporales.
  - Rendimiento: Respuesta en tiempo real para consultas de disponibilidad y

actualizaciones de inventario.
- Seguridad y privacidad: Cumplimiento de normas de protección de datos de los usuarios.
- Diseño responsivo: Acceso desde navegadores web y aplicaciones móviles para distintos dispositivos.
- Escalabilidad: Soportar el crecimiento de usuarios y farmacias sin degradación del rendimiento.
- Interoperabilidad: Posibilidad de integrarse con servicios externos.

Especificación Complementaria

- *Historia de Revisiones*

Versión	Fecha	Descripción	Autores
Borrador de Inicio	15/09/25	Primer Borrador: Para refinarse durante la elaboración	- Equipo 3

- *Introducción*

Este documento es el repositorio de todos los requisitos del sistema de pedidos (OMS) Te Acerco Salud (TAS) que no se capturan en los casos de uso.

- *Funcionalidad*
  - Registro y gestión de errores

Registrar todos los errores en un archivo de registro (log).

  - Reglas de negocio conectables

En varios puntos de los escenarios de varios casos de uso soportar la capacidad de adaptar la funcionalidad del sistema con un conjunto de reglas.

  - Seguridad

Todo uso requiere la autenticación de los usuarios.

- *Facilidad de Uso*

Factores humanos

El cliente será capaz de seleccionar la sucursal y los medicamentos que necesita para su receta médica, por lo tanto:

- Se debe brindar un sistema de búsqueda amigable para el usuario en el cual pueda encontrar los medicamentos.

- El sistema debe de contener un diseño practico y fácil de interactuar para seleccionar la sucursal de su preferencia.

- *Fiabilidad*

Capacidad de recuperación

Si ocurre un fallo en cualquier paso del proceso de subir receta, el sistema mostrará un mensaje de error y permitirá al usuario repetir la operación en el mismo momento o más tarde.

- *Rendimiento*

Los clientes quieren completar el proceso de surtido de receta muy rápido. por tanto:

- Cuando un paciente ingresa su receta, el sistema debe dar respuesta del estado del pedido sin demoras.
- El tiempo de carga de la página web y de la aplicación móvil debe ser corto, para que la experiencia sea fluida.
- El sistema debe soportar que cientos o miles de pacientes ingresen recetas al mismo tiempo, sobre todo en horarios pico (mañanas o fines de semana o temporadas de enfermedades).

- *Interfaces*

Interfaces y hardware destacable

- Monitor
- Computadora
- Teclado y ratón
- Celular

Interfaces software

Para los servicios externos se requieren de diversas interfaces.

- *Reglas del Dominio (Negocio)*

ID	Regla	Grado de Variación	Fuente
Regla 1	Después de notificar al cliente sobre el surtido de su receta, se le da un plazo de máximo 2 días para recoger la receta surtida.	Media (Se puede cambiar el plazo máximo de recogida)	Política de TAS

Regla 2	El Farmacéutico deberá verificar que las recetas que contengan medicamentos controlados aparezcan en la receta original, el paciente deberá llegar con su Receta original, que contengan los datos completos y que la firma del médico autorizado para prescribir medicamentos o corresponda a lo plasmado en el contenido de la receta.	Baja (Aplica solamente a los medicamentos controlados)	Secretaría de Salud
Regla 3	Se debe enviar notificaciones automáticas al usuario sobre el estado del pedido: recibido, en preparación, listo para recoger	Baja	Política de TAS
Regla 4	El monto total de la receta se pagará físicamente en la sucursal al momento de recibir la receta surtida.	Baja	Política de TAS
Regla 5	El Paciente siempre recibe todos los medicamentos de su receta en la sucursal seleccionada, aun cuando los medicamentos son surtidos desde otras sucursales	Media (Depende del inventario y coordinación entre las sucursales)	Política de TAS
Regla 6	El Paciente deberá ingresar solamente una tarjeta para realizar su registro en el sistema.	Baja	Política de TAS
Regla 7	Las recetas que no sean recogidas se le aplicarán una multa de 10% del monto total de la receta, y se	Media (Se puede modificar el porcentaje)	Política de TAS

	aplicará a la tarjeta del paciente.		
Regla 8	La receta no podrá ser generada si no se cuenta con stock suficiente de algún medicamento en los inventarios de las sucursales participantes.	Media (Depende del stock de los inventarios)	Política de TAS

Modelo de Casos de Uso

- *Historia de Revisiones*

Versión	Fecha	Descripción	Autores
Borrador de Inicio	15/09/25	Primer Borrador: Para refinarse durante la elaboración	- Equipo 3
Corrección del 1er avance	20/09/25	Corrección de la lista actor – semántica y objetivo	- Equipo 3
Corrección del 2do avance	08/10/25	Corrección del caso de uso CU-01 y Nuevo caso de USO CU-02	- Equipo 3

- *Lista Actor – Semántica*

Actor	Semántica
Paciente	Este es el principal usuario del sistema busca surtir sus recetas en una única farmacia, podrá registrarse e iniciar sesión.
Farmacéutico	Se encargará de administrar el inventario, administrar la recolección de los medicamentos y el surtido de la receta, la disponibilidad de la farmacia y la gestión de recetas que no fueron recogidas.

Administrador de Cadena	Es el encargado de agregar las distintas farmacias de su cadena y de gestionar los farmacéuticos de las farmacias.
Administrador General	Es el que administra y gestiona el Sistema TAS, permite registrar los administradores de las cadenas.

- *Lista Actor – Objetivo*

Actor	Objetivo
Paciente	– Registrarse – Iniciar Sesión – Subir Receta – Consultar el Estado de Receta – Recoger el surtido de Receta
Farmacéutico	– Preparar Receta – Gestionar Receta no recogida – Gestionar los Medicamentos del Inventario – Gestionar la Disponibilidad de la Sucursal
Administrador de Cadena	– Gestionar las Sucursales – Gestionar los Farmacéuticos en la Sucursal
Administrador General	– Gestionar los Administradores de Cadena

- *Lista de Casos de Uso*

Caso de uso: Registro de Paciente

- Actor: Paciente

Caso de uso: Inicio de Sesión de Paciente

- Actor: Paciente

Caso de uso: Subir Receta

- Actor: Paciente

Caso de uso: Consultar el Estado de Receta

- Actor: Paciente

Caso de uso: Recoger el surtido de Receta

- Actor: Paciente

Caso de uso: Preparar Receta

- Actor: Farmacéutico

Caso de uso: Gestionar Receta no recogida

- Actor: Farmacéutico

Caso de uso: Gestionar el Inventario

- Actor: Farmacéutico

Caso de uso: Gestionar la Disponibilidad de la Sucursal

- Actor: Farmacéutico

Caso de uso: Gestionar las Sucursales

- Actor: Administrador de Cadena

Caso de uso: Gestionar los Farmacéuticos de la Sucursal

- Actor: Administrador de Cadena

Caso de uso: Gestionar los Administradores de Cadenas

- Actor: Administrador General

- *Descripción Breve de Casos de Uso*

Registro de Paciente: Un Paciente ingresa por primera vez al Sistema, ingresa las credenciales que el Sistema requiere, el Sistema valida las credenciales y el Paciente queda registrado como un usuario Paciente.

Inicio de Sesión de Paciente: El Paciente quiere acceder al Sistema, el Paciente ingresa sus credenciales, el Sistema valida las credenciales y el Paciente ingresa al Sistema.

Subir receta: El Paciente tiene una Receta, selecciona la Sucursal de su preferencia, ingresa cada Medicamento al Sistema, el Sistema le muestra el contenido de la Receta, el Paciente confirma su Receta, el Sistema genera los pedidos para surtir la receta y la ruta, el Sistema le muestra al paciente un comprobante del registro de su receta.

Consultar el estado de una receta: El Paciente desea consultar el estado de su Receta, el Sistema le muestra la Receta, sus líneas de receta, y el estado actual de su Receta.

Recoger el surtido de una receta: El Paciente es notificado de que su receta ya está surtida completamente, el paciente llega a la sucursal y muestra su comprobante de su receta, el Farmacéutico inicia el proceso de entrega de receta, el Sistema muestra el total a pagar por la receta, el Paciente realiza su pago y el Farmacéutico entrega el surtido de la receta.

Preparar receta: El Sistema notifica al Farmacéutico que un paciente ha enviado una Receta, el Sistema a su vez proporciona la ruta de las sucursales que debe visitar para recoger los Medicamentos, el Farmacéutico organiza la recolección de los Medicamentos y cuando la Receta esta surtida el Farmacéutico cambia el estado de la receta como Listo para recoger, para que el Paciente pueda recogerla.

Gestionar Receta no recogida: El Farmacéutico desea consultar una receta no recogida, el Sistema muestra la lista de las recetas que no fueron recogidas, el Farmacéutico selecciona una Receta, el Farmacéutico confirma que la Receta no ha sido recogida, el Sistema actualiza el estado de la Receta y se aplica una Multa al Paciente, el Sistema devuelve los Medicamentos al inventario de las Sucursales que participaron en el surtido de la Receta y el Sistema muestra la Ruta de las Sucursales participantes.

Gestionar el Inventario: El Farmacéutico desea consultar el Inventario de la Sucursal donde se encuentra, puede agregar nuevos Medicamentos, actualizar las cantidades o eliminar Medicamentos que ya no se encuentran en la Sucursal, el Sistema actualiza el Inventario de la Sucursal.

Gestionar la disponibilidad de la Sucursal: El Farmacéutico desea cambiar la disponibilidad de la Farmacia (abierto o cerrado) para no recibir las Recetas de los Pacientes, el Sistema actualiza la disponibilidad de la Sucursal.

Gestionar las Sucursales: El Administrador de Cadena desea registrar nuevas Sucursales, modificar alguna información de las Sucursales o eliminar las que ya no se encuentren disponibles. El Sistema actualiza la Sucursal.

Gestionar los Farmacéuticos de la Sucursal: El Administrador de Cadena desea registrar los Farmacéuticos que operarán en la Sucursal seleccionada o eliminarlos. El Sistema actualiza el personal asignado de la Sucursal.

Gestionar los Administradores de Cadenas: El Administrador General desea registrar los nuevos administradores de cadenas o eliminarlos. El Sistema actualiza los Administradores de Cadenas.

- *Descripción Completa de Casos de Uso*

Caso de Uso CU-01: Subir Receta

Actor principal: Paciente

Personal involucrado e intereses:

- Paciente: Quiere que las entradas sean sencillas y rápidas, recibir el surtido de su receta en la farmacia seleccionada.
- Farmacéutico: Quiere recibir la receta y los pedidos necesarios para surtir la receta completamente.
- Sucursales: Desean mantener actualizado su inventario y participar en el surtido de la receta.

Precondiciones: El Paciente debe estar identificado y autenticado en el sistema.

Garantías de éxito (Postcondiciones): Se actualiza el inventario de las sucursales participantes. Se genera los pedidos y la ruta de las sucursales participantes. Se registra la receta para su surtido. Se notifica al paciente del registro de su receta y las sucursales participantes para el surtido.

Escenario principal de éxito (o flujo básico):

1. El Paciente inicia el proceso de una receta.
2. El Paciente ingresa la cédula del doctor y el sistema muestra las sucursales disponibles.
3. El Paciente selecciona la sucursal donde desea recibir el surtido de su receta y el Sistema muestra la sucursal seleccionada.
4. El Paciente ingresa el nombre del medicamento deseado y el sistema muestra los medicamentos que coincidan con dicho nombre.
5. El Paciente ingresa la cantidad y selecciona el medicamento deseado.
6. El Sistema registra la línea de receta y muestra la lista de los medicamentos solicitados.
   El Paciente repite los pasos 4-6 hasta que ingrese todos los medicamentos de su receta.
7. El Paciente confirma la receta.
8. El Sistema valida el stock de los medicamentos y actualiza los inventarios en la sucursal seleccionada o en la más cercana.
9. El Sistema genera los pedidos de los medicamentos solicitados y la ruta de las sucursales participantes.
10. El Sistema calcula la suma total de la receta, genera una fecha estimada para la recolección, le asigna un folio único y se registra la receta completa.
11. El Sistema envía una notificación al paciente sobre el registro y a las sucursales participantes indicando la receta, los pedidos y se muestra un comprobante del registro de la receta.

Extensiones (o flujos alternativos):

*a. En cualquier momento el Sistema falla

1. El Paciente reinicia el proceso, e intenta hacer el registro de receta de nuevo.

2a. Se ingresa un formato de cedula incorrecto

1. El Sistema muestra un error mencionando que el formato de cedula es incorrecto.

2a. No hay ninguna sucursal disponible en el Sistema

1. El Sistema muestra un error mencionando que no hay sucursales disponibles en el sistema.

3a. El Paciente selecciona una sucursal que ya no se encuentra disponible

1. El Sistema muestra un error mencionando la sucursal seleccionada no está disponible.

4a. El Paciente ingresa un nombre de un medicamento que no existe en el Sistema

1. El Sistema muestra un error mencionando que no se encontraron medicamentos con el nombre proporcionado.

5a. El Paciente selecciona un medicamento que ya ha sido agregado en la receta

En el paso 6 no se registra la línea de receta.

1. El Sistema actualiza la cantidad del medicamento solicitado de la receta.

6a. El Paciente desea eliminar algún medicamento de su receta.

1. El Paciente selecciona el medicamento que desea eliminar.
2. El Sistema elimina el medicamento seleccionado y muestra la lista actualizada de los medicamentos solicitados.

8a. El Sistema al validar que uno o más medicamentos solicitados no tienen stock disponible en los inventarios de las sucursales.

1. El Sistema muestra un error mencionando que no hay stock disponible para el medicamento.
2. El Paciente reinicia el proceso.

11a. El Sistema no puede enviar las notificaciones.

1. El Sistema muestra un error mencionando que no se pudo enviar las notificaciones.

11b. El Paciente desea ver el comprobante en formato PDF

1. El Sistema muestra el comprobante en PDF para su descarga.

Requisitos especiales:

- Interfaz de Usuario fácil de usar, el paciente debe poder navegar y completar cualquier proceso sin la necesidad de leer una guía.

- Tiempo de respuesta de un máximo 10 segundos al procesar la receta y mostrar su comprobante.

Lista de tecnologías y variaciones de datos:

No se han definido tecnologías ni variaciones de datos por el momento.

Frecuencia: Podría ser casi continuo, dependiendo de la cantidad de recetas ingresadas por los pacientes.

Caso de Uso CU-02: Gestionar receta no recogida

Actor principal: Farmacéutico

Personal involucrado e intereses:

- Paciente: Desea ser notificado de la penalización en caso de no recoger el surtido de su receta.
- Farmacéutico: Desea evitar la acumulación de recetas no recogidas.

Precondiciones: El Farmacéutico debe ser identificado y autenticado en el sistema.

Garantías de éxito (Postcondiciones): Los medicamentos registrados en la receta se devuelven a los inventarios de las sucursales involucradas. La receta se actualiza con el estado no entregado. Se genera una multa al paciente por incumplimiento. El Paciente es notificado de la multa.

Escenario principal de éxito (o flujo básico):

1. El Farmacéutico inicia la gestión de recetas no recogidas.
2. El Sistema muestra las recetas no recogidas de la sucursal del farmacéutico.
3. El Farmacéutico selecciona una receta para su revisión y el sistema muestra la receta.
4. El Farmacéutico confirma que la receta no fue recogida.
5. El Sistema actualiza el inventario de cada sucursal participante del surtido de la receta y actualiza el estado de la receta a “No entregado”.
6. El Sistema calcula la multa con un monto establecido y la asocia a la tarjeta del paciente.
7. El Sistema envía una notificación al paciente del estado final de su receta y la multa aplicada.
8. El Sistema muestra los pedidos y la ruta de las sucursales a visitar para la devolución de cada medicamento de la receta.

Extensiones (o flujos alternativos):

*a. En cualquier momento el Sistema falla

1. El Farmacéutico reinicia el proceso, e intenta hacer el proceso de receta no recogida nuevamente.

2a. No hay recetas con el estado no recogido.

1. El Sistema muestra un error mencionando que no hay ninguna receta no recogida en la sucursal.

4a. El Farmacéutico no actualizó el estado de la receta y el paciente llegó después.

1. El Farmacéutico marca la receta como recogida.
2. El Sistema actualiza el estado de la receta como “Recogido”

6a. El Sistema no puede procesar la multa al paciente.

1. El Sistema muestra un error mencionando que no se pudo procesar la multa a la tarjeta del paciente.
2. El Farmacéutico repite el paso 4 o intenta el mismo proceso más tarde.

7a. El Sistema no puede enviar la notificación al paciente.

1. El Sistema muestra un error mencionando que no se pudo enviar la notificación sobre el estado de la receta al paciente.
2. El Farmacéutico llama vía telefónica al paciente sobre el estado de la receta.

Requisitos especiales:

- Interfaz de Usuario fácil de usar, el farmacéutico debe poder navegar y completar cualquier proceso sin la necesidad de leer una guía.
- Tiempo de respuesta de un máximo 10 segundos al procesar el estado de la receta, aplicar la multa y mostrar la ruta.

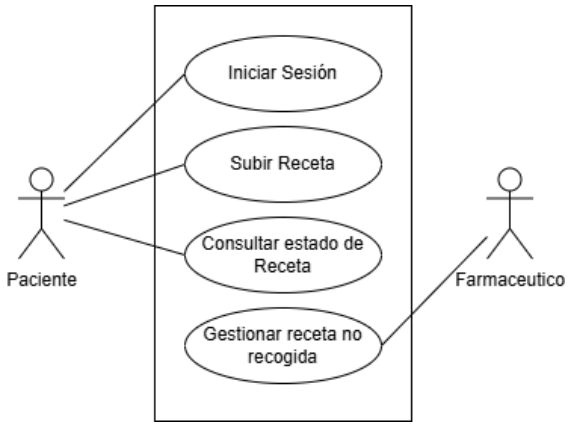
Lista de tecnologías y variaciones de datos:

No se han definido tecnologías ni variaciones de datos por el momento.

Frecuencia:

Podría ser casi continuo, dependiendo de la cantidad de recetas no recogidas por los pacientes.

- *Diagrama de Caso de Uso*



Glosario

- *Historia de Revisiones*

Versión	Fecha	Descripción	Autores
Borrador de Inicio	15/09/25	Primer Borrador: Para refinarse durante la elaboración	- Equipo 3

- *Definiciones*

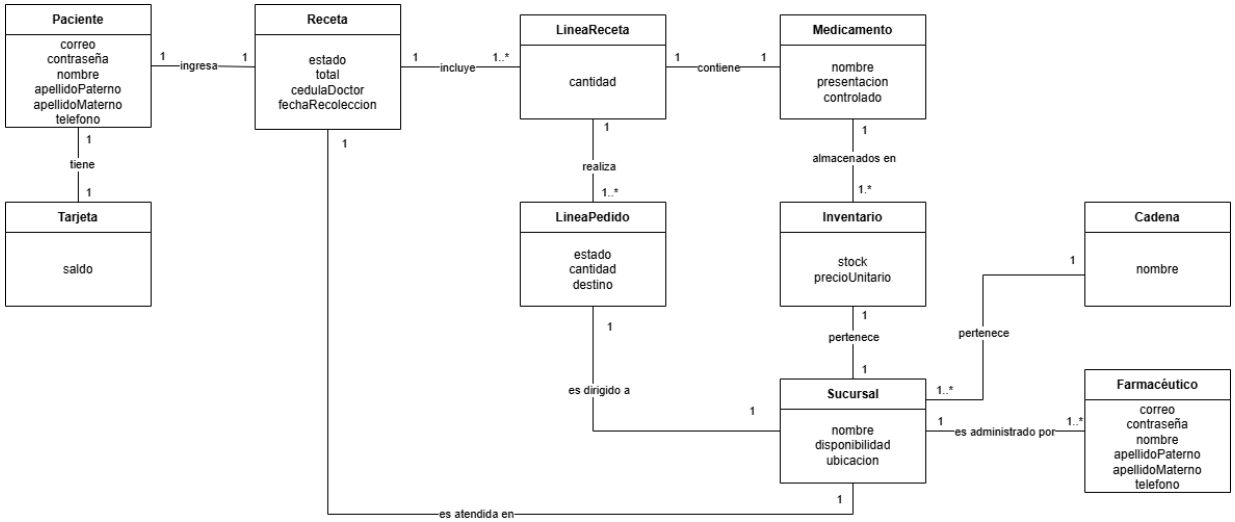
Término	Definición	Alias
Paciente	Usuario final del sistema que requiere surtir sus recetas médicas.	Usuario, Cliente
Cadena farmacéutica	Empresa que posee múltiples sucursales y participa en el sistema TAS.	Farmacia corporativa
Sucursal	Punto de venta individual de una cadena farmacéutica.	Tienda, Local, Farmacia
Farmacéutico	Personal encargado de la operación de cada sucursal.	Encargado, Personal de farmacia
Receta médica	Documento prescrito por un profesional de la salud que indica los medicamentos que debe recibir el paciente.	Prescripción, Orden médica
Pedido	Solicitud generada por el sistema para surtir los medicamentos de una receta con una sucursal cercana.	Orden de surtido
Inventario	Conjunto de productos y existencias disponibles en cada sucursal.	Stock, Existencias, Disponibilidad de medicamento
Algoritmo de	Mecanismo que asigna los medicamentos	Coordinación

surtido	faltantes de un pedido a una o varias sucursales colaboradoras.	de surtido
Notificación	Mensaje enviado al usuario para informar sobre el estado de su pedido o cambios en el sistema.	Aviso, Alerta
Sistema de gestión de pedidos	Sistema encargado de manejar los pedidos que necesitará realizar la sucursal donde recibe la receta del cliente.	OMS, Order Management System

Fase de Elaboración

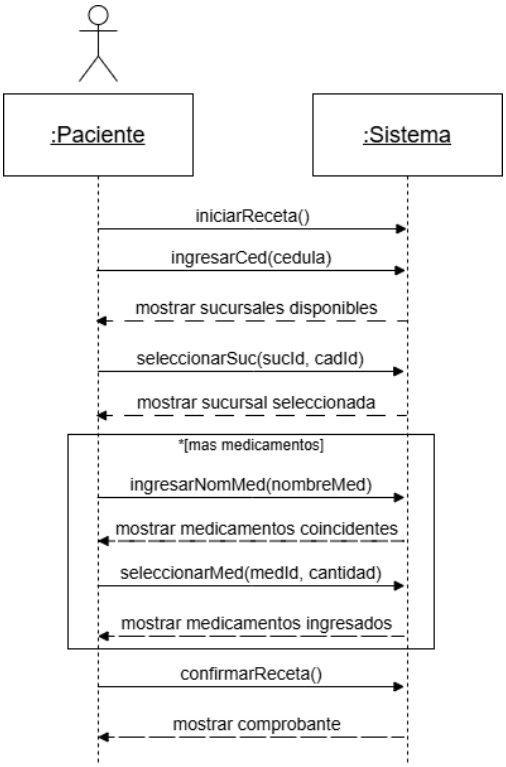
Análisis

- *Diagrama de Clases Conceptuales*

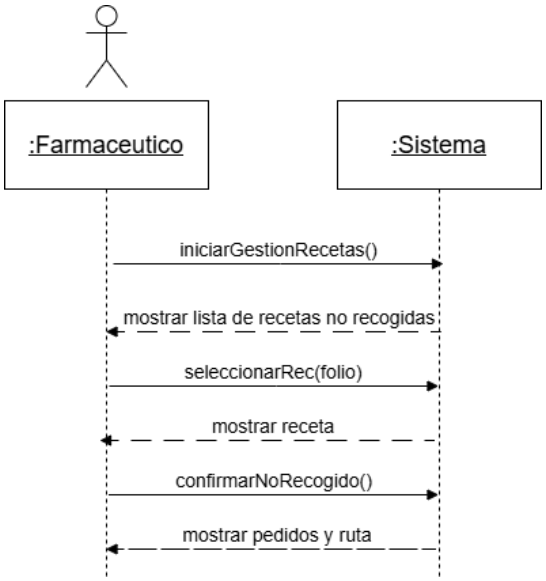


- *Diagramas de Secuencia del Sistema*

Caso de Uso CU-01: Subir receta



Caso de Uso CU-02: Gestionar receta no recogida



- *Contratos*

Caso de Uso CU-01: Subir receta

Contrato C01: iniciarReceta

Operación: iniciarReceta()

Referencias Cruzadas: Caso de uso CU-01: Ingreso de una receta

Precondiciones: Ninguna

Postcondiciones:

Se creó una instancia de Receta receta

receta.estado pasó a ser “En curso”

Receta receta se asoció con paciente

Contrato C02: ingresarCedula

Operación: ingresarCed(cedula)

Referencias Cruzadas: Caso de uso CU-01: Ingreso de una receta

Precondiciones: Hay una receta en curso

Postcondiciones:

receta.cedula pasó a ser cedula

Contrato C03: seleccionarSucursal

Operación: seleccionarSuc(sucId, cadId)

Referencias Cruzadas: Caso de uso CU-01: Ingreso de una receta

Precondiciones: Hay una receta en curso

Postcondiciones:

Se creó una instancia Cadena cad

Se creó una instancia Sucursal suc

Sucursal suc se asoció con cad

Receta receta se asoció con suc

Contrato C04: seleccionarMedicamento

Operación: seleccionarMed(medId, cantidad)

Referencias Cruzadas: Caso de uso CU-01: Ingreso de una receta

Precondiciones: Hay una receta en curso

Postcondiciones:

Se creó una instancia Medicamento med

Se creó una instancia LineaReceta linRec

linRec se asoció con med

linRec se asoció con receta

Contrato C05: confirmarReceta

Operación: confirmarReceta()

Referencias Cruzadas: Caso de uso CU-01: Ingreso de una receta

Precondiciones: Hay una receta en curso

Postcondiciones:

inv.stock pasó a ser inv.stock - linPed.cantidad a partir de la cantidad que puede surtir el inventario

Se creó una instancia LineaPedido linPed a partir de cada medicamento solicitado

linPed se asoció con lineRec

Caso de Uso CU-02: Gestionar receta no recogida

Contrato C06: seleccionarReceta

Operación: seleccionarRec(folio)

Referencias Cruzadas: Caso de uso CU-02: Gestionar receta no recogida

Precondiciones: Ninguna

Postcondiciones:

Se creó una instancia de Receta receta con base a la coincidencia del folio

Contrato C07: confirmarNoRecogido

Operación: confirmarNoRecogido()

Referencias Cruzadas: Caso de uso CU-02: Gestionar receta no recogida

Precondiciones: Hay una receta no recogida

Postcondiciones:

inv.stock pasó a ser (inv.stock + linPed.cantidad) de acuerdo cada uno de las líneas pedidos

receta.estado pasó a ser “No entregado”

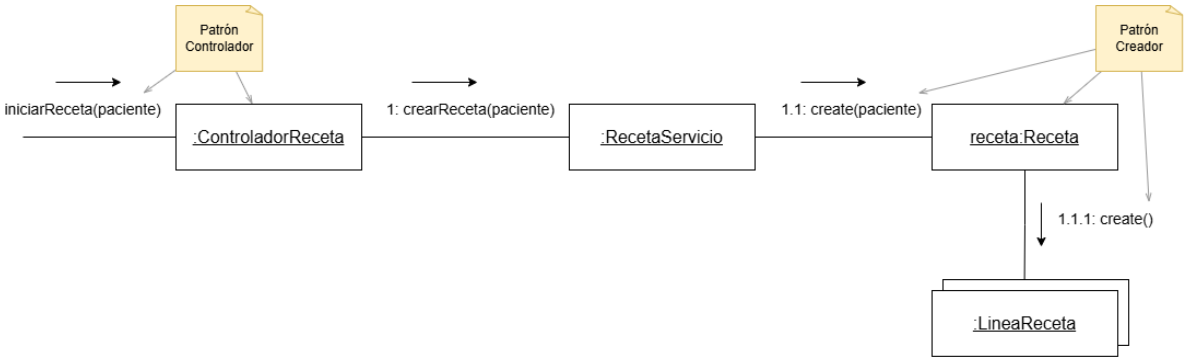
tarjeta.saldo pasó a ser tarjeta.saldo - (rec.total * 0.10)

Diseño

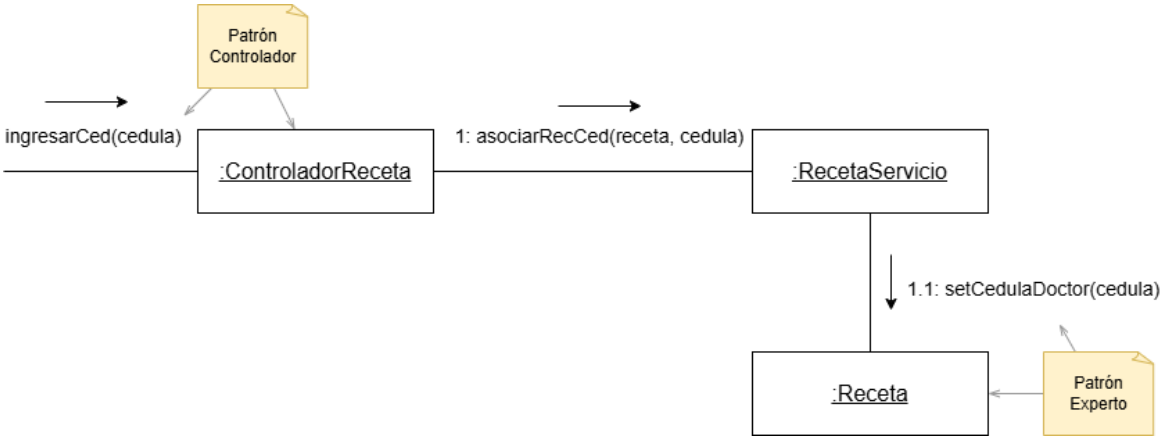
- *Diagramas de Interacción de Objetos*

Caso de Uso CU-01: Subir receta

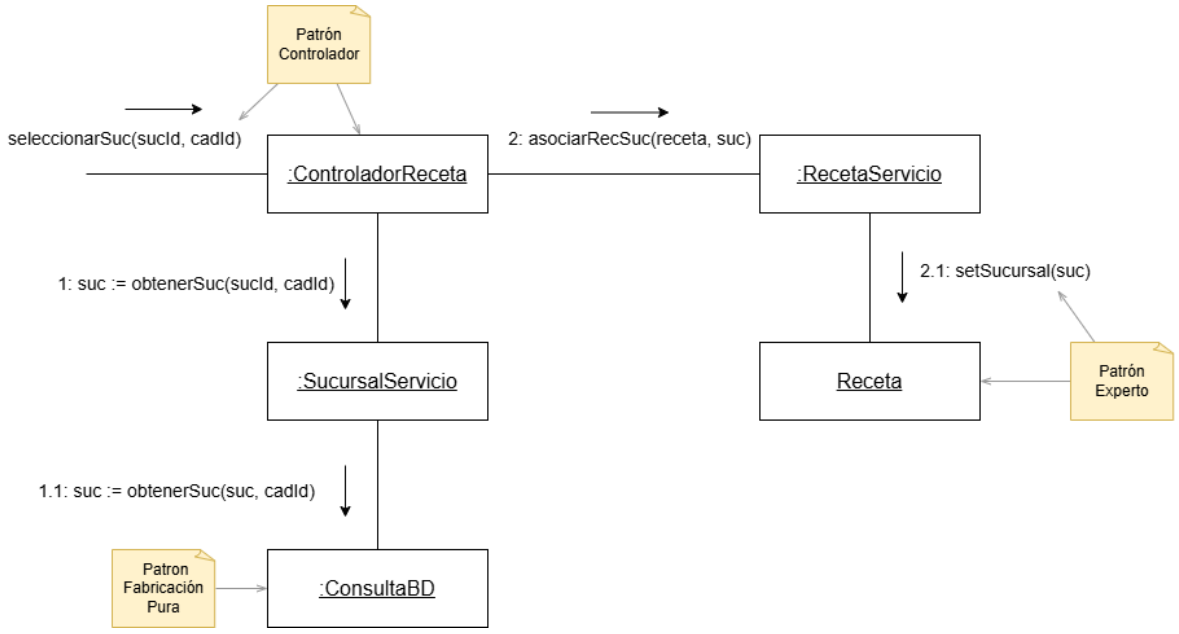
Contrato C01



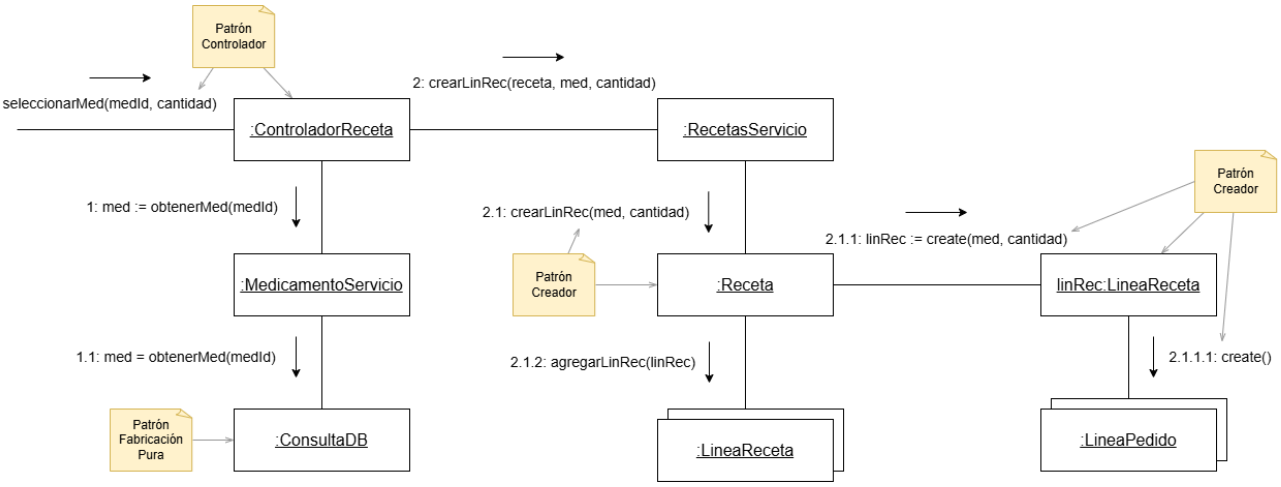
Contrato C02



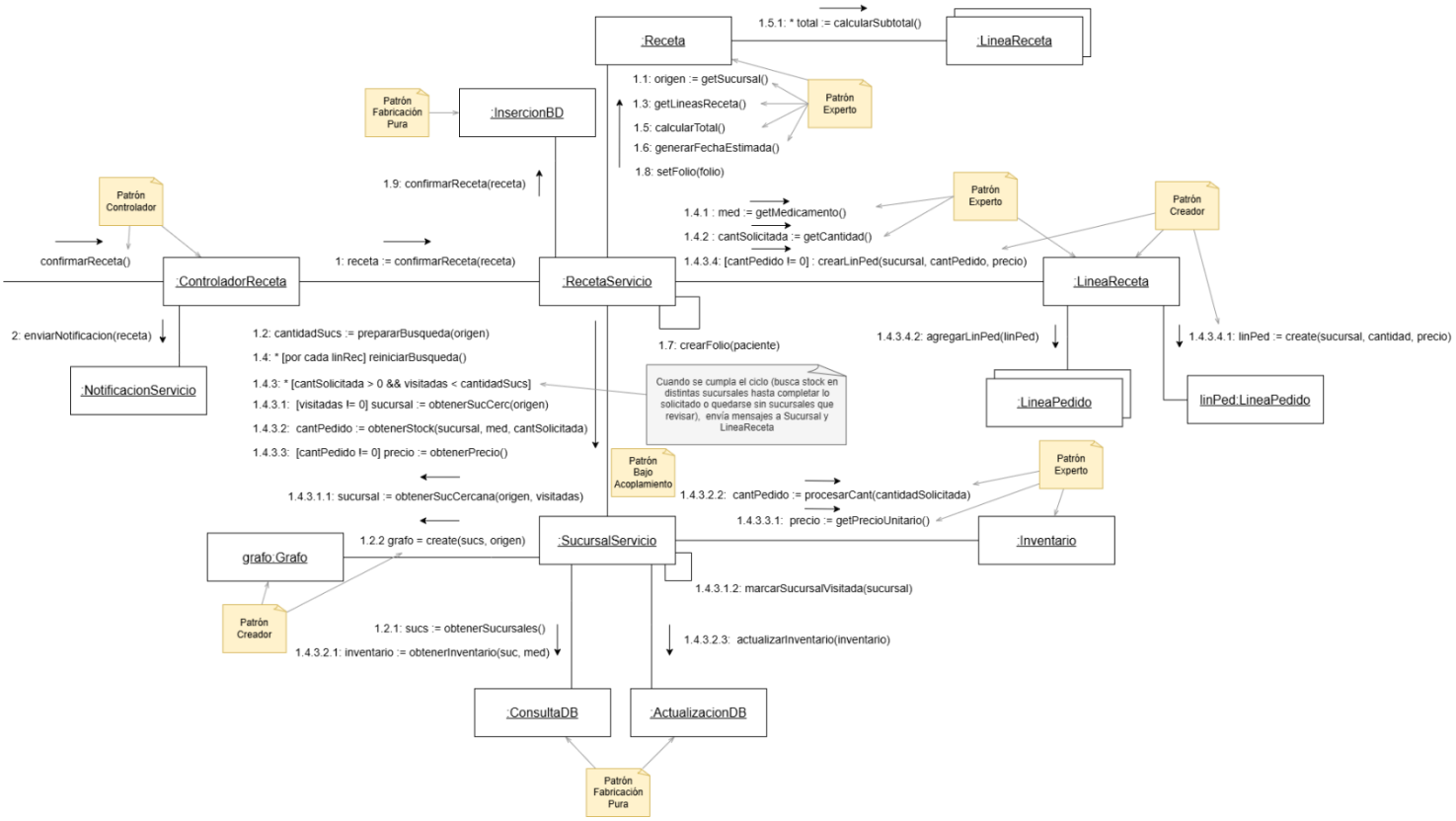
Contrato C03



Contrato C04

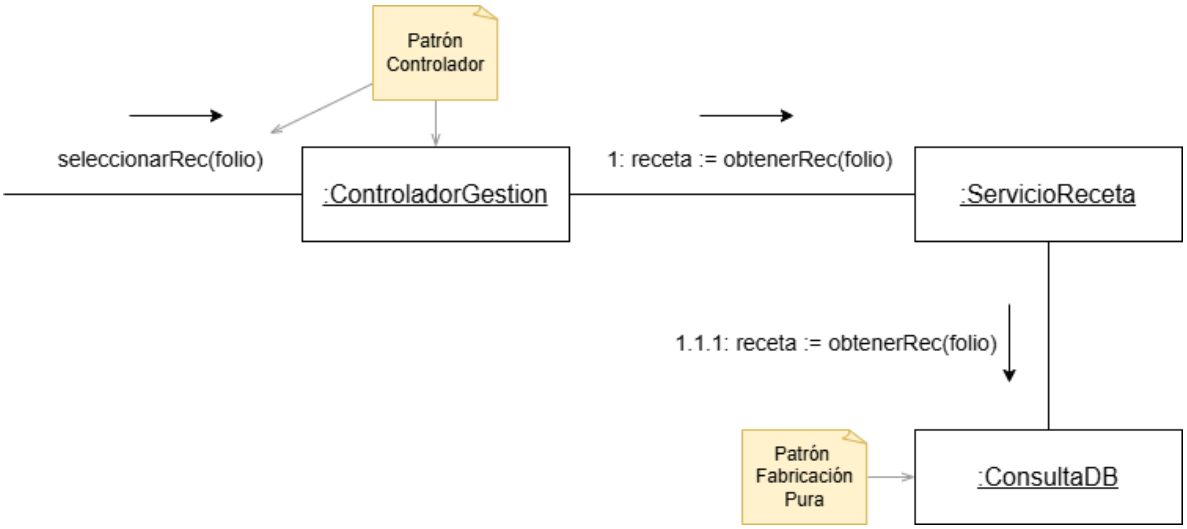


Contrato C05

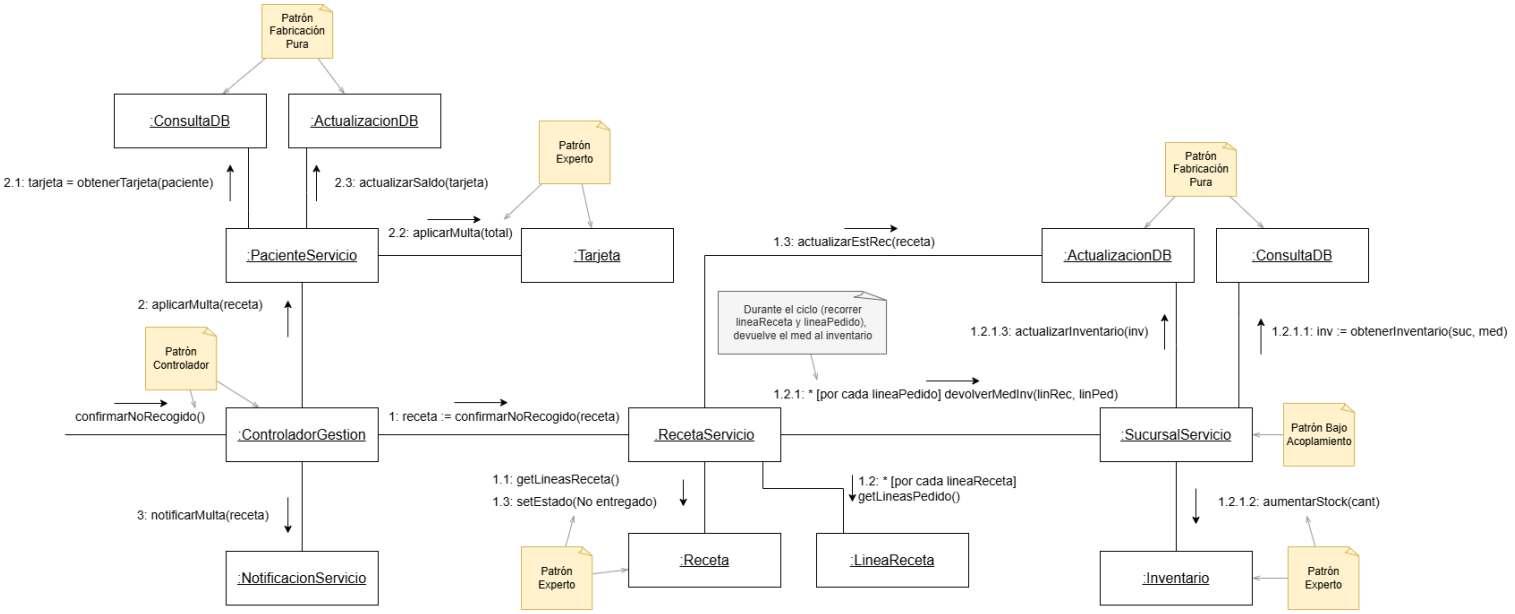


Caso de Uso CU-02: Gestionar receta no recogida

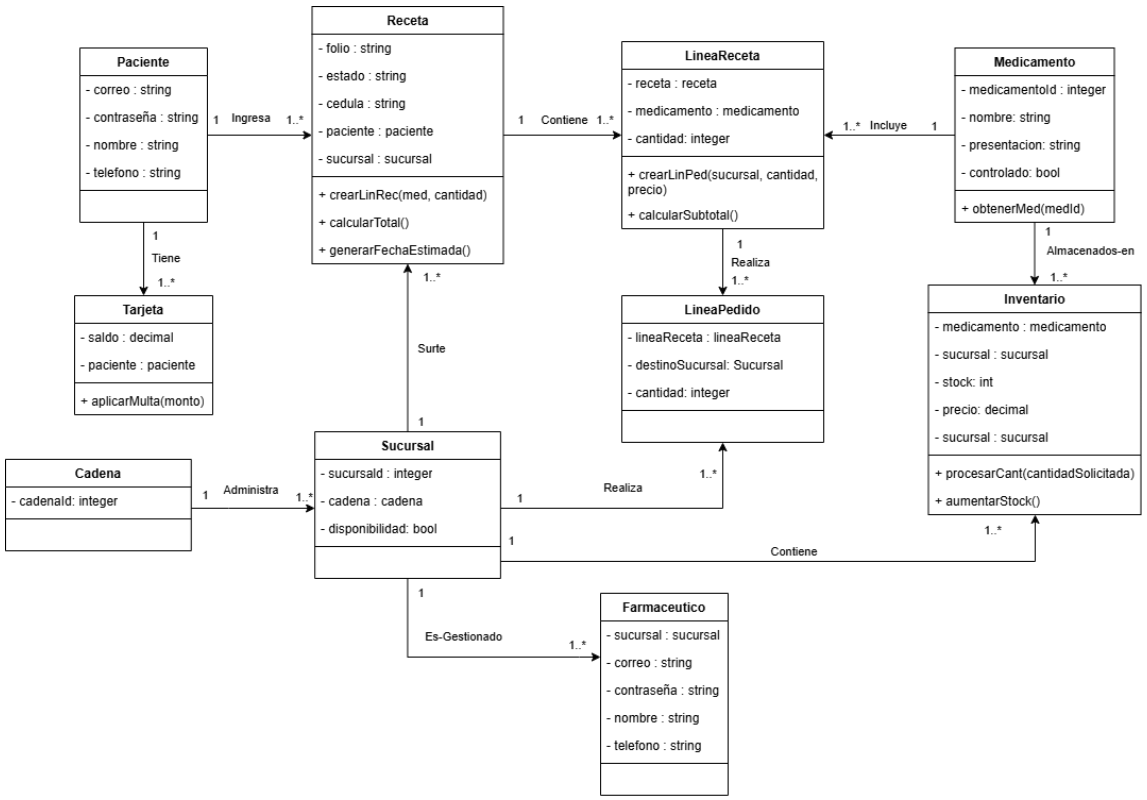
Contrato C06



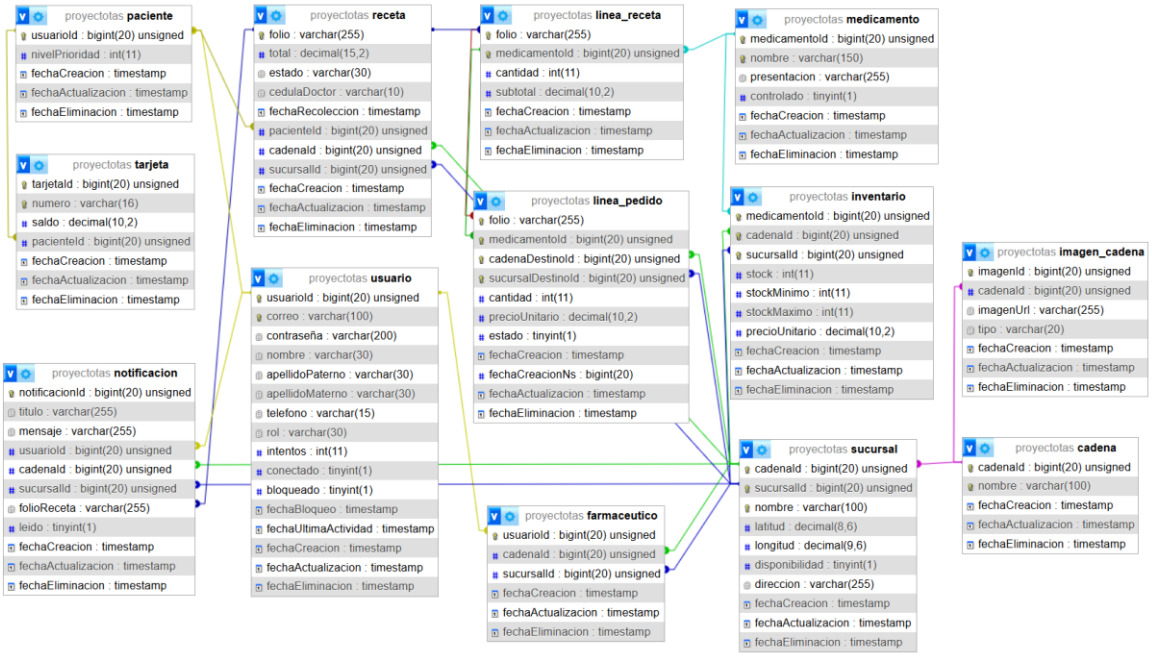
Contrato C07



- *Diagrama de Clases Software*

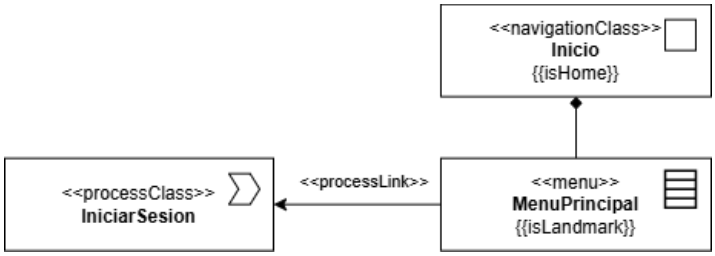


- *Modelo de la Base de Datos*

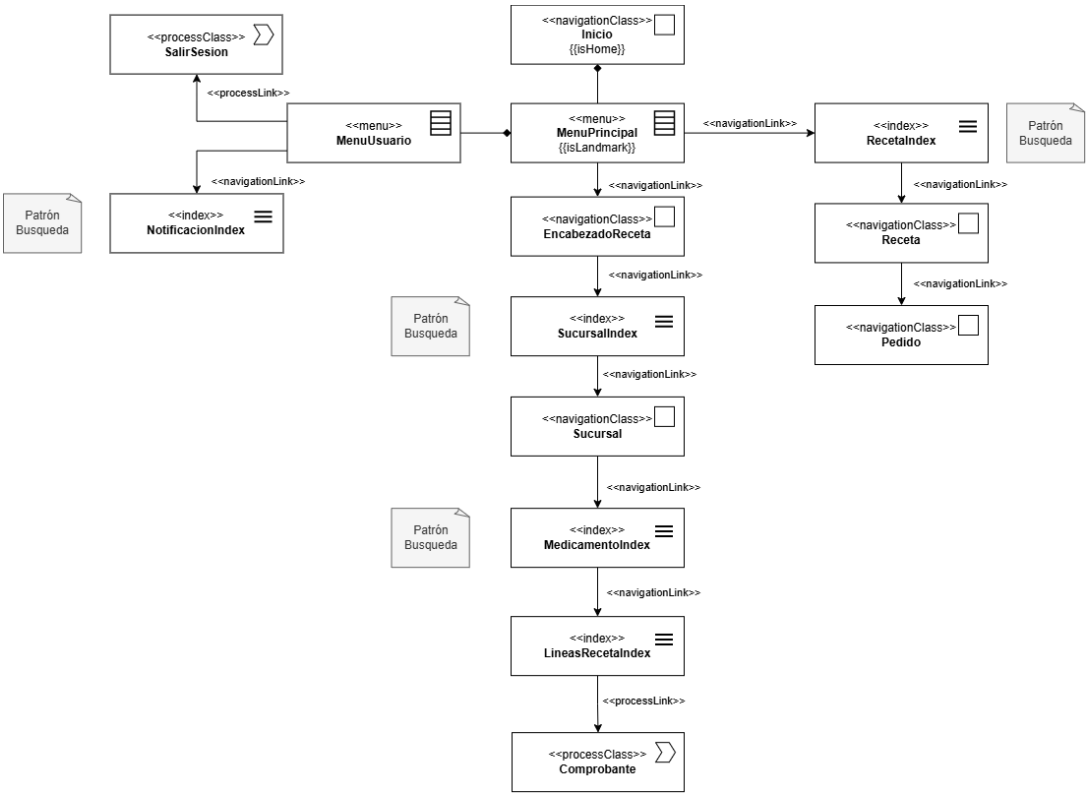


- *Modelo de Navegación*

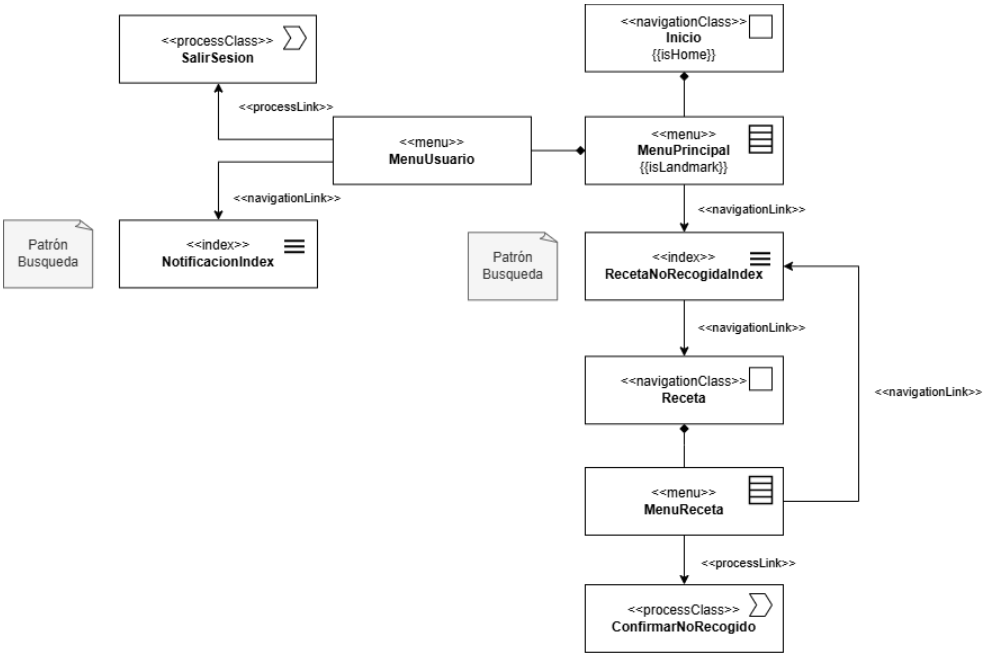
Usuario No autenticado



Usuario Paciente

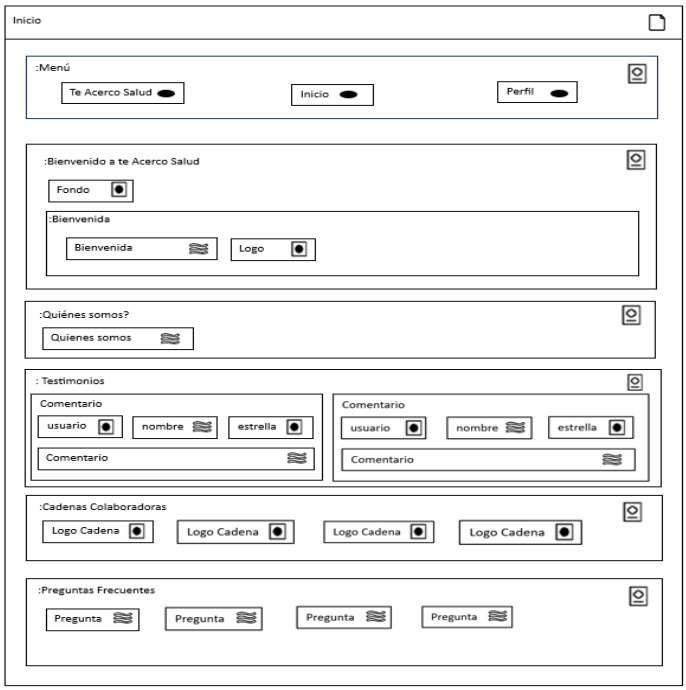


Usuario Farmacéutico

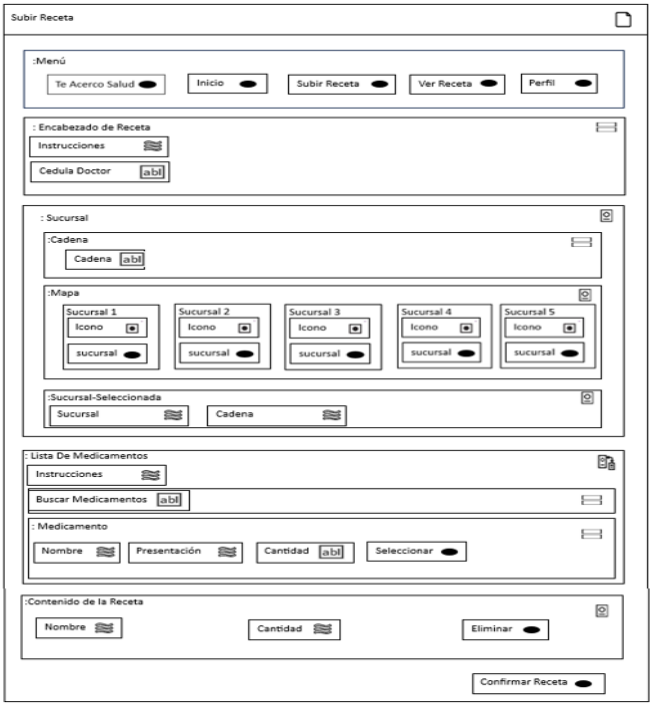


- *Modelo de Presentación*

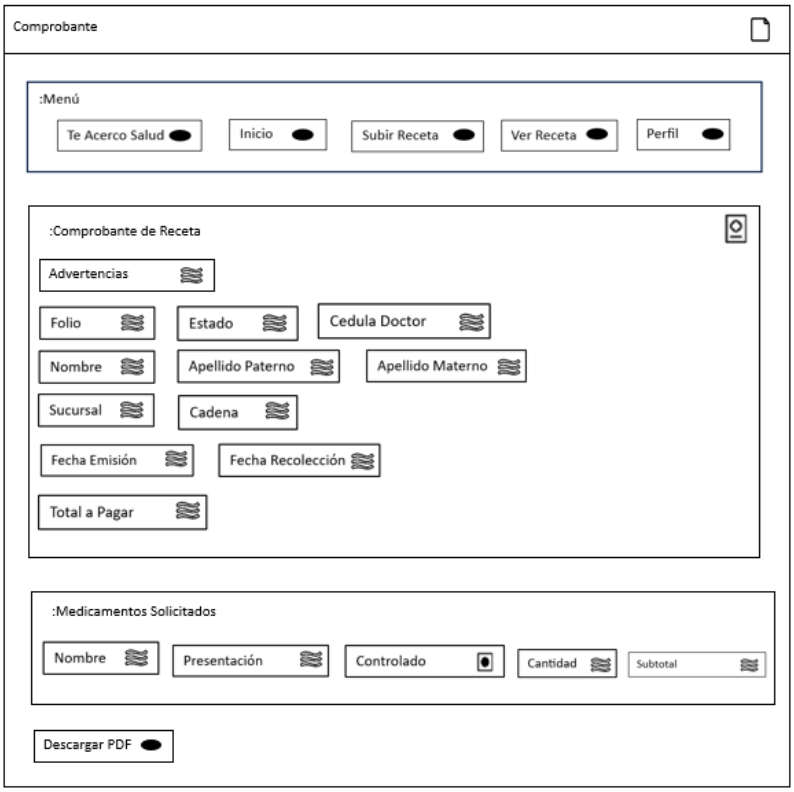
Presentación Inicio



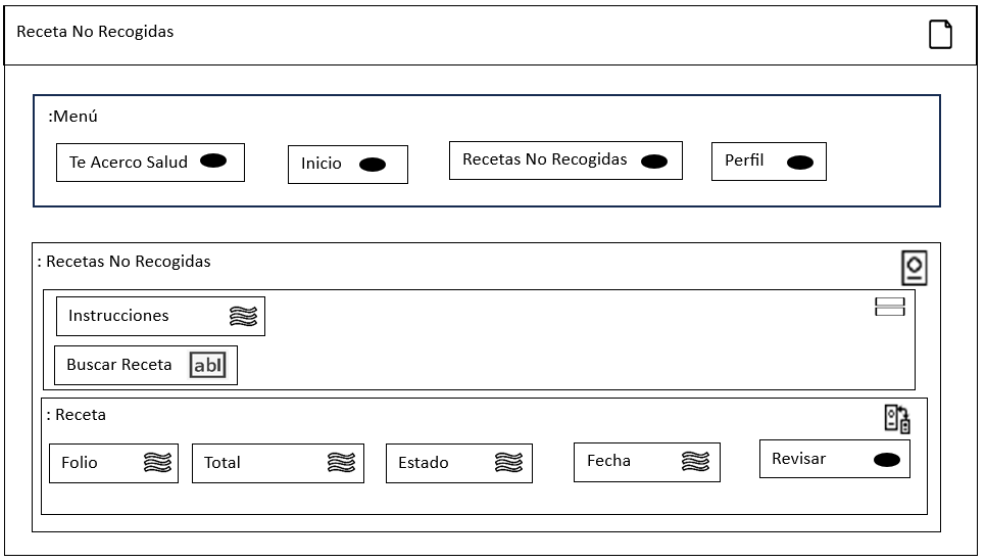
Presentación Subir Receta



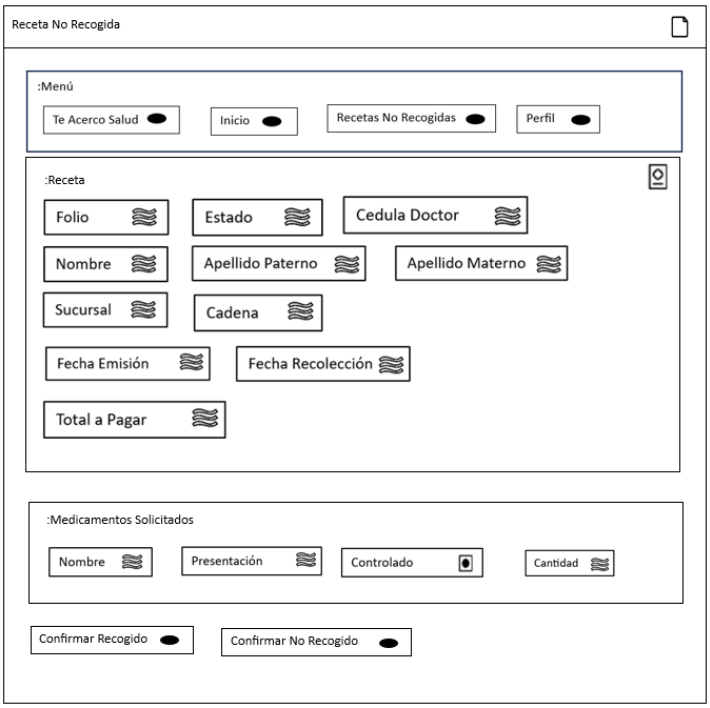
Presentación Comprobante



Presentación Recetas no Recogidas



Presentación Receta no recogida



Presentación Pedidos y ruta

Implementación

- *Código Fuente de la Estructura de Clases*

```
class Cadena {
    private $cadenaId;
    private $nombre;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($cadenaId = null, $nombre = null, $logoUrl =
null, $fechaCreacion = null, $fechaActualizacion = null, $fechaEliminacion =
null) {
        $this->cadenaId = $cadenaId;
        $this->nombre = $nombre;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
    }

    public function getCadenaId() { return $this->cadenaId; }
    public function getNombre() { return $this->nombre; }
    public function getLogoUrl() { return $this->logoUrl; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}

    public function setCadenaId($cadenaId) { $this->cadenaId = $cadenaId; }
    public function setNombre($nombre) { $this->nombre = $nombre; }
    public function setLogoUrl($logoUrl) { $this->logoUrl = $logoUrl; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }
}
```

```
class Farmaceutico extends Usuario {
    private $sucursal;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($sucursal = null, $usuarioId = null, $correo
= null, $contraseña = null, $nombre = null, $apellidoPaterno = null,
$apellidoMaterno = null, $telefono = null, $rol = null, $intentos = null,
$conectado = null, $bloqueado = null, $fechaBloqueo = null,
$fechaUltimaActividad = null, $fechaCreacion = null, $fechaActualizacion =
null, $fechaEliminacion = null) {
        parent::__construct($usuarioId, $correo, $contraseña, $nombre,
$apellidoPaterno, $apellidoMaterno, $telefono, $rol, $intentos, $conectado,
$bloqueado, $fechaBloqueo, $fechaUltimaActividad, $fechaCreacion,
$fechaActualizacion, $fechaEliminacion);
        $this->sucursal = $sucursal;
    }

    public function getSucursal() { return $this->sucursal; }
    public function setSucursal($sucursal) { $this->sucursal = $sucursal; }
}

class Inventario {
    private $medicamento;
    private $sucursal;
    private $stock;
    private $precioUnitario;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($medicamento = null, $sucursal = null,
$stock = null, $precioUnitario = null, $fechaCreacion = null,
$fechaActualizacion = null, $fechaEliminacion = null) {
        $this->medicamento = $medicamento;
        $this->sucursal = $sucursal;
        $this->stock = $stock;
        $this->precioUnitario = $precioUnitario;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
```

```
    }

    public function getMedicamento() { return $this->medicamento; }
    public function getSucursal() { return $this->sucursal; }
    public function getStock() { return $this->stock; }
    public function getPrecioUnitario() { return $this->precioUnitario; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}

    public function setMedicamento($medicamento) { $this->medicamento =
$medicamento; }
    public function setSucursal($sucursal) { $this->sucursal = $sucursal; }
    public function setStock($stock) { $this->stock = $stock; }
    public function setPrecioUnitario($precioUnitario) { $this-
>precioUnitario = $precioUnitario; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }

    public function procesarCant($cantidadSolicitada) {
        $cantPed = min($this->stock, $cantidadSolicitada);
        $this->stock -= $cantPed;
        return  $cantPed;
    }

    public function aumentarStock($cantidad) {
        $this->stock += $cantidad;
    }
}


class LineaPedido {
    private $lineaReceta;
    private $destinoSucursal;
    private $cantidad;
    private $estado;
```

```
    private $precioUnitario;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($lineaReceta = null, $destinoSucursal =
null, $cantidad = null, $precioUnitario = null, $estado = null,
$fechaCreacion = null, $fechaActualizacion = null, $fechaEliminacion = null)
{
        $this->lineaReceta = $lineaReceta;
        $this->destinoSucursal = $destinoSucursal;
        $this->cantidad = $cantidad;
        $this->precioUnitario = $precioUnitario;
        $this->estado = $estado;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
    }

    public function getLineaReceta() { return $this->lineaReceta; }
    public function getDestinoSucursal() { return $this->destinoSucursal; }
    public function getCantidad() { return $this->cantidad; }
    public function getPrecioUnitario() { return $this->precioUnitario; }
    public function getEstado() { return $this->estado; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}

    public function setLineaReceta($lineaReceta) { $this->lineaReceta =
$lineaReceta; }
    public function setDestinoSucursal($destinoSucursal) { $this-
>destinoSucursal = $destinoSucursal; }
    public function setCantidad($cantidad) { $this->cantidad = $cantidad; }
    public function setPrecioUnitario($precioUnitario) { $this-
>precioUnitario = $precioUnitario; }
    public function setEstado($estado) { $this->estado = $estado; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
```

```
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }
}


class LineaReceta {
    private $receta;
    private $medicamento;
    private $cantidad;
    private $subtotal;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;
    private $lineaPedidos;

    public function __construct($receta = null, $medicamento = null,
$cantidad = null, $subtotal = null, $fechaCreacion = null,
$fechaActualizacion = null, $fechaEliminacion = null) {
        $this->receta = $receta;
        $this->medicamento = $medicamento;
        $this->cantidad = $cantidad;
        $this->subtotal = $subtotal;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
        $this->lineaPedidos = [];
    }

    public function getReceta() { return $this->receta; }
    public function getMedicamento() { return $this->medicamento; }
    public function getCantidad() { return $this->cantidad; }
    public function getSubtotal() { return $this->subtotal; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}
    public function getLineasPedido() { return $this->lineaPedidos; }

    public function setReceta($receta) { $this->receta = $receta; }
    public function setMedicamento($medicamento) { $this->medicamento =
$medicamento; }
```

```
    public function setCantidad($cantidad) { $this->cantidad = $cantidad; }
    public function setSubtotal($subtotal) { $this->subtotal = $subtotal; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }
    public function setLineasPedido($lineasPedido) { $this->lineaPedidos =
$lineasPedido; }

    public function crearLinPed($sucursal, $cantidad, $precio) {
        $lineaPed = new LineaPedido( null, $sucursal, $cantidad, $precio);
        $lineaPed->setLineaReceta($this);
        $this->agregarLinPed($lineaPed);
    }

    public function agregarLinPed($linPed) {
        $this->lineaPedidos[] = $linPed;
    }

    public function reiniciarLineaPedido(){
        $this->lineaPedidos = [];
    }

    public function calcularSubtotal() {
        $subtotal = 0;
        foreach ($this->getLineasPedido() as $pedido) {
            $subtotal += $pedido->getCantidad() * $pedido-
>getPrecioUnitario();
        }
        $this->subtotal = $subtotal;
        return $subtotal;
    }
}

class Medicamento {
    private $medicamentoId;
    private $nombre;
    private $presentacion;
    private $controlado;
    private $fechaCreacion;
```

```
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($medicamentoId = null, $nombre = null,
$presentacion = null, $controlado = null, $fechaCreacion = null,
$fechaActualizacion = null, $fechaEliminacion = null) {
        $this->medicamentoId = $medicamentoId;
        $this->nombre = $nombre;
        $this->presentacion = $presentacion;
        $this->controlado = $controlado;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
    }

    public function getMedicamentoId() { return $this->medicamentoId; }
    public function getNombre() { return $this->nombre; }
    public function getPresentacion() { return $this->presentacion; }
    public function getControlado() { return $this->controlado; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}

    public function setMedicamentoId($medicamentoId) { $this->medicamentoId
= $medicamentoId; }
    public function setNombre($nombre) { $this->nombre = $nombre; }
    public function setPresentacion($presentacion) { $this->presentacion =
$presentacion; }
    public function setControlado($controlado) { $this->controlado =
$controlado; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }
    }
}
```

```php
class Notificacion
{
    private $notificacionId;
    private $titulo;
    private $mensaje;
    private $leido;
    private $usuario;
    private $sucursal;
    private $receta;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($notificacionId = null, $titulo = null,
$mensaje = null, $leido = null, $usuario = null, $sucursal = null, $receta =
null, $fechaCreacion = null, $fechaActualizacion = null, $fechaEliminacion =
null)
    {
        $this->notificacionId = $notificacionId;
        $this->titulo = $titulo;
        $this->mensaje = $mensaje;
        $this->leido = $leido;
        $this->usuario = $usuario;
        $this->sucursal = $sucursal;
        $this->receta = $receta;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
    }

    public function getNotificacionId() { return $this->notificacionId; }
    public function getTitulo() { return $this->titulo; }
    public function getMensaje() { return $this->mensaje; }
    public function getLeido() { return $this->leido; }
    public function getUsuario() { return $this->usuario; }
    public function getSucursal() { return $this->sucursal; }
    public function getReceta() { return $this->receta; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}
```

```
    public function setNotificacionId($notificacionId) { $this->notificacionId = $notificacionId; }
    public function setUsuario($usuario) { $this->usuario = $usuario; }
    public function setSucursal($sucursal) { $this->sucursal = $sucursal; }
    public function setReceta($receta) { $this->receta = $receta; }
    public function setTitulo($titulo) { $this->titulo = $titulo; }
    public function setMensaje($mensaje) { $this->mensaje = $mensaje; }
    public function setLeido($leido) { $this->leido = $leido; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion = $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this->fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this->fechaEliminacion = $fechaEliminacion; }
}

class Paciente extends Usuario {
    private $nivelPrioridad;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($nivelPrioridad = null, $usuarioId = null, $correo = null, $contraseña = null, $nombre = null, $apellidoPaterno = null, $apellidoMaterno = null, $telefono = null, $rol = null, $intentos = null, $conectado = null, $bloqueado = null, $fechaBloqueo = null, $fechaUltimaActividad = null, $fechaCreacion = null, $fechaActualizacion = null, $fechaEliminacion = null) {
        parent::__construct($usuarioId, $correo, $contraseña, $nombre, $apellidoPaterno, $apellidoMaterno, $telefono, $rol, $intentos, $conectado, $bloqueado, $fechaBloqueo, $fechaUltimaActividad, $fechaCreacion, $fechaActualizacion, $fechaEliminacion);
        $this->nivelPrioridad = $nivelPrioridad;
    }

    public function getNivelPrioridad() {
        return $this->nivelPrioridad;
    }

    public function setNivelPrioridad($nivelPrioridad) {
        $this->nivelPrioridad = $nivelPrioridad;
```

```
    }
}


class Receta {
    private $folio;
    private $total;
    private $estado;
    private $cedulaDoctor;
    private $fechaRecoleccion;
    private $paciente;
    private $sucursal;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;
    private $lineasReceta;

    public function __construct($folio = null, $total = null, $estado =
null, $cedulaDoctor = null, $fechaRecoleccion = null, $paciente = null,
$sucursal = null, $fechaCreacion = null, $fechaActualizacion = null,
$fechaEliminacion = null) {
        $this->folio = $folio;
        $this->total = $total;
        $this->estado = $estado ?? 'En curso';
        $this->cedulaDoctor = $cedulaDoctor;
        $this->fechaRecoleccion = $fechaRecoleccion;
        $this->paciente = $paciente;
        $this->sucursal = $sucursal;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
        $this->lineasReceta = [];
    }

    public function getFolio() { return $this->folio; }
    public function getTotal() { return $this->total; }
    public function getEstado() { return $this->estado; }
    public function getCedulaDoctor() { return $this->cedulaDoctor; }
    public function getFechaRecoleccion() { return $this->fechaRecoleccion;
}
    public function getPaciente() { return $this->paciente; }
    public function getSucursal() { return $this->sucursal; }
```

```
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}
    public function getLineasReceta() { return $this->lineasReceta; }

    public function setFolio($folio) { $this->folio = $folio; }
    public function setTotal($total) { $this->total = $total; }
    public function setEstado($estado) { $this->estado = $estado; }
    public function setCedulaDoctor($cedulaDoctor) { $this->cedulaDoctor =
$cedulaDoctor; }
    public function setFechaRecoleccion($fechaRecoleccion) { $this-
>fechaRecoleccion = $fechaRecoleccion; }
    public function setPaciente($paciente) { $this->paciente = $paciente; }
    public function setSucursal($sucursal) { $this->sucursal = $sucursal; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }
    public function setLineasReceta($lineasReceta) { $this->lineasReceta =
$lineasReceta; }
    public function mostrarMedIngresados() { return array_map(fn($linRec) =>
$linRec->toArray(), $this->lineasReceta); }

    public function crearLinRec($med, $cantidad) {
        // Caso Alternativo: Actualizar cantidad
        foreach ($this->lineasReceta as $linea) {
            if ($linea->getMedicamento()->getMedicamentoId() == $med-
>getMedicamentoId()) {
                $linea->setCantidad($linea->getCantidad() + $cantidad);
                return;
            }
        }

        $linRec = new LineaReceta(null, $med, $cantidad);
        $linRec->setReceta($this);
        $this->agregarLinRec($linRec);
    }
```

```
    private function agregarLinRec($linRec) {
        $this->lineasReceta[] = $linRec;
        $this->ordenarLineasReceta();
    }

    private function ordenarLineasReceta() {
        usort($this->lineasReceta, function($a, $b) {
            $nombreA = $a->getMedicamento()->getNombre();
            $nombreB = $b->getMedicamento()->getNombre();
            return strcmp($nombreA, $nombreB);
        });
    }

    public function eliminarLinRec($medId) {
        foreach ($this->lineasReceta as $i => $linea) {
            if ($linea->getMedicamento()->getMedicamentoId() == $medId) {
                unset($this->lineasReceta[$i]);
                $this->lineasReceta = array_values($this->lineasReceta);
                return;
            }
        }
    }

    public function calcularTotal() {
        $total = 0;
        foreach ($this->getLineasReceta() as $lineaReceta) {
            $total += $lineaReceta->calcularSubtotal();
        }
        $this->total = $total;
        return $total;
    }

    public function generarFechaEstimada() {
        $this->fechaRecoleccion = date('Y-m-d', strtotime('+7 days'));
    }
}

class Sucursal {
    private $sucursalId;
    private $cadena;
    private $nombre;
    private $latitud;
```

```
    private $longitud;
    private $disponibilidad;
    private $direccion;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($sucursalId = null, $cadena = null, $nombre
= null, $latitud = null, $longitud = null, $disponibilidad = null,
$direccion = null, $fechaCreacion = null, $fechaActualizacion = null,
$fechaEliminacion = null) {
        $this->sucursalId = $sucursalId;
        $this->cadena = $cadena;
        $this->nombre = $nombre;
        $this->latitud = $latitud;
        $this->longitud = $longitud;
        $this->disponibilidad = $disponibilidad;
        $this->direccion = $direccion;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
    }

    public function getSucursalId() { return $this->sucursalId; }
    public function getCadena() { return $this->cadena; }
    public function getNombre() { return $this->nombre; }
    public function getLatitud() { return $this->latitud; }
    public function getLongitud() { return $this->longitud; }
    public function getDisponibilidad() { return $this->disponibilidad; }
    public function getDireccion() { return $this->direccion; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}

    public function setSucursalId($sucursalId) { $this->sucursalId =
$sucursalId; }
    public function setCadena($cadena) { $this->cadena = $cadena; }
    public function setNombre($nombre) { $this->nombre = $nombre; }
    public function setLatitud($latitud) { $this->latitud = $latitud; }
    public function setLongitud($longitud) { $this->longitud = $longitud; }
```

```
    public function setDisponibilidad($disponibilidad) { $this-
>disponibilidad = $disponibilidad; }
    public function setDireccion($direccion) { $this->direccion =
$direccion; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }
}


class Tarjeta {
    private $tarjetaId;
    private $numero;
    private $saldo;
    private $paciente;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($tarjetaId = null, $numero = null, $saldo =
null, $paciente = null, $fechaCreacion = null, $fechaActualizacion = null,
$fechaEliminacion = null) {
        $this->tarjetaId = $tarjetaId;
        $this->numero = $numero;
        $this->saldo = $saldo;
        $this->paciente = $paciente;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
    }

    public function getTarjetaId() { return $this->tarjetaId; }
    public function getNumero() { return $this->numero; }
    public function getSaldo() { return $this->saldo; }
    public function getPaciente() { return $this->paciente; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
```

```
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}

    public function setTarjetaId($tarjetaId) { $this->tarjetaId =
$tarjetaId; }
    public function setNumero($numero) { $this->numero = $numero; }
    public function setSaldo($saldo) { $this->saldo = $saldo; }
    public function setPaciente($paciente) { $this->paciente = $paciente; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }

    public function aplicarMulta($monto) {
        $this->saldo = $this->saldo - ($monto * 0.10);
    }
}

class Usuario {
    private $usuarioId;
    private $correo;
    private $contraseña;
    private $nombre;
    private $apellidoPaterno;
    private $apellidoMaterno;
    private $telefono;
    private $rol;
    private $intentos;
    private $conectado;
    private $bloqueado;
    private $fechaBloqueo;
    private $fechaUltimaActividad;
    private $fechaCreacion;
    private $fechaActualizacion;
    private $fechaEliminacion;

    public function __construct($usuarioId = null, $correo = null,
$contraseña = null, $nombre = null, $apellidoPaterno = null,
$apellidoMaterno = null, $telefono = null, $rol = null, $intentos = null,
$conectado = null, $bloqueado = null, $fechaBloqueo = null,
```

```
$fechaUltimaActividad = null, $fechaCreacion = null, $fechaActualizacion =
null, $fechaEliminacion = null) {
        $this->usuarioId = $usuarioId;
        $this->correo = $correo;
        $this->contraseña = $contraseña;
        $this->nombre = $nombre;
        $this->apellidoPaterno = $apellidoPaterno;
        $this->apellidoMaterno = $apellidoMaterno;
        $this->telefono = $telefono;
        $this->rol = $rol;
        $this->intentos = $intentos;
        $this->conectado = $conectado;
        $this->bloqueado = $bloqueado;
        $this->fechaBloqueo = $fechaBloqueo;
        $this->fechaUltimaActividad = $fechaUltimaActividad;
        $this->fechaCreacion = $fechaCreacion;
        $this->fechaActualizacion = $fechaActualizacion;
        $this->fechaEliminacion = $fechaEliminacion;
    }

    public function getUsuarioId() { return $this->usuarioId; }
    public function getCorreo() { return $this->correo; }
    public function getContraseña() { return $this->contraseña; }
    public function getNombre() { return $this->nombre; }
    public function getApellidoPaterno() { return $this->apellidoPaterno; }
    public function getApellidoMaterno() { return $this->apellidoMaterno; }
    public function getTelefono() { return $this->telefono; }
    public function getRol() { return $this->rol; }
    public function getIntentos() { return $this->intentos; }
    public function getConectado() { return $this->conectado; }
    public function getBloqueado() { return $this->bloqueado; }
    public function getFechaBloqueo() { return $this->fechaBloqueo; }
    public function getFechaUltimaActividad() { return $this-
>fechaUltimaActividad; }
    public function getFechaCreacion() { return $this->fechaCreacion; }
    public function getFechaActualizacion() { return $this-
>fechaActualizacion; }
    public function getFechaEliminacion() { return $this->fechaEliminacion;
}

    public function setUsuarioId($usuarioId) { $this->usuarioId =
$usuarioId; }
```

```
    public function setCorreo($correo) { $this->correo = $correo; }
    public function setContraseña($contraseña) { $this->contraseña =
$contraseña; }
    public function setNombre($nombre) { $this->nombre = $nombre; }
    public function setApellidoPaterno($apellidoPaterno) { $this-
>apellidoPaterno = $apellidoPaterno; }
    public function setApellidoMaterno($apellidoMaterno) { $this-
>apellidoMaterno = $apellidoMaterno; }
    public function setTelefono($telefono) { $this->telefono = $telefono; }
    public function setRol($rol) { $this->rol = $rol; }
    public function setIntentos($intentos) { $this->intentos = $intentos; }
    public function setConectado($conectado) { $this->conectado =
$conectado; }
    public function setBloqueado($bloqueado) { $this->bloqueado =
$bloqueado; }
    public function setFechaBloqueo($fechaBloqueo) { $this->fechaBloqueo =
$fechaBloqueo; }
    public function setFechaUltimaActividad($fechaUltimaActividad) { $this-
>fechaUltimaActividad = $fechaUltimaActividad; }
    public function setFechaCreacion($fechaCreacion) { $this->fechaCreacion
= $fechaCreacion; }
    public function setFechaActualizacion($fechaActualizacion) { $this-
>fechaActualizacion = $fechaActualizacion; }
    public function setFechaEliminacion($fechaEliminacion) { $this-
>fechaEliminacion = $fechaEliminacion; }

    public function aumentarIntento() {
        $this->intentos++;
    }
}
```

Anexos

- *Segmentos de Código de Interacción de Objetos*

```
class ControladorReceta extends Controller
{
    private RecetaServicio $recetaServicio;
    private SucursalServicio $sucursalServicio;
    private MedicamentoServicio $medicamentoServicio;
    private NotificacionServicio $notificacionServicio;
```

```
    public function __construct()
    {
        $this->recetaServicio = new RecetaServicio();
        $this->sucursalServicio = new SucursalServicio();
        $this->medicamentoServicio = new MedicamentoServicio();
        $this->notificacionServicio = new NotificacionServicio();
    }

    public function iniciarReceta(Request $request)
    {
        try {
            $paciente = $request->session()->get('paciente');

            $this->recetaServicio->crearReceta($paciente);

            return view('paciente.subirReceta');

        } catch (Exception $e) {
            return view('paciente.subirReceta', ['error' => $e-
>getMessage()]);
        }
    }

    public function ingresarCed($cedula) {
        try {
            $receta = session()->get('receta');
            if ($receta->getEstado() !== 'En curso' || !$receta) {
                return response()->json(['error' => true, 'message' => 'No
hay una receta en curso']);
            }

            $this->recetaServicio->validarCed($cedula);

            $this->recetaServicio->asociarRecCed($receta, $cedula);

            $sucursales = $this->sucursalServicio->obtenerSucursales();
            return response()->json([ 'success' => true, 'sucursales' =>
$sucursales ]);
        } catch (Exception $e) {
            return response()->json(['error' => true, 'message' => $e-
>getMessage()]);
```

```
        }
    }

    public function seleccionarSuc($sucId, $cadId)
    {
        try {
            $receta = session()->get('receta');
            if ($receta->getEstado() !== 'En curso' || !$receta) {
                return response()->json(['error' => true, 'message' => 'No
hay una receta en curso']);
            }

            $suc = $this->sucursalServicio->obtenerSuc($sucId, $cadId);

            $this->recetaServicio->asociarRecSuc($receta, $suc);

            return response()->json(['success' => true, 'sucursal' => $suc-
>toArray()]);

        } catch (Exception $e) {
            return response()->json(['error' => true, 'message' => $e-
>getMessage()]);
        }
    }

    public function ingresarNomMed($nombreMed) {
        try {
            $receta = session()->get('receta');
            if ($receta->getEstado() !== 'En curso' || !$receta) {
                return response()->json(['error' => true, 'message' => 'No
hay una receta en curso']);
            }

            $medicamentos = $this->medicamentoServicio-
>obtenerMedicamentos($nombreMed);

            return response()->json(['success' => true, 'medicamentos' =>
$medicamentos]);

        } catch (Exception $e) {
            return response()->json(['error' => true, 'message' => $e-
>getMessage()]);
```

```
        }
    }

    public function seleccionarMed($medId, $cantidad)
    {
        try {
            $receta = session()->get('receta');
            if ($receta->getEstado() !== 'En curso' || !$receta) {
                return response()->json(['error' => true, 'message' => 'No
hay una receta en curso']);
            }

            $med = $this->medicamentoServicio->obtenerMed($medId);
            $this->recetaServicio->crearLinRec($receta, $med, $cantidad);

            return response()->json(['success' => true, 'medicamentos' =>
$receta->mostrarMedIngresados()]);

        } catch (Exception $e) {
            return response()->json(['error' => true, 'message' => $e-
>getMessage()]);
        }
    }

    public function eliminarMed($medId)
    {
        try {
            $receta = session()->get('receta');
            if ($receta->getEstado() !== 'En curso' || !$receta) {
                return response()->json(['error' => true, 'message' => 'No
hay una receta en curso']);
            }

            $receta->eliminarLinRec($medId);

            session()->put('receta', $receta);
            return response()->json(['success' => true, 'medicamentos' =>
$receta->mostrarMedIngresados()]);

        } catch (Exception $e) {
            return response()->json(['error' => true, 'message' => $e-
>getMessage()]);
```

```
        }
    }

    public function confirmarReceta() {
        try {
            $receta = session()->get('receta');
            if ($receta->getEstado() !== 'En curso' || !$receta) {
                return response()->json(['error' => true, 'message' => 'No
hay una receta en curso']);
            }

            $receta = $this->recetaServicio->confirmarReceta($receta);

            $this->notificacionServicio->enviarNotificacion($receta);

            session()->put('comprobante_receta', $receta);
            return redirect()->route('mostrarComprobante')->with('success',
'Receta confirmada exitosamente.');
        } catch (Exception $e) {
            return redirect()->route('iniciarReceta')->with('error', $e-
>getMessage());
        }
    }

    public function mostrarComprobante() {
        $receta = session()->get('comprobante_receta');
        if (!$receta) {
            return redirect()->route('iniciarReceta')->with('error', 'No hay
comprobante disponible.');
        }
        return view('paciente.comprobante', ['receta' => $receta]);
    }

    public function descargarComprobante()
    {
        try {
            $receta = session()->get('comprobante_receta');
            $html = view('paciente.comprobantePdf', compact('receta'))-
>render();
            $mpdf = new Mpdf();
            $mpdf->WriteHTML($html);
            $mpdf->Output('comprobante.pdf', 'I');
```

```
        } catch (\Throwable $th) {
            return redirect()->route('mostrarComprobante')->with('error',
'No se pudo generar el PDF.');
        }
    }
}

class ControladorGestion extends Controller
{
    private RecetaServicio $recetaServicio;
    private PacienteServicio $pacienteServicio;
    private NotificacionServicio $notificacionServicio;

    public function __construct()
    {
        $this->recetaServicio = new RecetaServicio();
        $this->pacienteServicio = new pacienteServicio();
        $this->notificacionServicio = new NotificacionServicio();
    }

    public function iniciarGestionRecetas() {
        try {
            $farmaceutico = session('farmaceutico');
            $sucursal = $farmaceutico->getSucursal();

            $recetasNoRecogidas = $this->recetaServicio-
>obtenerRecetasNoRecogidas($sucursal->getCadena()->getCadenaId(), $sucursal-
>getSucursalId());

            return view('farmaceutico.recetasNoRecogidas',
['recetasNoRecogidas' => $recetasNoRecogidas]);
        } catch (Exception $e) {
            return view('farmaceutico.recetasNoRecogidas')->with('error',
$e->getMessage());
        }
    }

    public function seleccionarRec($folio) {
        try {
            $receta = $this->recetaServicio->obtenerRec($folio);
            session()->put('recetaSeleccionada', $receta);
```

```
            return redirect()->route('mostrarReceta');
        } catch (Exception $e) {
            return view('farmaceutico.recetasNoRecogidas')->with('error',
$e->getMessage());
        }
    }

    public function mostrarReceta() {
        try {
            $receta = session('recetaSeleccionada');

            return view('farmaceutico.revisionReceta', ['receta' =>
$receta]);
        } catch (Exception $e) {
            return view('farmaceutico.revisionReceta')->with('error', $e-
>getMessage());
        }
    }

    public function confirmarRecogido() {
        try {
            $receta = session('recetaSeleccionada');
            if ($receta->getEstado() !== 'No recogida' || !$receta) {
                return redirect()->route('iniciarGestionRecetas')-
>with('error', 'La receta no está en estado No recogida.');
            }

            $this->recetaServicio->confirmarRecogido($receta);

            return redirect()->route('iniciarGestionRecetas')-
>with('success', 'Receta marcada como recogida correctamente.');
        } catch (Exception $e) {
            return redirect()->route('iniciarGestionRecetas')->with('error',
$e->getMessage());
        }
    }

    public function confirmarNoRecogido() {
        try {
            $receta = session('recetaSeleccionada');
            if ($receta->getEstado() !== 'No recogida' || !$receta) {
```

```
                return redirect()->route('iniciarGestionRecetas')-
>with('error', 'La receta no está en estado No recogida.');
            }

            $receta = $this->recetaServicio->confirmarNoRecogido($receta);
            $this->pacienteServicio->aplicarMulta( $receta);
            $this->notificacionServicio->notificarMulta($receta);

            return redirect()->route('mostrarPedidosRuta')->with('success',
'Receta marcada como no recogida correctamente y multa aplicada.');
        } catch (Exception $e) {
            if (str_contains($e->getMessage(), 'Error al notificar multa'))
{
                return redirect()->route('mostrarPedidosRuta')-
>with('error', $e->getMessage());
            }
            return redirect()->route('iniciarGestionRecetas')->with('error',
$e->getMessage());
        }
    }

    public function mostrarPedidosRuta() {
        try {
            $receta = session('recetaSeleccionada');

            $receta = $this->recetaServicio->obtenerPedidos($receta);

            $ruta = $this->recetaServicio->obtenerRuta($receta);

            return view('farmaceutico.pedidosRuta', ['receta' => $receta,
'ruta' => $ruta]);
        } catch (Exception $e) {
            return view('farmaceutico.pedidosRuta')->with('error', $e-
>getMessage());
        }
    }
}


class RecetaServicio
{
    private ConsultaDB $consultaDB;
```

```
    private ActualizacionDB $actualizacionDB;
    private InsercionDB $insercionDB;
    private SucursalServicio $sucursalServicio;

    public function __construct()
    {
        $this->consultaDB = ConsultaDB::getInstancia();
        $this->actualizacionDB = ActualizacionDB::getInstancia();
        $this->insercionDB = InsercionDB::getInstancia();
        $this->sucursalServicio = new sucursalServicio();
    }

    public function crearReceta($paciente) {
        $receta = new Receta(null, null, null, null, null, $paciente);
        session()->put('receta', $receta);
    }

    public function asociarRecCed($receta, $cedula) {
        $receta->setCedulaDoctor($cedula);
        session()->put('receta', $receta);
    }

    public function asociarRecSuc($receta, $suc) {
        $receta->setSucursal($suc);
        session()->put('receta', $receta);
    }

    public function crearLinRec($receta, $med, $cantidad) {
        $receta->crearLinRec($med, $cantidad);
        session()->put('receta', $receta);
    }

    public function validarCed($cedula)
    {
        if (!$cedula || empty($cedula)) {
            throw new Exception('La cédula es obligatoria.');
        }
        if (strlen($cedula) > 10) {
            throw new Exception('La cédula no puede tener más de 10
caracteres.');
        }
        if (!preg_match('/^[a-zA-Z0-9]+$/', $cedula)) {
```

```
            throw new Exception('La cédula no puede contener símbolos.');
        }
        return true;
    }

    public function confirmarReceta($receta)
    {
        $origen = $receta->getSucursal();
        $cantidadSucs = $this->sucursalServicio->prepararBusqueda($origen);

        try {
            $this->consultaDB->iniciarTransaccion();

            foreach ($receta->getLineasReceta() as $linRec) {
                $this->sucursalServicio->reiniciarBusqueda();
                $med = $linRec->getMedicamento();
                $cantSolicitada = $linRec->getCantidad();
                $visitadas = 0;

                while ($cantSolicitada > 0 && $visitadas < $cantidadSucs) {
                    if ($visitadas != 0) $sucursal = $this-
>sucursalServicio->obtenerSucCerc($origen);
                    else $sucursal = $origen;

                    $cantPedido = $this->sucursalServicio-
>obtenerStock($sucursal, $med, $cantSolicitada);
                    if ($cantPedido != 0) {
                        $cantSolicitada -= $cantPedido;
                        $precio = $this->sucursalServicio->obtenerPrecio();
                        $linRec->crearLinPed($sucursal, $cantPedido,
$precio);
                    }
                    $visitadas++;
                }

                if ($cantSolicitada > 0) {
                    $this->consultaDB->rollback();
                    throw new Exception('No hay stock disponible para el
medicamento: ' . $med->getNombre());
                }
            }
```

```
            $receta->calcularTotal();
            $receta->generarFechaEstimada();
            $folio = $this->crearFolio($receta->getPaciente());
            $receta->setFolio($folio);
            $this->insercionDB->confirmarReceta($receta);
            $this->consultaDB->hacerCommit();
            return $receta;

        } catch (Exception $e) {
            $this->consultaDB->rollback();
            Log::error('Error al confirmar receta: ' . $e->getMessage());
            throw $e;
        }
    }

    public function obtenerRecetasNoRecogidas($cadenaId, $sucursalId)
    {
        try {
            $recetas = $this->consultaDB-
>obtenerRecetasNoRecogidas($cadenaId, $sucursalId);
            return $recetas;
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function obtenerRec($folio)
    {
        try {
            $receta = $this->consultaDB->obtenerRec($folio);
            return $receta;
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function confirmarRecogido($receta)
    {
        try {
            $receta->setEstado('Recogido');
            $this->consultaDB->iniciarTransaccion();
            $this->actualizacionDB->actualizarEstRec($receta);
```

```
            $this->consultaDB->hacerCommit();
        } catch (Exception $e) {
            $this->consultaDB->rollback();
            throw $e;
        }
    }

    public function confirmarNoRecogido($receta)
    {
        try {
            $this->consultaDB->iniciarTransaccion();
            foreach ($receta->getLineasReceta() as $linRec) {
                foreach ($linRec->getLineasPedido() as $linPed) {
                    $this->sucursalServicio->devolverMedInv($linRec,
$linPed);
                }
            }
            $receta->setEstado('No entregado');
            $this->actualizacionDB->actualizarEstRec($receta);
            $this->consultaDB->hacerCommit();
            return $receta;
        } catch (Exception $e) {
            $this->consultaDB->rollback();
            throw $e;
        }
    }

    public function obtenerPedidos($receta) {
        try {
            $recetaConPedidos = $this->consultaDB->obtenerPedidos($receta);
            return $recetaConPedidos;
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function obtenerRuta($receta) {
        $pedidos = [];
        foreach ($receta->getLineasReceta() as $lineaReceta) {
            foreach ($lineaReceta->getLineasPedido() as $pedido) {
                $pedidos[] = $pedido;
            }
```

```
        }

        usort($pedidos, function($a, $b) {
            return $a->getFechaCreacionNs() <=> $b->getFechaCreacionNs();
        });

        $ruta = [];
        $repetidas = [];

        $ruta[] = $receta->getSucursal()->toArray();
        $repetidas[] = $receta->getSucursal()->getSucursalId() . '-' .
$receta->getSucursal()->getCadena()->getCadenaId();
        foreach ($pedidos as $pedido) {
            $sucursal = $pedido->getDestinoSucursal();
            $id = $sucursal->getSucursalId() . '-' . $sucursal->getCadena()-
>getCadenaId();
            if (!in_array($id, $repetidas)) {
                $repetidas[] = $id;
                $ruta[] = $sucursal->toArray();
            }
        }
        return $ruta;
    }

    public function obtenerRecetas($usuarioId){
        try {
            $recetas = $this->consultaDB->obtenerRecetas($usuarioId);
            return $recetas;
        } catch (Exception $e) {
            throw $e;
        }
    }


    public function crearFolio($paciente){
        $nombre = $paciente->getNombre();
        $apP = $paciente->getApellidoPaterno();
        $apM = $paciente->getApellidoMaterno();

        $iniciales =
            strtoupper(substr($nombre, 0, 1)) .
            strtoupper(substr($apP, 0, 1)) .
```

```
            strtoupper(substr($apM, 0, 1));

        $fecha = date("Ymd");

        $random =
strtoupper(substr(str_shuffle("ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghi
jklmnopqrstuvwxyz"), 0, 4));

        $folioGenerado = $iniciales . $fecha . $random;

        return $folioGenerado;
    }
}

class SucursalServicio
{
    private ConsultaDB $consultaDB;
    private ActualizacionDB $actualizacionDB;
    private Grafo $grafo;
    private $visitadasClaves = [];
    private Inventario $inventario;

    public function __construct()
    {
        $this->consultaDB = ConsultaDB::getInstancia();
        $this->actualizacionDB = ActualizacionDB::getInstancia();
    }

    public function obtenerSucursales()
    {
        try {
            $sucursales = $this->consultaDB->obtenerSucursales();
            return array_map(
                fn($sucursal) => $sucursal->toArray(),
                $sucursales
            );
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function obtenerSuc($sucId, $cadId)
```

```
    {
        try {
            $suc = $this->consultaDB->obtenerSuc($sucId, $cadId);
            return $suc;
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function prepararBusqueda($origen)
    {
        $sucs = $this->consultaDB->obtenerSucursales();
        $this->grafo = FactoryGrafo::construirGrafo($sucs, $origen);
        return count($sucs);
    }

    public function reiniciarBusqueda()
    {
        $this->visitadasClaves = [];
    }

    public function obtenerSucCerc($origen)
    {
        $origenClave = FactoryGrafo::claveSucursal($origen);
        $sucursal = $this->grafo->obtenerSucCercana($origenClave, $this-
>visitadasClaves);
        // Log::info('Sucursal mas cercana: ' . ($sucursal ? $sucursal-
>getNombre() : ''));
        $this->marcarSucursalVisitada($sucursal);
        return $sucursal;
    }

    public function obtenerStock($suc, $med, $cantidadSolicitada){
        $inventario = $this->consultaDB->obtenerInventario($suc, $med);

        if($inventario === null || $inventario->getStock() <= 0){
            return 0;
        }

        $cantPedido = $inventario->procesarCant($cantidadSolicitada);

        $this->inventario = $inventario;
```

```
        $this->actualizacionDB->actualizarInventario($inventario);

        return $cantPedido;
    }

    public function obtenerPrecio(){
        return $this->inventario->getPrecioUnitario();
    }

    private function marcarSucursalVisitada($sucursal)
    {
        $clave = FactoryGrafo::claveSucursal($sucursal);
        if (!in_array($clave, $this->visitadasClaves)) {
            $this->visitadasClaves[] = $clave;
        }
    }

    public function devolverMedInv($linRec, $linPed) {
        try {
            $suc = $linPed->getDestinoSucursal();
            $med = $linRec->getMedicamento();
            $cant = $linPed->getCantidad();
            $inv = $this->consultaDB->obtenerInventario($suc, $med);
            if ($inv === null) {
                throw new Exception('Inventario no encontrado de ' . $suc-
>getNombre() . ' para el medicamento ' . $med->getNombre());
            }
            $inv->aumentarStock($cant);
            $this->actualizacionDB->actualizarInventario($inv);
        } catch (Exception $e) {
            throw $e;
        }
    }
}

class MedicamentoServicio
{
    private ConsultaDB $consultaDB;

    public function __construct()
    {
        $this->consultaDB = ConsultaDB::getInstancia();
```

```
    }

    public function obtenerMedicamentos($nombreMed)
    {
        // Prevenir inyeccion SQL a-z y A-Z
        if (!preg_match('/^[a-zA-Z]+$/', $nombreMed)) {
            throw new Exception('El nombre del medicamento solo puede
contener letras.');
        }

        try {
            $medicamentos = $this->consultaDB-
>obtenerMedsNombre($nombreMed);
            return array_map(
                fn($med) => $med->toArray(),
                $medicamentos
            );
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function obtenerMed($medId)
    {
        try {
            $med = $this->consultaDB->obtenerMed($medId);
            return $med;
        } catch (Exception $e) {
            throw $e;
        }
    }
}

class NotificacionServicio
{
    private ConsultaDB $consultaDB;
    private ActualizacionDB $actualizacionDB;
    private InsercionDB $insercionDB;

    public function __construct()
    {
        $this->consultaDB = ConsultaDB::getInstancia();
```

```
        $this->actualizacionDB = ActualizacionDB::getInstancia();
        $this->insercionDB = InsercionDB::getInstancia();
    }

    public function enviarNotificacion($receta) {
        try {
            $notificaciones = [];

            $notificaciones[] = new Notificacion(
                null,
                'Receta registrada #'.$receta->getFolio(),
                'Tu receta ha sido registrada exitosamente',
                false,
                $receta->getPaciente(),
                null,
                $receta,
            );

            $notificaciones[] = new Notificacion(
                null,
                'Nueva receta para surtir #'.$receta->getFolio(),
                'Hay una nueva receta para surtir en esta sucursal.',
                false,
                null,
                $receta->getSucursal(),
            );

            foreach ($receta->getLineasReceta() as $lineaReceta) {
                foreach ($lineaReceta->getLineasPedido() as $lineaPedido) {
                    $notificaciones[] = new Notificacion(
                        null,
                        'Pedido de receta #'.$receta->getFolio(),
                        'Prepara el medicamento "' . $lineaReceta-
>getMedicamento()->getNombre() . '" (Cantidad: ' . $lineaPedido-
>getCantidad() . ').',
                        false,
                        null,
                        $lineaPedido->getDestinoSucursal(),
                    );
                }
            }
```

```
            $this->insercionDB->guardarNotificaciones($notificaciones);
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function obtenerNotificacionesPaciente($usuarioId)
    {
        try {
            return $this->consultaDB-
>obtenerNotificacionesPaciente($usuarioId);
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function obtenerNotificacionesFarmaceutico($cadenaId,
$sucursalId)
    {
        try {
            return $this->consultaDB-
>obtenerNotificacionesFarmaceutico($cadenaId, $sucursalId);
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function marcarComoLeido($notificacionId)
    {
        try {
            $this->actualizacionDB-
>marcarNotificacionComoLeido($notificacionId);
        } catch (Exception $e) {
            throw $e;
        }
    }

    public function notificarMulta($receta)
    {
        try {
            $notificacion = new Notificacion(
                null,
```

```
                'Receta no recogida, Multa aplicada',
                'Se ha aplicado una multa a su tarjeta asociada, por no
recoger la receta #'.$receta->getFolio(),
                false,
                $receta->getPaciente(),
                null,
                $receta,
            );

            $this->insercionDB->guardarNotificaciones([$notificacion]);
        } catch (Exception $e) {
            throw New Exception("Error al notificar multa: " . $e-
>getMessage() . " Telefono del paciente: " . $receta->getPaciente()-
>getTelefono());
        }
    }
}

class PacienteServicio
{
    private ConsultaDB $consultaDB;
    private ActualizacionDB $actualizacionDB;

    public function __construct()
    {
        $this->consultaDB = ConsultaDB::getInstancia();
        $this->actualizacionDB = ActualizacionDB::getInstancia();
    }

    public function aplicarMulta($receta)
    {
        try {
            $this->consultaDB->iniciarTransaccion();
            $paciente = $receta->getPaciente();
            $tarjeta = $this->consultaDB->obtenerTarjeta($paciente);
            $tarjeta->aplicarMulta($receta->getTotal());
            $this->actualizacionDB->actualizarSaldo($tarjeta);
            $this->consultaDB->hacerCommit();
        } catch (Exception $e) {
            $this->consultaDB->rollback();
            throw $e;
        }
```

```
    }
}
```

- *Lista de Tecnologías Utilizada*

1. Metodología de desarrollo: PU (Proceso Unificado), SCRUM.
2. CASE: Enterprise Architect.
3. Notación de modelado: UML y UWE.
4. Marco de desarrollo: LARAVEL (MVC -Modelo Vista Controlador-).
5. Paradigma de desarrollo: Orientado a objetos.
6. Flutter
7. Sistema Gestor de Base de Datos (Orientado a Objetos, Relacional).
8. Servicio web.