UNIVERSIDAD AUTÓNOMA GABRIEL RENÉ MORENO

FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

TECNOLOGÍA WEB (INF513-SA)

Proyecto No.1 - Sistema vía Mail Básico

"SISTEMA MAIL PARA LA CLINICA VETERINARIA C& B"

Grupo #9

Autores:

Condori Soraide Jackelin	215057473
Mejía Orihuela Marina	215059514

Docente:

M.SC. ING. EVANS BALCÁZAR VEIZAGA

SANTA CRUZ – BOLIVIA

Santa Cruz de la Sierra - Estado Plurinacional de Bolivia
Universidad Autónoma Gabriel René Moreno (U.A.G.R.M.)
Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones (F.I.C.C.T.)

Tecnología Web (INF513-SA)
Docente:
M.SC. ING. EVANS BALCÁZAR VEIZAGA

PROYECTOS GESTIÓN 1-2023
Entrega Proyecto No.1 - Sistema vía Mail Básico
SISTEMA MAIL PARA LA ADMINISTRACION DE CONDOMINIOS
Grupo #9

Autores:
Condori Soraide Jackeline
Mejía Orihuela Marina

Santa Cruz de la Sierra - Estado Plurinacional de Bolivia

Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones Av. Busch, Ciudad Universitaria, Módulo 236.
Telf. - Fax: (591) 3 - 3553636
E-mail: f_icct@uagrm.edu.bo
Pag.-Web: https://www.uagrm.edu.bo/facultades/ficct
Santa Cruz, Bolivia

Impreso en Bolivia
Printed in Bolivia

INDICE DE CONTENIDO

TABLA DE ILUSTRACIONES

1 Introducción

La medicina veterinaria se define como la aplicación de la medicina en los animales no humanos. Esta se ocupa en la prevención, diagnóstico, tratamiento de enfermedades, trastornos y lesiones que pueden llegar a pasar en cualquier situación a los animales sea en la casa del dueño o fuera de ella.

La veterinaria “C & B” es una clínica veterinaria, que se encarga en el manejo y manipulación de animales domésticos que se encuentran en los hogares de cada cliente, contando con médicos veterinarios, que se encargan del diagnóstico de la salud y estado del animal proporcionando una calidad de trabajo en su atención, y técnicos veterinarios, que es el encargado del manejo, administración y organización de una clínica veterinaria y conocimientos de los procesos para un desarrollo exitoso.

Si bien la comunicación tradicional entre doctores y técnicos es muy fluida al momento de asignar tareas y servicios que corresponden a los pacientes. En momentos que mas se precisas la informacion de una situacion la comunicación tradicional puede llegar a fallar. En ese caso se consulto mediante encuentas que herramientas serian las mas adecuadas para solucionar problemas, se escogio la mas adecuado que es este presente proyecto, donde se asignara de manera eficaz los servicios y vacunas a los pacientes y quienes seran los responsables en su debida atencion y en su diagnostico

2 Antecedentes y Justificaciones

3 El problema

3.1 Definición del problema

La Veterinaria “C & B” al ser una clínica, que presta servicios de prevenir, diagnosticar, dar tratamiento y supervisar el estado del avance de la salud del paciente. Este no posee un plantel administrativo para cubrir las actividades de la organización y asignación de los tratamiento o servicios que se planean darle a la variedad de pacientes que se reciben en un tiempo determinado. Como una desventaja en el momento de la organización de consultas, que doctor veterinario se haya encargado de cierto paciente, mostrando datos como recetas, vacunas y tratamientos sigue este para su salud. De esta manera se precisa una manera eficiente para los veterinarios en la consulta de estos datos para sus pacientes.

3.2 Situación problemática

La clínica veterinaria “C & B” para la consulta, listado y la vista de los datos de ciertos pacientes, se tiene un retraso enorme en el tiempo de consulta de datos de estos, una mala asignación de datos de cada paciente de parte del sistema de la clínica y un listado de pacientes no ordenados de manera adecuada.

3.2.1.1 Situación deseada

La clínica veterinaria “C & B” realizara los mismos procesos que tienen retrasos en las consultas de datos, pero con la solución propuesta agilizara los procesos necesarios. Quitando del caminos los retrasos de tiempo que existen en la clinica al solicuitar informacion de los pacientes.

4 Objetivos

4.1 Objetivos General

Implementar un servidor de correo electrónico que nos permita consultar, listar y asignar datos de pacientes en la clínica veterinaria “C & B”.

4.2 Objetivos Especifico

- Definir los requisitos funcionales y no funcionales a traves de lo requerimientos.
- Analizar los requisitos para entender la situacion problematica.
- Diseñar una base de datos que se aescalable y con bajo acoplamiento.
- Implementar un server de correo que sea capaz de registrar, modificar y listar la informacion necesaria que se solicite.
- Implementar un sistema que intereactue con el personal de la veterianaria mediante correos electrocnicos para la consulta de pacientes.

5 Metodología

Para la compresión y presentación clara de cómo se realizó el proyecto se escogió la metodología PUDS para su desarrollo y UML para mostrar las actividades por medio de gráficos de cada proceso.

5.1 UML

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software como en el flujo de procesos en la fabricación.

- UML cumple con los siguientes requerimientos:
- Establecer una definición formal de un metamodelo común basado en el estándar MOF (Meta-Object Facility) que especifique la sintaxis abstracta del UML.
- Brindar una explicación detallada de la semántica de cada concepto de modelado UML
- Especificar los elementos de notación de lectura humana para representar los conceptos individuales de modelado UML, así como las reglas para combinarlos en una variedad de diferentes tipos de diagramas que corresponden a diferentes aspectos de los sistemas modelados.
- Definir formas que permitan hacer que las herramientas UML cumplan con esta especificación.

5.2 PUDS

El Proceso Unificado de Desarrollo de Software (PUDS) es un proceso iterativo e incremental caracterizado por casos de uso, estar centrado en la arquitectura y estar enfocado en el riesgo.

Este proceso permite el desarrollo de software de gran escala, al estar teniendo pruebas constantes y obteniendo respuesta con base en estas pruebas, esto con el fin de garantizar la calidad del resultado final. Sin embargo, esto ocasiona un incremento en la complejidad de la realización del proyecto.

A continuación, veremos las Fases, Flujo de Trabajo, y Actividades correspondientes al PUDS que se llevarán a cabo durante el desarrollo del sistema.

Fase de Inicio

- Captura de requisitos.
- Requisitos funcionales y no funcionales.
- Identificar y describir actores y casos de uso
- Priorizar casos de usos.
- Estructuración de casos de uso.
- Realizar los prototipos de interfaces de casos de uso.

Fase de Elaboración

- Análisis de elaboración.
- Análisis de casos de uso.
- Análisis de paquete.
- Análisis de arquitectura.
- Resultado; especificación de requisitos de software.
- Diseño de infractora.
- Diseño de casos de uso
- Diseño de arquitectura.
- Diseño de datos
- Resultado: Descripción de diseño de software.

- **Fase de Construcción**
- Implementación.
- Traducción de los modelos de diseño a lenguaje de programación.

6 Marco Teórico

6.1 Servidor de Correo

Los servidores de correo electrónico se encargan de la gestión del envío de correos. Esta gestión funciona mediante protocolos que se encargan de dar órdenes para identificar tanto al servidor emisor como receptor. Este servidor funciona principalmente con los servicios de SMTP que se encarga del envió correo electrónico y POP o IMAP que se encarga de recibir todo mensaje que es enviado. Sus ventajas del servidor son:

- Escalabilidad
- Seguridad
- Estabilidad
- Adaptabilidad

6.2 Correo Electrónico

Correo electrónico, o en inglés e-mail (electronic mail), es un servicio de red que permite a los usuarios enviar y recibir mensajes rápidamente (también denominados mensajes electrónicos o cartas electrónicas) mediante sistemas de comunicación electrónicos. Principalmente se usa el nombre de “correo electrónico” para denominar al sistema que provee este servicio en Internet, mediante el protocolo SMTP, aunque por extensión también puede verse aplicado a sistemas análogos que usen otras tecnologías. Por mediode mensajes de correo electrónico se puede enviar, no solamente texto, sino todo tipo dedocumentos digitales. Su eficiencia, conveniencia y bajo costo están logrando que el correo electrónico desplace al correo ordinario para muchos usos habituales.

6.3 SMTP Y POP3

6.3.1 Sockets

Un socket (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red, se define, por tanto, como el punto final en una conexión.

El término socket es también usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo.

Para que dos procesos puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

- Que un proceso sea capaz de localizar al otro.
- Que ambos procesos sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.
- Para ello son necesarios los dos recursos que originan el concepto de socket:
- Un par de direcciones del protocolo de red (dirección IP, si se utiliza el protocolo TCP/IP), que identifican la computadora de origen y la remota.
- Un par de números de puerto, que identifican a un programa dentro de cada computadora. Para ello son necesarios los dos recursos que originan el concepto de socket:

- Un par de direcciones del protocolo de red (dirección IP, si se utiliza el protocolo TCP/IP), que identifican la computadora de origen y la remota.
- Un par de números de puerto, que identifican a un programa dentro de cada computadora.

Los sockets permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por uno de los procesos que se denomina programa "cliente". El segundo

proceso espera a que otro inicie la comunicación, por este motivo se denomina programa "servidor".

Cuando un cliente conecta con el servidor se crea un nuevo socket, de esta forma, el servidor puede seguir esperando conexiones en el socket principal y comunicarse con el cliente conectado, de igual manera se establece un socket en el cliente en un puerto local.

Una aplicación servidor normalmente escucha por un puerto específico esperando una petición de conexión de un cliente, una vez que se recibe, el cliente y el servidor se conectan de forma que les sea posible comunicarse entre ambos. Durante este proceso, el cliente es asignado a un número de puerto, mediante el cual envía peticiones al servidor y recibe de este las respuestas correspondientes.

Similarmente, el servidor obtiene un nuevo número de puerto local que le servirá para poder continuar escuchando cada petición de conexión del puerto original. De igual forma une un socket a este puerto local.

6.3.2 SMTP

El protocolo SMTP (Protocolo simple de transferencia de correo) es el protocolo estándar que permite la transferencia de correo de un servidor a otro mediante una conexión punto a punto. Éste es un protocolo que funciona en línea, encapsulado en una trama TCP/IP. El correo se envía directamente al servidor de correo del destinatario. El protocolo SMTP funciona con comandos de textos enviados al servidor SMTP (al puerto 25 de manera predeterminada).A cada comando enviado por el cliente (validado por la cadena de caracteres

ASCII CR/LF, que equivale a presionar la tecla Enter) le sigue una respuesta del servidor SMTP compuesta por un número y un mensaje descriptivo.

6.3.2.1 Retransmisión

Cuando se envía un email a través del protocolo de retransmisión SMTP, lo que se produce es la validación de una serie de comandos de texto (de la cadena de caracteres ASCII), que posteriormente son enviados a un servidor SMTP. Por lo general, se utilizan los puertos 25 o 587.

En este proceso no entra en juego el contenido del correo electrónico, sino que la atención del lenguaje SMTP define exclusivamente en la transmisión.

Cada vez que se envía un email mediante el protocolo SMTP, se abre una nueva sesión del servicio de retransmisión SMTP. A continuación, se llevan a cabo una serie de intercambios de información entre el cliente de email y el servidor SMTP de destino, como si de una conversación se tratara.

6.3.2.2 Descripción del protocolo

SMTP es un protocolo orientado a la conexión basado en texto, en el que un remitente de correo se comunica con un receptor de correo electrónico mediante la emisión de secuencias de comandos y el suministro de los datos necesarios en un canal de flujo de datos ordenado fiable, normalmente un protocolo de control de transmisión de conexión (TCP). Una sesión

SMTP consiste en comandos originados por un cliente SMTP (el agente de inicio, emisor o transmisor) y las respuestas correspondientes del SMTP del servidor (el agente de escucha, o receptor) para que la sesión se abra y se intercambian los parámetros de la sesión. Una sesión puede incluir cero o más transacciones SMTP. Una transacción de SMTP se compone de tres secuencias de comando / respuesta.

Los cuales son:

- **MAIL:** Comando para establecer la dirección de retorno, también conocido como ReturnPath, remitente o sobre. Esta es la dirección para mensajes de despedida.
- **RCPT:** Comando, para establecer un destinatario de este mensaje. Este mandato puede emitirse varias veces, una para cada destinatario. Estas direcciones son también parte de la envolvente.
- **DATA:** Para enviar el mensaje de texto. Este es el contenido del mensaje, en lugar de su envoltura. Se compone de una cabecera de mensaje y el cuerpo del mensaje separado por una línea en blanco. DATA es en realidad un grupo de comandos, y el servidor responde dos veces: una vez para el comando de datos adecuada, para reconocer que está listo para recibir el texto, y la segunda vez después de la secuencia final de los datos, para aceptar o rechazar todo el mensaje.

6.3.2.3 Comandos

La conversación se produce mediante comandos de texto muy simples. Los más comunes son los siguientes:

- **HELO:** Para abrir una sesión con el servidor.
- **EHLO:** Para abrir una sesión, en el caso de que el servidor soporte extensiones definidas en el RFC 1651.
- **MAIL FROM:** Para indicar quien envía el mensaje.
- **RCPT TO:** Para indicar el destinatario del mensaje.
- **DATA:** Para indicar el comienzo del mensaje, este finalizará cuando haya una línea únicamente con un punto.
- **QUIT:** Para cerrar la sesión.

La conversación se produce mediante comandos de texto muy simples. Los más comunes son los siguientes:

- **HELO:** Para abrir una sesión con el servidor.
- **EHLO:** Para abrir una sesión, en el caso de que el servidor soporte extensiones definidas en el RFC 1651.
- **MAIL FROM:** Para indicar quien envía el mensaje.
- **RCPT TO:** Para indicar el destinatario del mensaje.
- **DATA:** Para indicar el comienzo del mensaje, este finalizará cuando haya una línea únicamente con un punto.
- **QUIT:** Para cerrar la sesión.

6.3.3 POP3

El protocolo POP (Protocolo de oficina de correos), como su nombre lo indica, permite recoger el correo electrónico en un servidor remoto (servidor POP). Es necesario para las personas que no están permanentemente conectadas a Internet, ya que así pueden consultar sus correos electrónicos recibidos sin que ellos estén conectados. Existen dos versiones principales de este protocolo, POP2 y POP3, que usan los puertos 109 y 110 respectivamente, y que funcionan utilizando comandos de texto radicalmente diferentes. Al igual que con el

protocolo SMTP, el protocolo POP (POP2 y POP3) funciona con comandos de texto enviados al servidor POP. Cada uno de estos comandos enviados por el cliente (validados por la cadena CR/LF) está compuesto por una palabra clave, posiblemente acompañada por uno o varios argumentos, y está seguido por una respuesta del servidor POP compuesta por un número y un mensaje descriptivo. Por lo tanto, el protocolo POP3 administra la autenticación utilizando el nombre de usuario y la contraseña. Sin embargo, esto no es seguro, ya que las contraseñas, al igual que los correos electrónicos, circulan por la red como texto sin codificar (de manera no cifrada). Además, el protocolo POP3 bloquea las bandejas de entrada durante el acceso, lo que significa que es imposible que dos usuarios accedan de manera simultánea a la misma bandeja de entrada.

7 Desarrollo

7.1 Captura de requisitos

7.1.1 Identificación de casos de uso

7.1.2 Lista de actores

- Administrador: persona que envía comunicados a los integrantes de los condominios
- Cliente: persona que es dueño del paciente(Mascota)

7.1.3 Identificación de casos de uso

- CU1. Gestión de Usuarios (Administrativos, Clientes, Pacientes).
- CU2. Gestionar Inventarios(Almacén-Activo).
- CU3. Gestionar Productos.
- CU4. Gestionar Servicios (Vacunación, Consultas).
- CU5. Gestionar Recetas.
- CU6. Gestionar Historial
- CU7. Gestionar Pagos
- CU8. Gestionar Reportes y Estadísticas

7.2 Priorización de Casos de Uso

Lista de Casos de Uso	Prioridad
CU1. Gestión de Usuarios	Alta
CU2. Gestionar Inventarios	Alta
CU3. Gestionar Productos.	media
CU4. Gestionar Servicio	media
CU5. Gestionar Recetas.	media
CU6. Gestionar Historial	media
CU7. Gestionar Pagos	Alta
CU8. Gestionar Reportes y Estadísticas	Baja

Tabla 1. Tabla de priorización de casos de uso

7.3 Detalle de casos de uso

7.3.1 CU1. Gestión de Usuarios

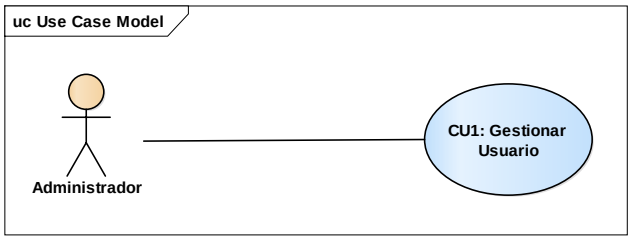


Figure 1. Diagrama de caso de uso -CU1.Getionar usuarios

Caso de uso	CU1: Gestionar Usuarios
Proposito	Su propósito es registra, modificar, listar y eliminar los usuarios del sistema
Descripcion	El sistema permitirá registro, modificación, listado y eliminación de los usuarios
Actor	Administrator
Actor iniciador	Administrator
Precondicion	Ninguna
Flujo principal	1. Se envía por correo electrónico el comando correspondiente para gestionar un usuario. (“usuario_crear[]”, “usuario_todos[]”,”usuario_uno[]”, “usuario_editar[]”, “usuario_eliminar[]”). 2. El servidor lee el correo. 3. Se valida los datos 4. El servidor envia la respuesta
Excepcion	Comandos invalidos Parametros incorrectos Fallo en la conexion con la base de datos

Tabla 2 Diagrama de caso de uso -CU1.Getionar usuarios

CU2. Gestionar Inventarios

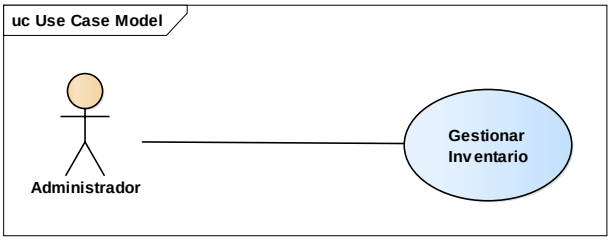


Figure 2Diagrama de caso de uso - CU2.Getionar inventarios

Caso de uso	CU2: Gestionar Inventarios
Proposito	Su propósito es registrar, modificar, listar y eliminar los inventarios
Descripcion	El sistema permitirá registro, modificación, listado y eliminación de los inventarios
Actor	Administrator
Actor iniciador	Administrator
Precondicion	Ningana
Flujo principal	5. Se envía por correo electrónico el comando correspondiente para gestionar un usuario. (“inventario_crear[]”, “inventario_todos[]”,”inventario_uno[]”, “inventario_editar[]”, “inventario_eliminar[]”). 6. El servidor lee el correo. 7. Se valida los datos 8. El servidor envia la respuesta
Excepcion	Comandos inválidos Parámetros incorrectos Fallo en la conexión con la base de datos

Tabla 3Tabla descripcion de caso de uso -CU2. Gestionar inventarios

7.3.2 CU3. Gestionar Productos.

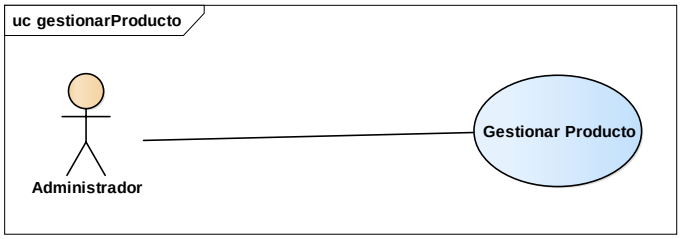


Figure 3 Diagrama de caso de uso -CU3.Getionar productos

Caso de uso	CU3: Gestionar productos
Proposito	Su propósito es registra, modificar, listar y eliminar los productos del sistema
Descripcion	El sistema permitirá registro, modificación, listado y eliminación de los productos
Actor	Administrador
Actor iniciador	Administrador
Precondicion	Ninguna
Flujo principal	9. Se envía por correo electrónico el comando correspondiente para gestionar un usuario. (“producto_crear[]”, “producto_todos[]”,”producto_uno[]”, “producto_editar[]”, “producto_eliminar[]”). 10. El servidor lee el correo. 11. Se valida los datos 12. El servidor envia la respuesta
Excepcion	Comandos invalidos Parametros incorrectos Fallo en la conexion con la base de datos

Tabla 4Tabla descripcion de caso de uso -CU3. Gestionar productos

7.3.3 CU4. Gestionar Servicios

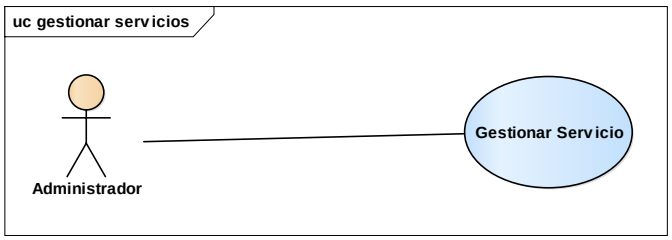


Figure 4Diagrama de caso de uso -CU4.Getionar servicios

Caso de uso	CU4: Gestionar Servicios
Proposito	Su propósito es registra, modificar, listar y eliminar los usuarios del sistema
Descripcion	El sistema permitirá registro, modificación, listado y eliminación de los usuarios
Actor	Administrator
Actor iniciador	Administrator
Precondicion	Ninguna
Flujo principal	13. Se envía por correo electrónico el comando correspondiente para gestionar un usuario. (“servicio_crear[]”, “servicio_todos[]”,”servicio_uno[]”, “servicio_editar[]”, “servicio_eliminar[]”). 14. El servidor lee el correo. 15. Se valida los datos 16. El servidor envia la respuesta
Excepcion	Comandos invalidos Parametros incorrectos Fallo en la conexion con la base de datos

Tabla 5. Tabla descripcion de caso de uso -CU4. Gestionar servicios

7.3.4 CU5. Gestionar Recetas.

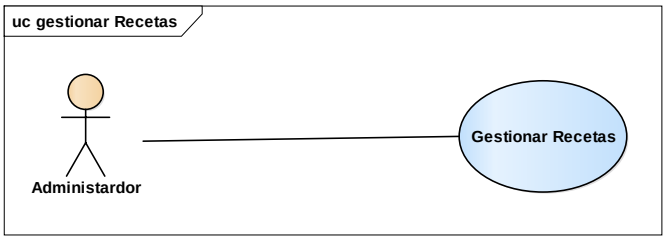


Figure 5Diagrama de caso de uso -CU5.Getionar recetas

Caso de uso	CU5: Gestionar Recetas
Proposito	Su propósito es registra, modificar, listar y eliminar los usuarios del sistema
Descripcion	El sistema permitirá registro, modificación, listado y eliminación de los usuarios
Actor	Administrator
Actor iniciador	Administrator
Precondicion	Ninguna
Flujo principal	17. Se envía por correo electrónico el comando correspondiente para gestionar un usuario. (“receta_crear[]”, “receta_todos[]”,”receta_uno[]”, “receta_editar[]”, “receta_eliminar[]”). 18. El servidor lee el correo. 19. Se valida los datos 20. El servidor envia la respuesta
Excepcion	Comandos invalidos Parametros incorrectos Fallo en la conexion con la base de datos

Tabla 6Tabla descripcion de caso de uso –CU5. Gestionar recetas

7.3.5 CU6. Gestionar Historial

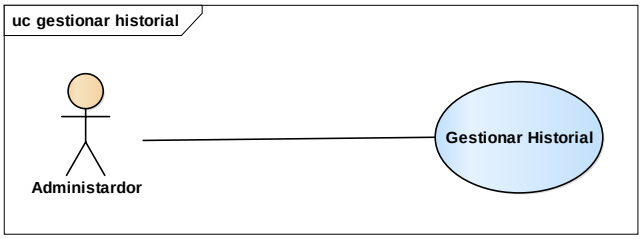


Figure 6Diagrama de caso de uso -CU6.Getionar historial

Caso de uso	CU6: Gestionar Historial
Proposito	Su proposito es registra, modificar, listar y eliminar los historiales del sistema
Descripcion	El sistema permitira registro, modificacion, listado y eliminacion de los usuarios
Actor	Administrador
Actor iniciador	Administrador
Precondicion	Ninguna
Flujo principal	21. Se envía por correo electrónico el comando correspondiente para gestionar un usuario. (“historal_crear[]”, “historial_todos[]”,”historial_uno[]”, “historial_editar[]”, “historial_eliminar[]”). 22. El servidor lee el correo. 23. Se valida los datos 24. El servidor envia la respuesta
Excepcion	Comandos invalidos Parametros incorrectos Fallo en la conexion con la base de datos

Tabla 7Tabla descripcion de caso de uso -CU6. Gestionar historial

7.3.6 CU7. Gestionar Pagos

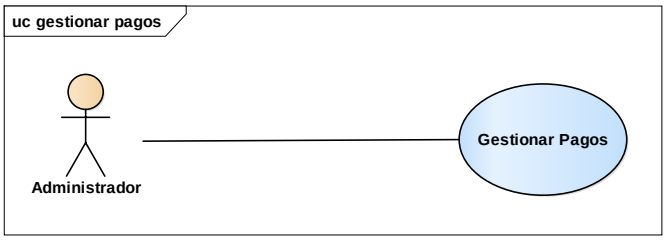


Figure 7 Diagrama de caso de uso -CU7.Getionar pagos

Caso de uso	CU1: Gestionar Usuarios
Proposito	Su propósito es registra, modificar, listar y eliminar los usuarios del sistema
Descripcion	El sistema permitirá registro, modificación, listado y eliminación de los usuarios
Actor	Administrador
Actor iniciador	Administrador
Precondicion	Ninguna
Flujo principal	25. Se envía por correo electrónico el comando correspondiente para gestionar un usuario. (“pago_crear[]”, “pago_todos[]”,pago_uno[]”, “pago_editar[]”, “pago_eliminar[]”). 26. El servidor lee el correo. 27. Se valida los datos 28. El servidor envia la respuesta
Excepcion	Comandos inválidos Parámetros incorrectos Fallo en la conexión con la base de datos

Tabla 8 Tabla descripción de caso de uso -CU7. Gestionar pagos

7.3.7 CU8. Gestionar Reportes y Estadísticas

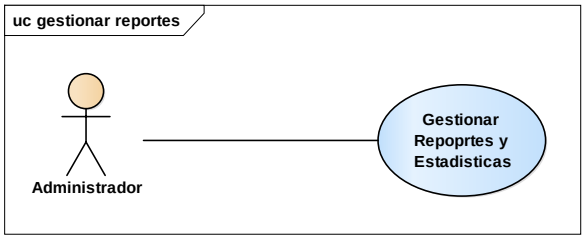


Figure 8Diagrama de caso de uso -CU8.Getionar reportes y estadistica

Caso de uso	CU1: Gestionar Usuarios
Proposito	Su propósito es registra, modificar, listar y eliminar los usuarios del sistema
Descripcion	El sistema permitirá registro, modificación, listado y eliminación de los usuarios
Actor	Administrador
Actor iniciador	Administrador
Precondicion	Ninguna
Flujo principal	29. Se envía por correo electrónico el comando correspondiente para gestionar un usuario. (“reporte_crear[]”, “reporte_todos[]”,”reporte_uno[]”, “reporte_editar[]”, “reporte_eliminar[]”). 30. El servidor lee el correo. 31. Se valida los datos 32. El servidor envia la respuesta
Excepcion	Comandos inválidos Parámetros incorrectos Fallo en la conexión con la base de datos

Tabla 9 Tabla descripcion de caso de uso -CU8. Gestionar reportes

7.4 Estructuración de modelo de casos de uso

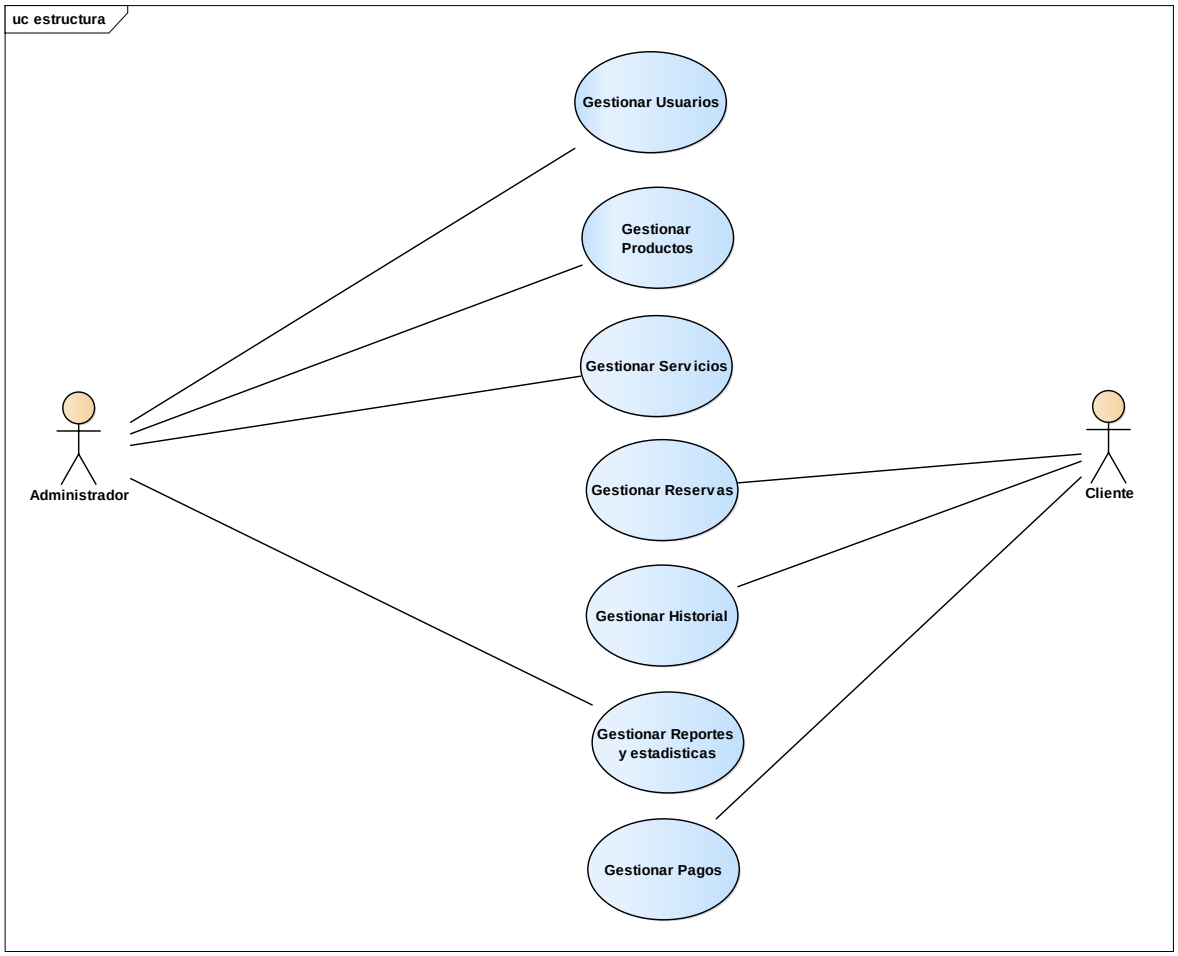


Tabla 10 Diagrama General de Casos de Usos

7.5 Análisis

7.5.1 Diagramas de comunicación

7.5.1.1 CU1. Gestión de Usuarios

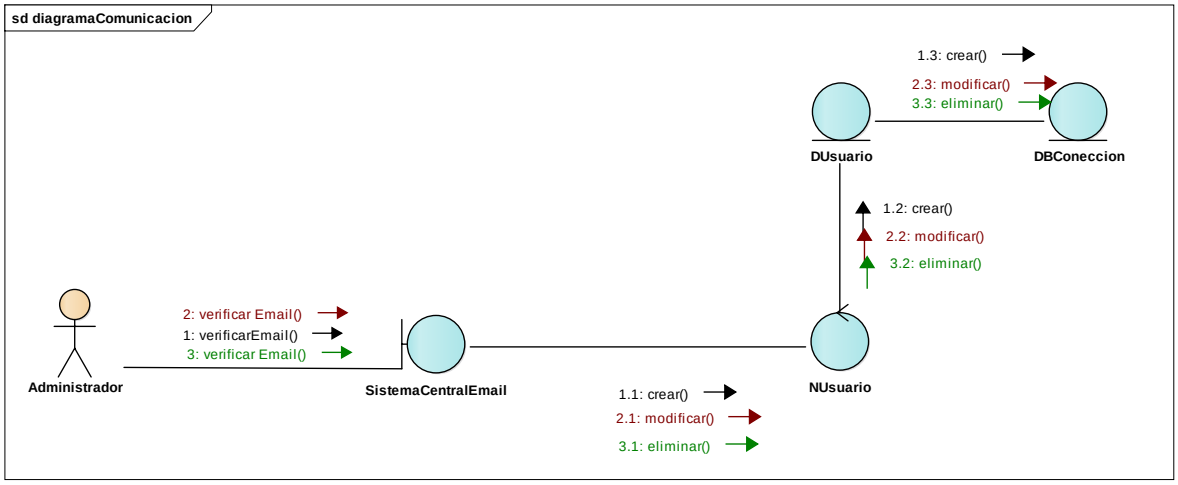


Figure 9. Diagrama de comunicación CU 1

7.5.1.2 CU2. Gestionar Inventarios

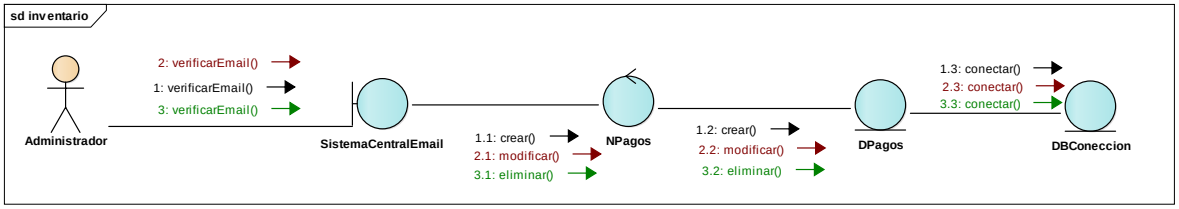


Figure 10. Diagrama de comunicación CU2

7.5.1.3 CU3. Gestionar Productos.

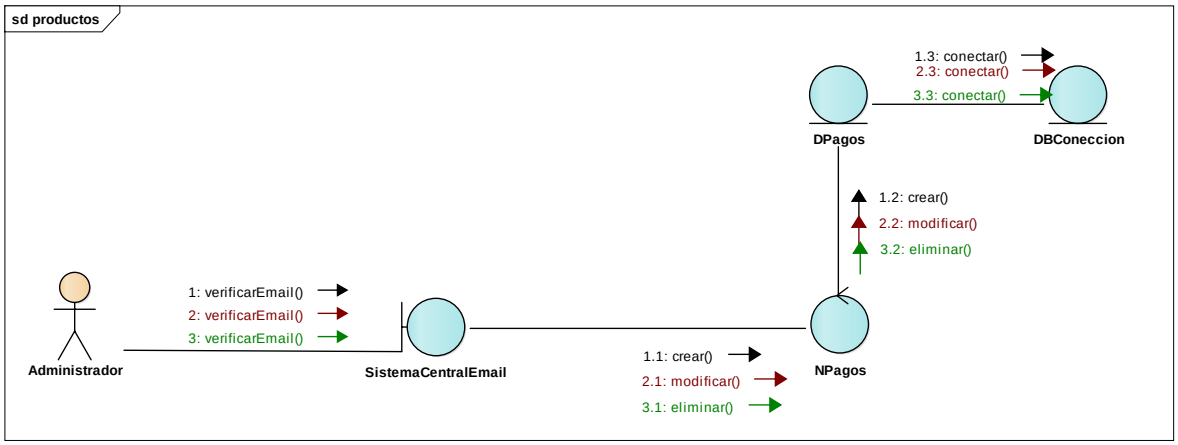


Figure 11. Diagrama de comunicación CU3

7.5.1.4 CU4. Gestionar Servicios

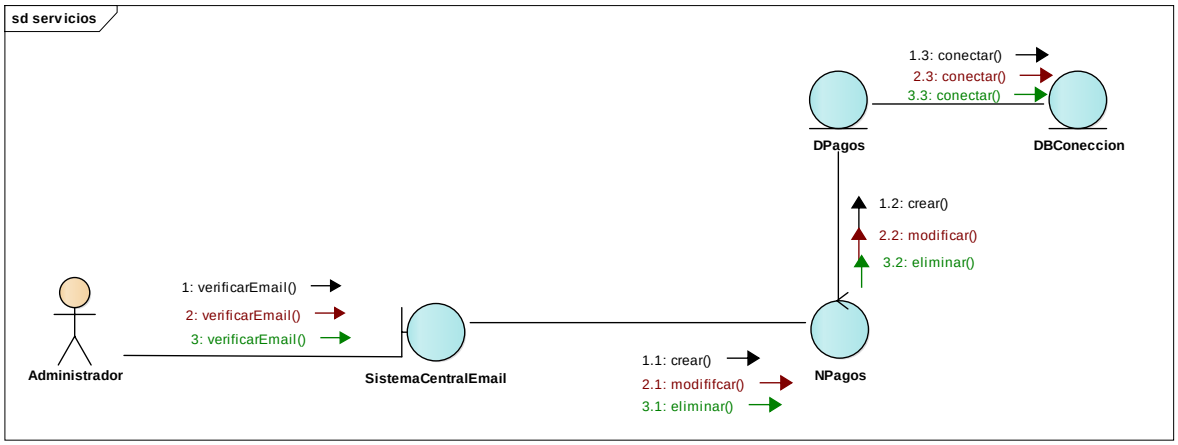


Figure 12. Diagrama de comunicación CU4

7.5.1.5 CU5. Gestionar Recetas.

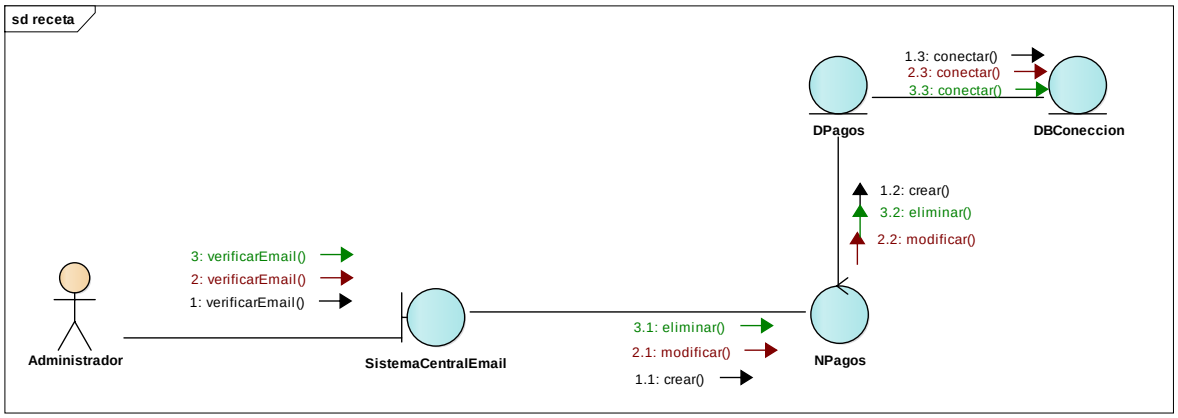
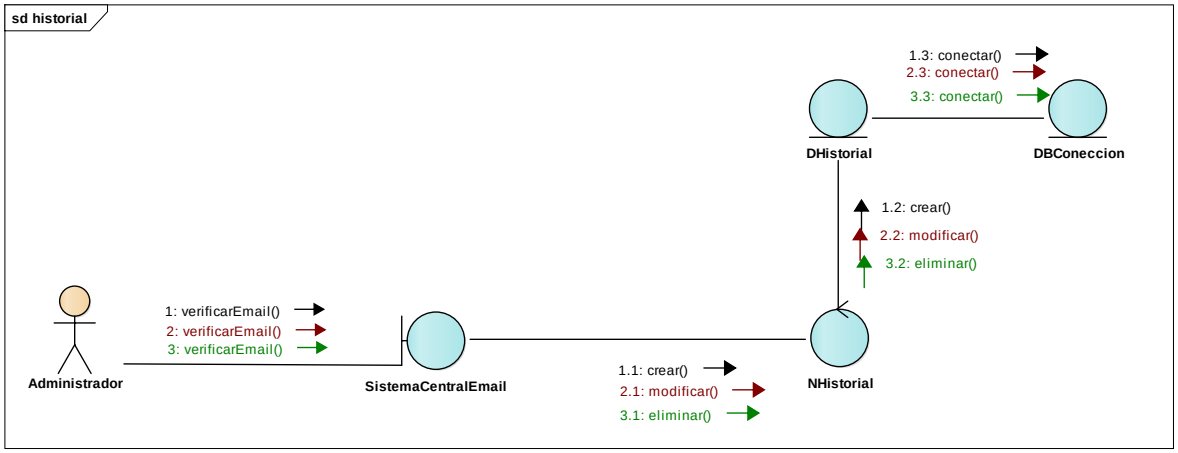


Figure 13. Diagrama de comunicación CU5

7.5.1.6 CU6. Gestionar Historial



7.5.1.7 CU7. Gestionar Pagos

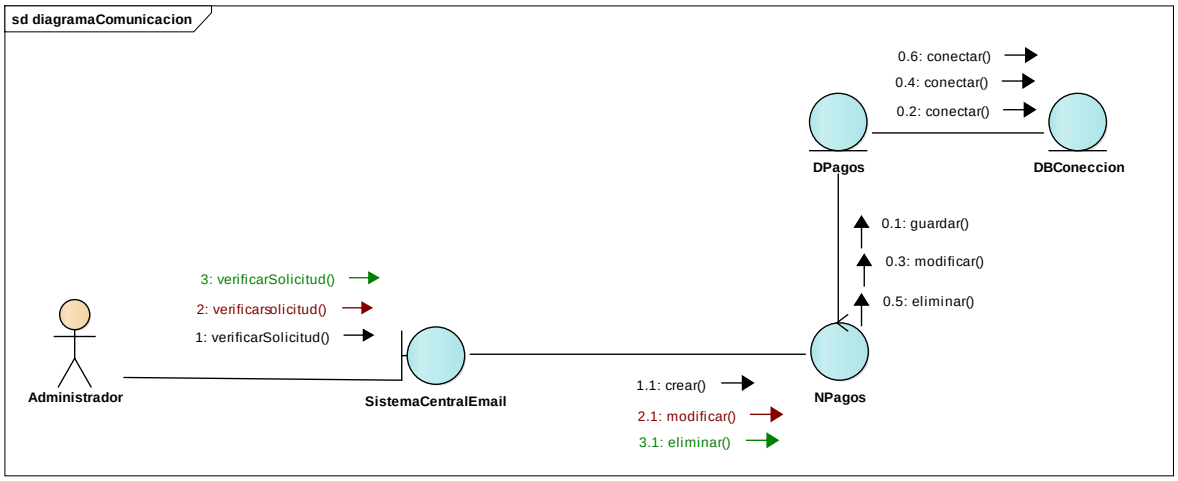


Figure 14. Diagrama de comunicación CU7

7.5.2 Gestionar Reportes y Estadísticas

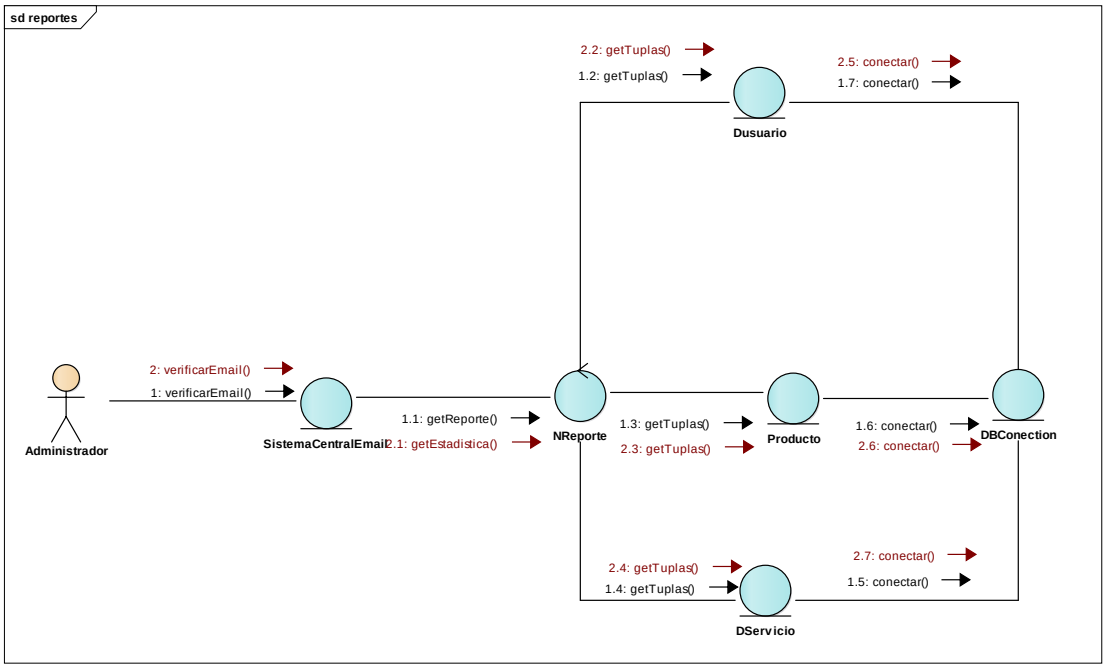


Figure 15. Diagrama de comunicación CU8

7.6 Diseño

7.6.1 Diseño de la arquitectura

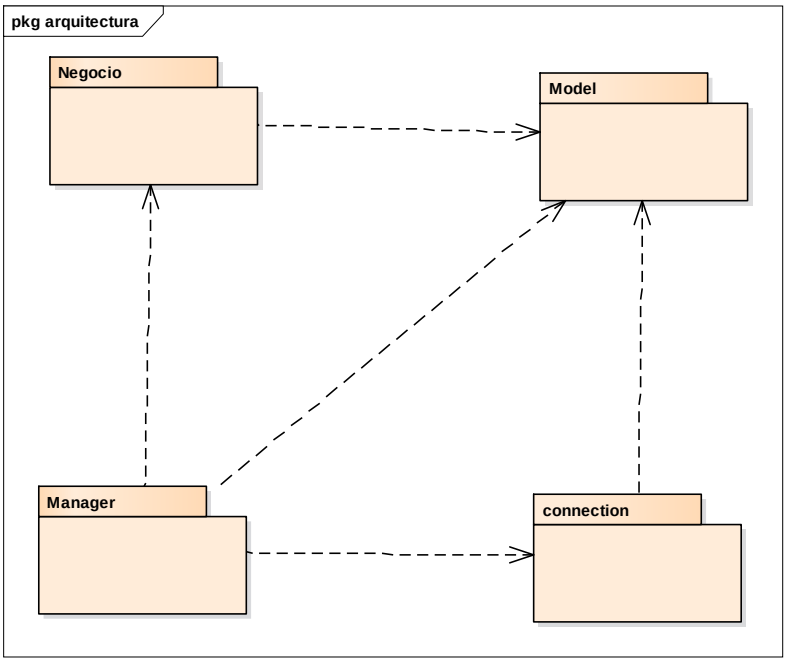
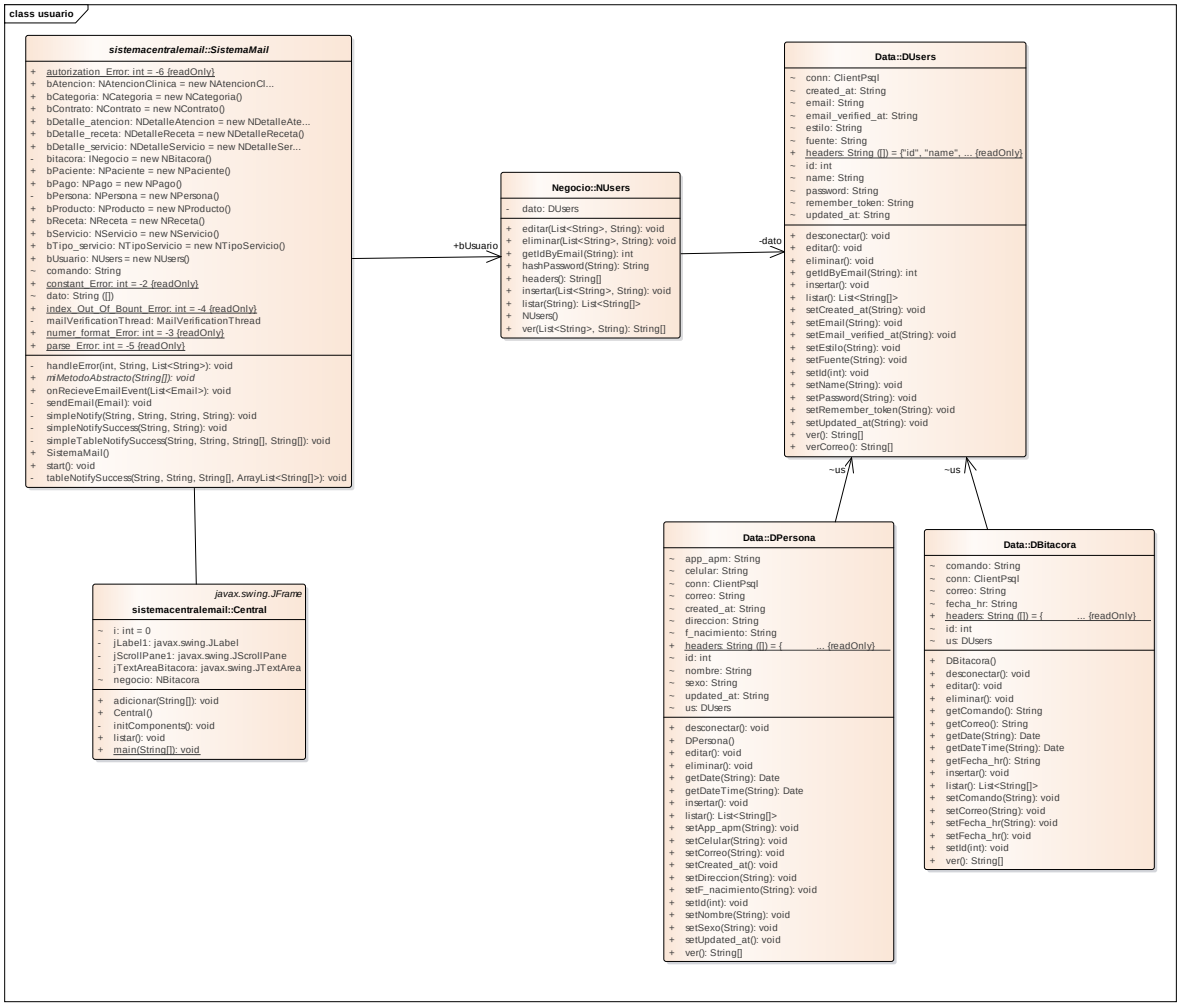


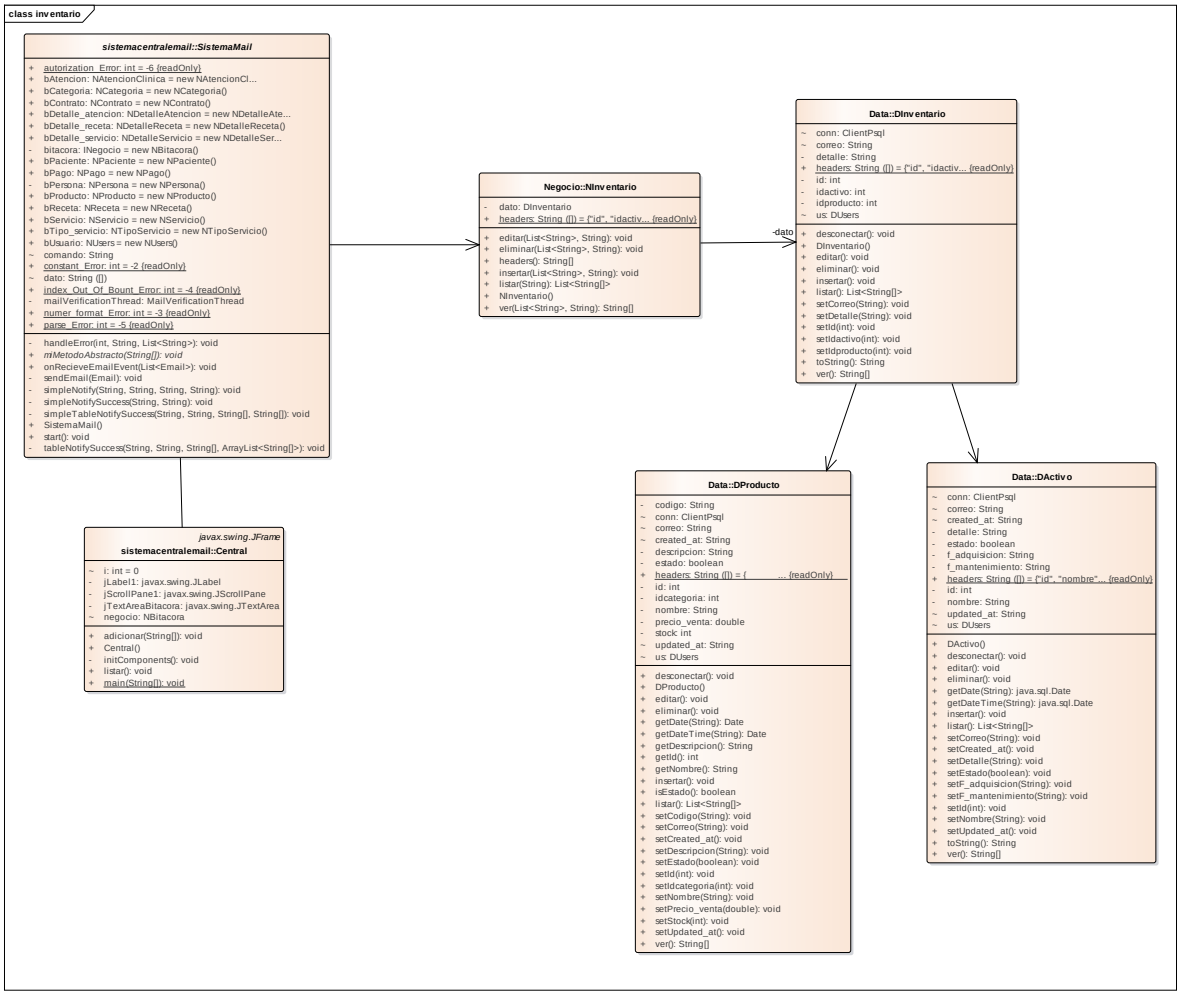
Figure 16. Diagrama de Paquetes

7.7 Diagramas de clases dinámico

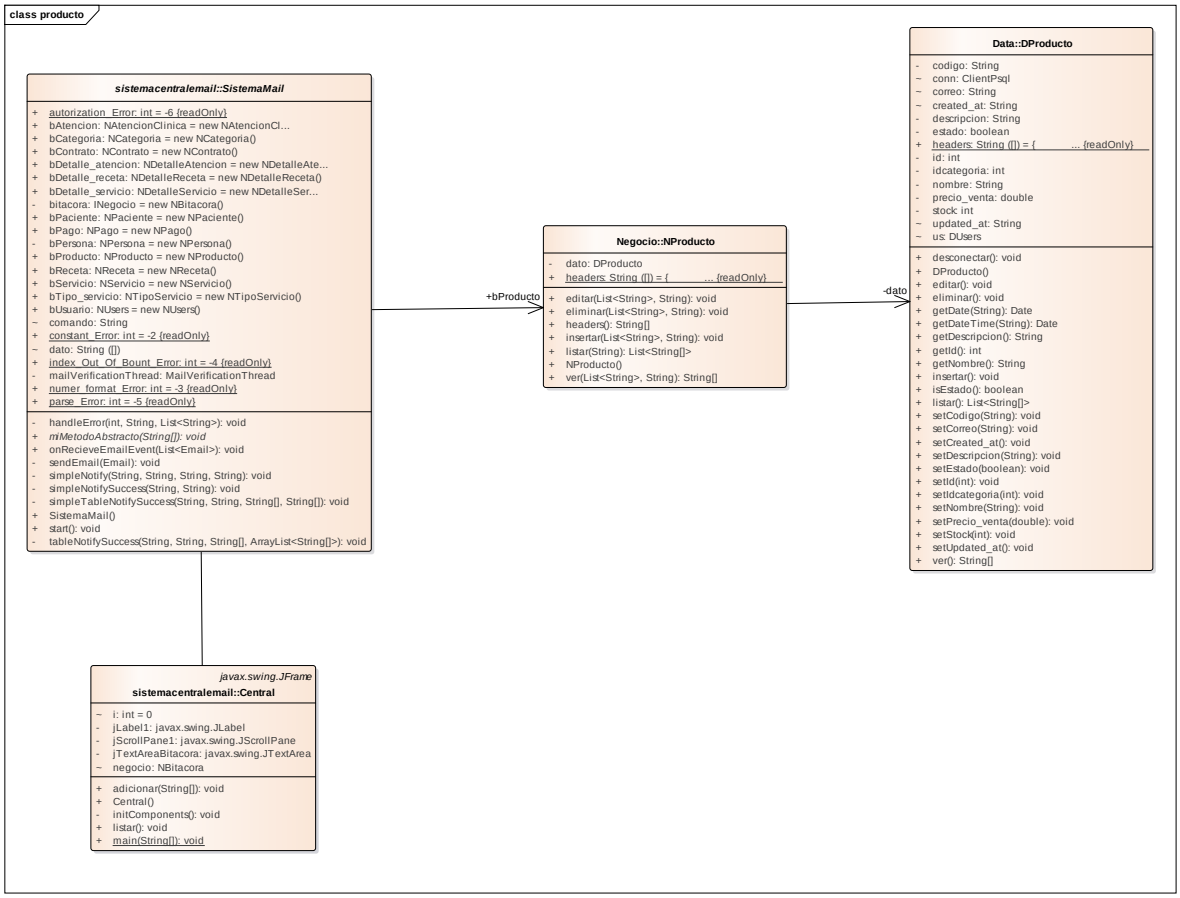
7.7.1 CU1. Gestión de Usuarios



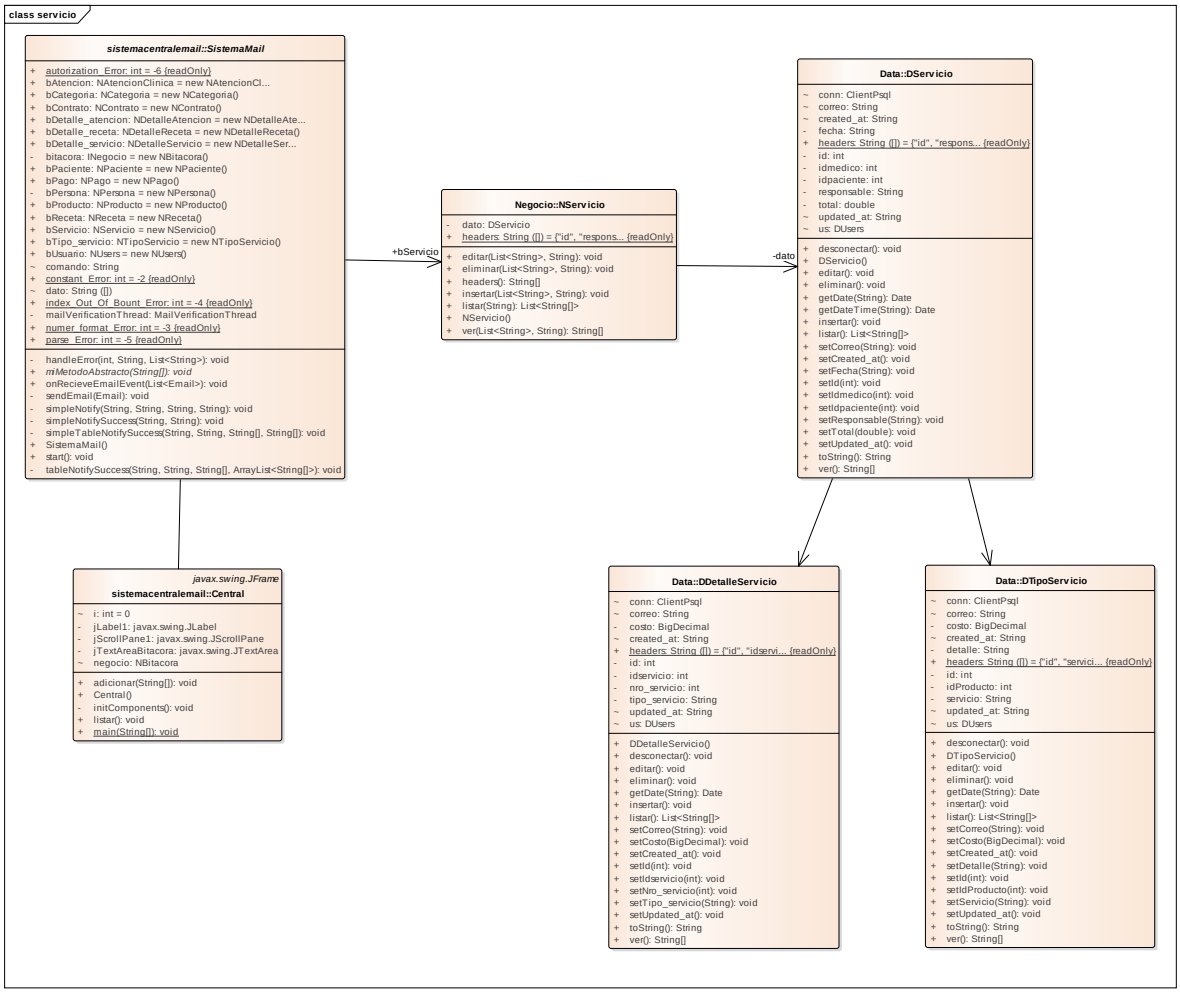
7.7.2 CU2. Gestionar Inventarios



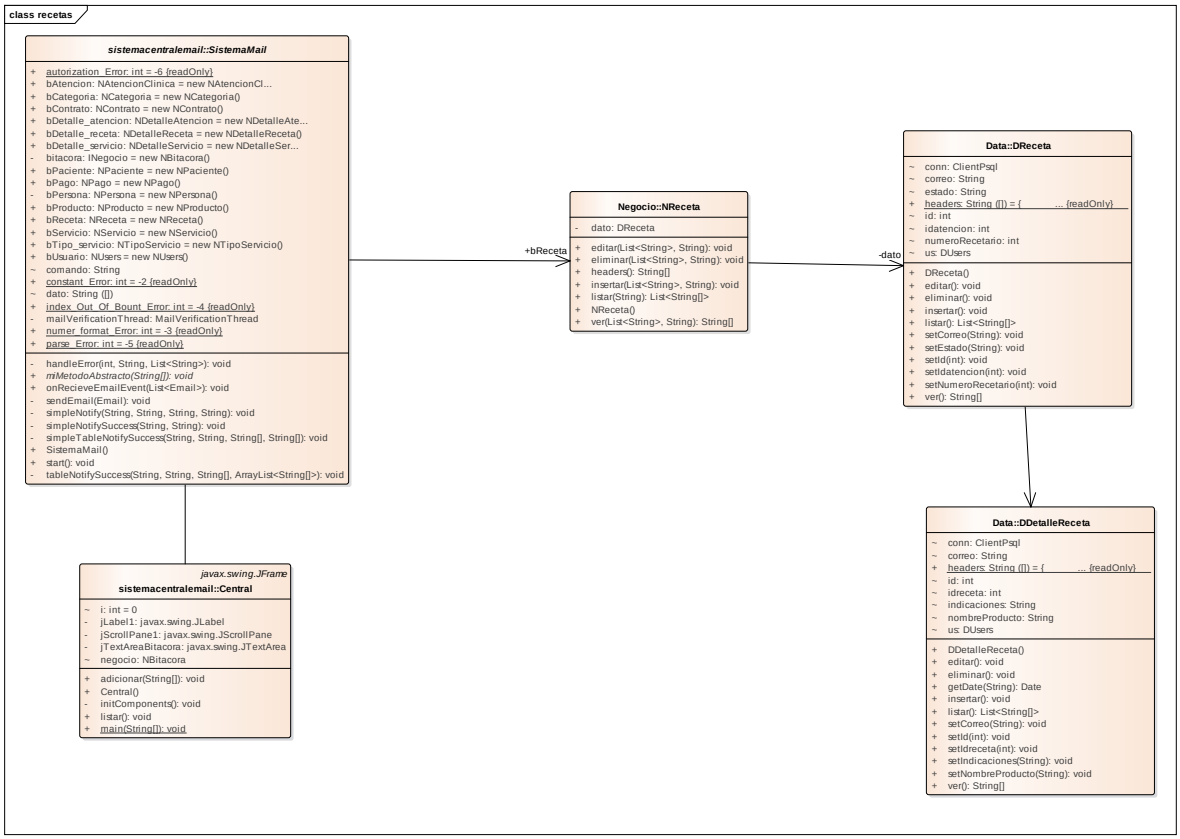
7.7.3 CU3. Gestionar Productos.



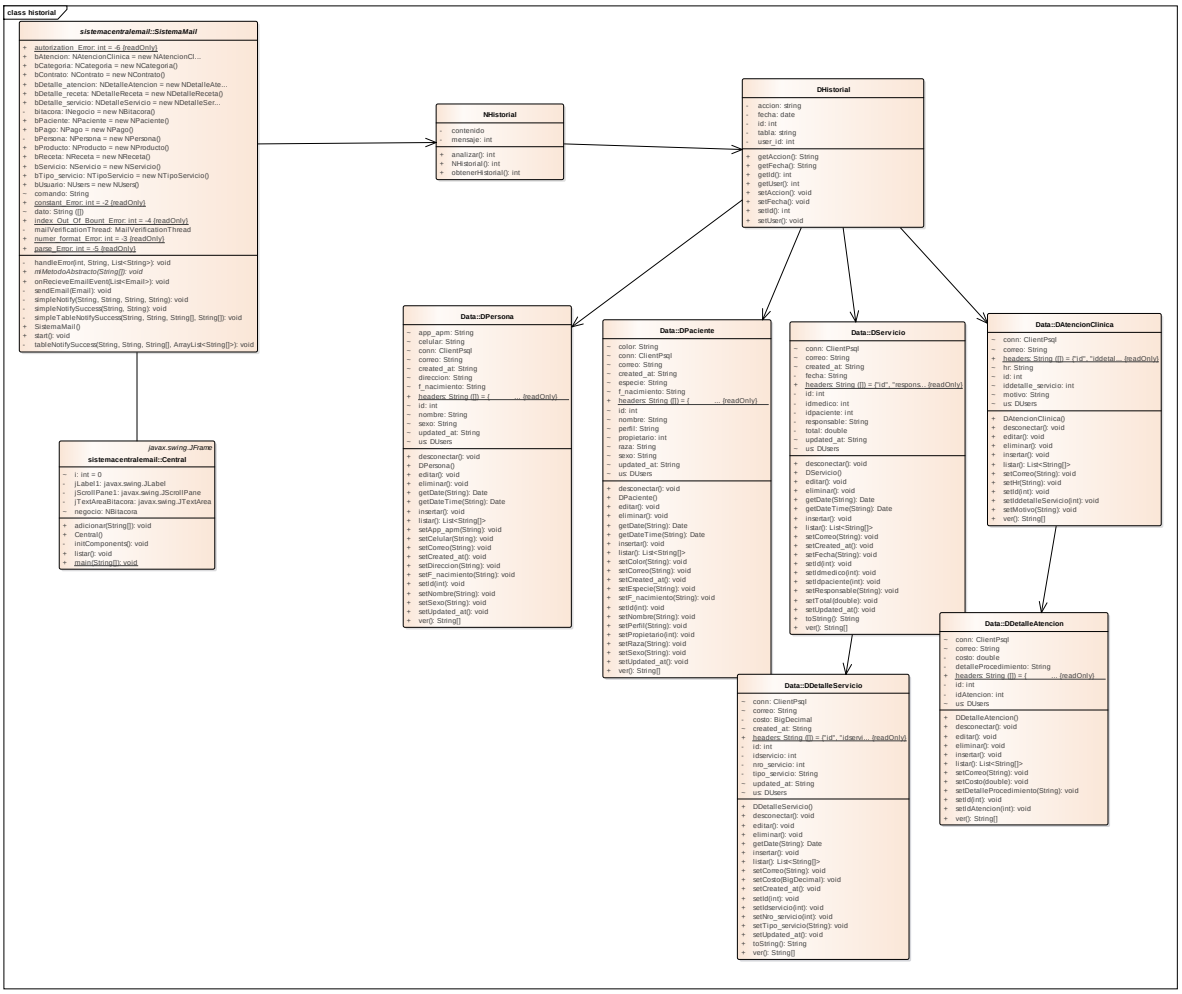
7.7.4 CU4. Gestionar Servicios



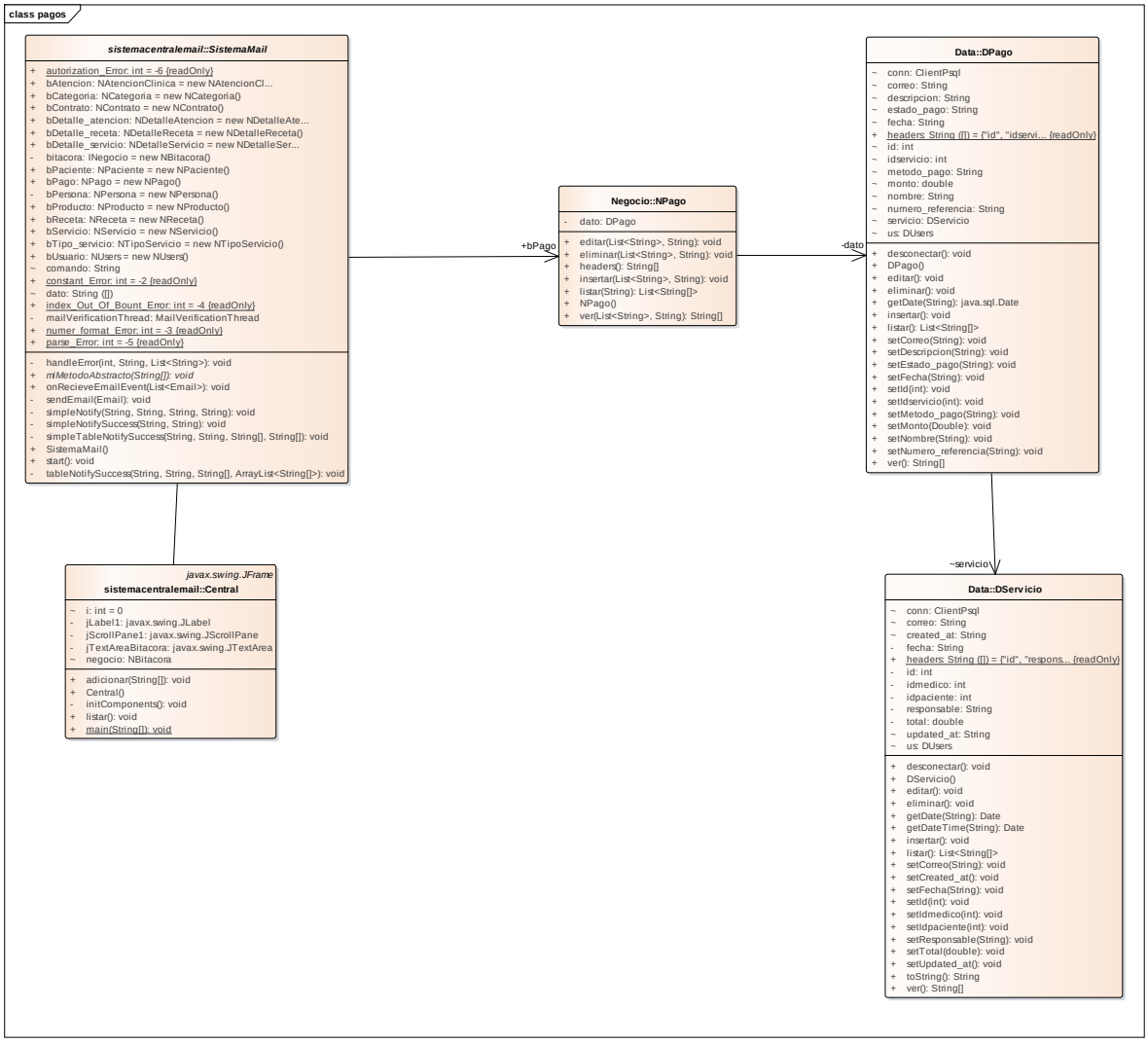
7.7.5 CU5. Gestionar Recetas.



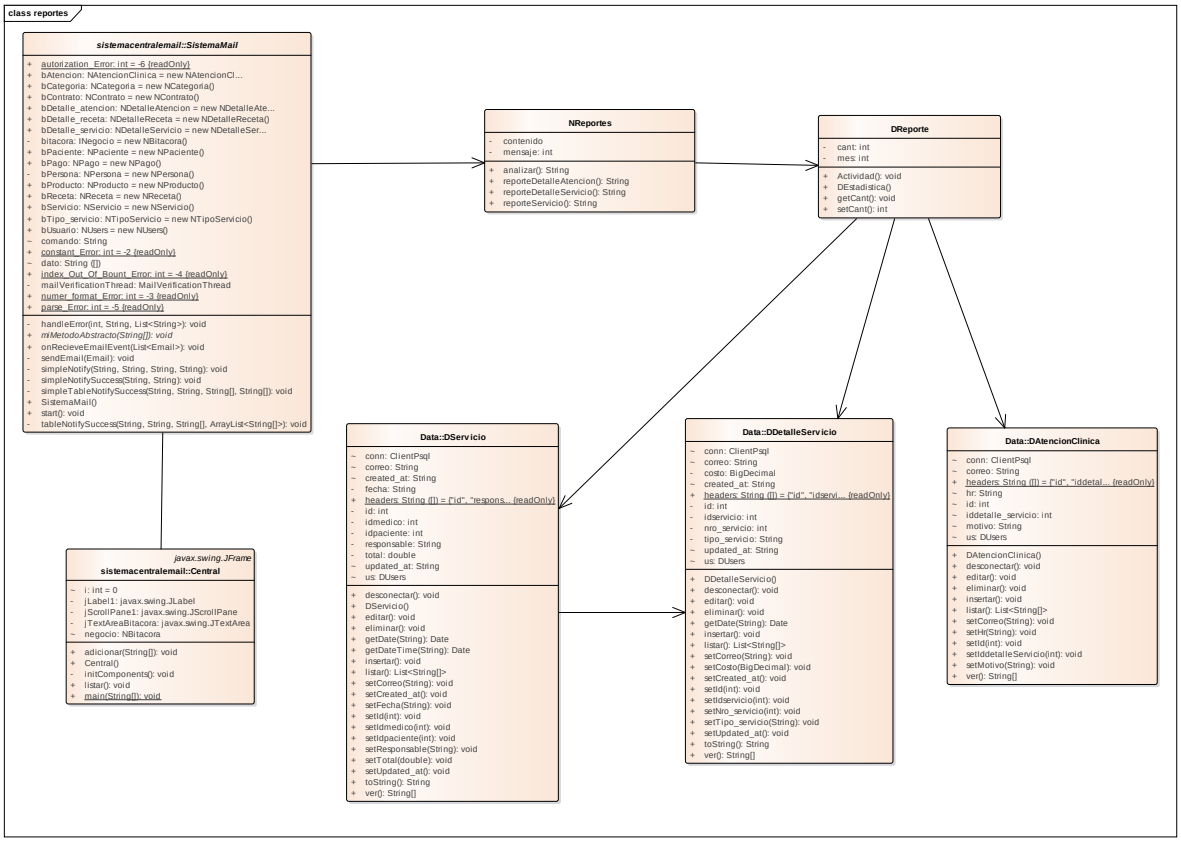
7.7.6 CU6. Gestionar Historial



7.7.7 CU7. Gestionar Pagos



7.7.8 CU8. Gestionar Reportes y Estadísticas



7.8 Diagramas de secuencia

7.8.1 CU1. Gestión de Usuarios

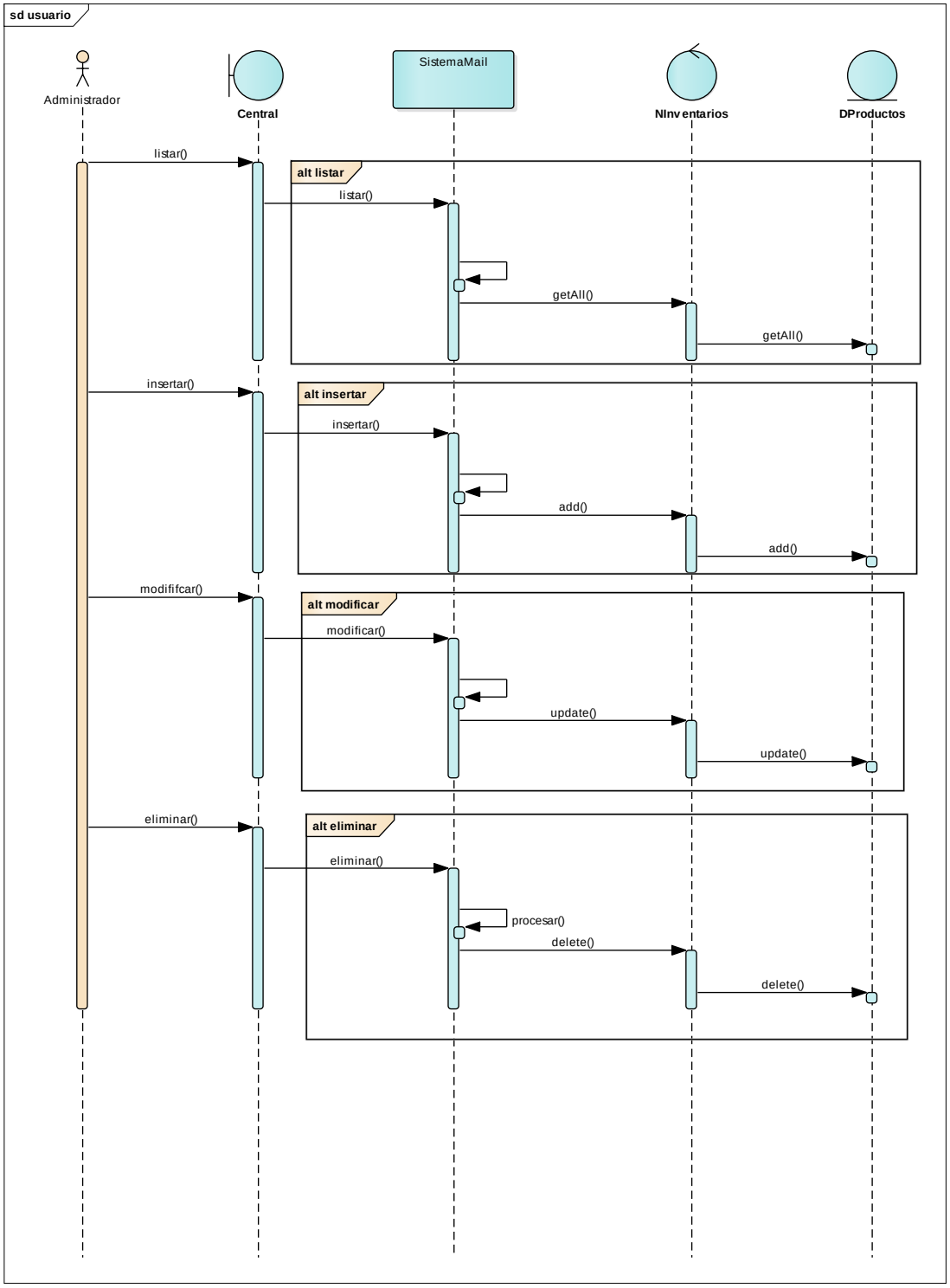


Figure 17 Diagrama de secuencia CU1

7.8.2 CU2. Gestionar Inventarios

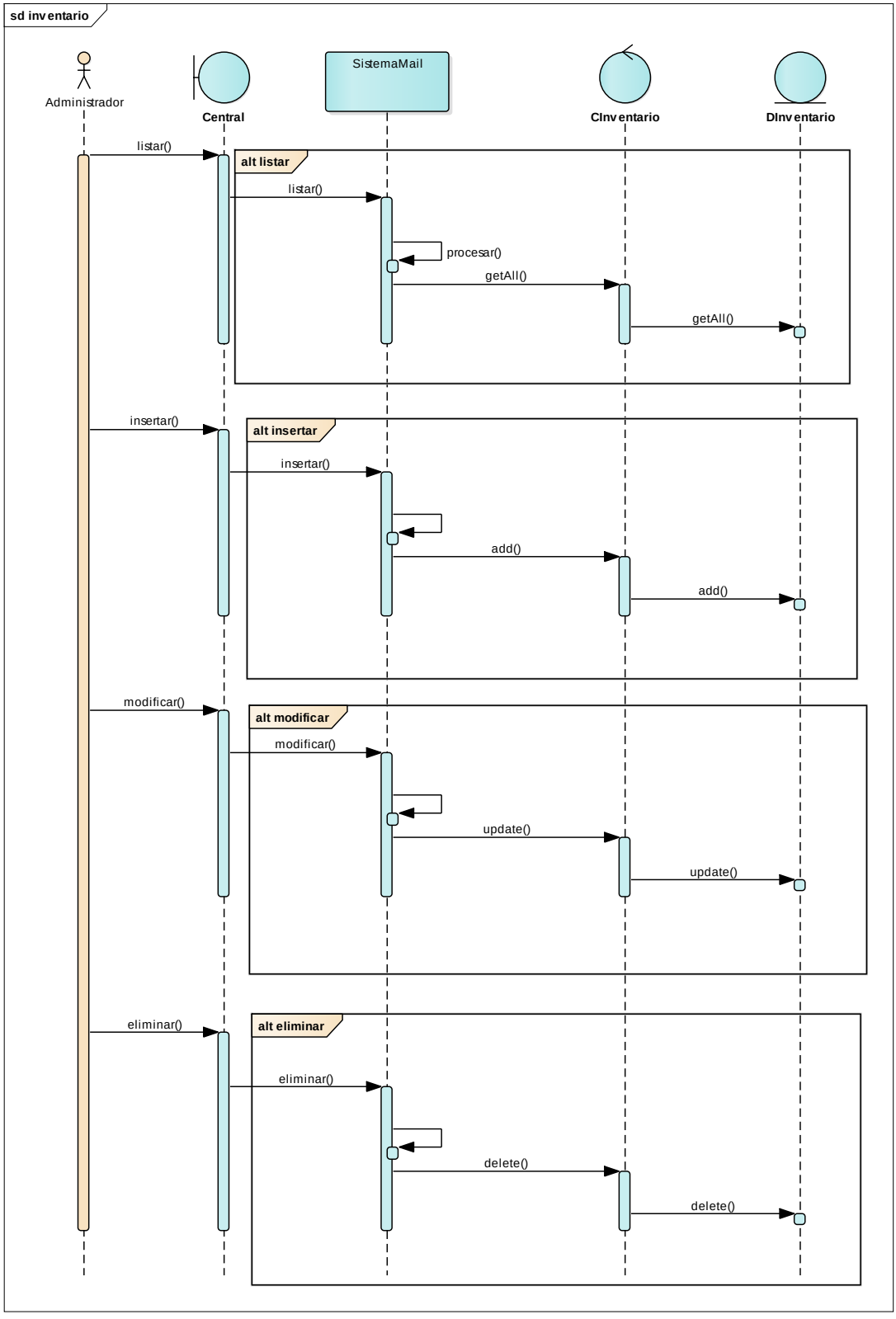


Figure 18 Diagrama de secuencia CU2

7.8.3 CU3. Gestionar Productos.

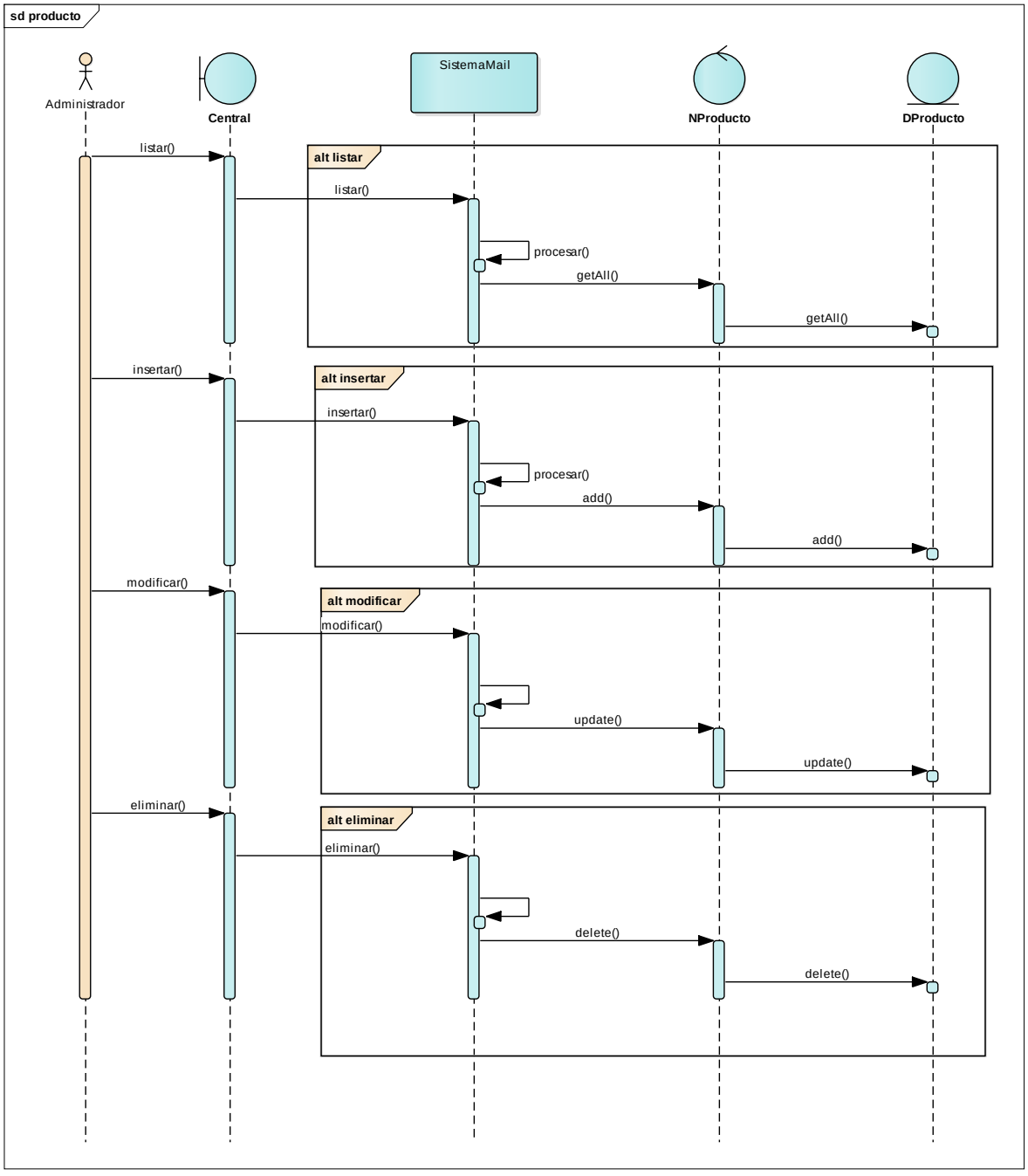


Figure 19 Diagrama de secuencia CU3

7.8.4 CU4. Gestionar Servicios

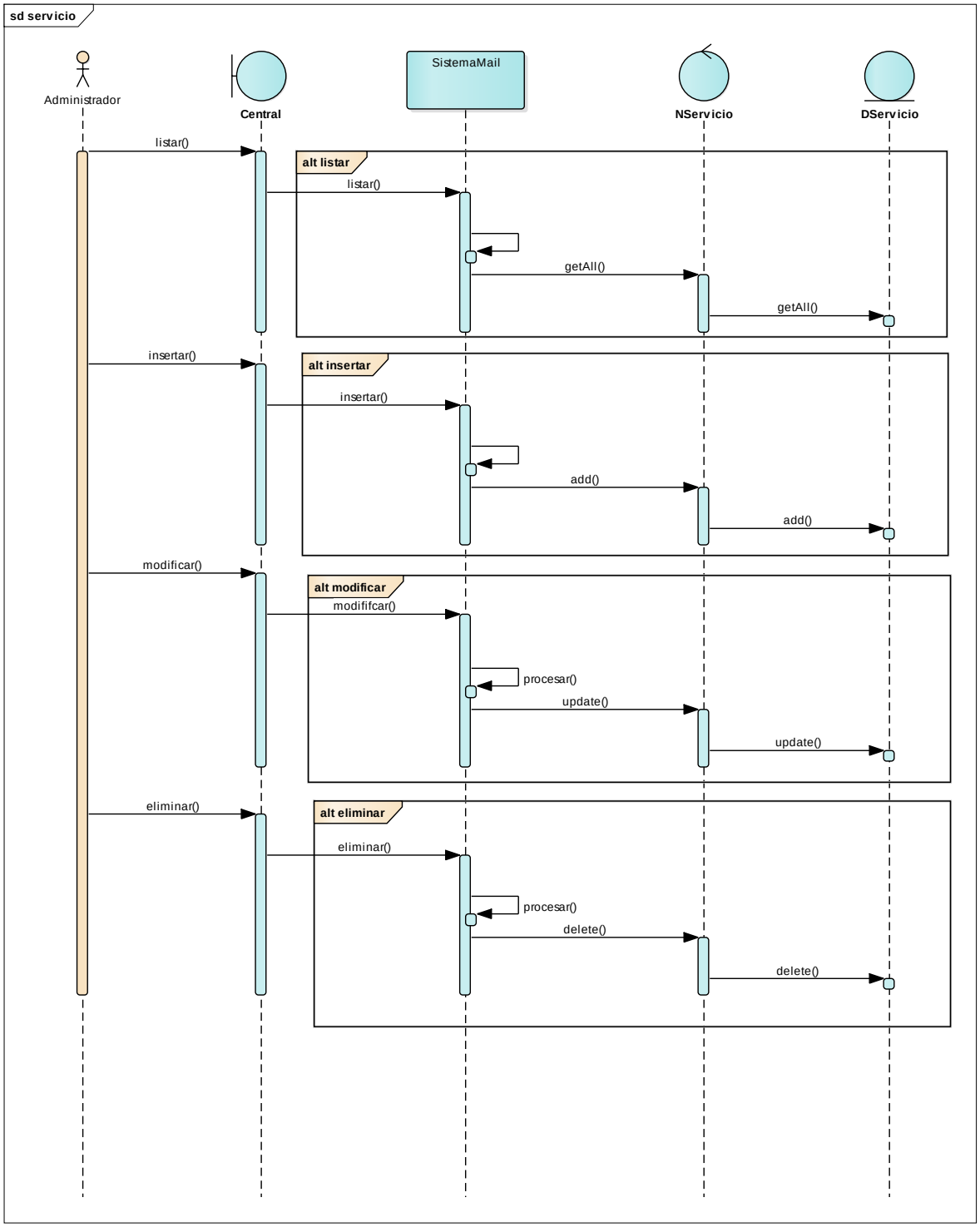


Figure 20 Diagrama de secuencia CU4

7.8.5 CU5. Gestionar Recetas.

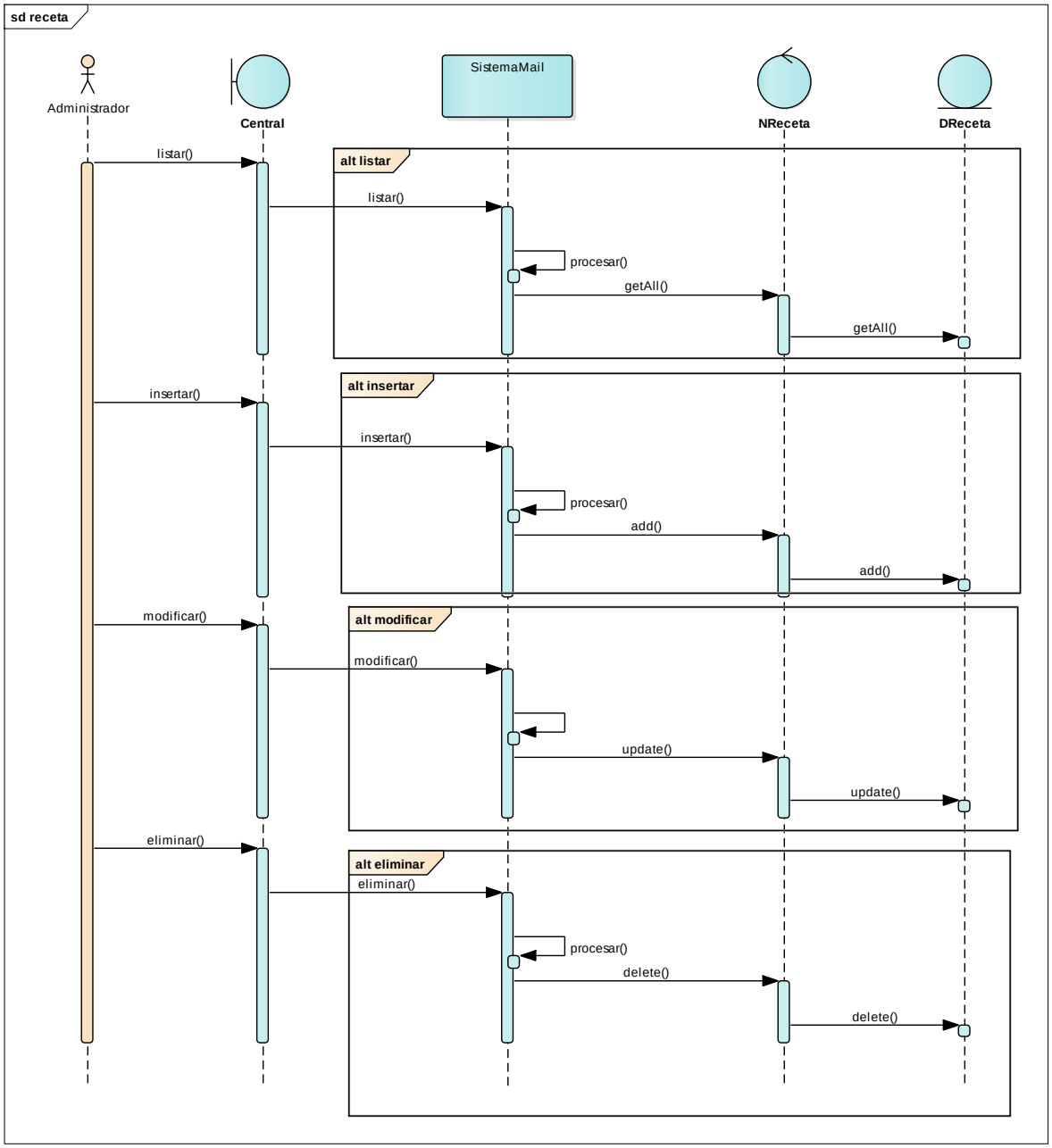
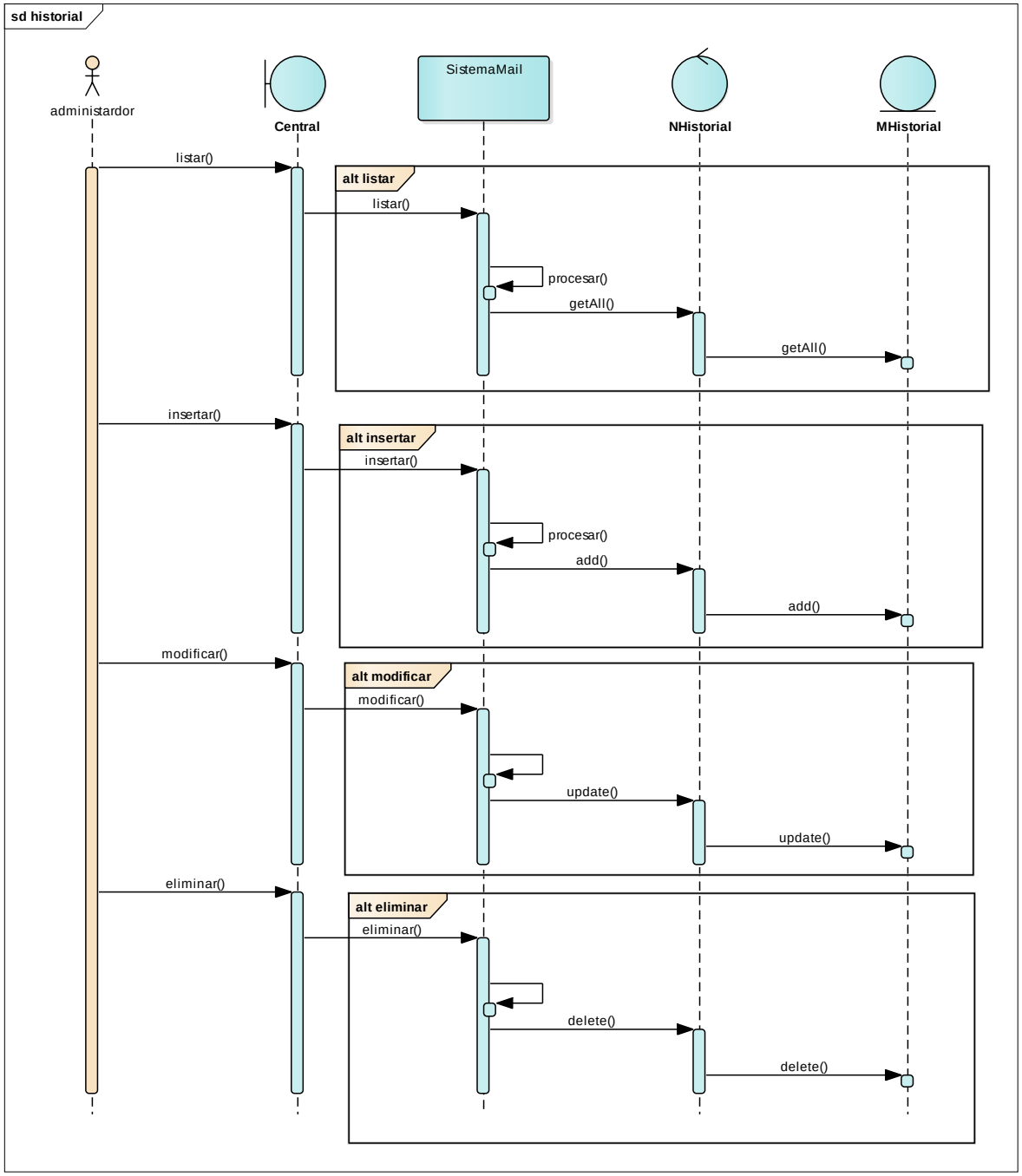


Figure 21 Diagrama de secuencia CU5

7.8.6 CU6. Gestionar Historial



7.8.7 CU7. Gestionar Pagos

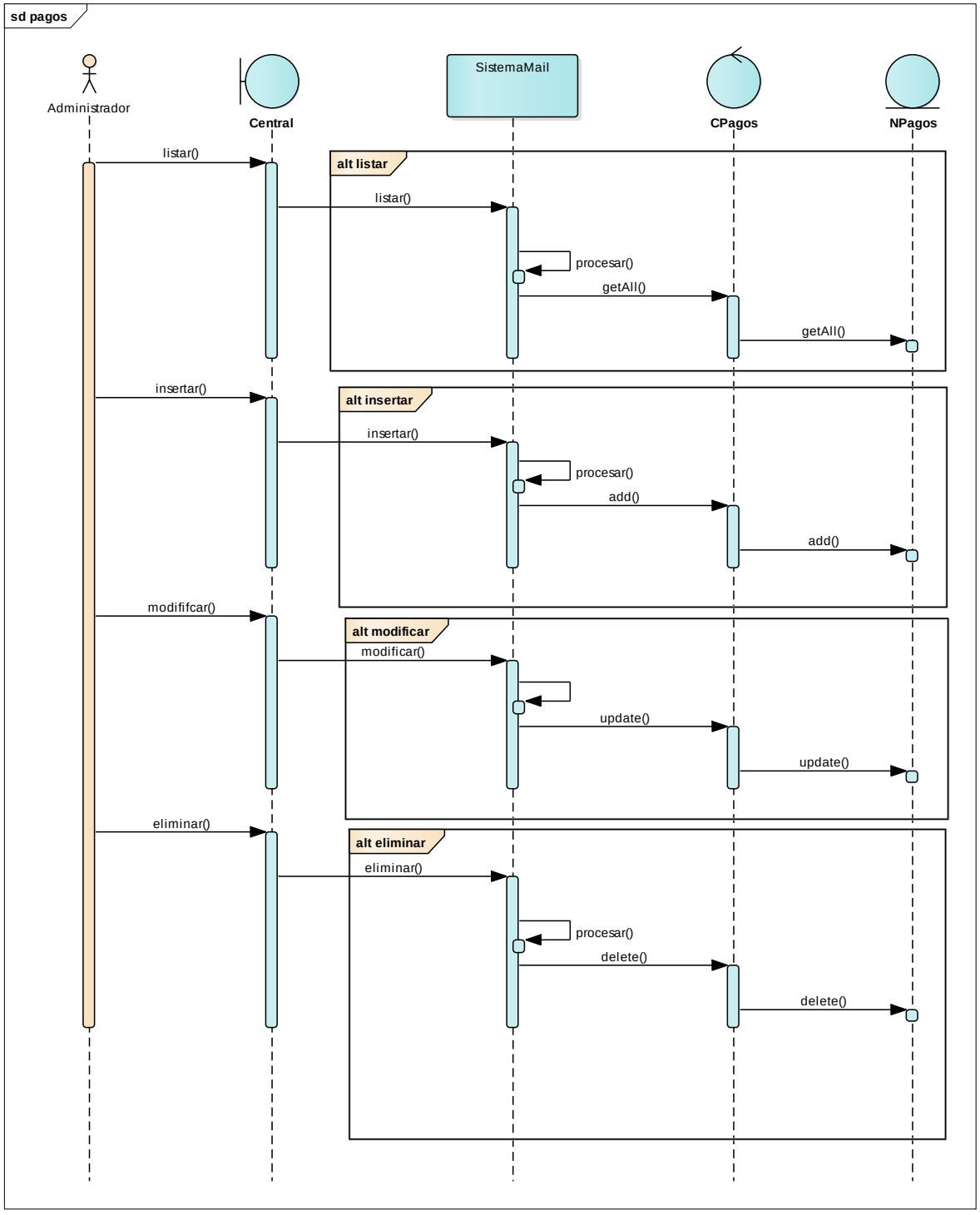


Figure 22 Diagrama de secuencia CU7

7.8.8 CU8. Gestionar Reportes y Estadísticas

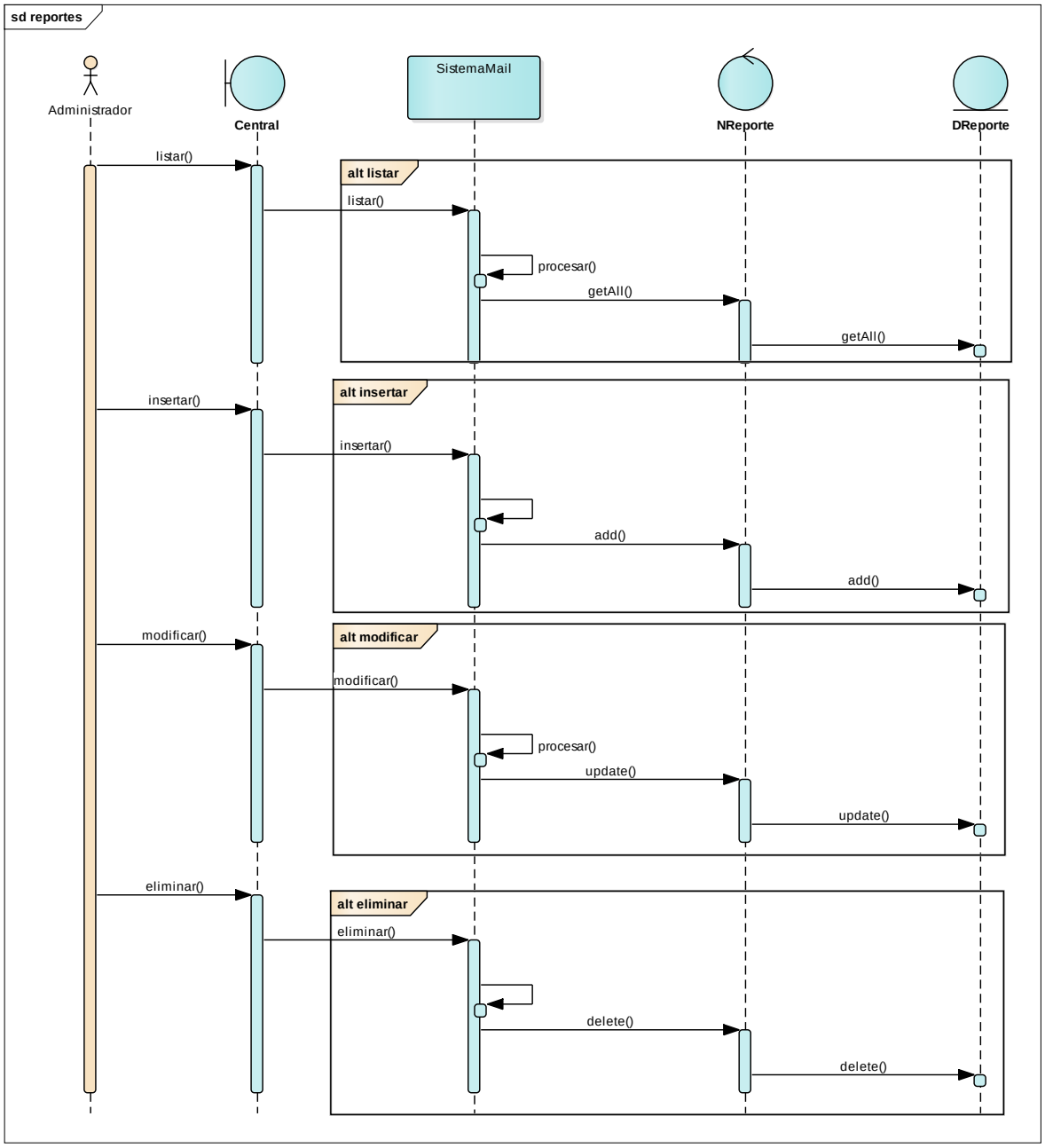
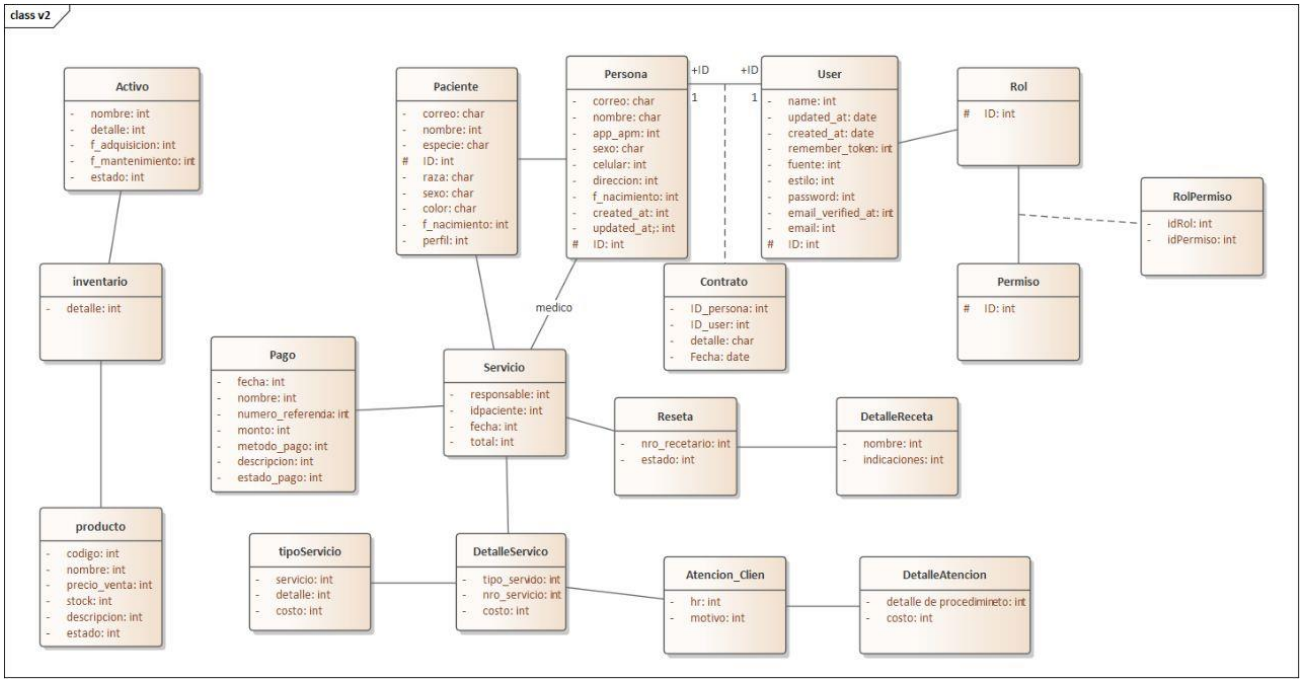


Figure 23 Diagrama de secuencia CU18

7.9 Diseño de la base de datos

7.9.1 Diseño conceptual



7.9.2 Diseño lógico

■ Llave primaria ■ Llave foránea

Pacientes

id	color	correo	especie	F_nacimiento	nombre	perfil	Raza	sexo

Persona

id	celular	correo	direccion	F_nacimiento	nombre	sexo

Contrato

id	detalle	fecha	Id_persona	Id_user

Usuario

id	email	estilo	fuente	name	password

Rol Permiso

id	Id_rol	Id_Permiso

Activo

id	nombre	detalle	F_adquisicion	F_mantenimiento

Inventario

id	detalle

Producto

id	codigo	nombre	Precio_venta	stock	descripcion	estado

Pago

id	fecha	nombre	Numero_referencia	monto	Método_pago	descripcion	Estado_pago

Tipo servicio

id	servicio	detalle	costo

Servicio

id	responsable	**Id_paciente**	Fecha	total

Detalle servicio

id	Tipo_servicio	Nro_servicio	costo

Receta

id	Nro_recetario	estado

Atención cliente

id	hora	motivo

Detalle receta

id	nombre	indicaciones

Detalle atención

id	Detalle_procedimiento	costo

7.9.3 Diseño físico

7.9.4 Script de la base de datos

```
CREATE DATABASE veterinaria DEFAULT CHARACTER SET utf8;

USE tecno;

CREATE TABLE permisos (

 id int PRIMARY KEY AUTO_INCREMENT,

 nombre char(150) NOT NULL

);

CREATE TABLE roles (

 id int PRIMARY KEY AUTO_INCREMENT,

 nombre char(150) NOT NULL,

 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

CREATE TABLE rol_permisos (

 permiso_id int NOT NULL,

 FOREIGN KEY(permiso_id) REFERENCES permisos(id) ON UPDATE
CASCADE ON DELETE CASCADE,
```

```
 rol_id int NOT NULL,

 FOREIGN KEY(rol_id) REFERENCES roles(id) ON UPDATE CASCADE ON
DELETE CASCADE

);

CREATE TABLE personas (

 id int PRIMARY KEY AUTO_INCREMENT,

 nombres char(150) NOT NULL,

 apellidos char(150) NOT NULL,

 sexo char(150) NOT NULL,

 direccion char(150) NOT NULL,

 email char(17) NOT NULL,

 telefono integer NOT NULL,

 f_Nacimiento Date NOT NULL,

 estado boolean DEFAULT TRUE,

 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

CREATE TABLE Activo (

 id int PRIMARY KEY AUTO_INCREMENT,

 nombre char(150) NOT NULL,
```

```
 detalle char(150) NOT NULL,

 f_adquisicion Date NULL,

 f_mantenimiento Date NULL,

 estado Boolean char(150) NOT NULL,

 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,



);

CREATE TABLE users (

 id int PRIMARY KEY AUTO_INCREMENT,

 nombre char(150) NOT NULL,

 email char(150) NOT NULL UNIQUE,

 pass char(150) NOT NULL,

 estado boolean DEFAULT TRUE,

 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 rol_id int NOT NULL,

 FOREIGN KEY(rol_id) REFERENCES roles(id) ON UPDATE CASCADE ON
DELETE CASCADE,

 persona_id int NOT NULL,
```

```
 FOREIGN KEY(persona_id) REFERENCES personas(id) ON UPDATE
CASCADE ON DELETE CASCADE

);

CREATE TABLE Inventario (

 id int PRIMARY KEY AUTO_INCREMENT,

 detalle char(150) NOT NULL,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 empresa_id int NOT NULL,

 FOREIGN KEY(activo_id) REFERENCES activo(id) ON UPDATE
CASCADE ON DELETE CASCADE

);

CREATE TABLE producto (

 id int PRIMARY KEY AUTO_INCREMENT,

 nombre char(150) NOT NULL,

 precion_venta integer NOT NULL,

 stock integer NOT NULL,

 descripcion char(150) NOT NULL,

 estado char(150) NOT NULL,



);
```

```
CREATE TABLE pago (

 id int PRIMARY KEY AUTO_INCREMENT,

 nombre char(150) NOT NULL,

 numero_referencia int NOT NULL,

 monto int NOT NULL,

 fecha  DATE NOT NULL,

 metodo_pago char(150) NOT NULL,

 descripcion char(150) NOT NULL,

 estado_pago char(150) NOT NULL,

hora_programada TIME NOT NULL,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 servicio_id int NOT NULL,

 FOREIGN KEY(servicio_id) REFERENCES servicio(id) ON UPDATE
CASCADE ON DELETE CASCADE,

);
```

```
CREATE TABLE servicio (

 id int PRIMARY KEY AUTO_INCREMENT,

 responsable char(150) NOT NULL,

 paciente_id int NOT NULL,

 FOREIGN KEY(paciente_id) REFERENCES paciente(id) ON UPDATE
CASCADE ON DELETE CASCADE,

 total int NOT NULL,

 fecha  DATE NOT NULL,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

CREATE TABLE Receta (

 id int PRIMARY KEY AUTO_INCREMENT,

 numero_recetario int NOT NULL,

 estado char(150) NOT NULL,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

FOREIGN KEY(detalleReceta_id) REFERENCES detalleReceta(id) ON
UPDATE CASCADE ON DELETE CASCADE,

 detalleReceta_id int NOT NULL,
```

```
 FOREIGN KEY(solicitud_id) REFERENCES solicitudes(id) ON
UPDATE CASCADE ON DELETE CASCADE,

 user_id int NOT NULL,

 FOREIGN KEY(user_id) REFERENCES users(id) ON UPDATE CASCADE
ON DELETE CASCADE

);

CREATE TABLE detalleAtencion (

 id int PRIMARY KEY AUTO_INCREMENT,

 detalleProcedimental char(150) NOT NULL,

 costo int NOT NULL,

 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

);

CREATE TABLE atencionClinica (

 id int PRIMARY KEY AUTO_INCREMENT,

 nombre char(150) NOT NULL,

 motivo char(150) NOT NULL,

 hora int NOT NULL,

 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 detalleAtencion_id int NOT NULL,
```

```
 FOREIGN          KEY(detalleAtencion_id)          REFERENCES
detalleAtencion(id) ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE bitacoras (

 id int PRIMARY KEY AUTO_INCREMENT,

 accion char(150) NOT NULL,

 tabla char(150) NOT NULL,

 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

 user_id int NOT NULL,

 FOREIGN KEY(user_id) REFERENCES users(id) ON UPDATE CASCADE
ON DELETE CASCADE

);
```

7.9.5 Tablas de volumen

usuarios

Atributo	Tipo de dato	Descripción	Tamaño	Nulo	Llave
id	Numérico entero	Identificador	4 bytes	No	Primaria
correo	Alfanumérico	Correo	50 caracteres	No	
contrasena	Alfanumérico	Contraseña	50 caracterres	No	
nombres	Alfanumerico	Nombres	50 caracteres	No	
apellidos	Alfanumérico	Apellidos	50 caracteres	No	

cumpleaños	Date	Fecha de nacimiendo	10 caracteres	No	
Creado_en	Date	Fecha de creacion	10 caracteres	No	

Persona

Atributo	**Tipo de dato**	**Descripción**	**Tamaño**	**Nulo**	**Llave**
id	Numérico entero	Identificador	4 bytes	No	Primaria
telefon	Alfanumerico	Telefono	8 caracteres	No	
genero	Caracter	Genero	1 caracter	No	

Usuario_id	Numerico entero	Identificador de usuario	4 bytes	No	Foránea
Creado_en	Date	Fecha de creacion	10 caracteres	No	

Cliente

Atributo	**Tipo de dato**	**Descripción**	**Tamaño**	**Nulo**	**Llave**
id	Numérico entero	Identificador	4 bytes	No	Primaria
telefono	Alfanumerico	telefono	8 caracteres	No	
Usuario_id	Numerico entero	Identificador de usuario	4 bytes	No	Foránea

Paciente

Atributo	Tipo de dato	Descripción	Tamaño	Nulo	Llave
id	Numérico entero	Identificador	4 bytes	No	Primaria
Nombre	Alfanumerico	Telefono	8 caracteres	No	
Especie	Caracter	Especie	1 caracter	No	
Raza	Numerico entero	raza	4 bytes	No	Foránea
sexo	Date	sexo	10 caracteres	No	

color	Caracter	Color	1 caracter	No	
F_nacimiento	Caracter	F_nacimiento	4 bytes	No	
perfil	Caracter	perfil	4 bytes	No	

Servicio

Atributo	**Tipo de dato**	**Descripción**	**Tamaño**	**Nulo**	**Llave**
id	Numérico entero	Identificador	4 bytes	No	Primaria
total	Numérico entero	total	8 caracteres	No	

responsable	Caracter	Genero	1 caracter	No	
paciente_id	Caracter	Identificador de paciente	4 bytes	No	Foránea
fecha	Date	Fecha de creacion	10 caracteres	No	

Pago

Atributo	**Tipo de dato**	**Descripción**	**Tamaño**	**Nulo**	**Llave**
id	Numérico entero	Identificador	4 bytes	No	Primaria
Nombre	caracter	Nombre	8 caracteres	No	

Numero_referencia	Numérico entero	Numero_referencia	1 caracter	No	
monto	Numérico entero	monto	4 bytes	No	
descripcion	caracter	descripcion	10 caracteres	No	

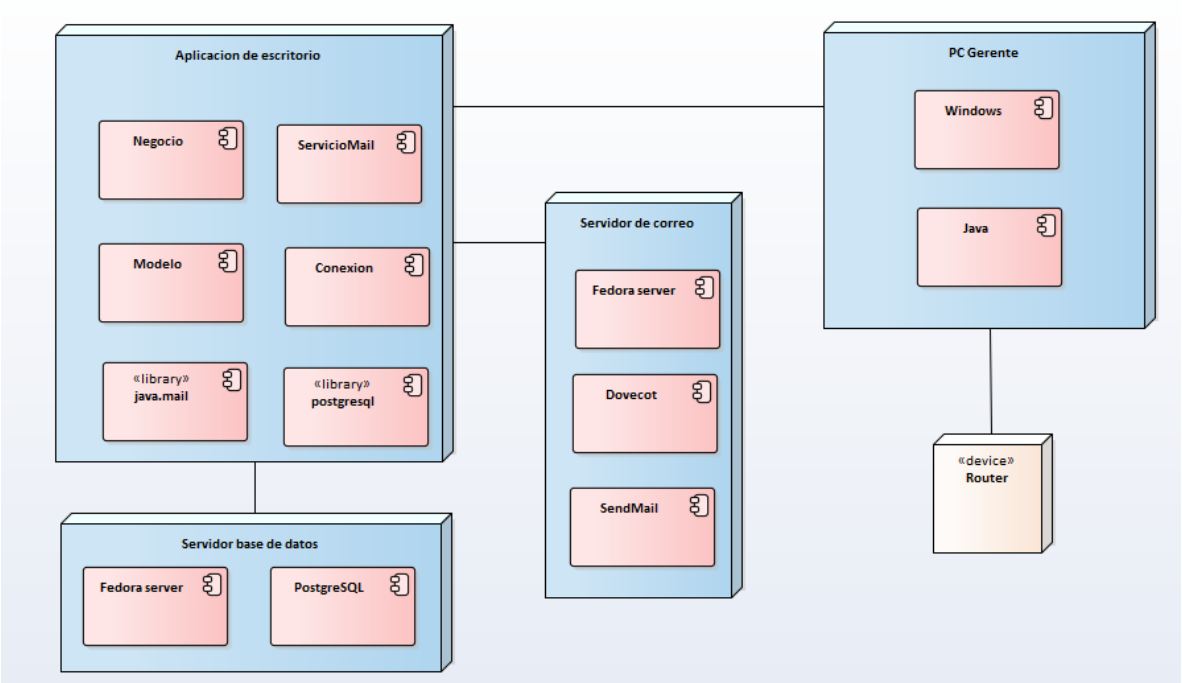
Inventario

Atributo	**Tipo de dato**	**Descripción**	**Tamaño**	**Nulo**	**Llave**
id	Numérico entero	Identificador	4 bytes	No	Primaria
detalle	Caracter	detalle	1 caracter	No	

Receta

Atributo	Tipo de dato	Descripción	Tamaño	Nulo	Llave
id	Numérico entero	Identificador	4 bytes	No	Primaria
Nro_receta	Nro_receta	total	8 caracteres	No	
estado	estado	Genero	1 caracter	No	

7.9.6 Diagrama de despliegue


Aplicacion de escritorio
Negocio
ServicioMail
Modelo
Conexion
«library»
java.mail
«library»
postgresql
Servidor base de datos
Fedora server
PostgreSQL
Servidor de correo
Fedora server
Dovecot
SendMail
PC Gerente
Windows
Java
«device»
Router

8 Resultado

Al momento de implementar el proyecto vía mail, se pudo obtener de manera satisfactoria los resultados esperados.

- Se pudo establecer conexión con el servidor mail al acceder al servicio POP3.

- Se pudo obtener respuesta al correo electrónico de “ayuda” enviando el comando “help!me” en el asunto del correo.

- Se pudo obtener resultados positivos para cada “comando”.

- Se pudo validar los comandos, entregando un mensaje de error cuando así lo amerite.

9 Conclusión y recomendaciones

9.1 Conclusión

Al finalizar el desarrollo del sistema de gestión por correo electrónico (vía mail básico) para el “Condominio”, destacamos las siguientes conclusiones:

- Se obtuvo información necesaria acerca de las políticas de negocio, su funcionamiento y los diferentes servicios que ofrecen.
- Se obtuvo información de los requisitos para la gestión de los condominios a través de los requerimientos del usuario.
- Se analizaron los requerimientos obtenidos refinándolos y estructurándolos, logrando como resultado el modelo de análisis para proseguir con el desarrollo.
- Se diseñó e implementó una base de datos en “PostgreSQL”, con la capacidad de almacenar la información de los requerimientos que se han establecido.
- Se diseñó e implementó el sistema utilizando el lenguaje de programación “JAVA”.

9.2 Recomendaciones

Debido a la necesidad y experiencia adquirida en el desarrollo del presente sistema de gestión, se sugiere lo siguiente:

- Mediante el proceso de desarrollo del sistema se obtuvo una documentación consistente. Se recomienda utilizar los modelos de la misma para cualquier posibilidad de extensión o adaptación de algún modulo del sistema según sea necesario, en función a los nuevos requerimientos que puedan surgir.
- Para poder obtener un funcionamiento óptimo y lograr la funcionalidad de la aplicación se recomienda contar con una óptima conexión a internet para poder interactuar correctamente con las funcionalidades del sistema, así como también con el servidor de correo.

10 Referencias

Meza González, J. D. (2018). *programarya*. Obtenido de https://www.programarya.com/CursosAvanzados/Java/Sockets#:~:text=Los%20sockets%20son%20un%20mecanismo,el%20lado %20del%20cliente%20y

Redondo Tejedor, B. (20 de Febrero de 2020). *¿Qué es un servidor SMTP y por qué se usa?* Obtenido de https://es.mailjet.com/blog/news/servidor-smtp/

Wikipedia. (s.f.). *Procesos Unificados, Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Proceso_unificado

Sistemas Distribuidos::Sockets en Java. (s.f.). Obtenido de https://www.infor.uva.es/~fdiaz/sd/doc/java.net.pdf

Wikipedia. (s.f.). *Correo electrónico, Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Correo_electr%C3%B3nico

Wikipedia. (s.f.). *Modelo de procesamiento de correo, Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Protocolo_para_transferencia_simple_de_correo#Modelo_de_p rocesamiento_de_correo

Wikipedia. (s.f.). *Procesos Unificados, Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Proceso_unificado

Wikipedia. (s.f.). *Protocolo de oficina de correo, Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Protocolo_de_oficina_de_correo

Wikipedia. (s.f.). *Sistema informático, Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Sistema_inform%C3%A1tico#:~:text=Un%20sistema%20infor m%C3%A1tico%20(SI)%20es,hardware%2C%20software%20y%20personal%20inform% C3%A1tico.