

UNIVERSIDAD AUTÓNOMA “GABRIEL RENÉ MORENO”
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA
COMPUTACIÓN Y TELECOMUNICACIONES



Grupo 32

SISTEMA WEB PARA LA ASIGNACIÓN DE HORARIOS, AULAS,
MATERIAS, GRUPOS Y ASISTENCIA DOCENTE PARA CADA GESTIÓN
ACADÉMICA DE LA FACULTAD.

Integrantes:

- MARCES GUTIERREZ ERICK MIGUEL - 223043257
- AUAD CASTILLO MIGUEL ANDRES - 221043756

Nro. Grupo: 32

DOCENTE: Ing. Garzón Cuellar Angélica

ÍNDICE

Contenido

ÍNDICE.....	2
1 PERFIL	5
1.1. INTRODUCCIÓN	5
1.2. OBJETIVOS	7
1.2.1. Objetivo General.....	7
1.2.2. Objetivo Específicos	7
1.3. DESCRIPCIÓN DEL PROBLEMA.....	8
1.4. ALCANCE	10
2. MARCO TEÓRICO.....	12
2.1. Sistemas de Gestión Académica.....	12
2.2. Asignación de Horarios y Aulas	12
2.3. Control de Asistencia Docente	13
2.4. Aplicaciones Web.....	13
2.5. Arquitectura Cliente–Servidor	13
2.6. Base de Datos Relacional	14
2.7. Metodología de Desarrollo (PUDS).....	14
3. MODELO DE NEGOCIO	15
DIAGRAMA DE ACTIVIDADES	15
PROCESO: GENERAR HORARIO	15
PROCESO: ASIGNAR/EDITAR HORARIO MANUAL	16
PROCESO: REGISTRAR ASISTENCIA.....	17
PROCESO: GENERAR REPORTE DE ASISTENC.....	18
PROCESO: GESTION DE USUARIOS, ROLES Y PERMISOS.....	19
4. FLUJO DE TRABAJO: CAPTURA DE REQUISITOS.....	20
4.1 IDENTIFICAR ACTORES Y CASOS DE USO	20

4.1.1 ACTORES	20
4.1.2. CASOS DE USO	20
4.2 PRIORIZAR CASOS DE USO ESTADO.....	21
4.3. DETALLAR UN CASO DE USO.....	23
CICLO #1.....	23
CICLO #2.....	34
4.4 ESTRUCTURAR MODELOS DE CASOS DE USO	43
CICLO 1.....	44
CICLO 2.....	44
DIAGRAMA GENERAL	45
5. FLUJO DE TRABAJO: ANÁLISIS	47
5.1. ANÁLISIS DE ARQUITECTURA	47
5.1.1. IDENTIFICAR PAQUETES.....	47
5.1.2. RELACIONAR PAQUETES Y CASOS DE USO	49
5.1.3. VISTA DE PAQUETE	52
5.2 ANÁLISIS DE CASO DE USO	55
5.2.1 DIAGRAMA DE COMUNICACIÓN	55
CICLO #1.....	55
CICLO #2.....	60
5.3 ANALIZAR UNA CLASE	64
CICLO #1.....	64
CICLO #2.....	69
5.4 ANALIZAR UN PAQUETE	73
6. FLUJO DE TRABAJO: DISEÑO	74
6.1 DISEÑO DE ARQUITECTURA	74
6.1.1 DISEÑO FÍSICO (DIAGRAMA DE DESPLIEGUE).....	74
6.1.2 DISEÑO LÓGICO (DIAGRAMA ORGANIZADO EN CAPAS)	74
6.2.1 DISEÑO DE DATOS LÓGICO.....	75
DIAGRAMA DE CLASE.....	75

6.2.2 DISEÑO DE DATOS FÍSICO	77
6.2. DISEÑAR CASOS DE USO	110
DIAGRAMA DE SECUENCIA	110
CICLO#1.....	110
CICLO #2.....	122
7. FLUJO DE TRABAJO: IMPLEMENTACIÓN	128
7.1) Herramientas de desarrollo de la aplicación WEB	128
7.1.1 Lenguaje de Programación y Framework Principal	128
7.1.2 Base de Datos	128
7.1.3 Plataforma de Despliegue y Sistema Operativo	129
7.1.4 Otras Herramientas y Control de Versiones	129
7.2 IMPLEMENTACIÓN DE LA ARQUITECTURA DEL SISTEMA.....	130
7.3 IMPLEMENTACIÓN DE LA ARQUITECTURA DEL SUBSISTEMAS	130
CONCLUSIÓN	131
8. RECOMENDACIÓN	131
9. BIBLIOGRAFÍA.....	133
ENLACES DEL PROYECTO:	133
Enlace del Proyecto en GitHub	133
Código QR del Proyecto en GitHub	133
Enlace a la Aplicación Web – Software Web	134
Codigo QR de la Aplicación Web – Software Web.....	134

1 PERFIL

1.1. INTRODUCCIÓN

En la actualidad, las instituciones educativas enfrentan grandes desafíos en la planificación y gestión de sus recursos académicos. Uno de los procesos más complejos dentro de la administración universitaria es la correcta asignación de horarios, aulas, grupos y docentes, ya que requiere coordinar diversos factores simultáneamente: disponibilidad de aulas, carga horaria del personal docente, asignaturas ofertadas y necesidades de cada carrera.

En la Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones (FICCT), estos procesos suelen realizarse de manera manual o semiautomatizada, utilizando hojas de cálculo, documentos dispersos o registros físicos. Esta forma de trabajo, aunque tradicional, se ha vuelto insuficiente frente al crecimiento constante de la población estudiantil, la diversificación de materias y la necesidad de disponer de información en tiempo real.

El manejo manual de los horarios conlleva problemas recurrentes, como la asignación duplicada de aulas, choques de horarios entre grupos o docentes, pérdida de tiempo en la elaboración de cronogramas y dificultad para realizar ajustes cuando surgen imprevistos. Además, el control de asistencia docente se realiza generalmente por medios físicos, lo que genera demoras en la entrega de reportes y falta de transparencia en el seguimiento de las actividades académicas. Estas deficiencias no solo afectan la gestión administrativa, sino también la organización general del proceso de enseñanza-aprendizaje.

La implementación de un sistema web integral que automatice estas tareas representa una solución moderna, eficiente y sostenible. Este sistema permitirá planificar los horarios de manera inteligente, asignar aulas de acuerdo con la disponibilidad real, registrar digitalmente la asistencia docente y generar reportes estadísticos que faciliten la toma de decisiones. Asimismo, centralizará toda la información académica en una sola plataforma, accesible desde cualquier dispositivo conectado a internet, garantizando la transparencia, trazabilidad y confiabilidad de los datos.

El proyecto denominado “Sistema Web para la Asignación de Horarios, Aulas, Grupos y Asistencia Docente” tiene como propósito fundamental digitalizar y

optimizar los procesos de planificación académica en la FICCT, contribuyendo a una administración más ordenada, eficiente y acorde con las exigencias tecnológicas actuales. El sistema estará diseñado bajo una arquitectura cliente-servidor, donde el frontend será desarrollado en React.js para ofrecer una interfaz moderna, dinámica y adaptable, mientras que el backend (en desarrollo posterior) permitirá manejar la lógica de negocio y el almacenamiento seguro de datos.

El desarrollo de este sistema se realizará aplicando la metodología PUDS (Proceso Unificado de Desarrollo de Software), complementada con modelos UML que representarán de manera gráfica las fases de análisis, diseño, implementación y pruebas. Esta metodología garantiza un desarrollo iterativo e incremental, asegurando la calidad del producto final y su alineación con los requerimientos funcionales

de la Facultad.

En definitiva, esta propuesta busca no solo resolver los problemas actuales de planificación y control, sino también modernizar la gestión académica, sentando las bases para futuras ampliaciones que incluyan la participación estudiantil, la integración con sistemas institucionales y la generación de indicadores de desempeño académico.

1.2. OBJETIVOS

1.2.1. Objetivo General

Desarrollar una aplicación web, que permita gestionar la programación académica de la carga horaria de cada gestión, asignación de horarios, aulas, grupos, docente y asistencia, optimizando la administración y trazabilidad en la Facultad FICCT

1.2.2. Objetivo Específicos

- **Automatizar la generación y validación de horarios** evitando cruces y conflictos entre aulas, grupos y docentes.
- **Facilitar el registro digital de la asistencia docente**, mediante formularios electrónicos o lectura de códigos QR para asegurar trazabilidad.
- **Integrar reportes estadísticos estáticos y dinámicos** sobre carga horaria, ausencias, y disponibilidad de aulas, permitiendo su exportación a formatos PDF y Excel.
- **Implementar una interfaz web intuitiva, adaptable y moderna (responsive)**, optimizada para su uso en dispositivos móviles, tabletas y computadoras.
- **Permitir acceso controlado y seguro según roles de usuario**, diferenciando los perfiles de administrador, docente y autoridad académica.
- **Diseñar un sistema de búsqueda y filtrado avanzado** que permita localizar rápidamente docentes, materias, aulas o grupos según criterios personalizados.
- **Incorporar alertas y notificaciones automáticas** para advertir sobre conflictos de horario, uso duplicado de aulas o cargas horarias excedidas.
- **Implementar un calendario visual interactivo**, que muestre los horarios por semana o mes y facilite la visualización global de la planificación académica.
- **Habilitar la importación y exportación masiva de datos** (por ejemplo, en formato CSV), para simplificar la carga inicial de información de docentes, materias y grupos.
- **Registrar un historial de cambios o auditoría del sistema**, que almacene las modificaciones realizadas por los usuarios, mejorando la transparencia administrativa.
- **Agregar un sistema de etiquetado de aulas según recursos disponibles**, como capacidad, equipamiento tecnológico o ubicación, para optimizar su asignación.
- **Desarrollar un panel de control o dashboard de indicadores**, que muestre métricas clave sobre asistencia docente, disponibilidad de aulas y eficiencia en la planificación.

1.3. DESCRIPCIÓN DEL PROBLEMA

La Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones (FICCT) enfrenta una serie de dificultades relacionadas con la planificación y gestión académica, especialmente en los procesos de asignación de horarios, aulas, grupos y control de asistencia docente. A pesar de los avances tecnológicos y de la disponibilidad de herramientas informáticas, gran parte de estas actividades continúan realizándose de manera manual o utilizando medios poco eficientes, como hojas de cálculo y documentos en papel.

Esta forma de trabajo ha generado una serie de problemas estructurales que afectan directamente la calidad de la gestión académica y administrativa. En primer lugar, la asignación manual de horarios conlleva a frecuentes conflictos, como la superposición de clases en un mismo espacio físico, la programación de un mismo docente en dos grupos simultáneamente o el uso inadecuado de aulas con capacidades o equipamientos inadecuados. La ausencia de un sistema automatizado que valide estos conflictos de manera inmediata provoca errores que deben ser corregidos posteriormente, consumiendo tiempo y esfuerzo por parte del personal administrativo.

En segundo lugar, la falta de una base de datos centralizada complica la consulta y actualización de información. Cada unidad académica o encargado maneja sus propios registros, lo que dificulta la trazabilidad y el control. Esto no solo retrasa la elaboración de los horarios semestrales, sino que también impide la generación de reportes confiables sobre la carga horaria de los docentes, la disponibilidad de aulas o el número de grupos activos por materia. En algunos casos, las decisiones se basan en información incompleta o desactualizada, lo que afecta la planificación global de la Facultad.

Otro aspecto crítico es el control de asistencia docente, el cual en muchos casos se realiza mediante firmas manuales en planillas físicas. Este método es vulnerable a errores, extravíos o manipulaciones, y no permite un seguimiento oportuno del cumplimiento de las horas de clase. Además, la consolidación de la información para los reportes mensuales o semestrales requiere un esfuerzo administrativo considerable, con un alto riesgo de inconsistencias o pérdidas de datos.

El problema se agrava ante el crecimiento constante del número de materias, grupos y docentes en la FICCT. A medida que aumenta la complejidad académica, la gestión manual se vuelve insostenible. Las tareas de elaboración de horarios, control de asistencia y validación de carga horaria se transforman en procesos lentos, repetitivos y propensos a errores, afectando la eficiencia y la transparencia institucional. Asimismo, la falta de

digitalización limita la capacidad de supervisión de las autoridades, quienes no pueden acceder a información consolidada en tiempo real ni evaluar el cumplimiento de las actividades docentes de manera precisa.

Todo esto repercute directamente en la experiencia del docente y del personal administrativo, quienes enfrentan dificultades para acceder a sus horarios actualizados, verificar sus cargas horarias o justificar sus asistencias. También impide a las autoridades contar con indicadores claros para la toma de decisiones estratégicas sobre el uso de infraestructura, la distribución de la carga académica o la planificación de futuras gestiones. En síntesis, el problema central radica en la ausencia de un sistema informático integral que permita automatizar, centralizar y controlar los procesos relacionados con la gestión académica. Esta carencia genera duplicidad de esfuerzos, pérdida de tiempo, errores en la planificación, falta de trazabilidad en la asistencia docente y escasa capacidad para generar reportes confiables.

Por tanto, resulta imperativo desarrollar una solución tecnológica moderna y segura, que brinde una plataforma unificada para la gestión de horarios, aulas, grupos y asistencia docente, eliminando los procesos manuales, mejorando la eficiencia operativa y garantizando la transparencia en la administración académica de la Facultad FICCT.

1.4. ALCANCE

El presente sistema web abarca el desarrollo de una plataforma integral destinada a gestionar y automatizar los procesos académicos relacionados con la planificación de horarios, aulas, grupos y asistencia docente en la Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones (FICCT).

El proyecto busca optimizar la administración académica, mejorar la trazabilidad de la información y reducir la carga de trabajo manual de los responsables de planificación.

El alcance del sistema comprende los siguientes aspectos funcionales y técnicos:

- **Gestión de Docentes:**

El sistema permitirá registrar, editar y administrar la información de los docentes, incluyendo sus datos personales, carga horaria asignada, materias que imparten, y grupos a los que pertenecen. Asimismo, se contemplará la validación de la disponibilidad horaria de cada docente para evitar conflictos o sobrecarga académica.

- **Gestión de Materias, Grupos y Carreras:**

Se implementará un módulo para registrar las materias de cada carrera, junto con los grupos respectivos. Este módulo permitirá relacionar docentes con materias y grupos, asegurando una estructura académica coherente y fácilmente actualizable.

- **Gestión de Aulas:**

El sistema contemplará la administración de aulas, considerando su capacidad, ubicación, tipo (laboratorio, aula teórica, etc.) y equipamiento. También permitirá registrar su disponibilidad en los distintos períodos, facilitando la asignación eficiente y evitando conflictos de espacio.

- **Asignación de Horarios:**

Se desarrollará una funcionalidad para asignar horarios de forma manual o automática, validando posibles conflictos entre docentes, grupos o aulas. El módulo incluirá un sistema de verificación de disponibilidad, alertas preventivas y una vista general de los horarios semanales o mensuales.

- **Control de Asistencia Docente:**

El sistema permitirá registrar la asistencia de los docentes de manera digital y en tiempo real, mediante formularios electrónicos o códigos QR únicos generados por el sistema. Este proceso garantizará la trazabilidad, exactitud de los registros y disponibilidad inmediata de la información.

- **Búsqueda y Filtrado Avanzado:**

Se integrará una herramienta de búsqueda y filtrado que permitirá localizar docentes, materias, grupos o aulas mediante criterios como carrera, semestre, tipo de aula o nombre del docente. Esto agilizará las tareas administrativas y reducirá los errores de gestión.

- **Reportes Administrativos:**

El sistema generará reportes automáticos sobre horarios, asistencia, disponibilidad de aulas, carga horaria docente y estadísticas de uso. Los reportes podrán exportarse en formatos PDF y Excel, tanto de forma estática (resúmenes) como dinámica (filtros personalizados).

- **Gestión de Usuarios y Roles:**

Se establecerá un sistema de acceso controlado mediante autenticación y roles, definiendo niveles de permisos diferenciados: administrador, docente y autoridad académica. Cada usuario tendrá acceso únicamente a las funciones que correspondan a su rol dentro del sistema.

- **Historial de Cambios y Auditoría:**

El sistema almacenará un registro de todas las acciones relevantes realizadas por los usuarios, como modificaciones de horarios, asistencia o configuración del sistema. Este registro permitirá mantener una trazabilidad total y garantizar la transparencia de los procesos administrativos.

- **Panel de Indicadores y Estadísticas:**

Se implementará un panel o dashboard visual que mostrará métricas clave, como número de aulas ocupadas, asistencia promedio de docentes, cantidad de conflictos detectados o nivel de uso de los recursos académicos. Este componente facilitará la toma de decisiones informadas por parte de las autoridades.

- **Importación y Exportación Masiva de Datos:**

Se incluirá la opción de cargar información masivamente mediante archivos CSV o Excel, facilitando el ingreso inicial de datos de docentes, materias, grupos o aulas. También permitirá exportar dichos datos para respaldo o transferencia entre sistemas.

- **Escalabilidad y Ampliación Futura:**

La arquitectura del sistema estará diseñada para permitir futuras ampliaciones, incluyendo la integración de módulos adicionales para la gestión de estudiantes, control de evaluaciones, planificación de carreras u otras unidades académicas dentro de la universidad.

2. MARCO TEÓRICO

El presente proyecto se sustenta en conceptos teóricos y tecnológicos relacionados con la gestión académica, el desarrollo de aplicaciones web, las bases de datos relacionales y las metodologías modernas de ingeniería de software. El propósito de este marco teórico es establecer los fundamentos que respaldan el diseño, desarrollo e implementación del sistema web denominado “Sistema para la Asignación de Horarios, Aulas, Grupos y Asistencia Docente”, destinado a optimizar la planificación académica en la Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones (FICCT).

2.1. Sistemas de Gestión Académica

Un sistema de gestión académica es una aplicación informática que permite administrar de manera centralizada los procesos académicos de una institución educativa.

Estos sistemas facilitan la organización de información sobre docentes, materias, grupos, horarios, aulas y asistencia, evitando duplicidades y mejorando la trazabilidad de los datos.

Su finalidad principal es automatizar tareas administrativas y ofrecer información precisa y actualizada para la toma de decisiones institucionales.

Entre sus beneficios destacan:

- Reducción de errores humanos en la planificación de horarios.
- Mejora de la eficiencia en el uso de recursos físicos (aulas).
- Acceso rápido y confiable a la información.
- Transparencia en los procesos académicos.

El sistema propuesto se enmarca dentro de esta categoría, orientándose a la planificación académica docente y la gestión automatizada de horarios y asistencia.

2.2. Asignación de Horarios y Aulas

La asignación de horarios constituye un problema de planificación común en instituciones educativas, en el cual se deben coordinar diversos recursos (aulas, docentes, grupos, materias) sin que existan conflictos de tiempo o disponibilidad.

Para ello, los sistemas modernos implementan algoritmos de validación y asignación automática, que analizan la información y generan soluciones óptimas de manera eficiente.

Este proceso requiere definir:

- Las restricciones (por ejemplo, un docente no puede dictar dos clases simultáneamente).
- Los recursos (aulas, horarios disponibles, grupos).
- Los criterios de optimización (minimizar conflictos, aprovechar mejor los espacios).

El sistema propuesto automatiza estas validaciones, asegurando que las asignaciones cumplan con todas las restricciones académicas y administrativas definidas por la Facultad.

2.3. Control de Asistencia Docente

El control de asistencia docente es un proceso fundamental en la gestión académica, ya que permite verificar el cumplimiento de la carga horaria asignada a cada profesor.

Tradicionalmente, este control se realiza mediante registros físicos o planillas firmadas, lo cual es propenso a errores y carece de trazabilidad.

El sistema propuesto reemplaza este método con un registro digital, basado en formularios electrónicos o códigos QR que validan automáticamente la fecha, hora y usuario que realiza la marcación.

Esto garantiza transparencia, seguridad y disponibilidad inmediata de la información, además de permitir generar reportes automáticos de asistencia.

2.4. Aplicaciones Web

Una aplicación web es un sistema de software que se ejecuta en un navegador y permite la interacción entre usuarios y servidores a través de Internet.

Las aplicaciones web modernas se caracterizan por su accesibilidad, ya que no requieren instalación, y por su capacidad de funcionar en distintos dispositivos.

El sistema se desarrollará utilizando React.js como tecnología principal del frontend, lo que permitirá construir una interfaz moderna, dinámica y completamente responsive (adaptable a celulares, tablets o computadoras).

El backend se implementará mediante una API REST desarrollada en Laravel (PHP), encargada de manejar la lógica de negocio y la conexión con la base de datos.

La información será almacenada en un sistema de base de datos relacional MySQL o PostgreSQL, garantizando integridad, consistencia y seguridad de los datos.

2.5. Arquitectura Cliente–Servidor

El sistema se basará en una arquitectura Cliente–Servidor, modelo que separa las funciones del sistema en dos partes:

- **Cliente:** donde se encuentra la interfaz que el usuario utiliza para interactuar con la aplicación (React.js).
- **Servidor:** que procesa las solicitudes, aplica las reglas de negocio y gestiona la base de datos.

Este tipo de arquitectura facilita la escalabilidad, la seguridad y la mantenibilidad del sistema, permitiendo futuras ampliaciones sin afectar el funcionamiento general.

2.6. Base de Datos Relacional

La base de datos será el componente encargado de almacenar toda la información académica: docentes, materias, aulas, grupos, horarios y registros de asistencia.

Se utilizará un **modelo relacional**, que organiza la información en tablas interconectadas mediante claves primarias y foráneas.

Esto permite realizar consultas complejas, mantener la integridad de los datos y evitar duplicaciones.

Se aplicarán principios de **normalización** para optimizar el diseño y mejorar la eficiencia del sistema.

2.7. Metodología de Desarrollo (PUDS)

El desarrollo del sistema seguirá la **Metodología PUDS (Proceso Unificado de Desarrollo de Software)**, la cual estructura el trabajo en fases iterativas: **Inicio, Elaboración, Construcción e Implementación**.

Cada fase incluye actividades de análisis, diseño, implementación y pruebas, asegurando que el producto final cumpla con los requisitos definidos y mantenga la calidad del software.

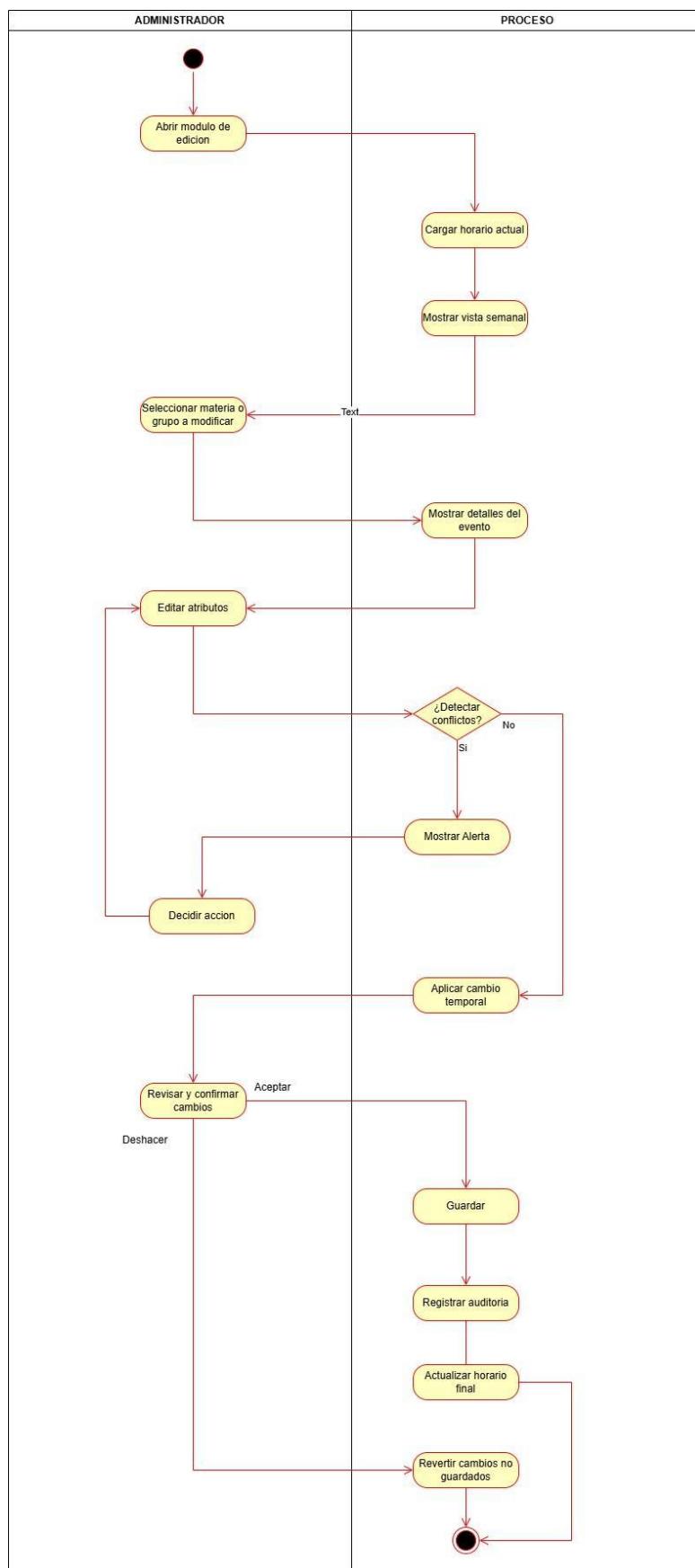
Los beneficios de aplicar PUDS son:

- Control y seguimiento del avance del proyecto.
- Claridad en los requerimientos funcionales.
- Reducción de errores durante el desarrollo.
- Documentación completa y trazable en cada fase.

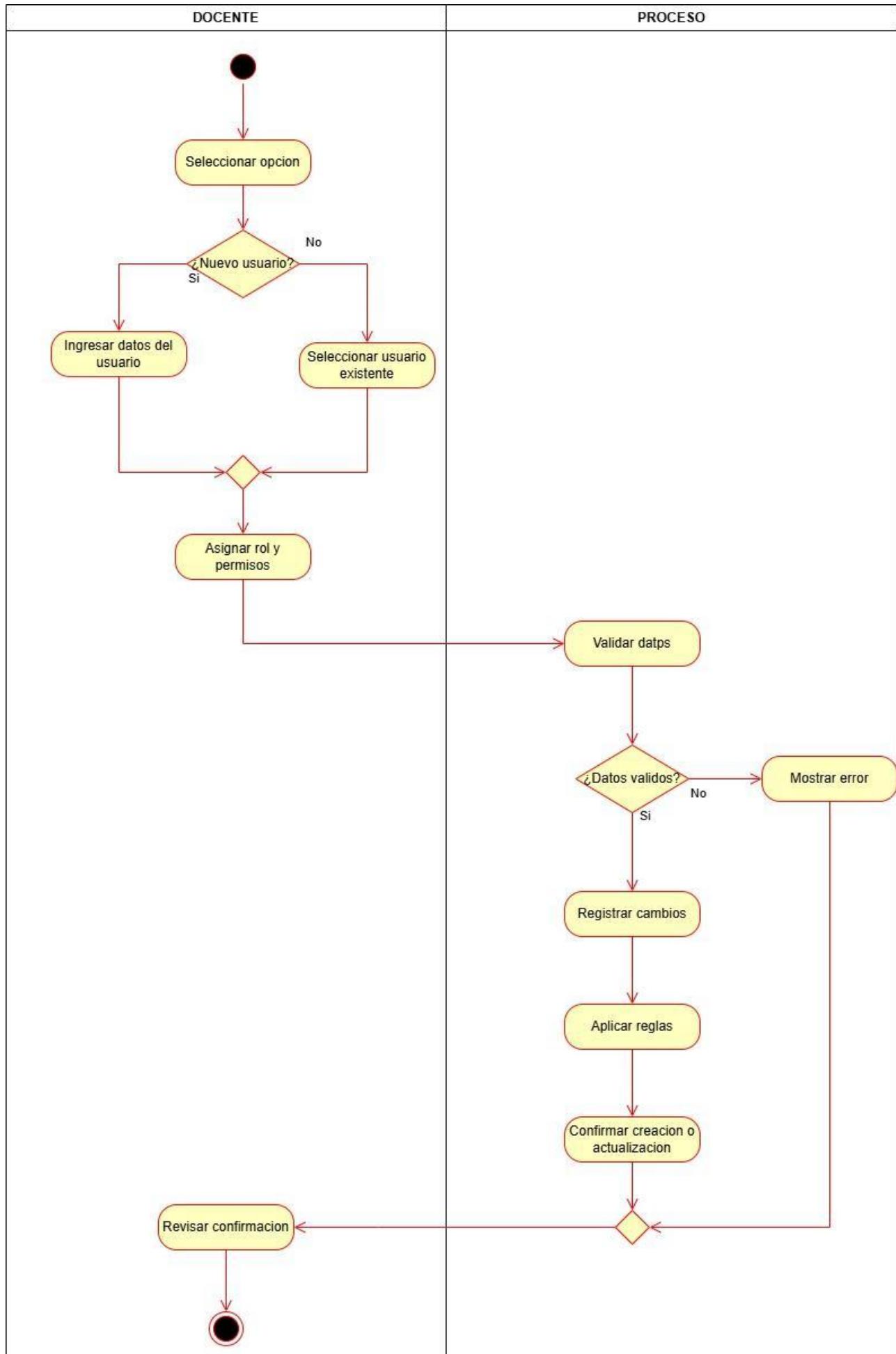
3. MODELO DE NEGOCIO

DIAGRAMA DE ACTIVIDADES

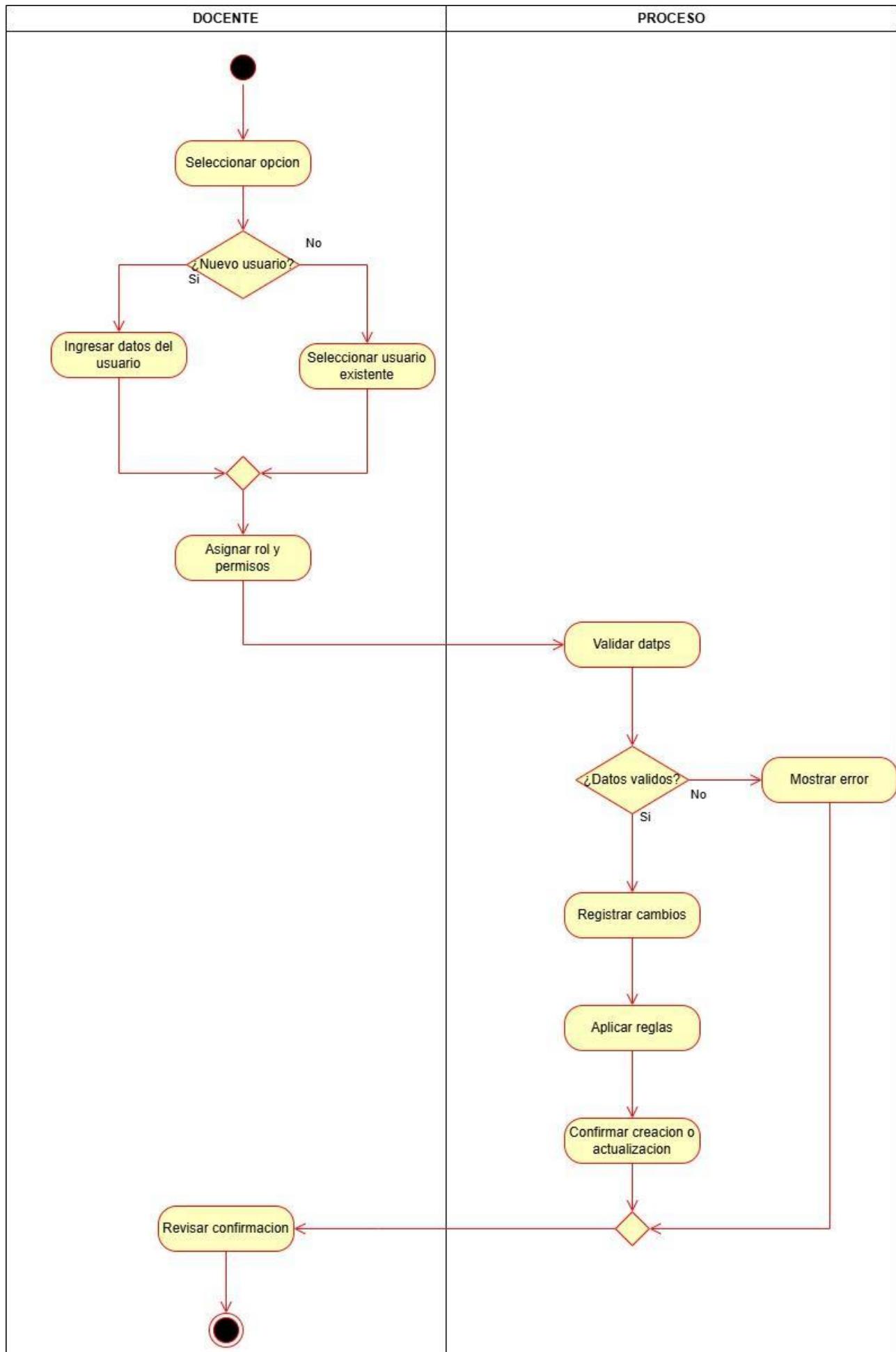
PROCESO: GENERAR HORARIO



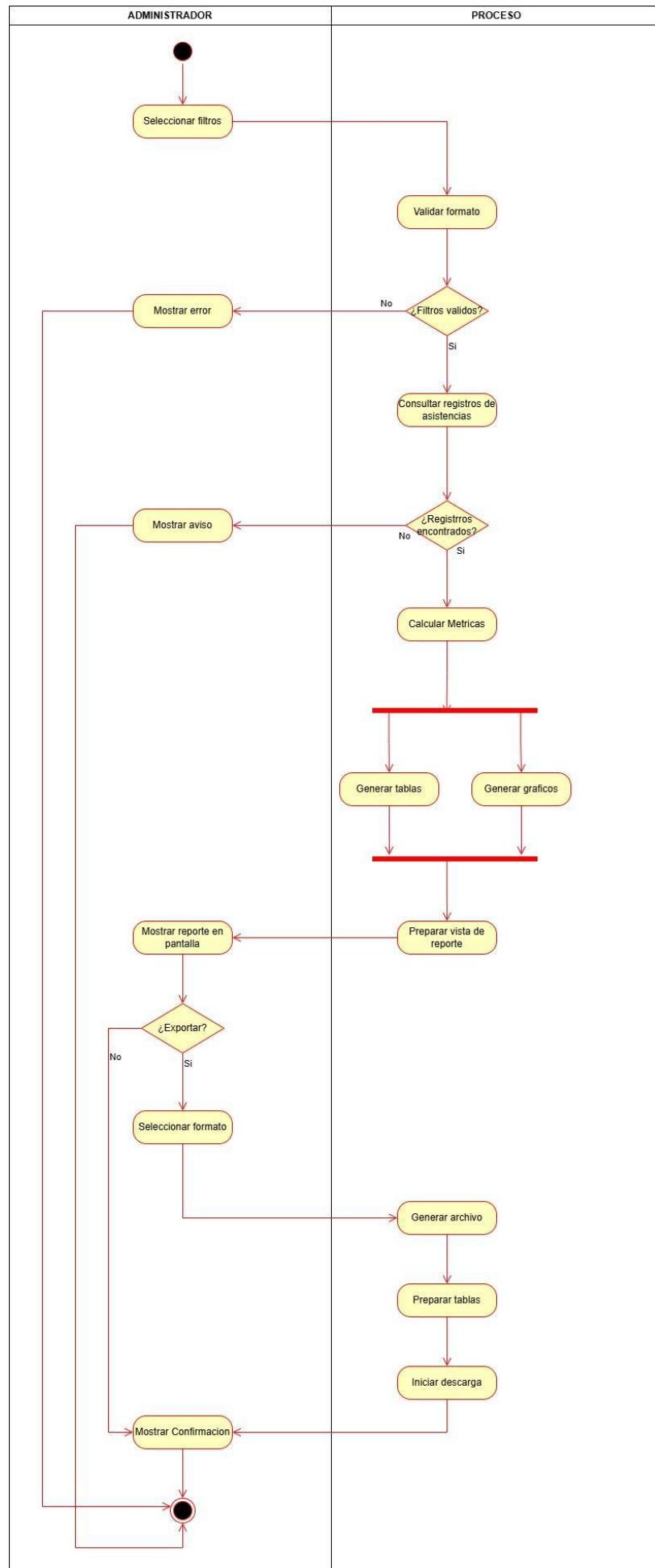
PROCESO: ASIGNAR/EDITAR HORARIO MANUAL



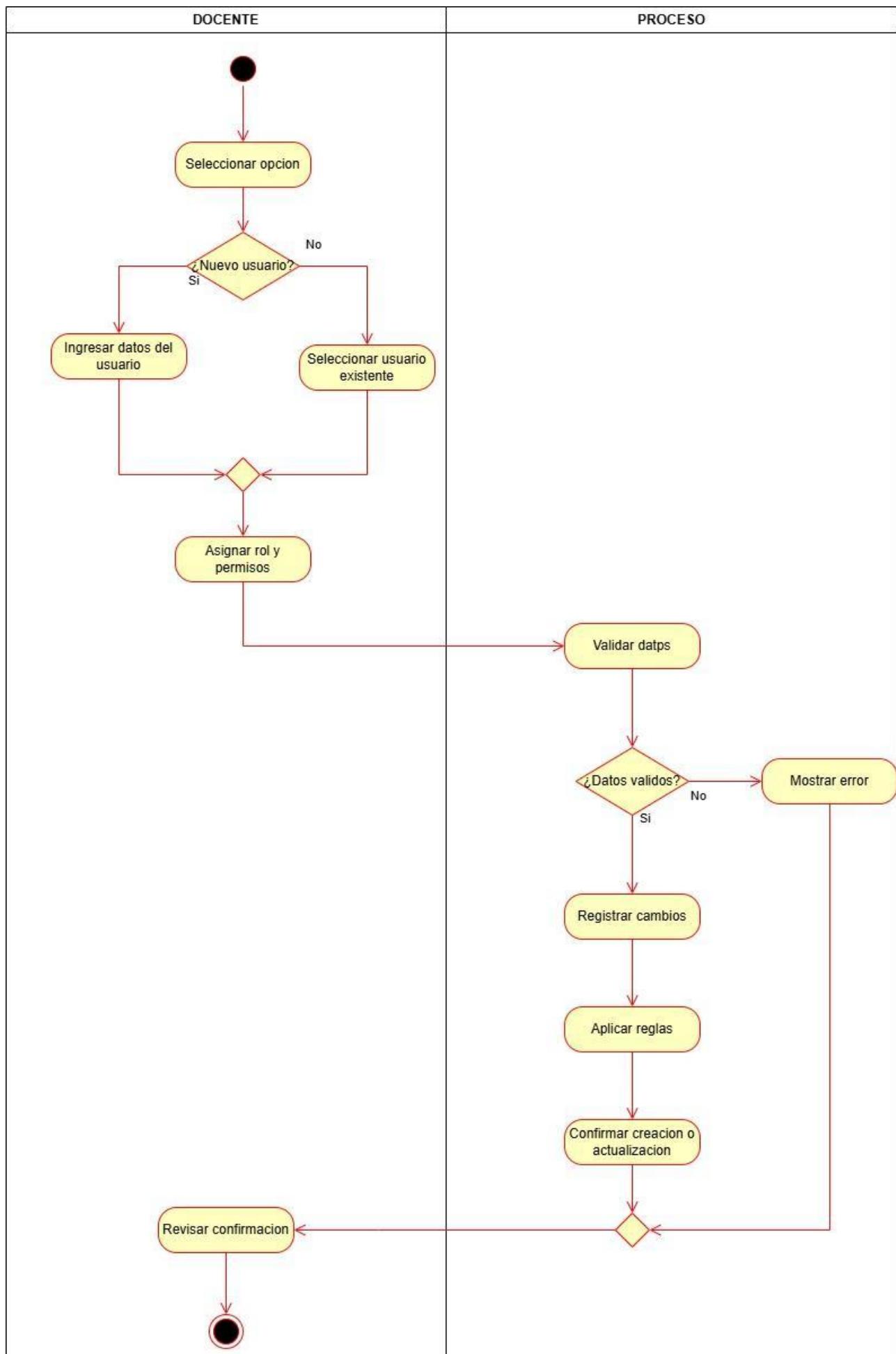
PROCESO: REGISTRAR ASISTENCIA



PROCESO: GENERAR REPORTE DE ASISTENCIA



PROCESO: GESTION DE USUARIOS, ROLES Y PERMISOS



4. FLUJO DE TRABAJO: CAPTURA DE REQUISITOS

4.1 IDENTIFICAR ACTORES Y CASOS DE USO

4.1.1 ACTORES

- **Administrador**
- **Docente**
- **Autoridad**

4.1.2. CASOS DE USO

- **CU1:** Registrar Historial de Cambios
- **CU2:** Consultar Historial de Acciones
- **CU3:** Gestionar Docente
- **CU4:** Gestionar Carga Horaria del Docente
- **CU5:** Gestionar Materia
- **CU6:** Gestionar Grupo
- **CU7:** Generar Horario
- **CU8:** Consultar Horarios Semanales
- **CU9:** Generar Reportes
- **CU10:** Consultar Aulas Disponibles
- **CU11:** Exportar Reporte a PDF/Excel
- **CU12:** Importación de Usuarios(CSV/XLSX)
- **CU13:** Iniciar Sesión
- **CU14:** Cerrar Sesión
- **CU15:** Gestionar Usuario
- **CU16:** Gestionar Perfil
- **CU17:** Gestionar Rol
- **CU18:** Gestionar Permiso
- **CU19:** Gestionar Periodos Académicos
- **CU20:** Gestionar Aula
- **CU21:** Consultar Horario por Docente
- **CU22:** Generar QR Asistencia
- **CU23:** Registrar Asistencia por QR
- **CU24:** Solicitar Licencia

4.2 PRIORIZAR CASOS DE USO ESTADO

Estado

- Aprobado: Es usado cuando el caso de uso está entre los principales en construirse.
- Incluido: Es usado cuando el caso de uso está entre los próximos a realizarse o tomarse en cuenta.
- Propuesto: No fue considerado inicialmente.

Prioridad

- Normal: El caso de uso se realiza inicialmente.
- Crítico: Cuando el caso de uso requiere más información que le será proporcionada de otro caso de uso.
- Significativo: Conforme a la necesidad de ver información con las vistas posibles.

Riesgo

- Normal: Se usa en los casos de uso básicos de registros de datos.
- Crítico: Cuando el caso de uso necesita de mucha información para realizarse.
- Accesorio: Cuando el caso de uso es base para otros.

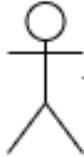
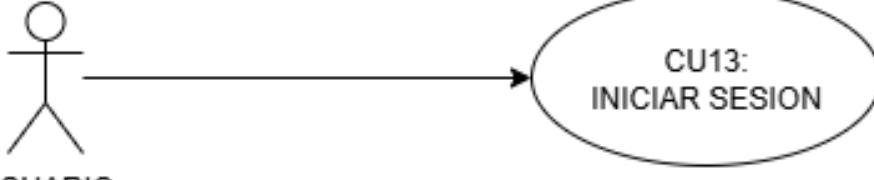
Nro	Caso de Uso	Actor(es)	Estado	Prioridad	Riesgo	Ciclo
CU1 3	Iniciar Sesión	Administrador, Docente, Autoridad	Aprobado	Crítico	Normal	Ciclo 1
CU1 4	Cerrar Sesión	Administrador, Docente, Autoridad	Aprobado	Normal	Normal	Ciclo 1
CU3	Gestionar Docente	Administrador	Aprobado	Crítico	Normal	Ciclo 1
CU4	Gestionar carga horaria de docente	Administrador	Aprobado	Crítico	Crítico	Ciclo 1
CU2 0	Gestionar Aula	Administrador	Aprobado	Normal	Accesorio	Ciclo 1
CU5	Gestionar Materia	Administrador	Aprobado	Normal	Accesorio	Ciclo 1
CU6	Gestionar Grupo	Administrador	Aprobado	Normal	Accesorio	Ciclo 1
CU1 7	Gestionar Rol	Administrador	Incluido	Crítico	Crítico	Ciclo 1
CU1 8	Gestionar Permiso	Administrador	Incluido	Crítico	Crítico	Ciclo 1
CU1 2	Importación Usuario	Administrador	Propuesto	Significativo	Crítico	Ciclo 1

	(CSV/XLSX)					
CU1 5	Gestionar Usuario	Administrador	Aprobado	Crítico	Crítico	Ciclo 1
CU1 9	Gestionar Periodos Académicos	Administrador	Aprobado	Normal	Accesorio	Ciclo 1
CU1	Registrar Historial de Cambios	Administrador, Docente, Autoridad.	Incluido	Significativo	Normal	Ciclo 1
CU2	Consultar Historial de Acciones	Administrador	Propuesto	Significativo	Normal	Ciclo 1
CU8	Consultar Horarios Semanales	Docente, Autoridad	Incluido	Significativo	Normal	Ciclo 2
CU2 1	Consultar Horario por Docente	Docente, Autoridad	Incluido	Significativo	Normal	Ciclo 2
CU9	Generar Reportes	Administrador, Autoridad	Incluido	Significativo	Accesorio	Ciclo 2
CU1 1	Exportar Reporte a PDF/Excel	Administrador, Autoridad	Incluido	Normal	Accesorio	Ciclo 2
CU2 4	Solicitar Licencia	Docente	Incluido	Significativo	Normal	Ciclo 2
CU1 0	Consultar aulas disponibles	Administrador	Incluido	Normal	Normal	Ciclo 2
CU1 6	Gestionar Perfil	Administrador, Docente	Incluido	Normal	Normal	Ciclo 2
CU7	Generar Horario	Administrador, Autoridad.	Aprobado	Crítico	Crítico	Ciclo 2
CU2 2	Generar QR	Docente	Aprobado	Critic	Normal	Ciclo 2
CU2 3	Registrar Asistencia por QR	Docente	Aprobado	Crítico	Crítico	Ciclo 2

4.3. DETALLAR UN CASO DE USO

CICLO #1

CU13: INICIAR SESION

CU13: INICIAR SESION	
 USUARIO	
Propósito:	Permitir que los usuarios (Administrador, Docente o Autoridad) accedan al sistema de acuerdo con sus credenciales y roles asignados.
Resumen:	El usuario ingresa su nombre de usuario y contraseña. El sistema valida los datos contra la base de usuarios registrados. Si son correctos, se concede acceso al panel correspondiente según su rol. En caso contrario, se muestra un mensaje de error.
Actores:	Administrador, Docente, Autoridad
Actor iniciador:	Usuario (Administrador, Docente o Autoridad)
Flujo principal:	<ul style="list-style-type: none">● El usuario accede a la página de inicio de sesión● Ingresá sus credenciales (usuario y contraseña).● El sistema valida los datos● Si los datos son correctos, el sistema redirige al panel del rol correspondiente● El sistema registra la hora y fecha del inicio de sesión.
Precondición:	El usuario debe estar previamente registrado en el sistema.

CU14: CERRAR SESIÓN

CU14: CERRAR SESIÓN	
<pre> graph LR USUARIO((USUARIO)) --> CU13([CU13: INICIAR SESION]) CU13 -- "<<extend>>" --> CU14([CU14: CERRAR SESION]) </pre>	
Propósito:	Permitir que el usuario finalice su sesión de manera segura en el sistema.
Resumen:	Una vez que el usuario ha terminado sus actividades, puede cerrar su sesión para evitar accesos no autorizados. El sistema elimina la sesión activa y redirige a la pantalla de inicio.
Actores:	Administrador, Docente, Autoridad
Actor iniciador:	Usuario (Administrador, Docente o Autoridad)
Flujo principal:	<ul style="list-style-type: none"> El usuario selecciona la opción “Cerrar sesión”. El sistema elimina las credenciales activas. Se muestra un mensaje de confirmación de cierre. El sistema redirige a la pantalla de inicio de sesión.
Precondición:	El usuario debe tener una sesión activa en el sistema.
Propósito:	Permitir que el usuario finalice su sesión de manera segura en el sistema.
Resumen:	Una vez que el usuario ha terminado sus actividades, puede cerrar su sesión para evitar accesos no

autorizados.

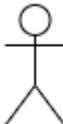
CU3: GESTIONAR DOCENTE

CU3: GESTIONAR DOCENTE	
 ADMINISTRADOR	CU3: GESTIONAR DOCENTE
Propósito:	Registrar, editar, eliminar y consultar la ficha de los docentes.
Resumen:	Permite al administrador mantener la información actualizada de los docentes, incluyendo datos personales, categoría, título y carga horaria.
Actores:	Administrador
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">• El administrador accede al módulo “Docentes”.• Selecciona la acción deseada (Registrar, Editar o Eliminar).• Ingresa o modifica los datos del docente.• Guarda los cambios• El sistema actualiza la información en la base de datos.
Precondición:	El administrador debe haber iniciado sesión.

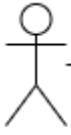
CU4: GESTIONAR CARGA HORARIA DEL DOCENTE

CU4: GESTIONAR CARGA HORARIA DEL DOCENTE	
 ADMINISTRADOR	CU4: GESTIONAR CARGA HORARIA DEL DOCENTE
Propósito:	Asignar y modificar la carga horaria de los docentes según las materias y grupos asignados.
Resumen:	El administrador gestiona las horas de trabajo de los docentes, asegurando que no excedan el máximo permitido y que las asignaciones sean válidas.
Actores:	Administrador
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">• El administrador accede al módulo “Carga Horaria”.• Selecciona al docente correspondiente.• Asigna materias y horarios.• El sistema valida que no se supere el límite máximo.• Guarda los cambios.
Precondición:	Deben existir docentes, materias y periodos registrados.

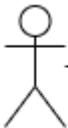
CU20: GESTIONAR AULA

CU20: GESTIONAR AULA	
 ADMINISTRADOR	CU20: GESTIONAR AULA
Propósito:	Registrar, modificar o eliminar la información de las aulas disponibles.
Resumen:	El administrador puede mantener actualizada la información de las aulas (nombre, ubicación, capacidad, estado).
Actores:	Administrador
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">● 1. El administrador accede al módulo “Aulas”.● 2. Selecciona “Registrar”, “Editar” o “Eliminar”.● 3. Ingresa o actualiza los datos del aula.● 4. Guarda los cambios.● 5. El sistema actualiza la base de datos.
Precondición:	El administrador debe haber iniciado sesión.

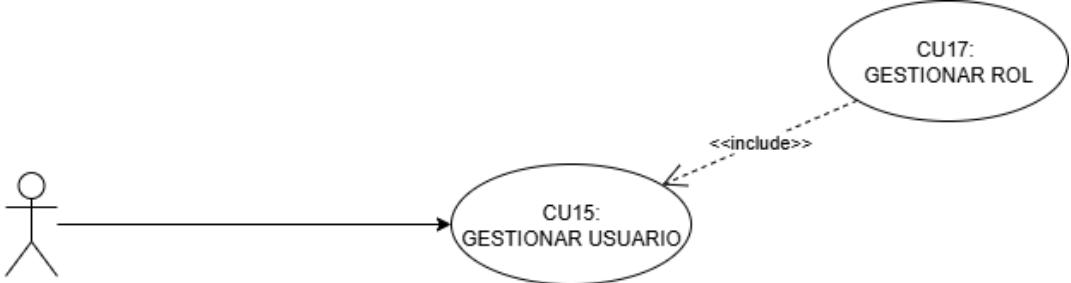
CU5: GESTIONAR MATERIA

CU5: GESTIONAR MATERIA	
 ADMINISTRADOR	CU5: GESTIONAR MATERIA
Propósito:	Registrar, modificar o eliminar materias.
Resumen:	El administrador gestiona las materias del sistema, vinculándolas a las carreras y definiendo su carga horaria.
Actores:	Administrador
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">● 1. El administrador accede al módulo “Materias”.● 2. Selecciona “Registrar nueva materia” o “Editar”.● 3. Ingresa o actualiza los datos.● 4. Guarda los cambios.● 5. El sistema actualiza la información en la base de datos.
Precondición:	Deben existir carreras registradas.

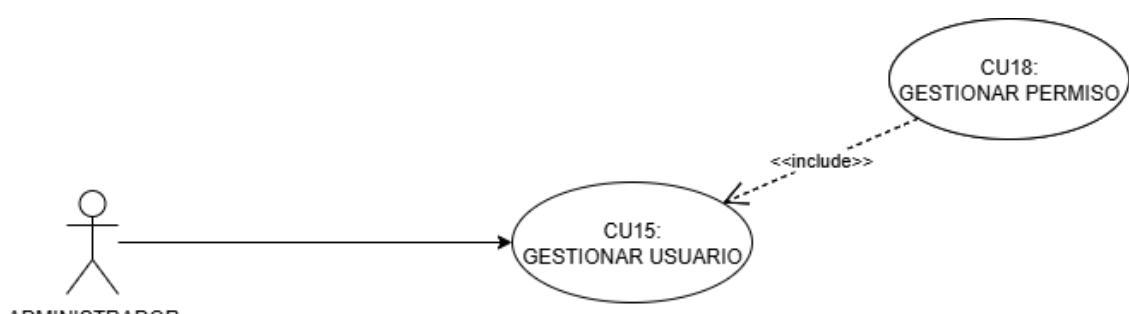
CU6: GESTIONAR GRUPO

CU6: GESTIONAR GRUPO	
 ADMINISTRADOR	→ CU6: GESTIONAR GRUPO
Propósito:	Registrar, modificar o eliminar grupos de materias.
Resumen:	El administrador puede crear nuevos grupos, asignar docentes y definir horarios según la materia.
Actores:	Administrador
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">• 1. El administrador accede al módulo “Grupos”.• 2. Selecciona “Nuevo grupo” o “Editar”.• 3. Asigna materia, docente y horario.• 4. Guarda los cambios.• 5. El sistema registra las modificaciones.
Precondición:	Deben existir docentes y materias registrados.

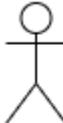
CU17: GESTIONAR ROL

CU17: GESTIONAR ROL	
	ADMINISTRADOR
Propósito:	Crear, modificar o eliminar roles dentro del sistema.
Resumen:	Permite al administrador definir los diferentes roles que se asignarán a los usuarios (por ejemplo: Administrador, Docente, Autoridad).
Actores:	Administrador, Sistema
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">● 1. El administrador accede al módulo “Roles”.● 2. Selecciona “Nuevo rol” o “Editar rol”.● 3. Define el nombre y descripción del rol.● 4. Guarda los cambios.● 5. El sistema registra el rol para su asignación.
Precondición:	El administrador debe haber iniciado sesión.

CU18: GESTIONAR PERMISO

CU18: GESTIONAR PERMISO	
	ADMINISTRADOR
Propósito:	Definir los permisos de acceso para cada rol.
Resumen:	El administrador puede configurar qué funciones o módulos están disponibles para cada rol, permitiendo el control de accesos en el sistema.
Actores:	Administrador
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">• 1. El administrador accede al módulo “Permisos”.• 2. Selecciona el rol al que desea asignar permisos.• 3. Marca o desmarca los módulos accesibles.• 4. Guarda la configuración.• 5. El sistema actualiza los permisos del rol.
Precondición:	Deben existir roles registrados en el sistema.

CU15: GESTIONAR USUARIO

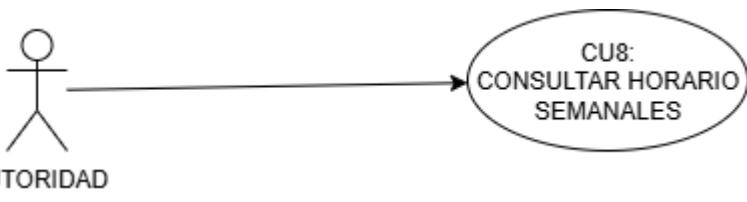
CU15: GESTIONAR USUARIO	
 ADMINISTRADOR	CU15: GESTIONAR USUARIO
Propósito:	Crear, editar o eliminar usuarios del sistema.
Resumen:	Permite al administrador gestionar las cuentas de usuario, asignando roles, credenciales y estado de activación.
Actores:	Administrador
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">● 1. El administrador accede al módulo “Usuarios”.● 2. Selecciona “Nuevo usuario” o “Editar”.● 3. Ingresa los datos del usuario (nombre, correo, rol, contraseña).● 4. Guarda los cambios.● 5. El sistema actualiza la base de datos.
Precondición:	Deben existir roles definidos en el sistema.

CU19: GESTIONAR PERIODOS ACADEMICOS

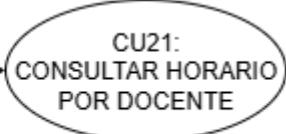
CU19: GESTIONAR PERIODOS ACADEMICOS	
 ADMINISTRADOR	CU19: GESTIONAR PERIODOS ACADEMICOS
Propósito:	Administrar los periodos académicos del sistema.
Resumen:	El administrador puede crear, editar o cerrar periodos académicos, vinculándolos a horarios y grupos.
Actores:	Administrador
Actor iniciador:	Administrador
Flujo principal:	<ul style="list-style-type: none">● 1. El administrador accede al módulo “Periodos Académicos”.● 2. Selecciona “Nuevo periodo” o “Editar”.● 3. Define las fechas de inicio y fin.● 4. Guarda los cambios.● 5. El sistema actualiza los registros.
Precondición:	El administrador debe haber iniciado sesión.

CICLO #2

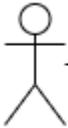
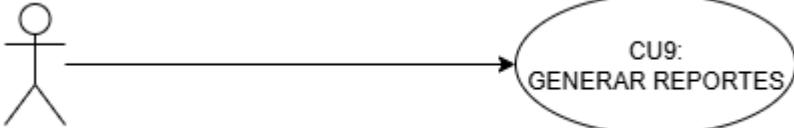
CU8: CONSULTAR HORARIOS SEMANALES

CU8: CONSULTAR HORARIOS SEMANALES	
AUTORIDAD	
Propósito	Permitir que el docente o autoridad visualice el horario completo de clases y actividades académicas de la semana.
Resumen	El sistema muestra los horarios semanales registrados para cada docente o grupo, incluyendo materias, aulas y períodos. Facilita la planificación y control académico.
Actores	Docente, Autoridad.
Actor iniciador	Docente o Autoridad.
Flujo principal	<ul style="list-style-type: none">● 1. El usuario accede al módulo de horarios.● 2. Selecciona el periodo o semana a consultar.● 3. El sistema obtiene la información desde la base de datos.● 4. Se muestran los horarios detallados (materia, aula, hora, grupo).● 5. El usuario puede imprimir o exportar si lo desea.
Precondición	El usuario debe haber iniciado sesión y existir horarios previamente generados en el sistema.

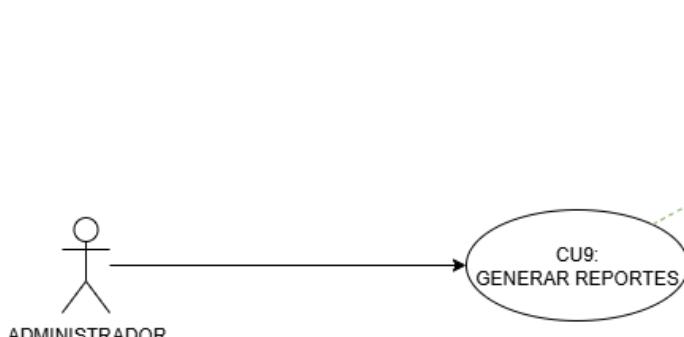
CU21: CONSULTAR HORARIO POR DOCENTE

CU21: CONSULTAR HORARIO POR DOCENTE	
 AUTORIDAD	
Propósito	Mostrar el horario asignado a un docente específico para fines de control o supervisión.
Resumen	Permite que un docente o autoridad consulte la carga horaria y distribución de clases del docente seleccionado.
Actores	Docente, Autoridad.
Actor iniciador	Autoridad (principalmente) o Docente.
Flujo principal	<ul style="list-style-type: none">● 1. El usuario accede a la sección de consulta por docente.● 2. Selecciona el docente deseado.● 3. El sistema muestra el horario completo asignado.● 4. Puede generar una versión imprimible o descargarla.
Precondición	El docente debe existir en el sistema y tener horarios asignados.

CU9: GENERAR REPORTES

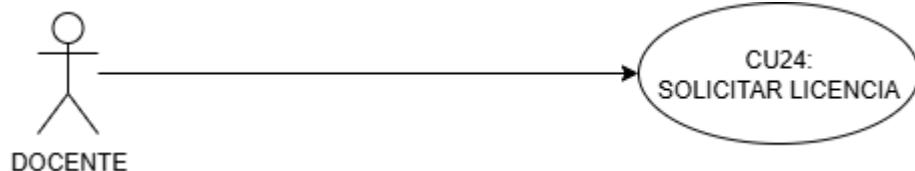
CU9: GENERAR REPORTES	
 ADMINISTRADOR	
Propósito	Permitir la generación de reportes administrativos o académicos a partir de los datos del sistema.
Resumen	El administrador o autoridad selecciona el tipo de reporte (horarios, asistencia, licencias, etc.) y el sistema genera un documento visualizable o exportable.
Actores	Administrador, Autoridad.
Actor iniciador	Administrador.
Flujo principal	<ul style="list-style-type: none">● 1. El usuario accede al módulo de reportes.● 2. Selecciona el tipo de reporte y los filtros deseados.● 3. El sistema procesa los datos y genera el reporte.● 4. Se muestra una vista previa con opción de exportar.
Precondición	El usuario debe estar autenticado y existir información registrada para generar el reporte.

CU11: EXPORTAR REPORTE A PDF/EXCEL

CU11: EXPORTAR REPORTE A PDF/EXCEL	
 <p>ADMINISTRADOR</p> <p>CU9: GENERAR REPORTES</p> <p>CU11: EXPORTAR REPORTE A PDF/EXCEL</p> <p><<extend>></p>	
Propósito	Permitir al usuario exportar los reportes generados a formatos PDF o Excel para su almacenamiento o impresión.
Resumen	Este caso de uso complementa al de “Generar Reportes”, ofreciendo opciones de exportación para facilitar el análisis y documentación.
Actores	Administrador, Autoridad.
Actor iniciador	Administrador o Autoridad.
Flujo principal	<ul style="list-style-type: none">● 1. Desde la vista de un reporte generado, el usuario elige la opción “Exportar”.● 2. Selecciona el formato deseado (PDF o Excel).● 3. El sistema genera y descarga el archivo.● 4. El usuario guarda o imprime el documento.
Precondición	Debe haberse generado previamente un reporte válido (CU9).

CU24: SOLICITAR LICENCIA

CU24: SOLICITAR LICENCIA



Propósito	Permitir que un docente solicite una licencia o permiso, indicando fechas y motivo.
Resumen	El docente completa un formulario de solicitud que será posteriormente revisado por la autoridad correspondiente.
Actores	Docente.
Actor iniciador	Docente.
Flujo principal	<ul style="list-style-type: none">● 1. El docente accede al módulo de licencias.● 2. Selecciona el tipo de licencia y las fechas.● 3. Adjunta documentación si es necesario.● 4. Envía la solicitud.● 5. El sistema registra la solicitud para revisión.
Precondición	El docente debe estar autenticado y con perfil activo.

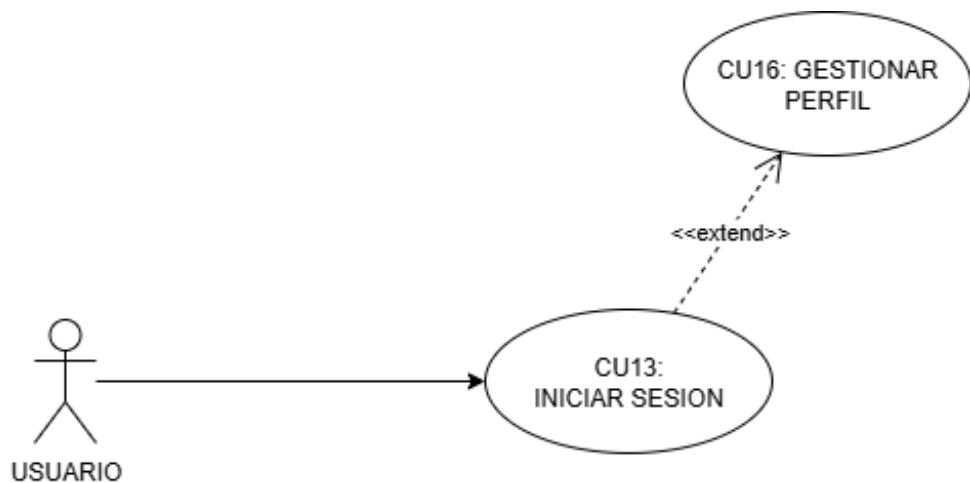
CU10: CONSULTAR AULAS DISPONIBLES

CU10: CONSULTAR AULAS DISPONIBLES

 ADMINISTRADOR	CU10: CONSULTAR AULAS DISPONIBLES
Propósito	Permitir al administrador conocer qué aulas están disponibles en un horario o periodo determinado.
Resumen	Facilita la gestión de espacios físicos verificando disponibilidad según la carga horaria existente.
Actores	Administrador.
Actor iniciador	Administrador.
Flujo principal	<ul style="list-style-type: none"> ● 1. El administrador accede al módulo de aulas. ● 2. Selecciona el día y la hora. ● 3. El sistema consulta la base de datos y muestra las aulas libres. ● 4. El administrador puede asignarlas o reservarlas.
Precondición	Deben existir aulas registradas y horarios configurados en el sistema.

CU16: GESTIONAR PERFIL

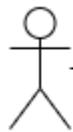
CU16: GESTIONAR PERFIL



Propósito	Permitir que el usuario (administrador o docente) actualice su información personal y credenciales de acceso.
Resumen	El usuario puede modificar sus datos básicos, foto, contraseña y preferencias de cuenta.
Actores	Administrador, Docente.
Actor iniciador	Usuario autenticado (Administrador o Docente).
Flujo principal	<ul style="list-style-type: none">● 1. El usuario accede a su perfil.● 2. Edita los campos deseados.● 3. El sistema valida la información.● 4. Se guardan los cambios y se notifica al usuario.
Precondición	El usuario debe haber iniciado sesión.

CU7: GENERAR HORARIO

CU7: GENERAR HORARIO



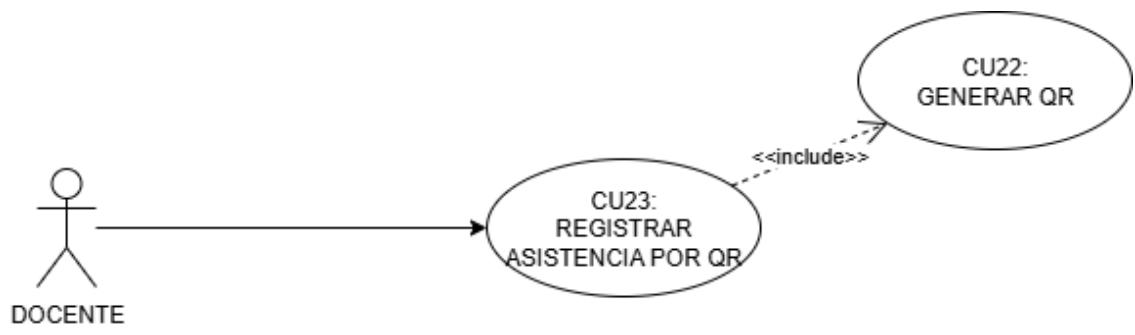
ADMINISTRADOR

CU7:
GENERAR HORARIO

Propósito	Crear y organizar horarios de clases según materias, grupos, aulas y docentes disponibles.
Resumen	El administrador o autoridad utiliza la información académica para generar horarios automáticos o manuales.
Actores	Administrador, Autoridad.
Actor iniciador	Administrador.
Flujo principal	<ul style="list-style-type: none">● 1. El administrador ingresa al módulo de horarios.● 2. Selecciona los parámetros (materias, docentes, grupos, aulas).● 3. El sistema valida la disponibilidad.● 4. Se genera el horario y se guarda en la base de datos.● 5. Se notifica a los docentes.
Precondición	Deben existir docentes, materias, grupos y aulas previamente registrados.

CU22: GENERAR QR

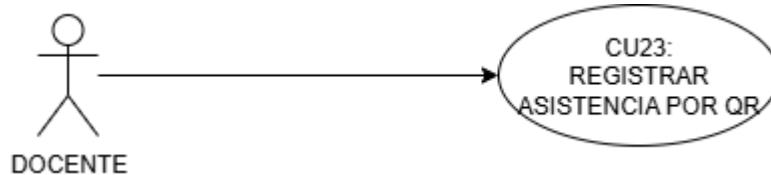
CU22: GENERAR QR



Propósito	Generar códigos QR únicos para el registro de asistencia de clases.
Resumen	El sistema crea un código QR asociado a la sesión de clase que será escaneado por el docente o estudiantes.
Actores	Docente.
Actor iniciador	Docente.
Flujo principal	<ul style="list-style-type: none">● 1. El docente ingresa al módulo de asistencia.● 2. Selecciona la clase o sesión.● 3. El sistema genera el código QR.● 4. El código se muestra o imprime para su uso en clase.
Precondición	Debe existir una clase o sesión registrada en el horario.

CU23: REGISTRAR ASISTENCIA POR QR O FORMULARIO WEB

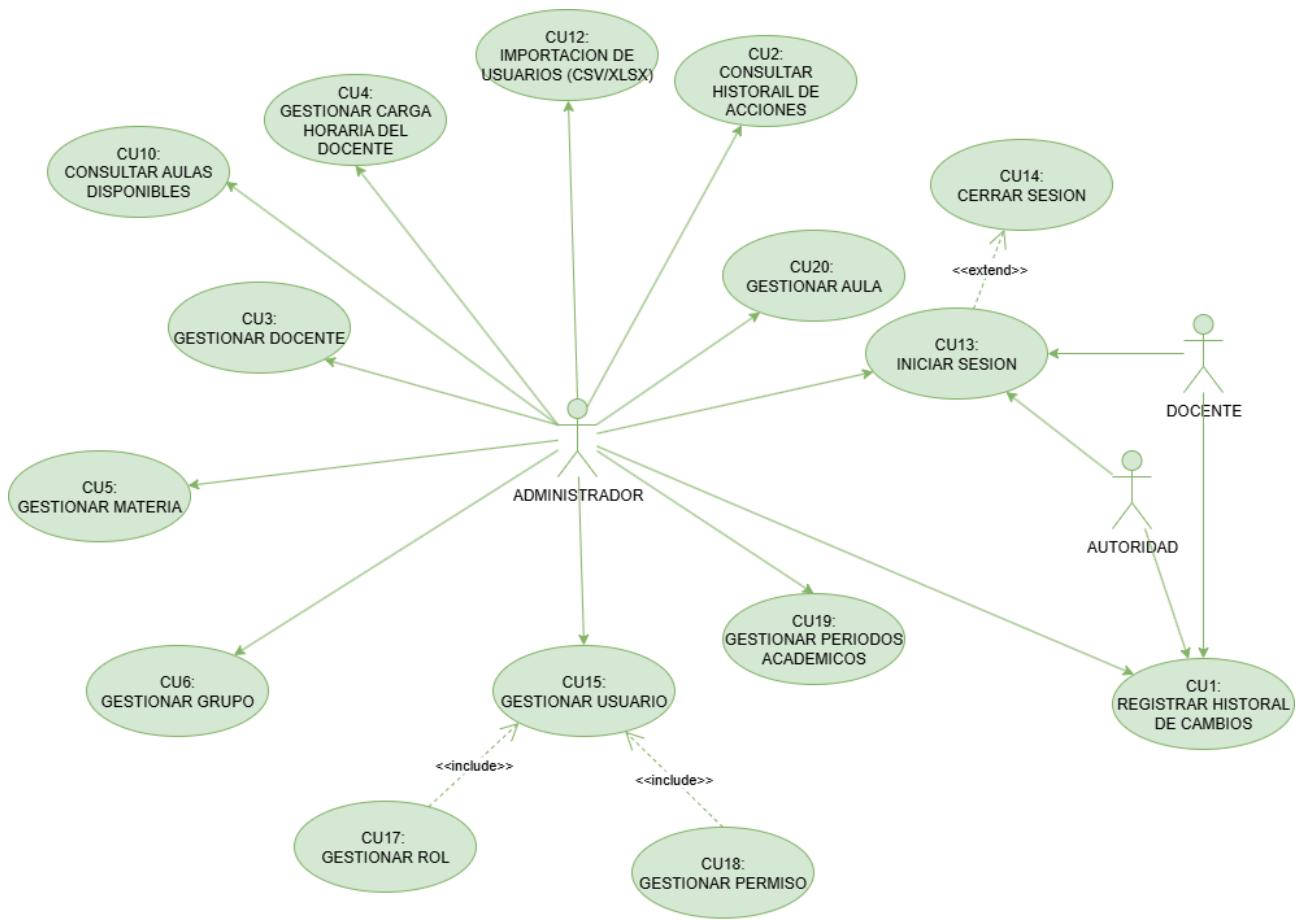
CU23: REGISTRAR ASISTENCIA POR QR O FORMULARIO WEB



Propósito	Registrar automáticamente la asistencia mediante el escaneo del código QR generado para la clase.
Resumen	Cuando el QR es escaneado, el sistema valida la información y marca la asistencia del docente o estudiante en la sesión correspondiente.
Actores	Docente.
Actor iniciador	Docente.
Flujo principal	<ul style="list-style-type: none"> ● 1. El usuario escanea el código QR. ● 2. El sistema valida la sesión y el usuario. ● 3. Registra la asistencia. ● 4. Confirma el registro exitoso.
Precondición	El código QR debe haber sido generado previamente (CU22) y el usuario debe estar autenticado.

4.4 ESTRUCTURAR MODELOS DE CASOS DE USO

CICLO 1



CICLO 2

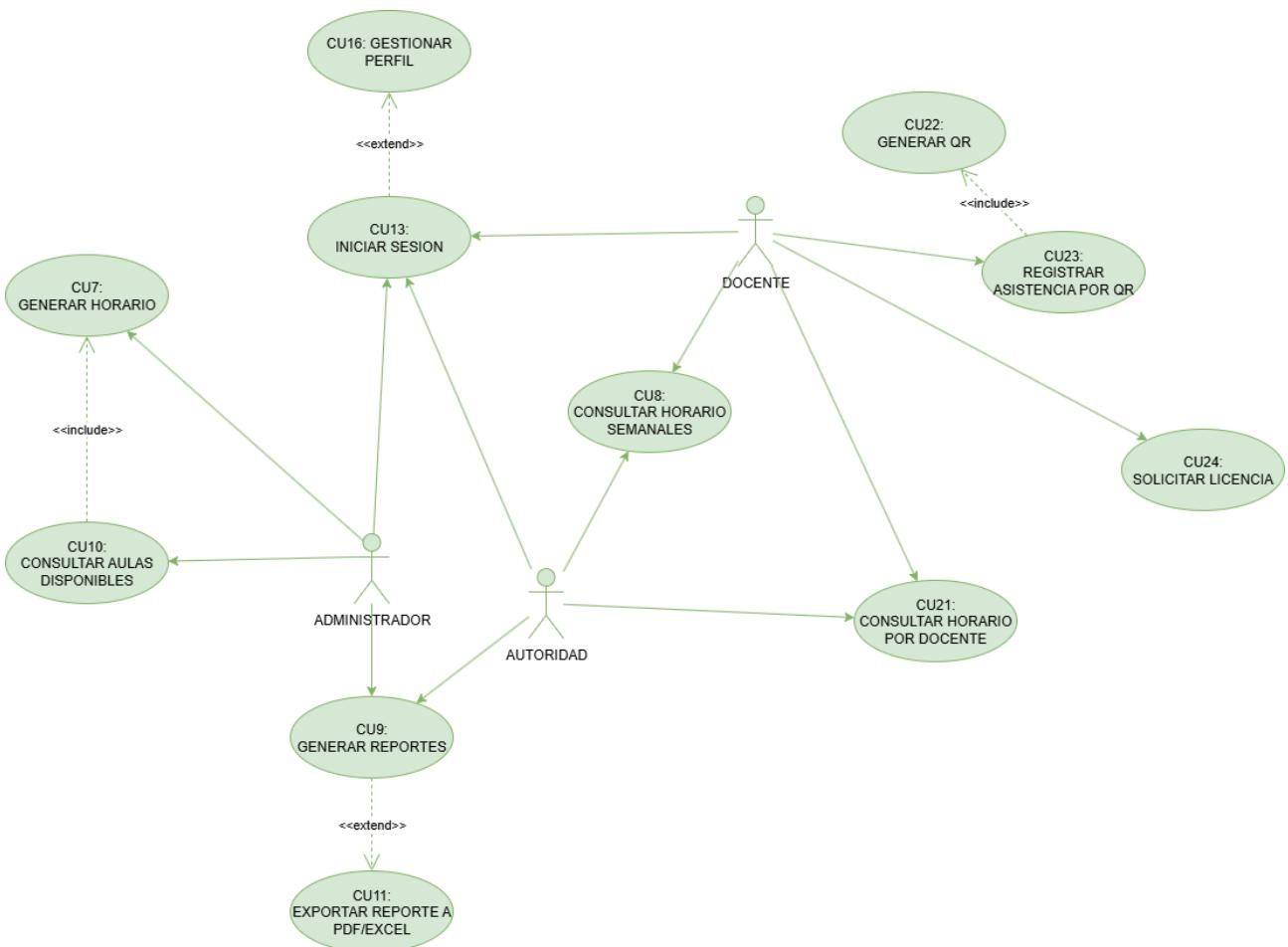
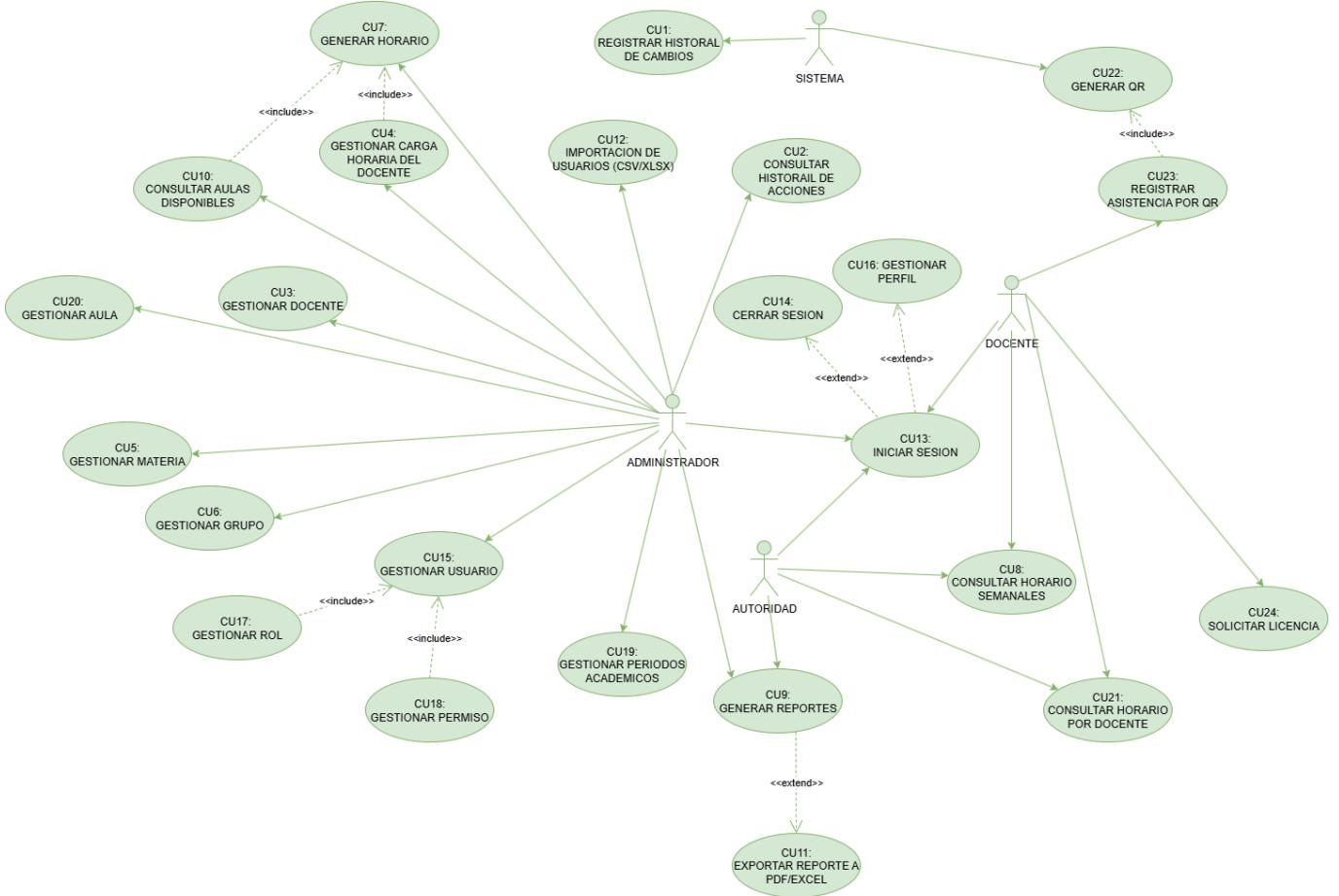


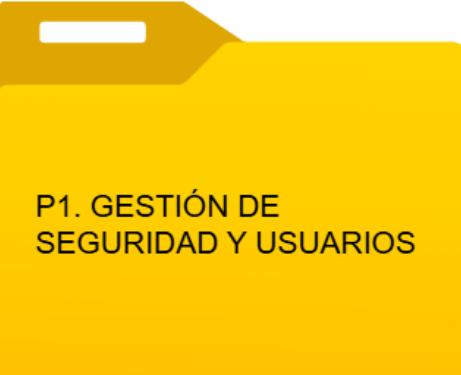
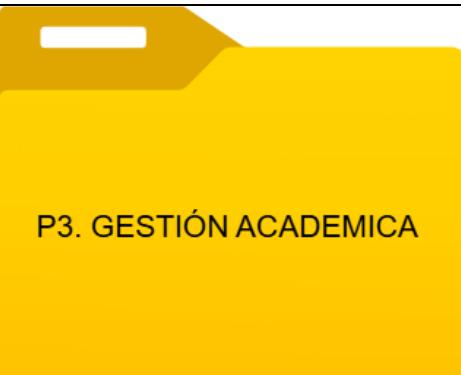
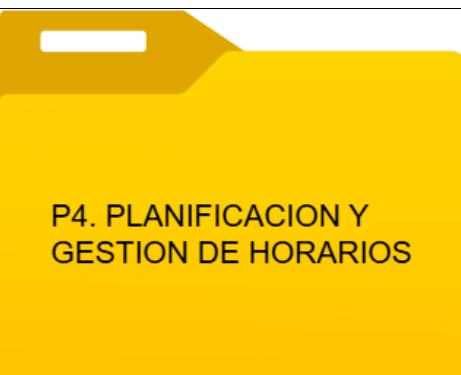
DIAGRAMA GENERAL

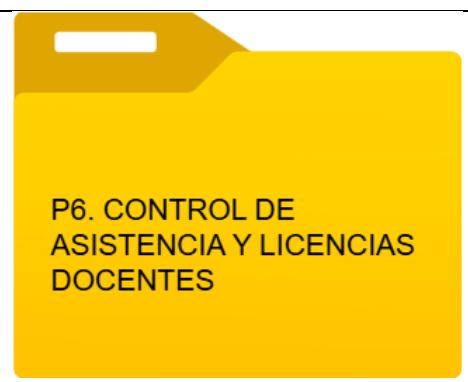


5. FLUJO DE TRABAJO: ANÁLISIS

5.1. ANÁLISIS DE ARQUITECTURA

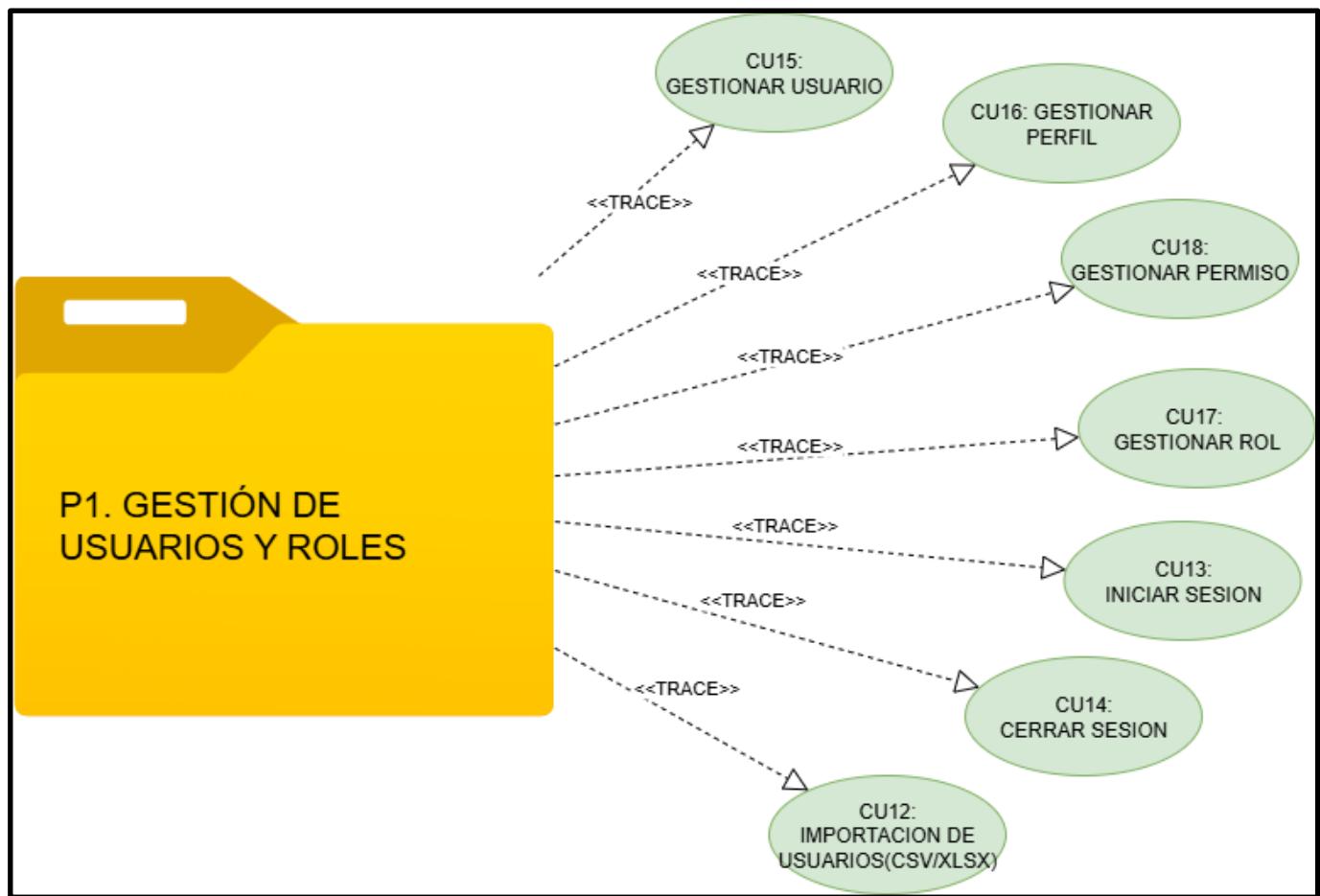
5.1.1. IDENTIFICAR PAQUETES

 <p>P1. GESTIÓN DE SEGURIDAD Y USUARIOS</p>	<p>Este paquete se encarga del control de autenticación y autorización de los actores. Permite el registro, actualización y eliminación de usuarios, la definición de roles, perfiles y permisos, así como el inicio y cierre de sesión.</p>
 <p>P2. AUDITORIA Y CONTROL DE CAMBIOS</p>	<p>Permite al sistema mantener trazabilidad de todas las operaciones críticas realizadas por los usuarios. Facilita la supervisión, auditoría y rendición de cuentas de las acciones en la plataforma. Su propósito es registrar, almacenar y consultar el historial de acciones realizadas en el sistema.</p>
 <p>P3. GESTIÓN ACADÉMICA</p>	<p>Permite mantener actualizada la información académica que sirve como base para la generación de horarios. Incluye docentes, carreras, materias, grupos, aulas y períodos académicos. Su propósito es administrar toda la estructura académica institucional necesaria para la asignación de horarios.</p>
 <p>P4. PLANIFICACIÓN Y GESTIÓN DE HORARIOS</p>	<p>Contiene los procesos de creación automática y manual de horarios, integrando información de docentes, aulas y materias. Permite también la consulta personalizada de horarios. Su propósito es generar, administrar y consultar horarios de clases y disponibilidad de aulas.</p>

 <p>P5. REPORTES Y EXPORTACIONES</p>	<p>Permite generar reportes personalizados sobre horarios, asistencias, docentes y períodos, con posibilidad de exportarlos a diferentes formatos. También posibilita la carga masiva de información. Su propósito es facilitar la generación y exportación de reportes académicos y administrativos</p>
 <p>P6. CONTROL DE ASISTENCIA Y LICENCIAS DOCENTES</p>	<p>Este paquete abarca la funcionalidad de control de asistencia y ausencias del personal docente, mediante el uso de códigos QR y gestión de licencias. Su propósito es gestionar la asistencia docente mediante QR y el manejo de licencias o ausencias.</p>

5.1.2. RELACIONAR PAQUETES Y CASOS DE USO

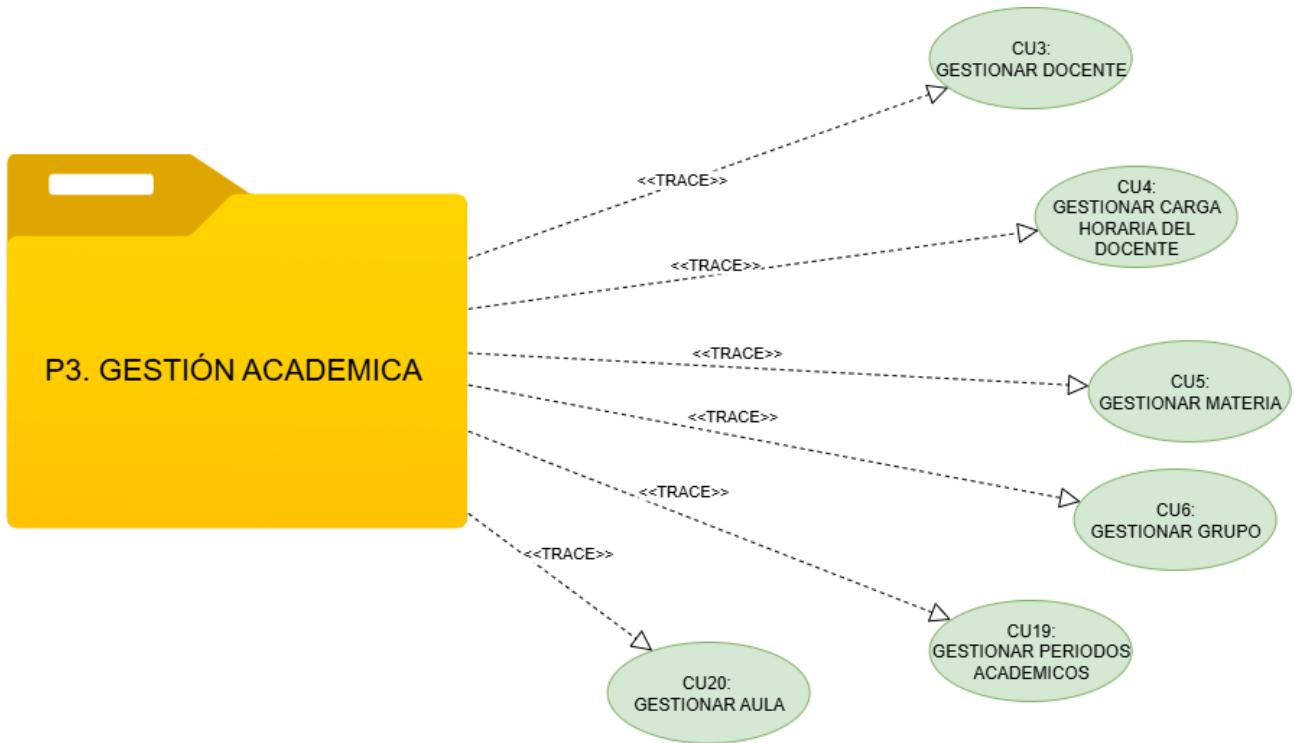
P1. GESTIÓN DE USUARIO Y ROLES



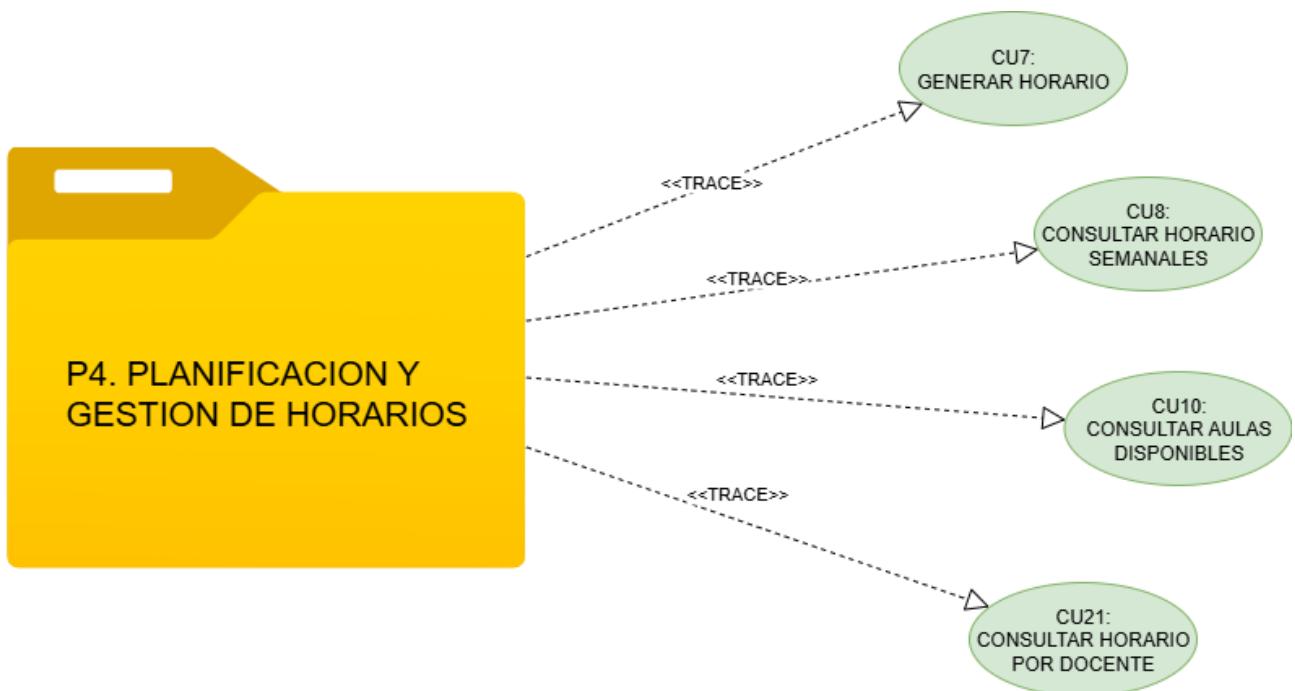
P2. AUDITORIA Y CONTROL DE CAMBIOS



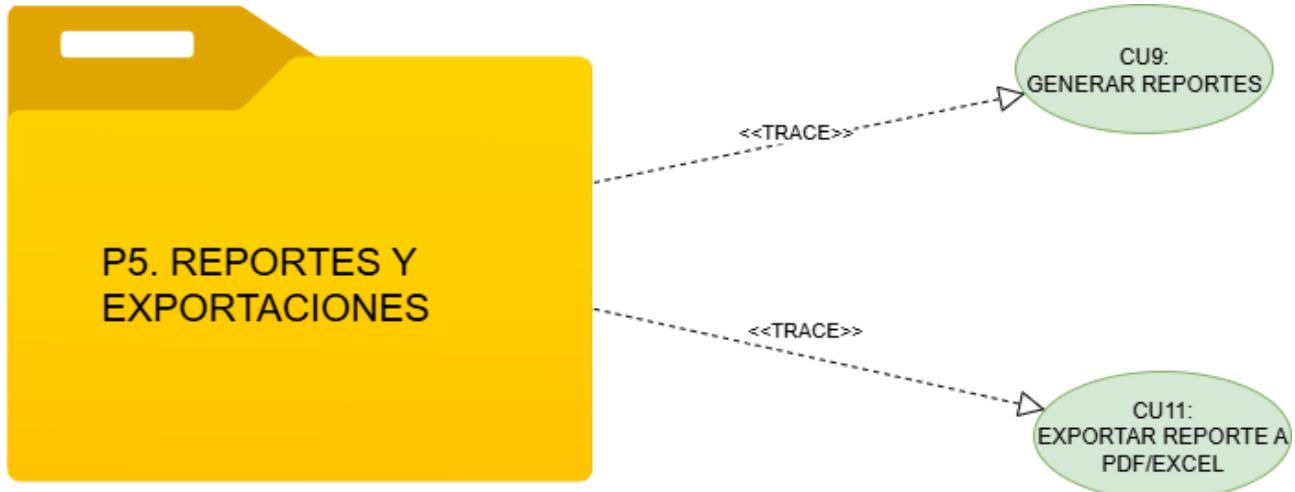
P3. GESTIÓN ACADÉMICA



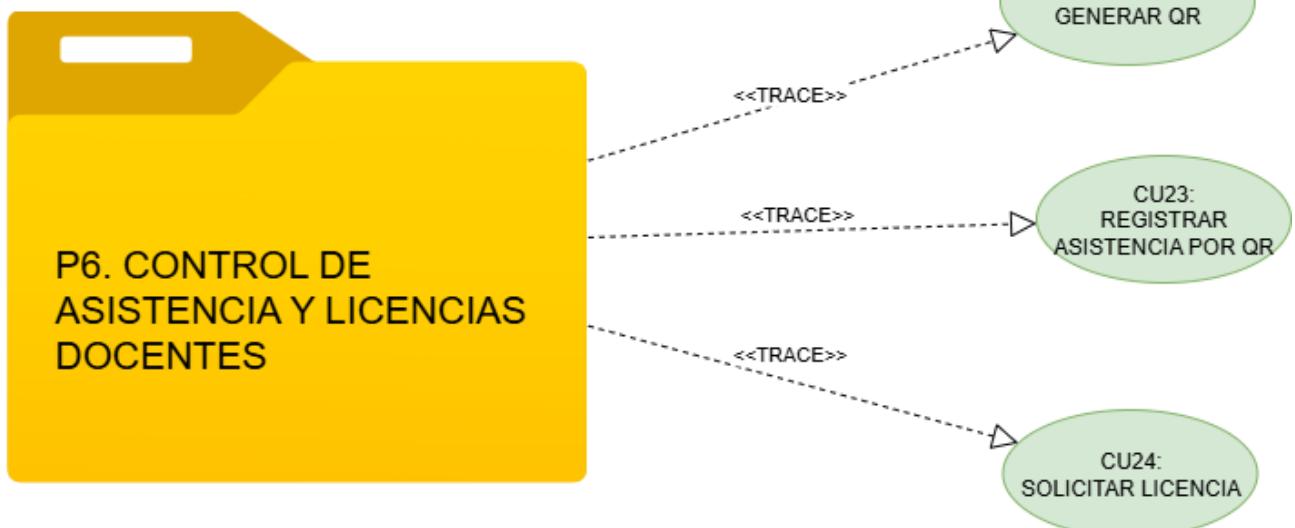
P4. PLANIFICACION Y GESTION DE HORARIOS



P5. ADMINISTRACIÓN DE SISTEMA

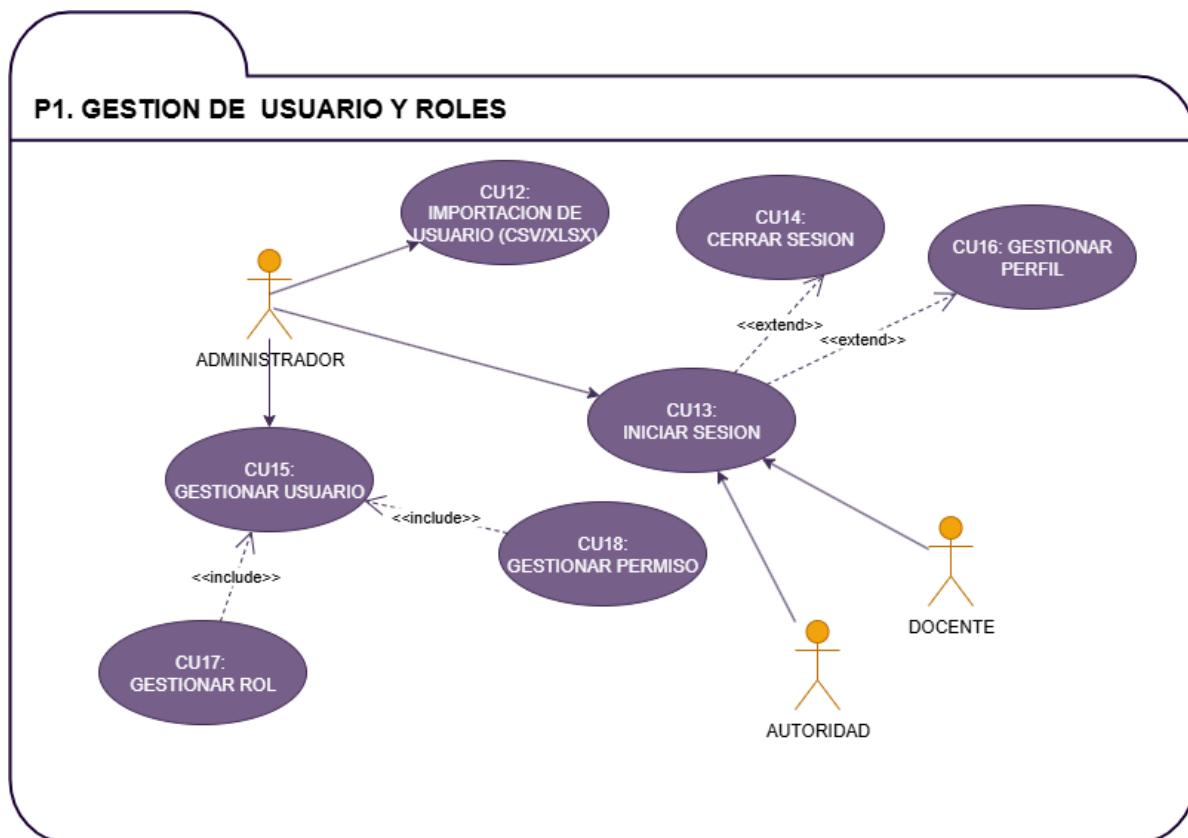


P6. CONTROL DE ASISTENCIA Y LICENCIAS DOCENTES

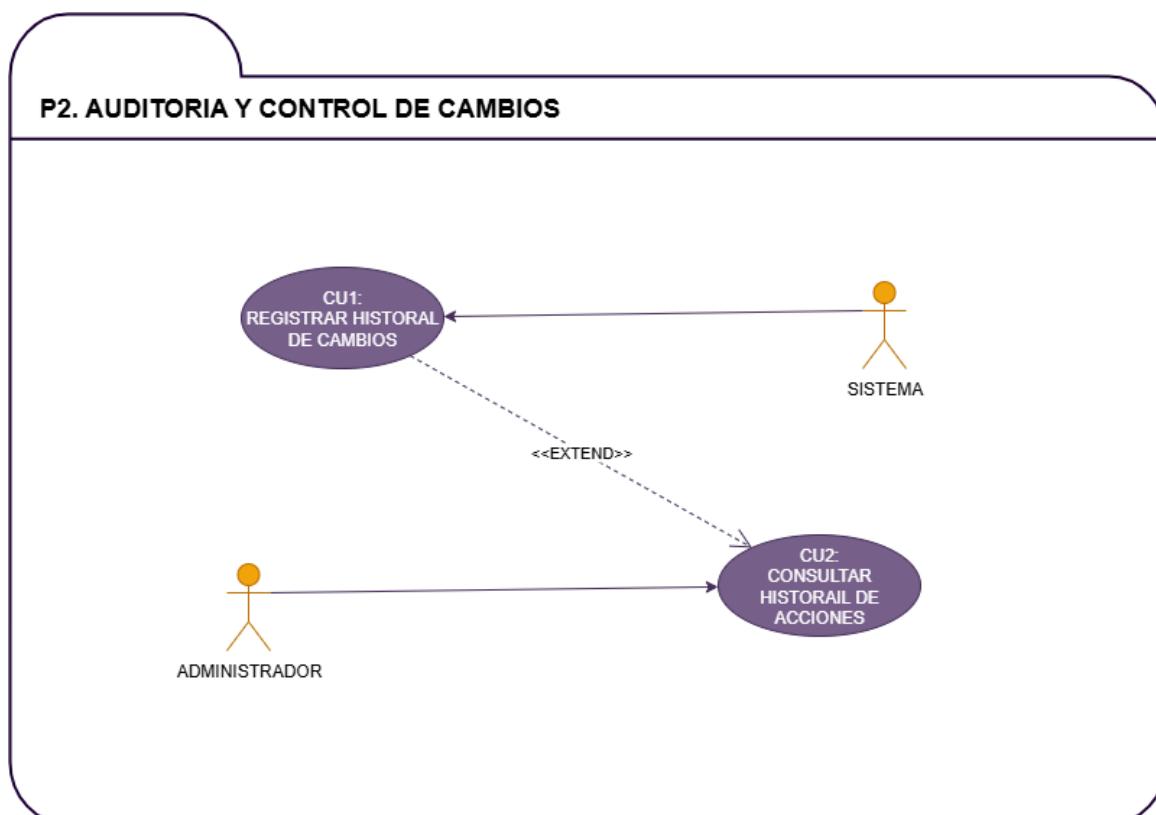


5.1.3. VISTA DE PAQUETE

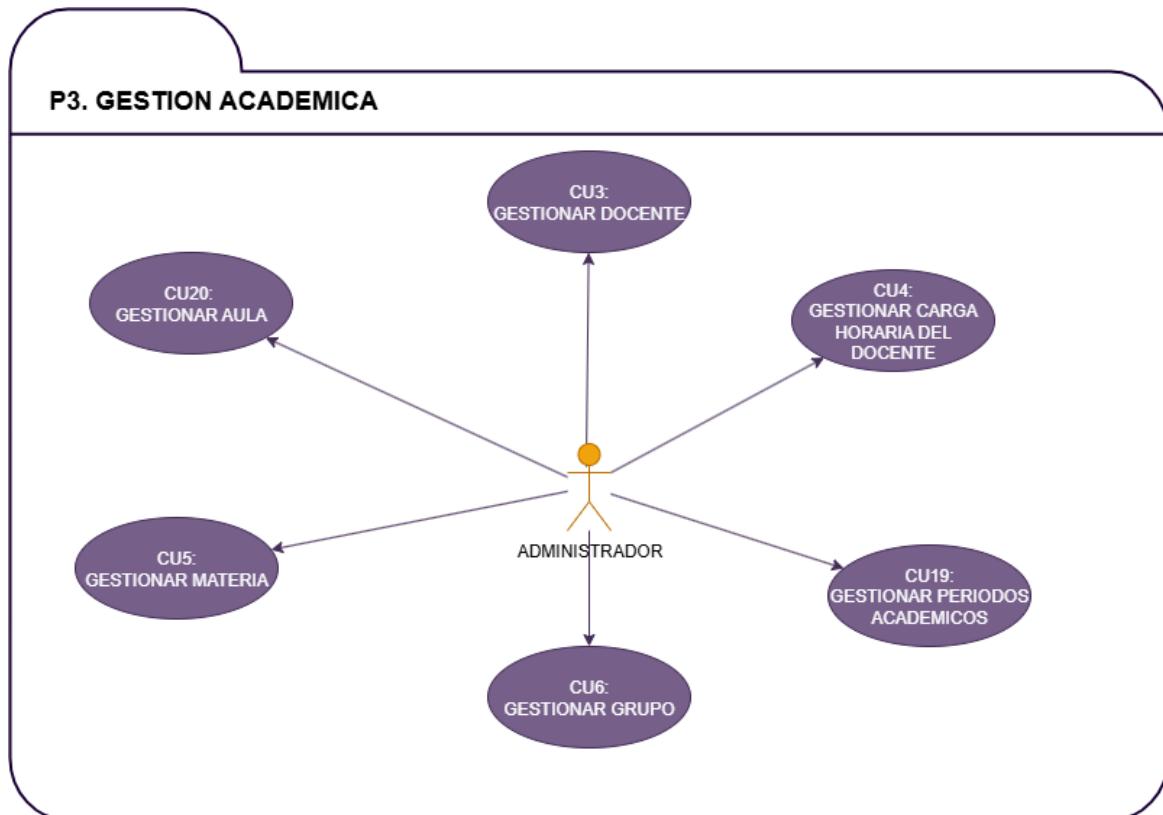
PAQUETE 1.- GESTION DE SEGURIDAD Y USUARIO



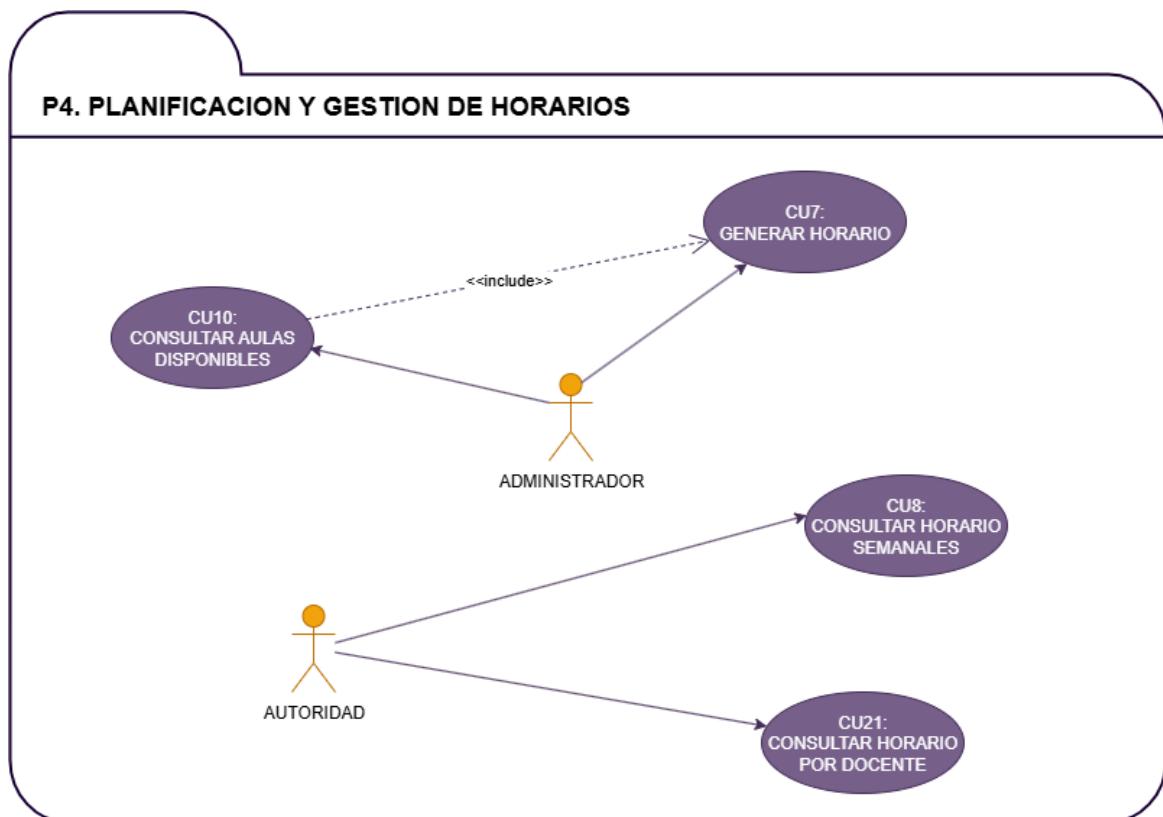
PAQUETE 2.- AUDITORIA Y CONTROL DE CAMBIOS



PAQUETE 3.- GESTIÓN ACADEMICA

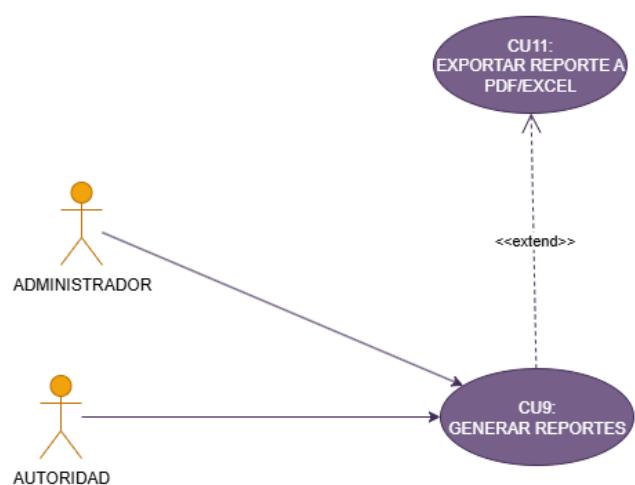


PAQUETE 4.- PLANIFICACION Y GESTION DE HORARIOS

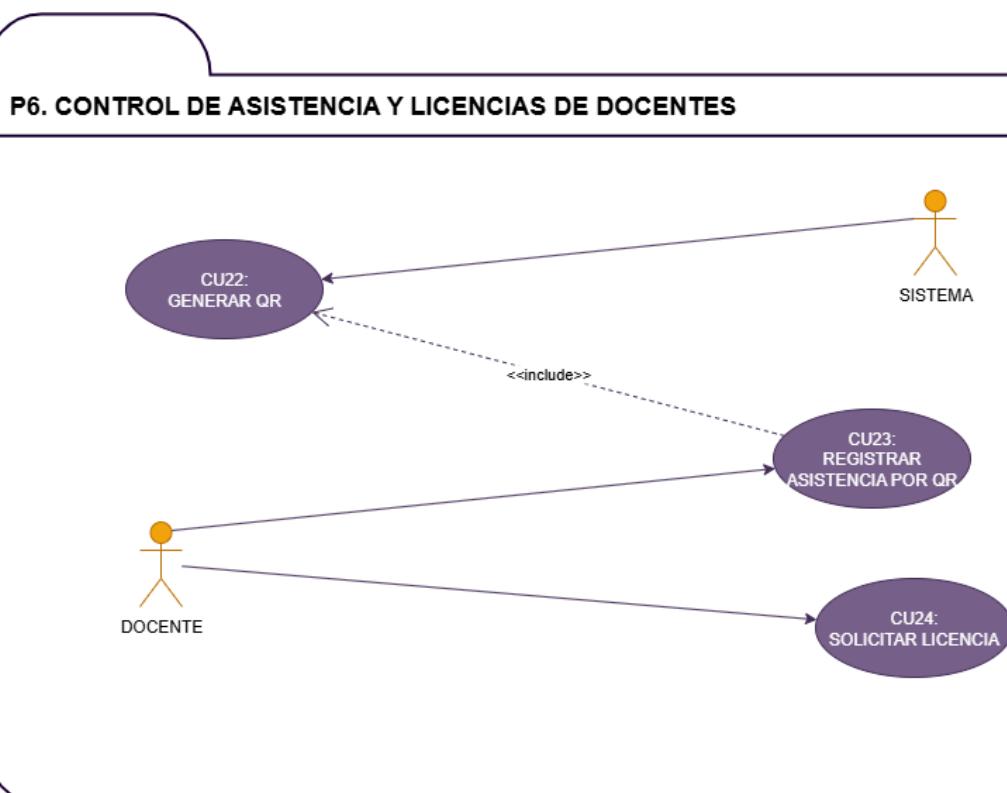


PAQUETE 5.- ADMINISTRACIÓN DE SISTEMA

P5. REPORTES Y EXPORTACIONES



PAQUETE 6.- CONTROL DE ASISTENCIA Y LICENCIAS DOCENTES

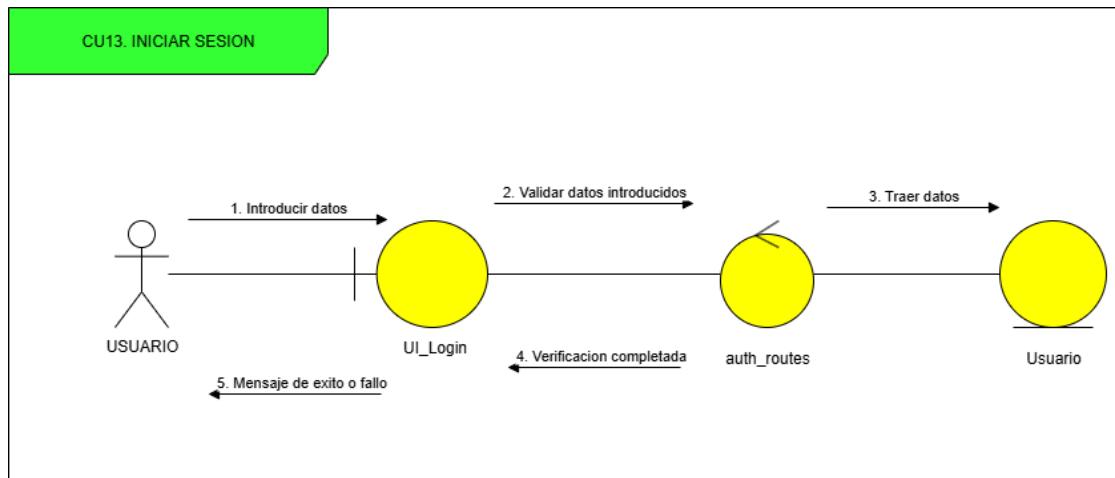


5.2 ANÁLISIS DE CASO DE USO

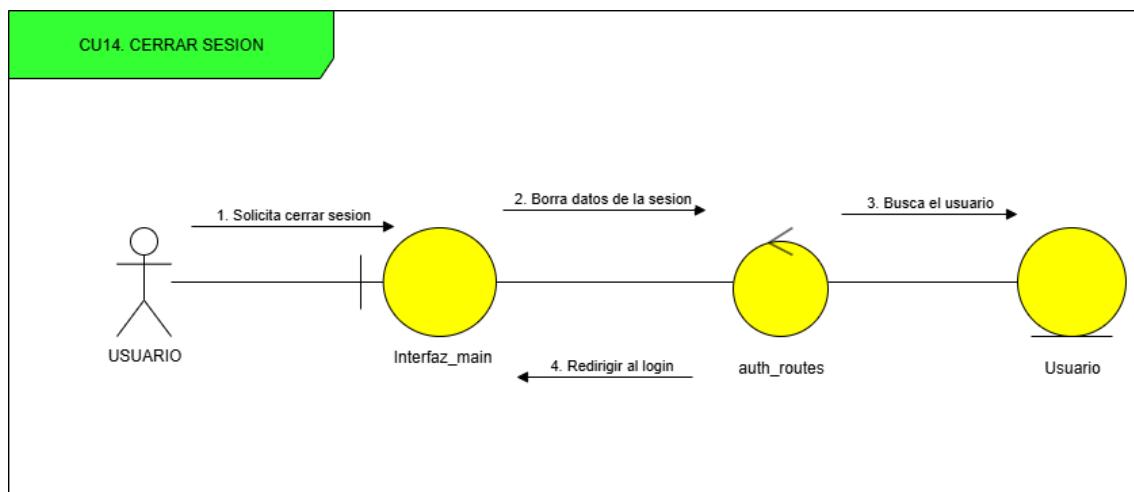
5.2.1 DIAGRAMA DE COMUNICACIÓN

CICLO #1

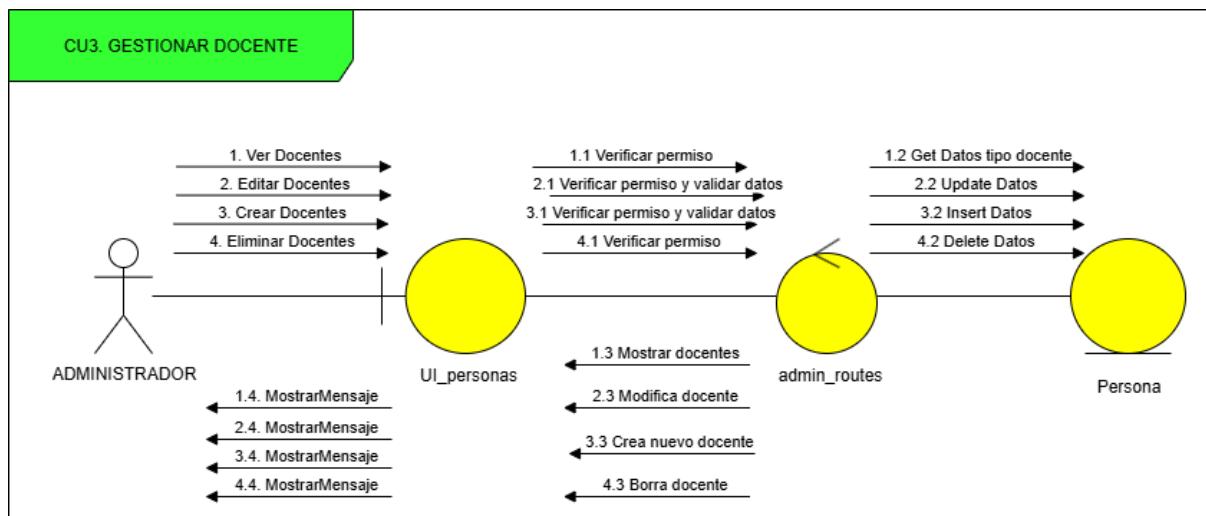
CU13. INICIAR SESIÓN



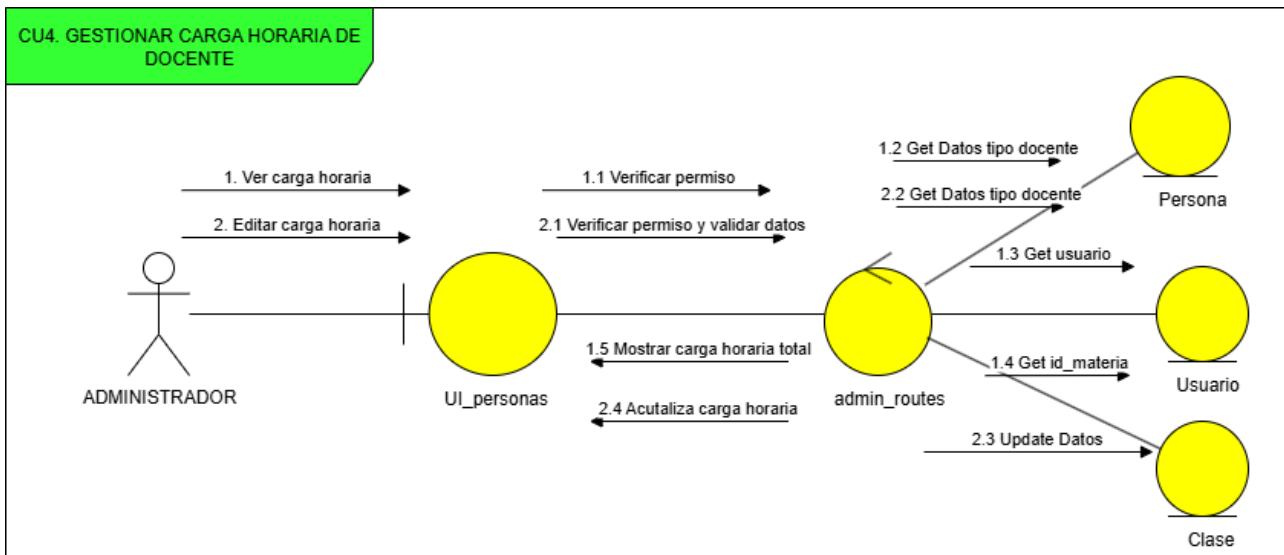
CU14. CERRAR SESIÓN



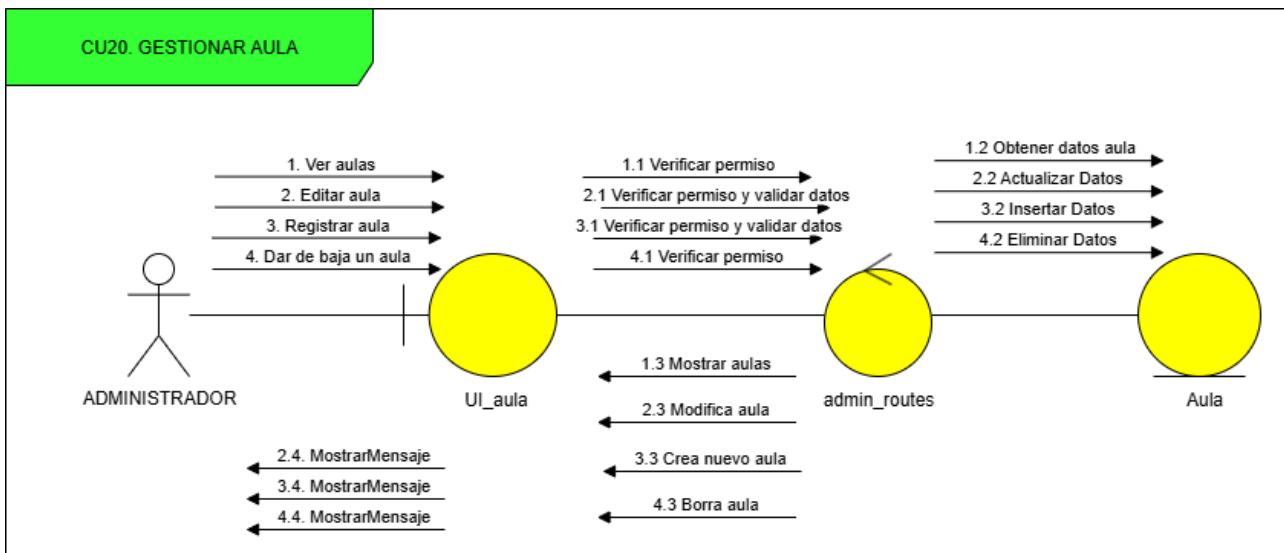
CU3. GESTIONAR DOCENTE



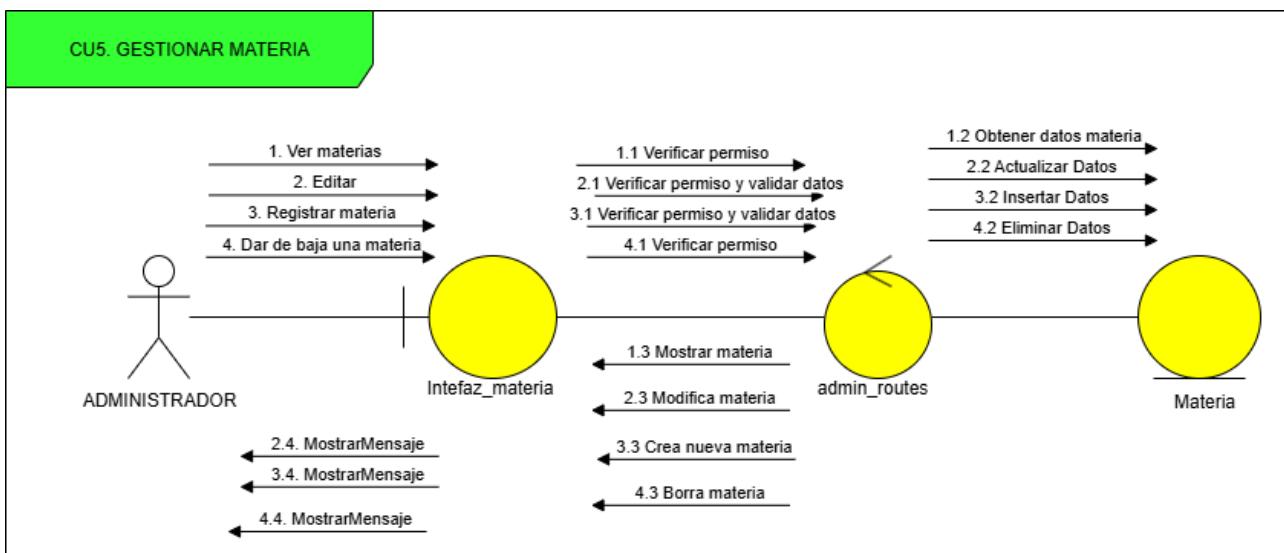
CU4. GESTIONAR CARGA HORARIA DE DOCENTE



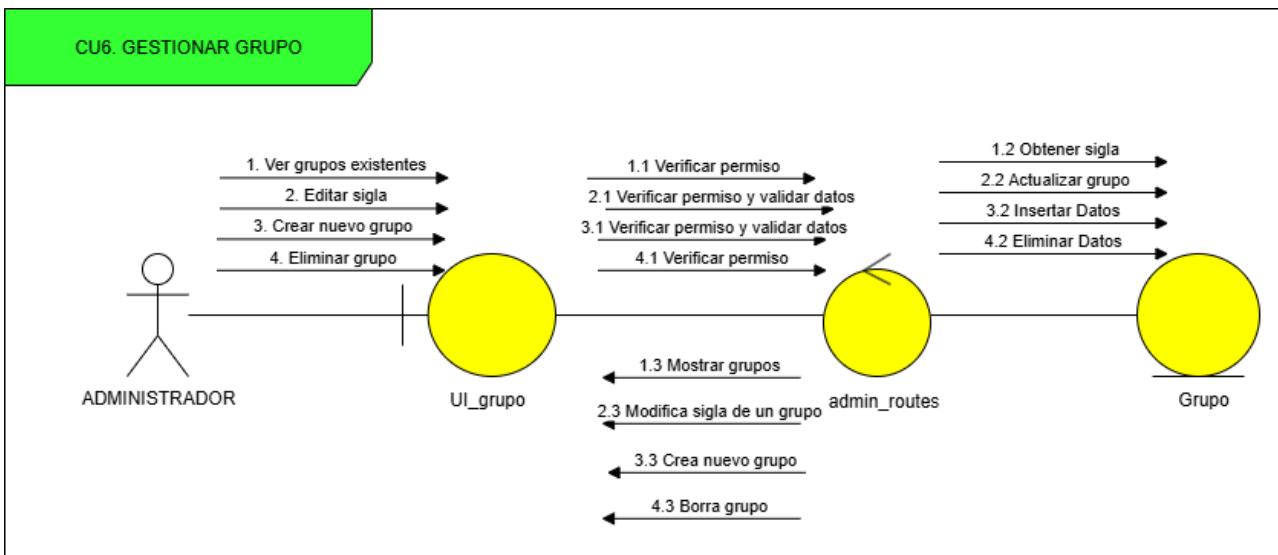
CU20. GESTIONAR AULA



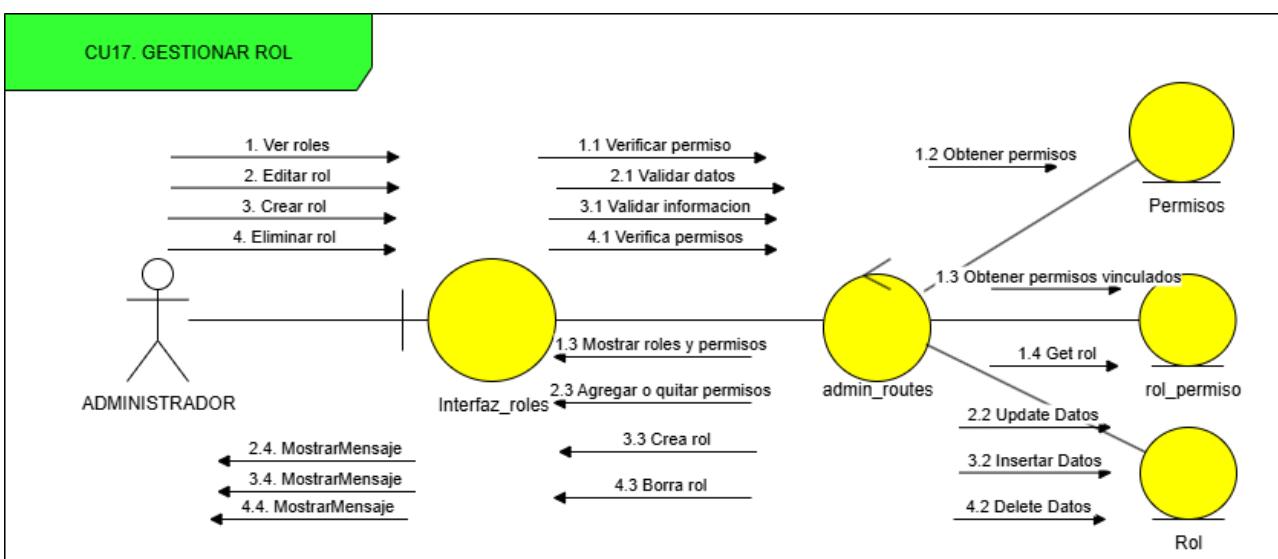
CU5. GESTIONAR MATERIA



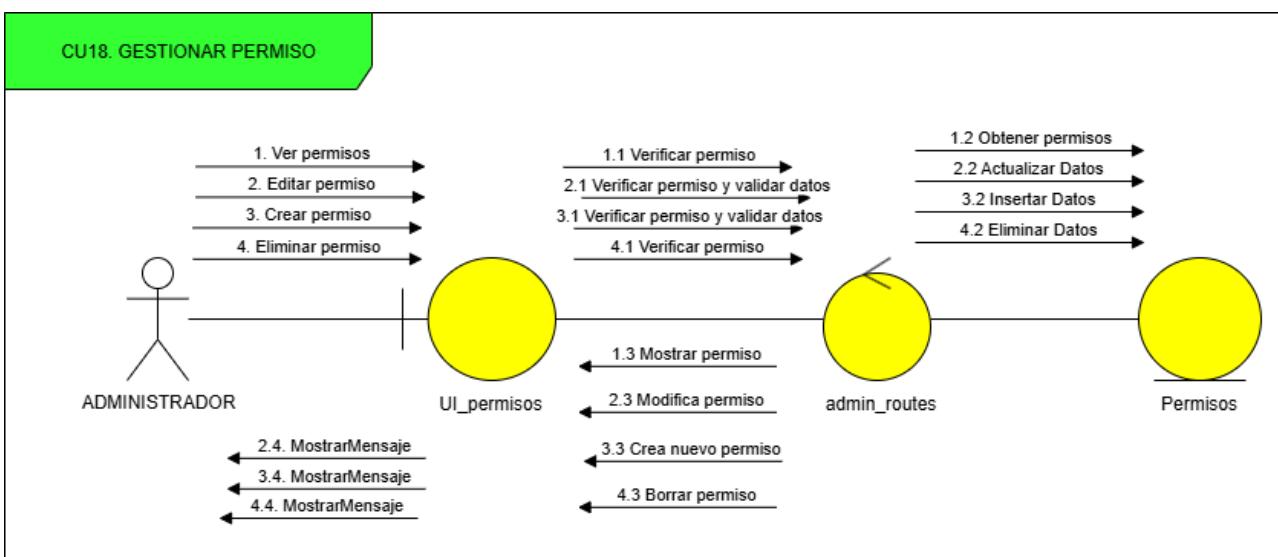
CU6. GESTIONAR GRUPO



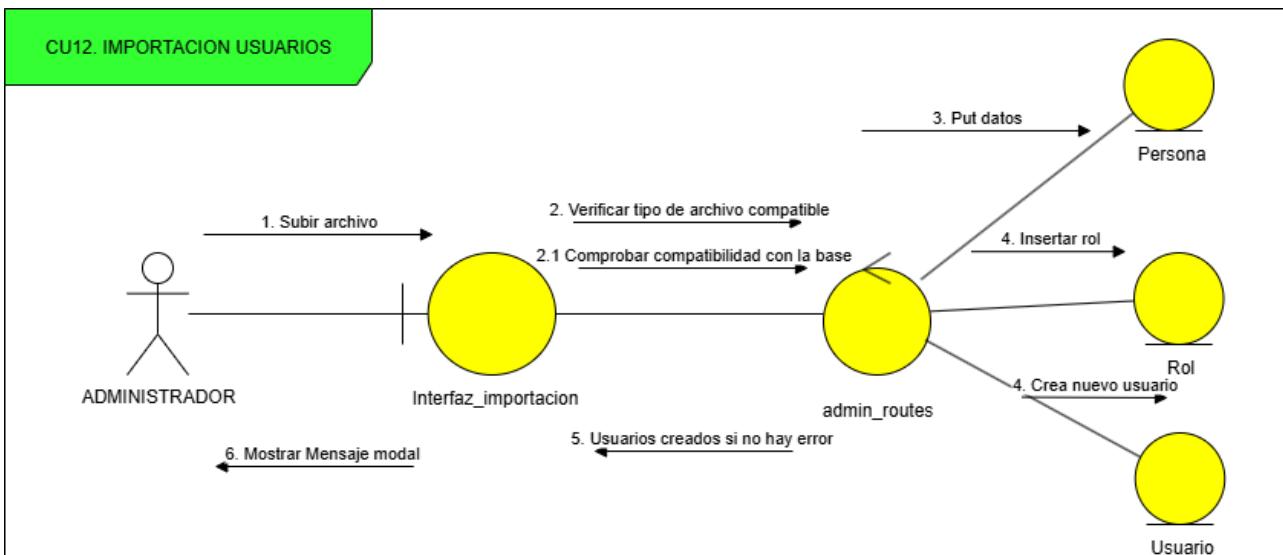
CU17. GESTIONAR ROL



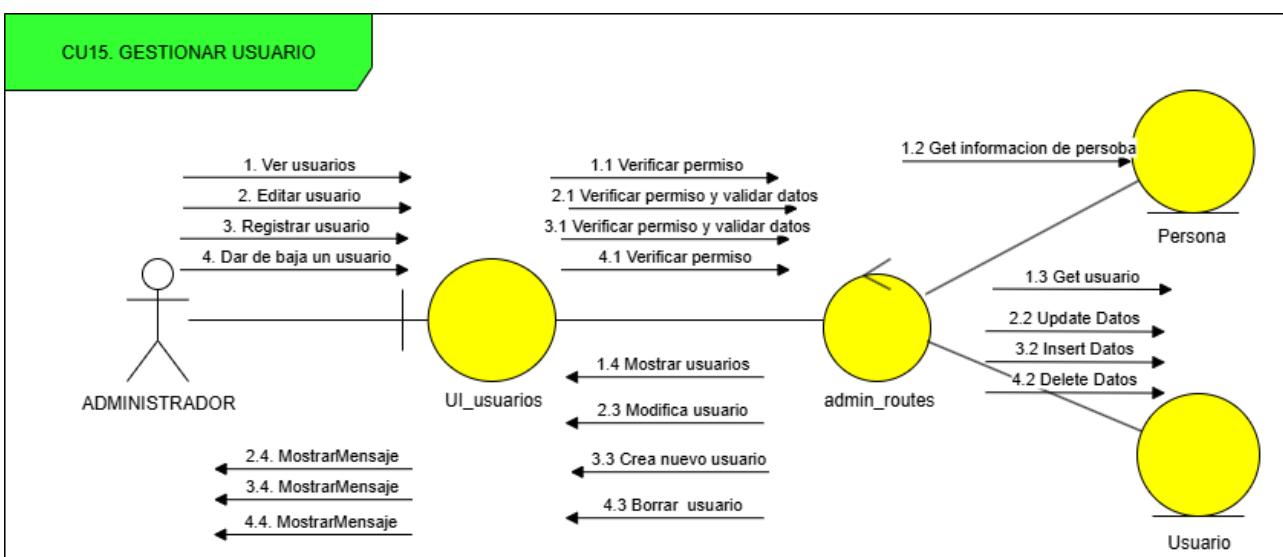
CU18. GESTIONAR PERMISO



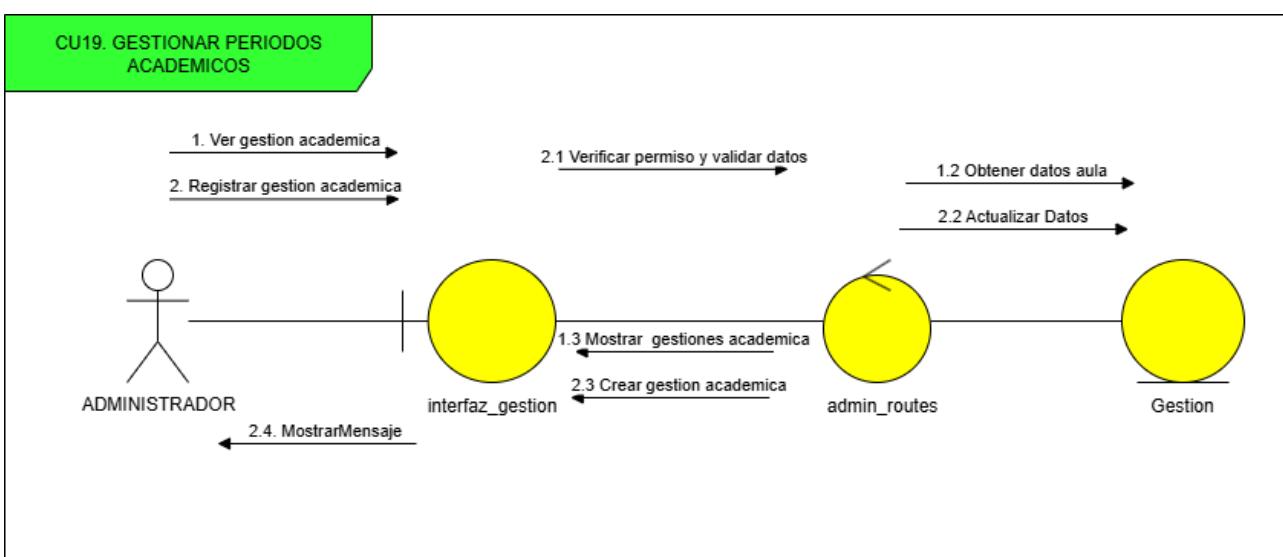
CU12. IMPORTAR USUARIOS



CU15. GESTIONAR USUARIO

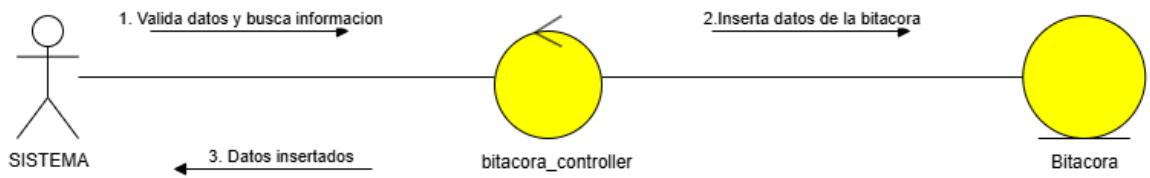


CU19. GESTIONAR PERIODOS ACADÉMICOS



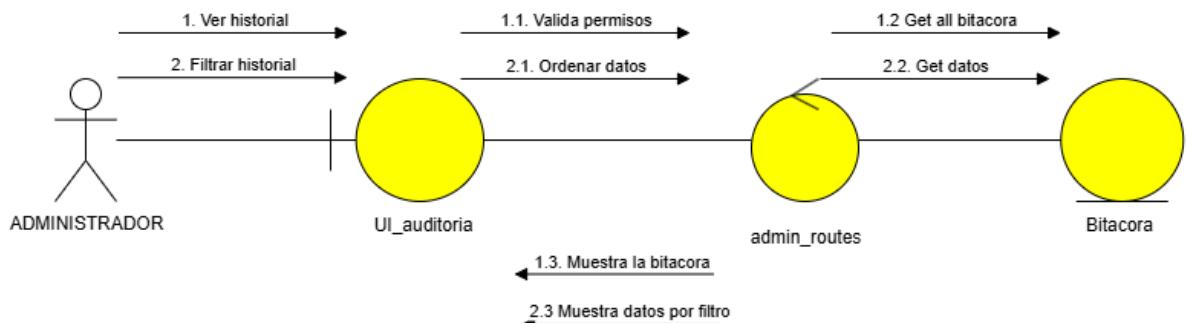
CU1. REGISTRAR HISTORIAL DE CAMBIOS

CU1. REGISTRAR HISTORIAL DE CAMBIOS



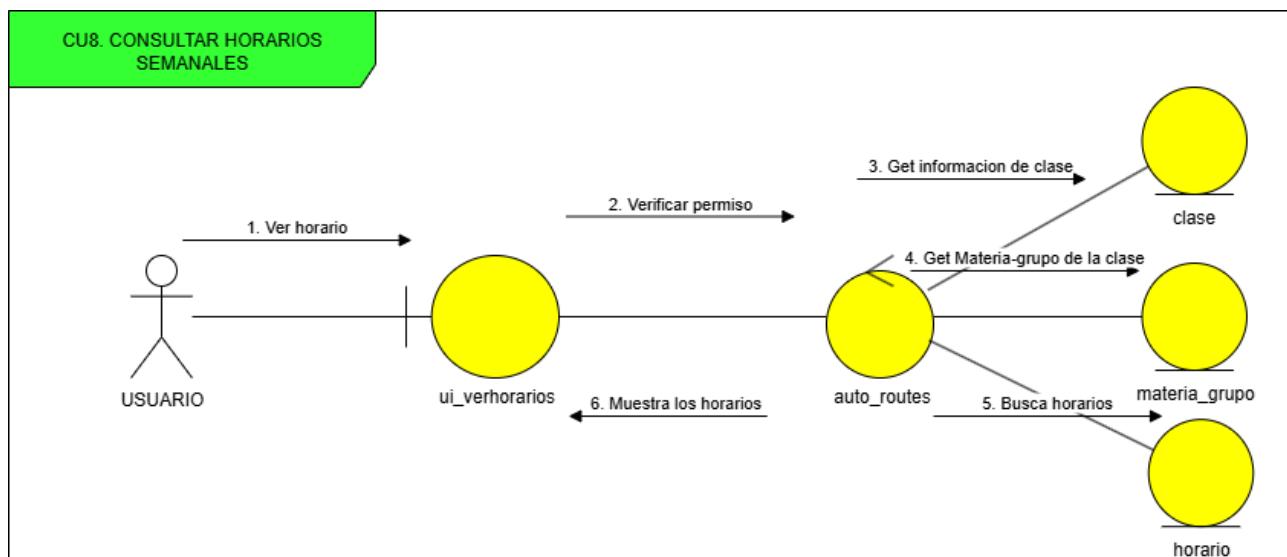
CU2. CONSULTAR HISTORIAL DE ACCIONES

CU2.1. Consultar historial de acciones

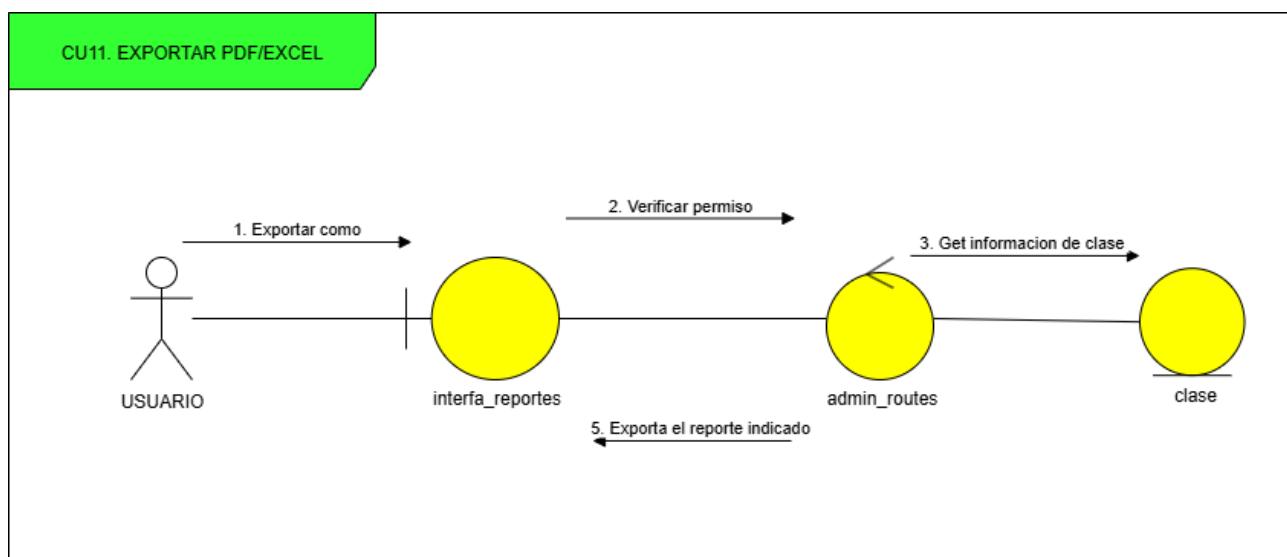


CICLO #2

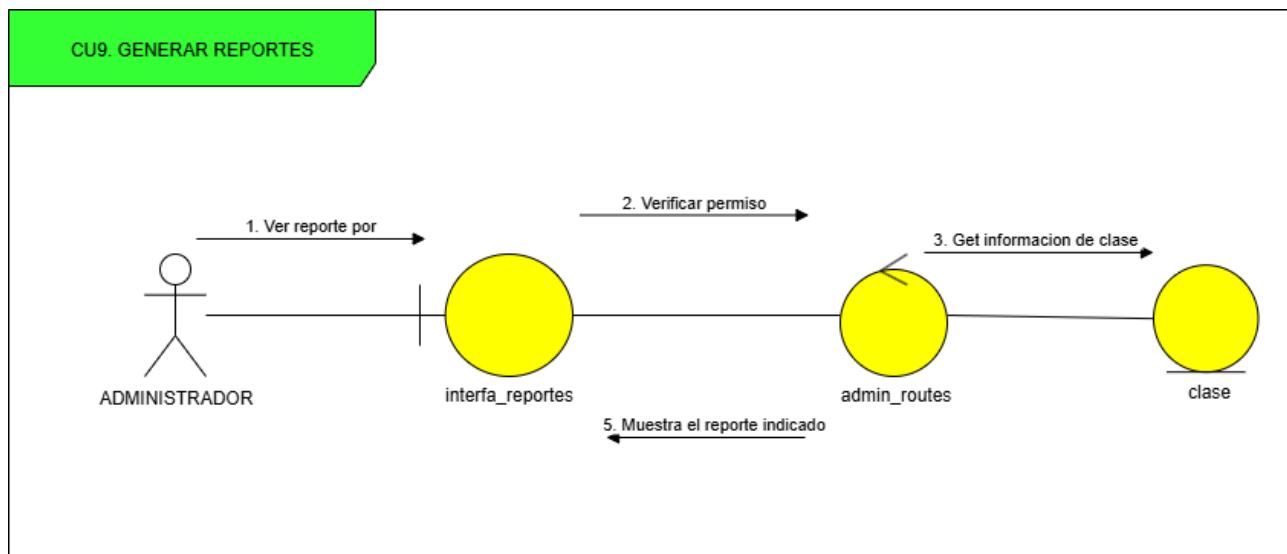
CU8: CONSULTAR HORARIOS SEMANALES



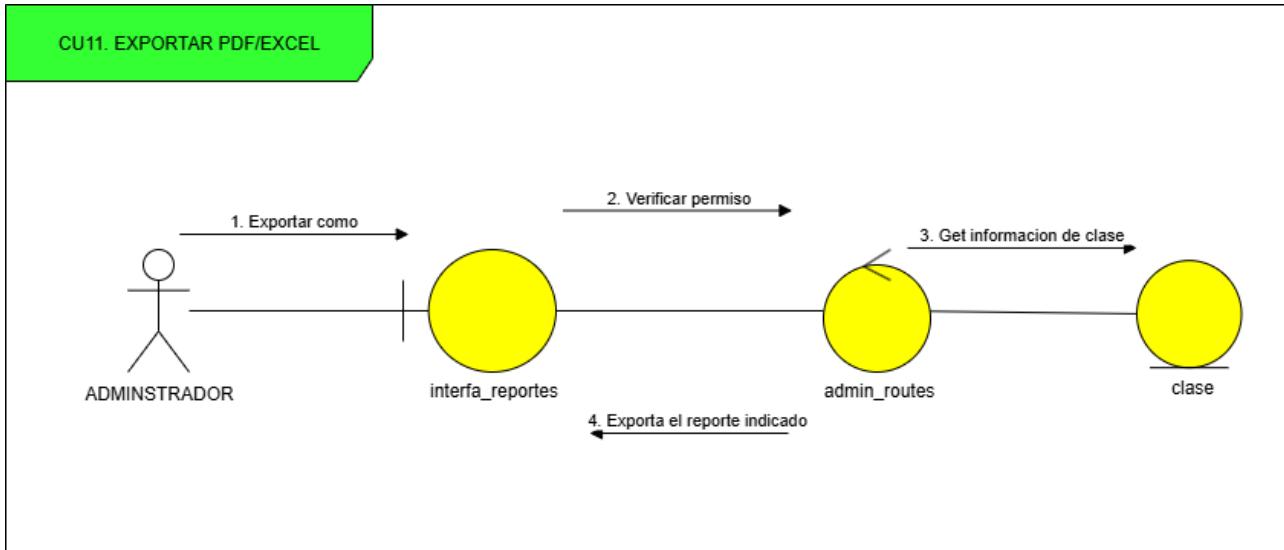
CU21: CONSULTAR HORARIO POR DOCENTE



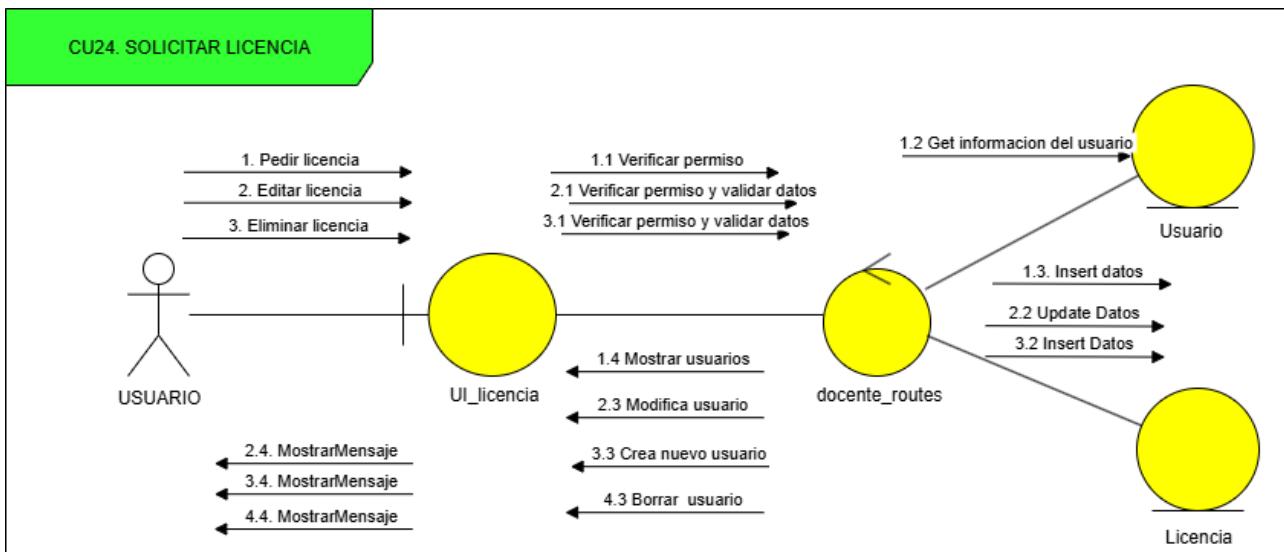
CU9: GENERAR REPORTES



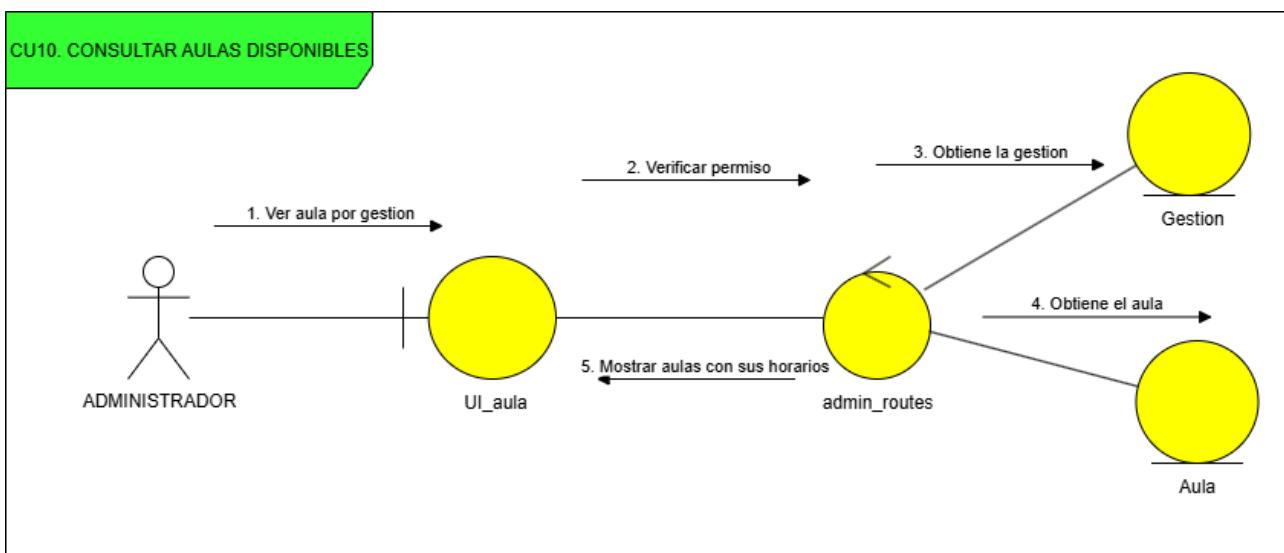
CU11: EXPORTAR REPORTE A PDF/EXCEL



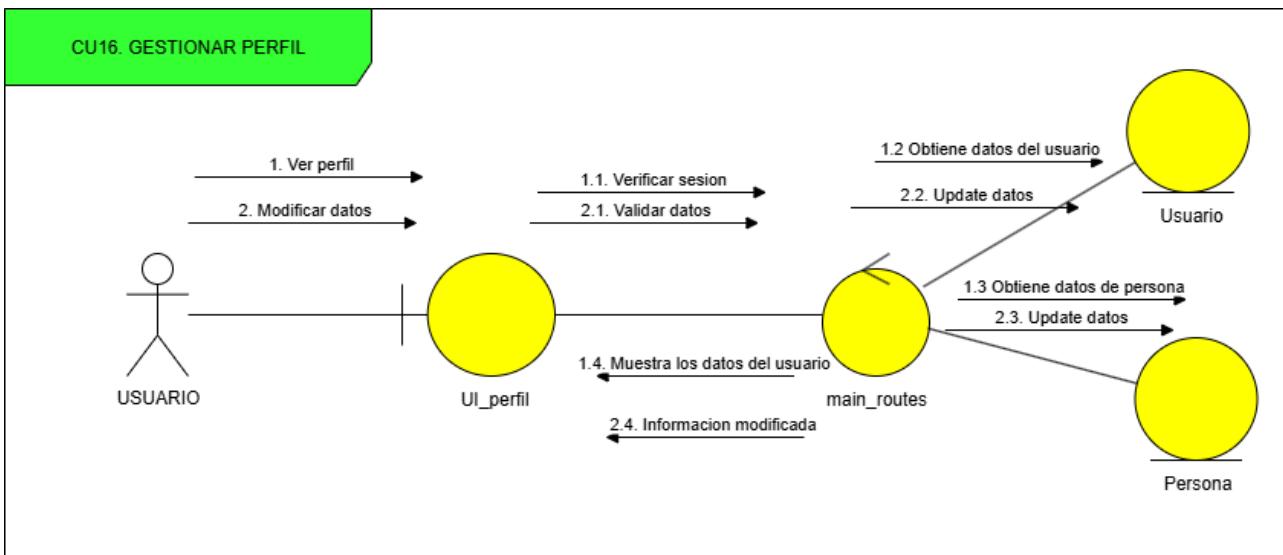
CU24: SOLICITAR LICENCIA



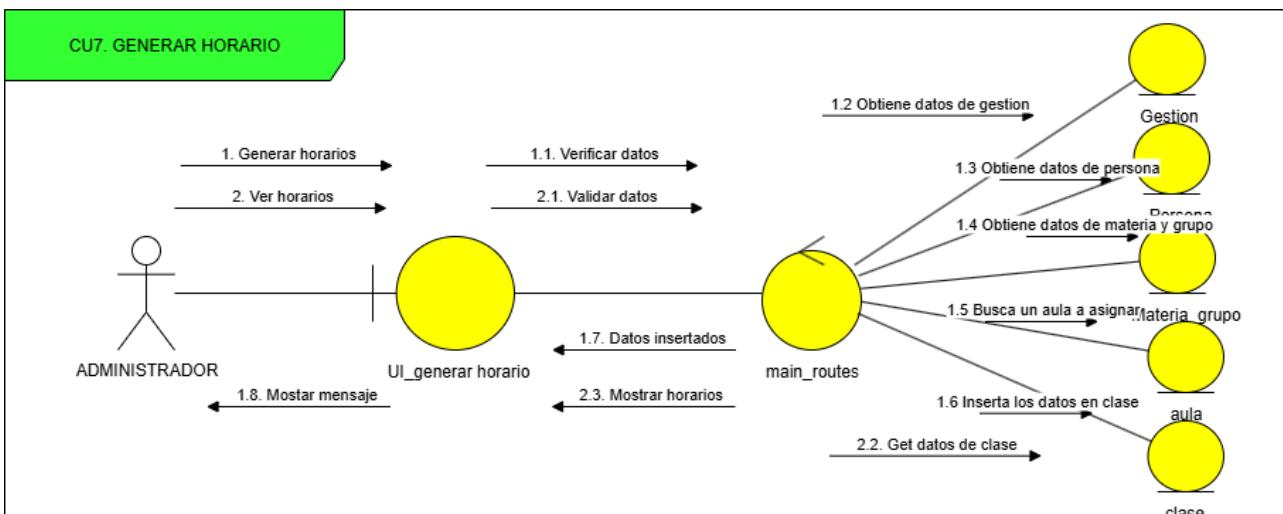
CU10: CONSULTAR AULAS DISPONIBLES



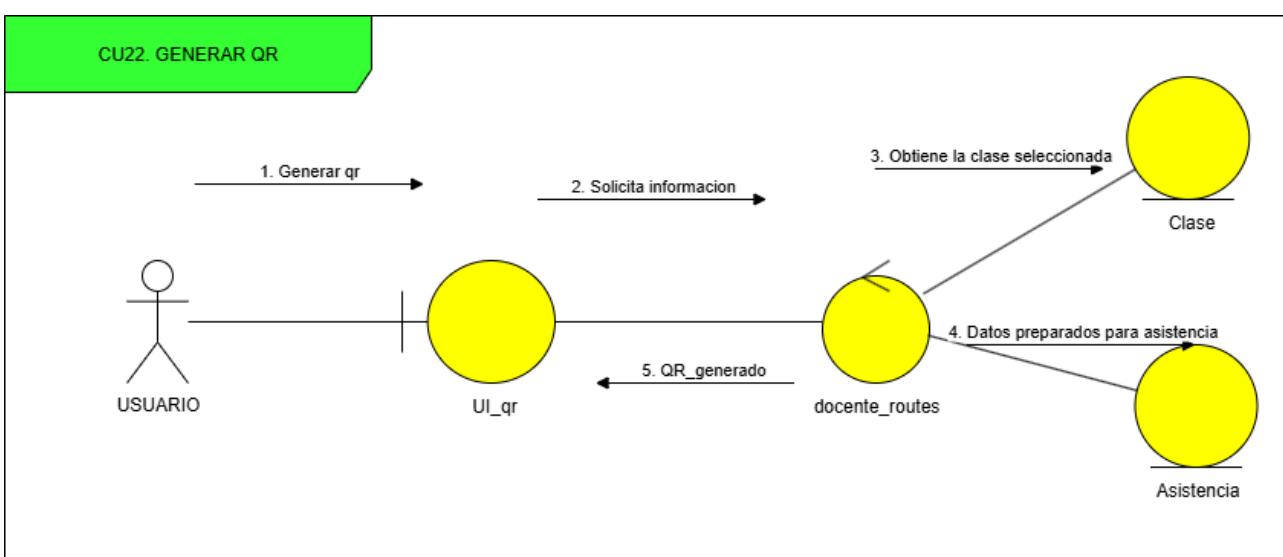
CU16: GESTIONAR PERFIL



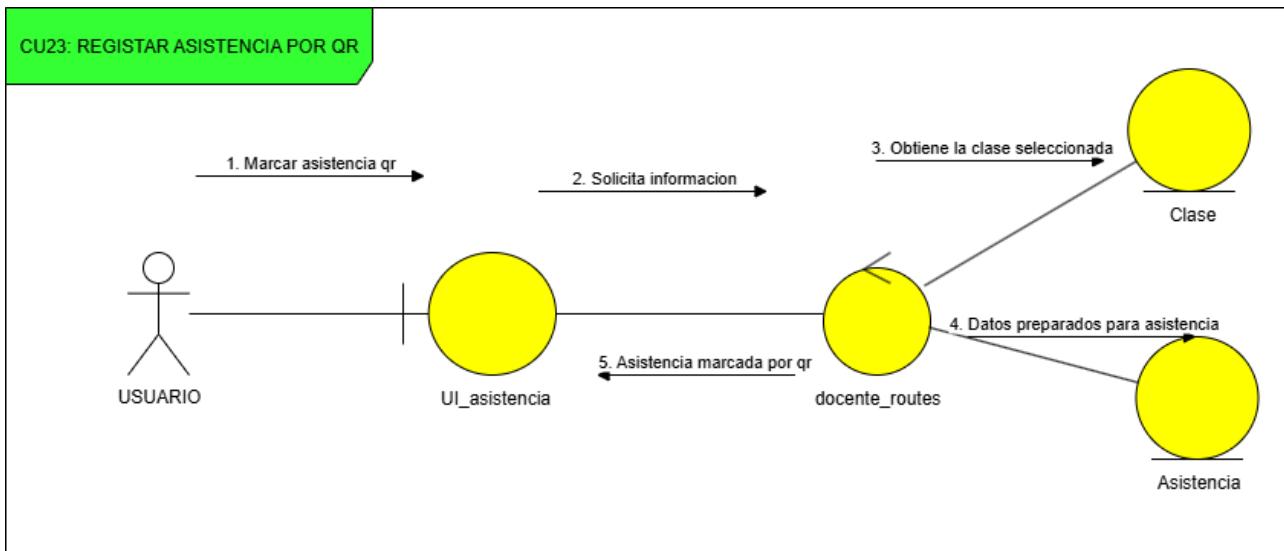
CU7: GENERAR HORARIO



CU22: GENERAR QR



CU23: REGISTRAR ASISTENCIA POR QR O FORMULARIO WEB



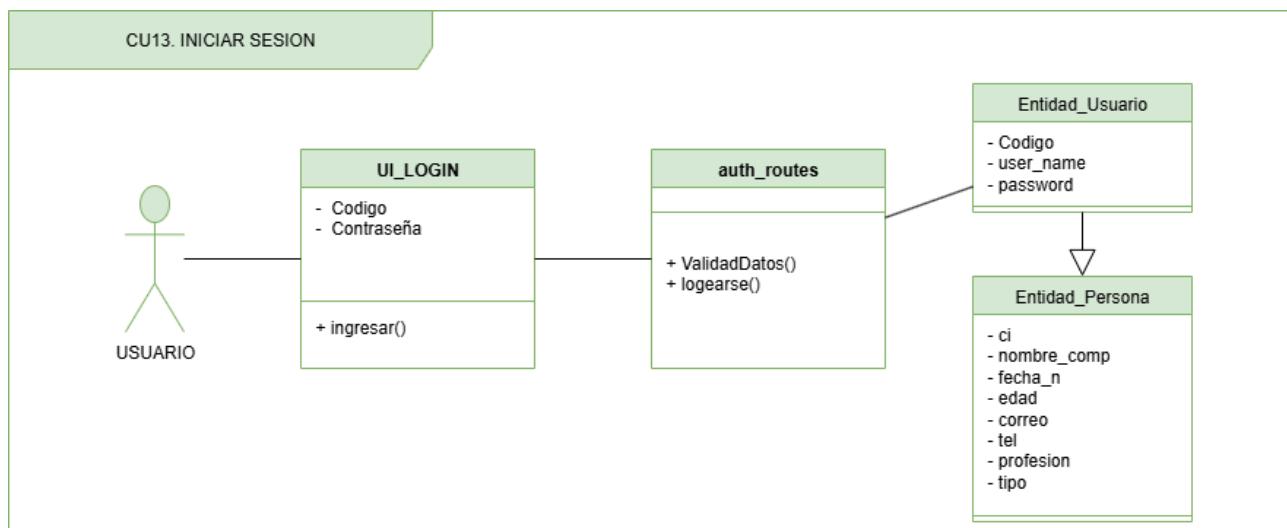
5.3 ANALIZAR UNA CLASE

CICLO #1

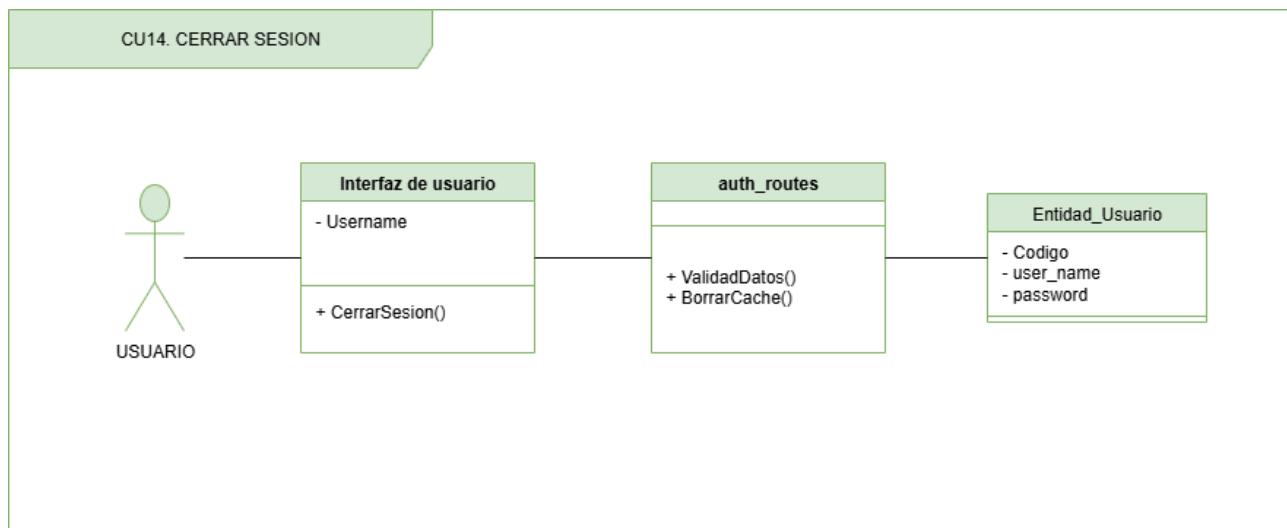
Modulos:

- Rutas de autenticación.- auth_routes.php
- Rutas principales.- main_routes.php
- Rutas de administrador.- admin_routes.php

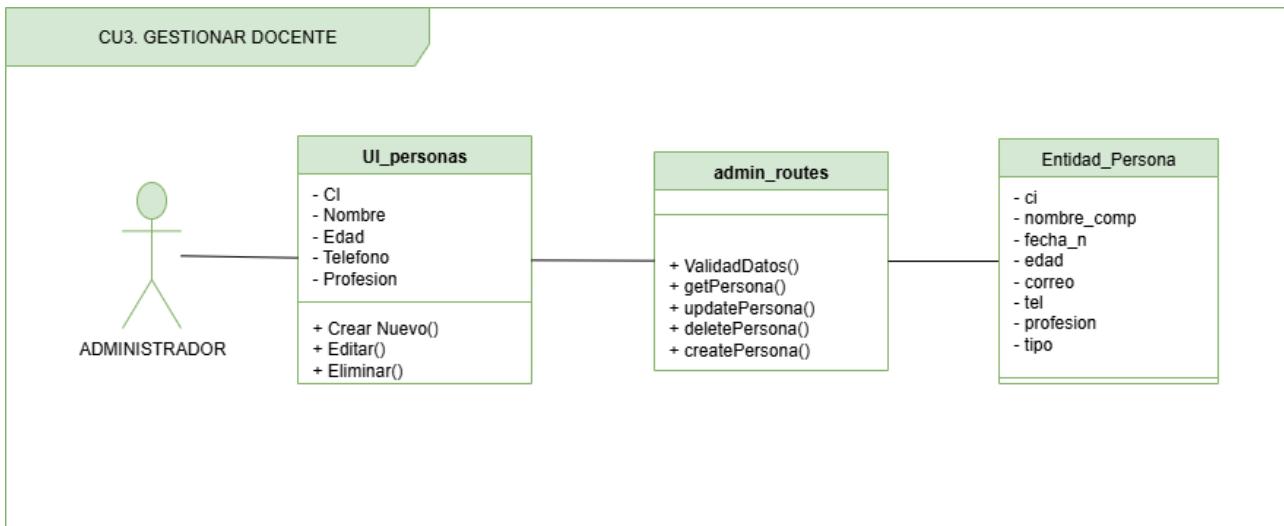
CU13. INICIAR SESIÓN



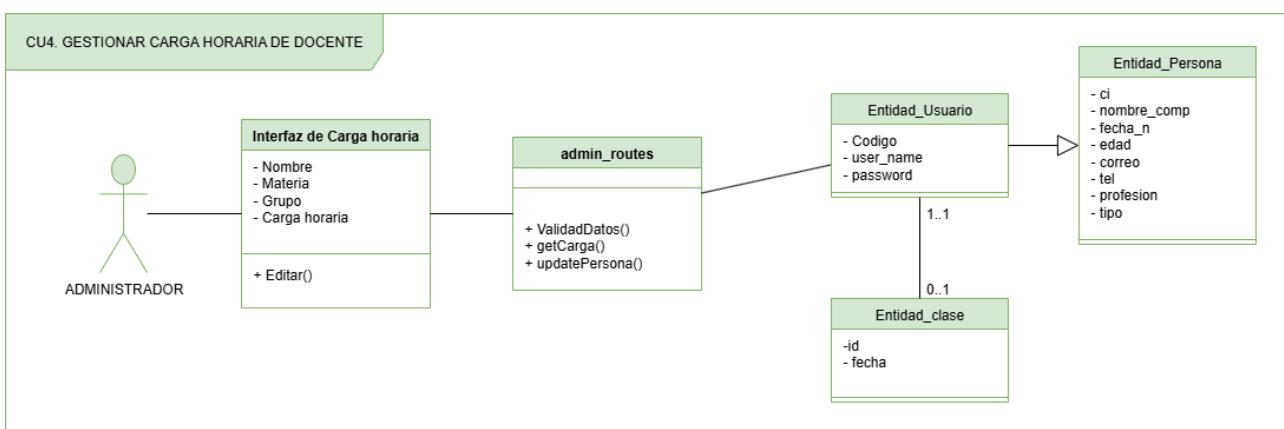
CU14. CERRAR SESIÓN



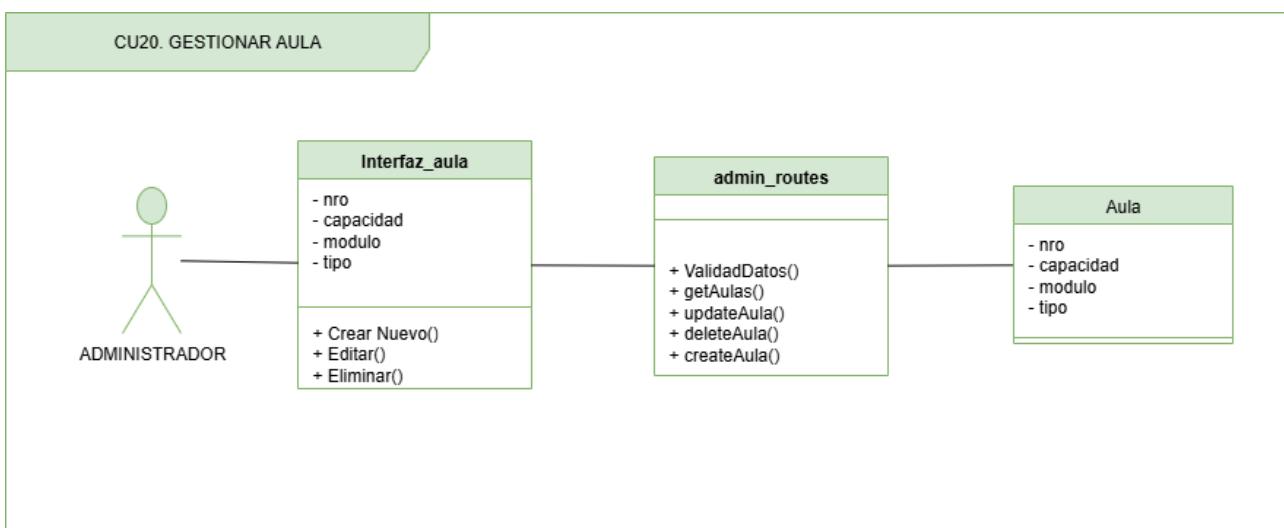
CU3. GESTIONAR DOCENTE



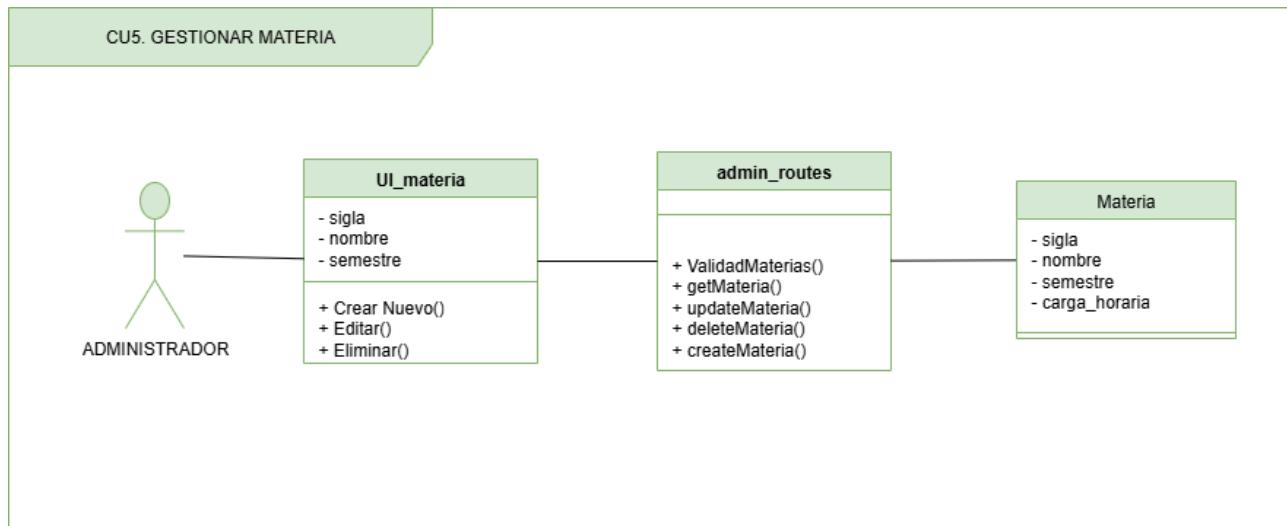
CU4. GESTIONAR CARGA HORARIA DE DOCENTE



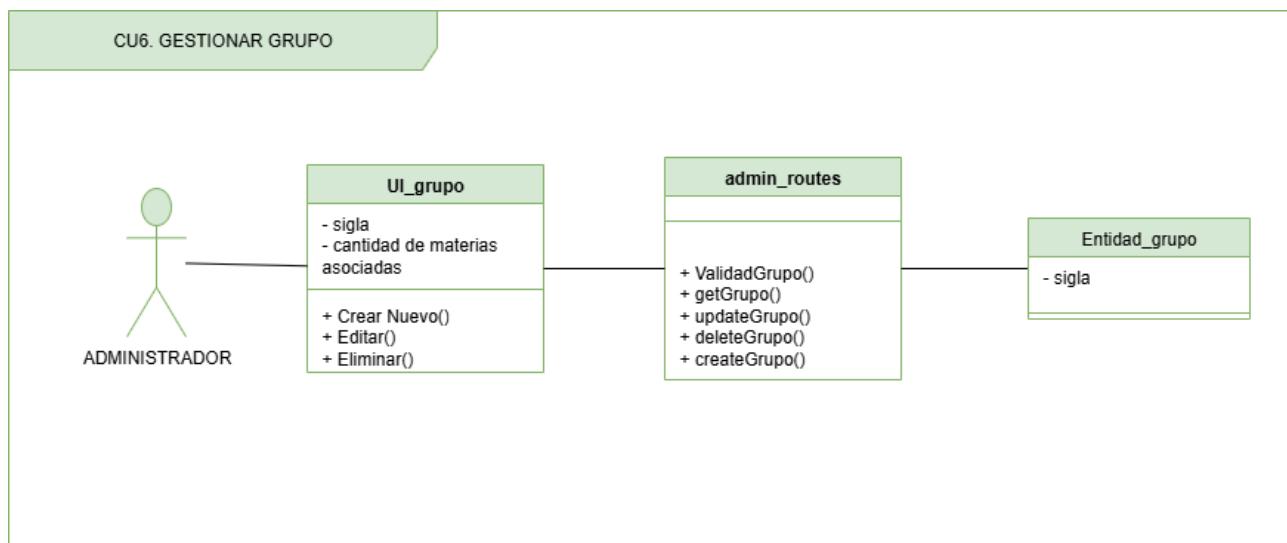
CU20. GESTIONAR AULA



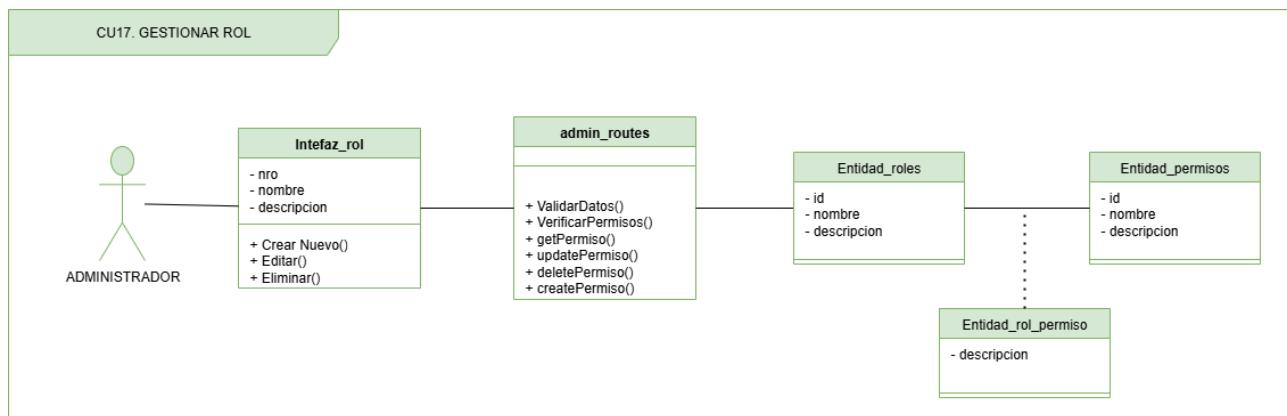
CU5. GESTIONAR MATERIA



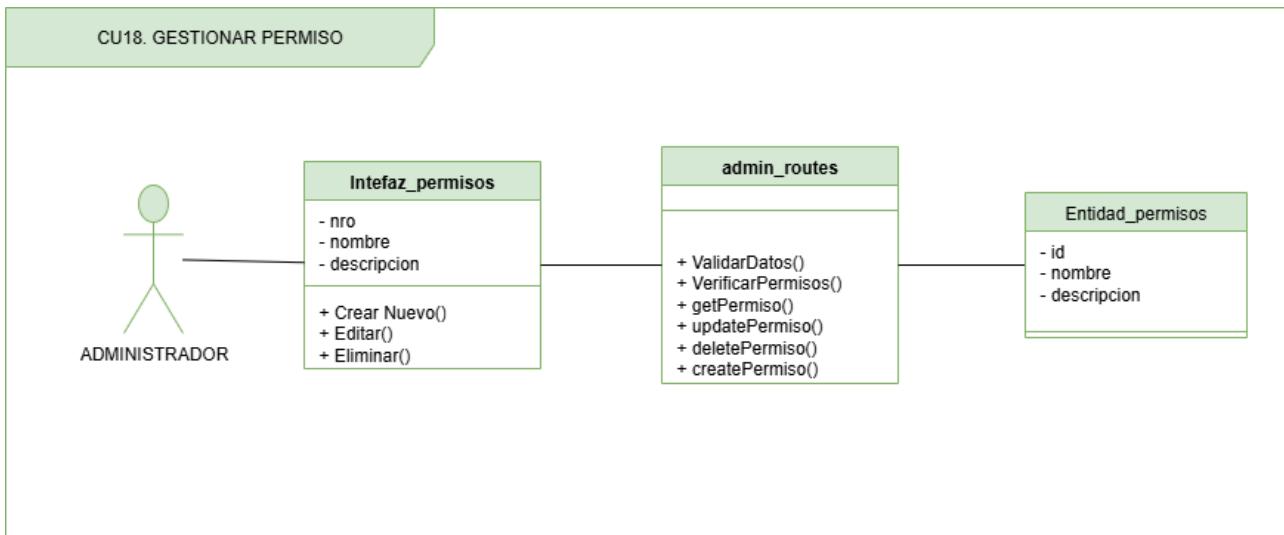
CU6. GESTIONAR GRUPO



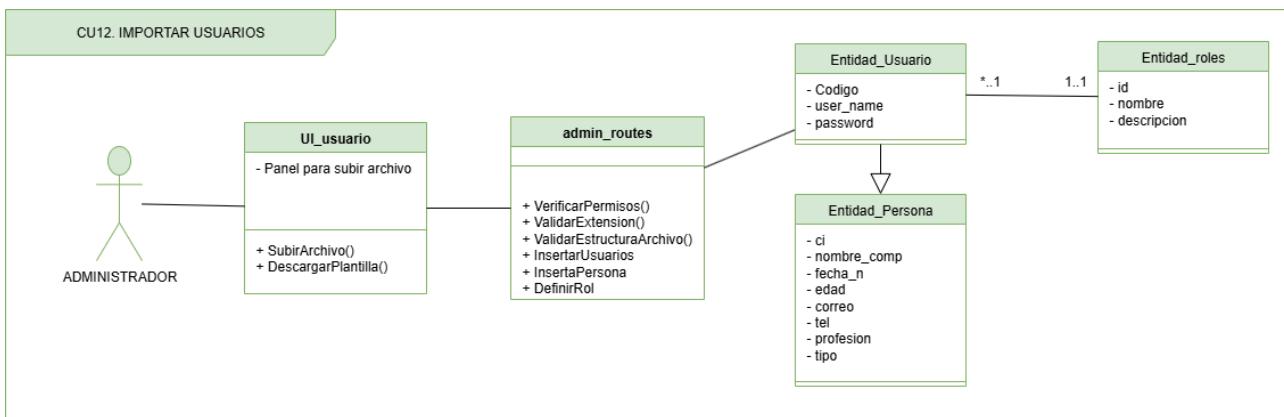
CU17. GESTIONAR ROL



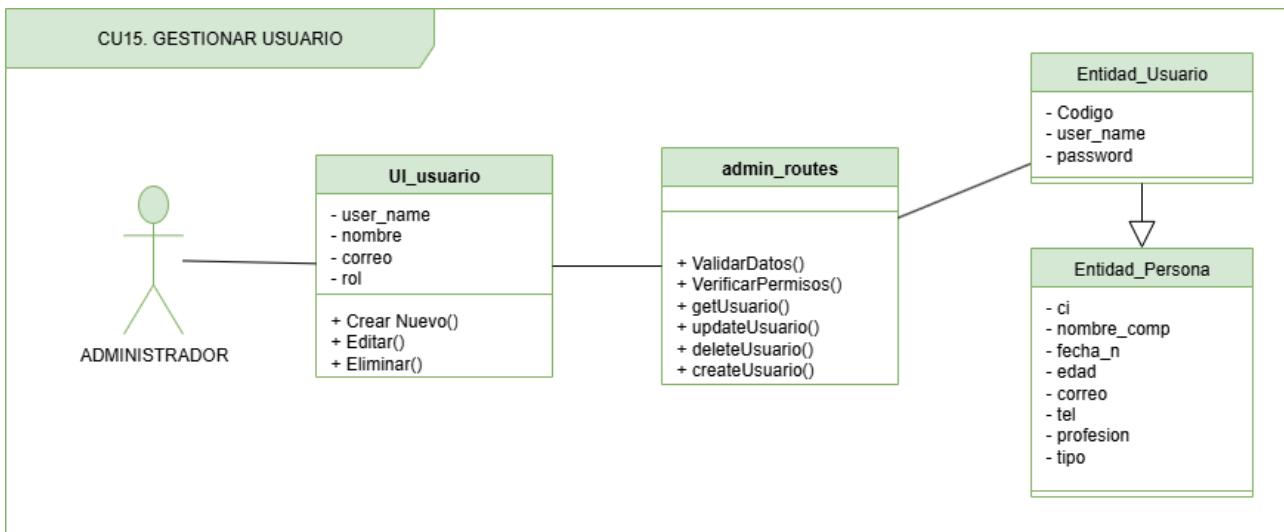
CU18. GESTIONAR PERMISO



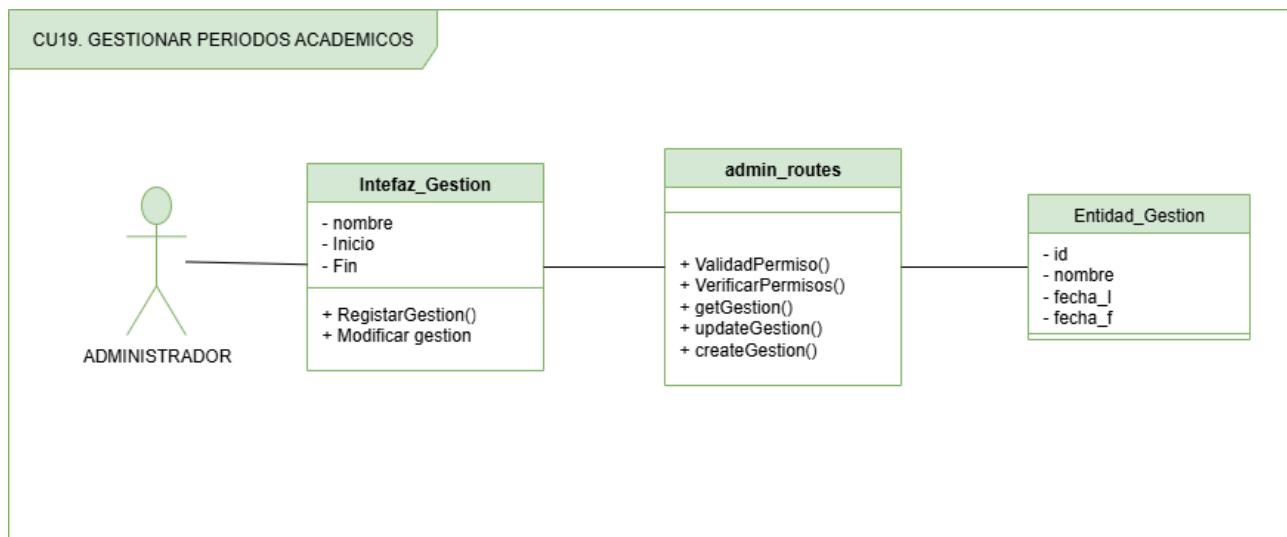
CU12. IMPORTAR USUARIOS



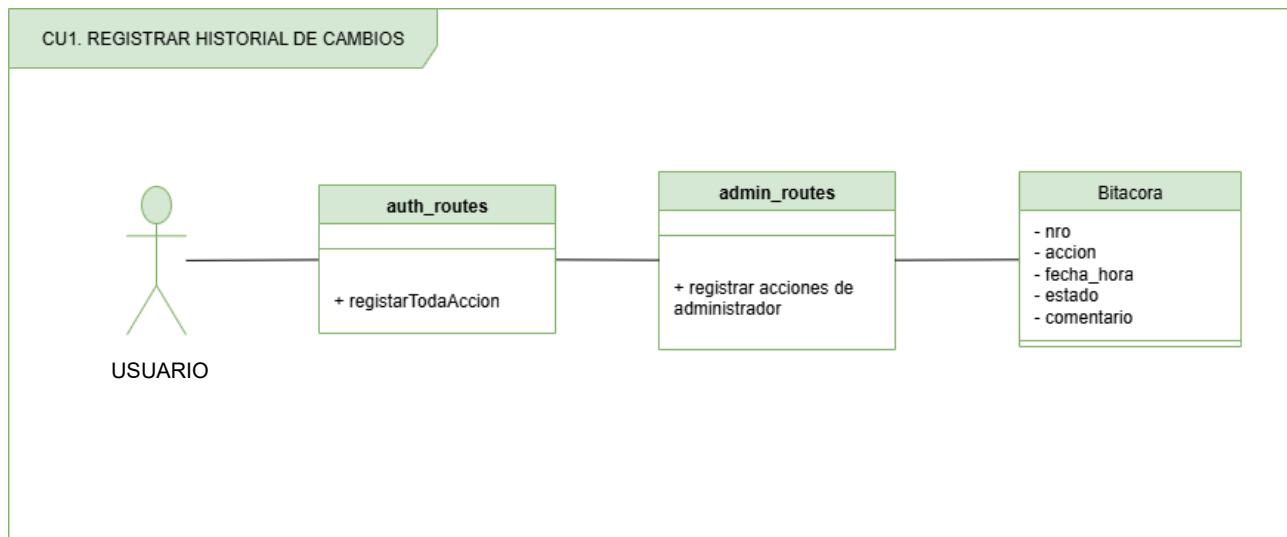
CU15. GESTIONAR USUARIO



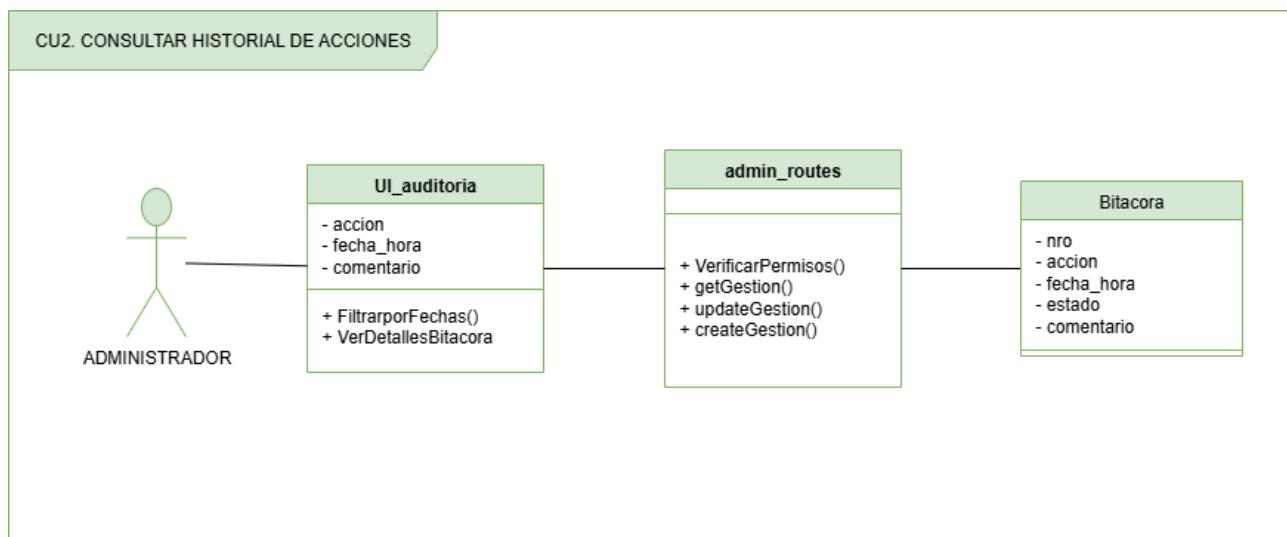
CU19. GESTIONAR PERIODOS ACADÉMICOS



CU1. REGISTRAR HISTORIAL DE CAMBIOS

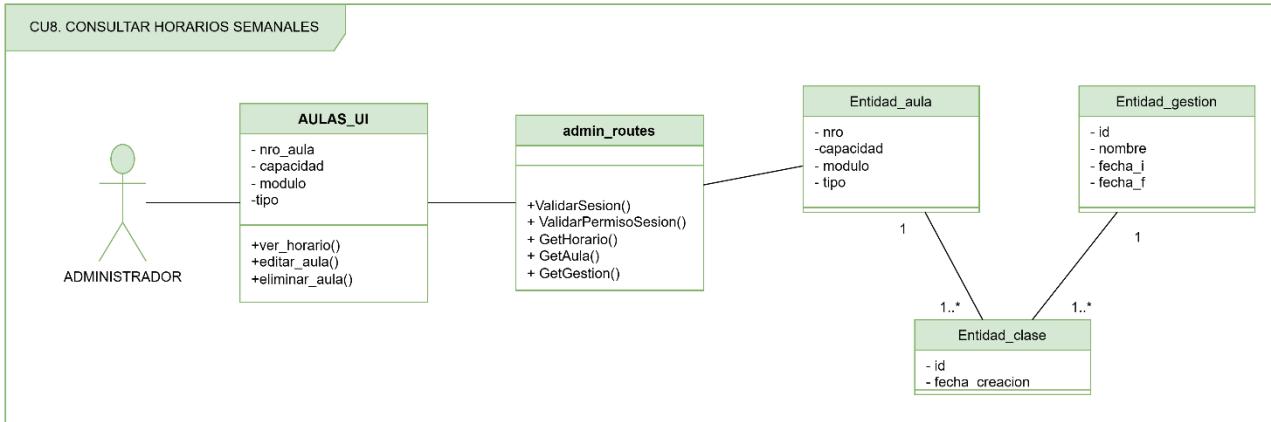


CU2. CONSULTAR HISTORIAL DE ACCIONES

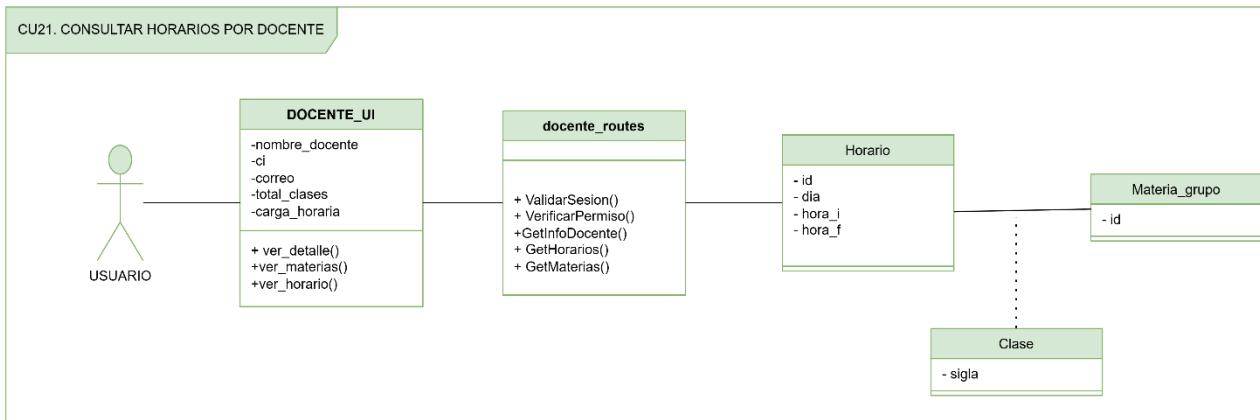


CICLO #2

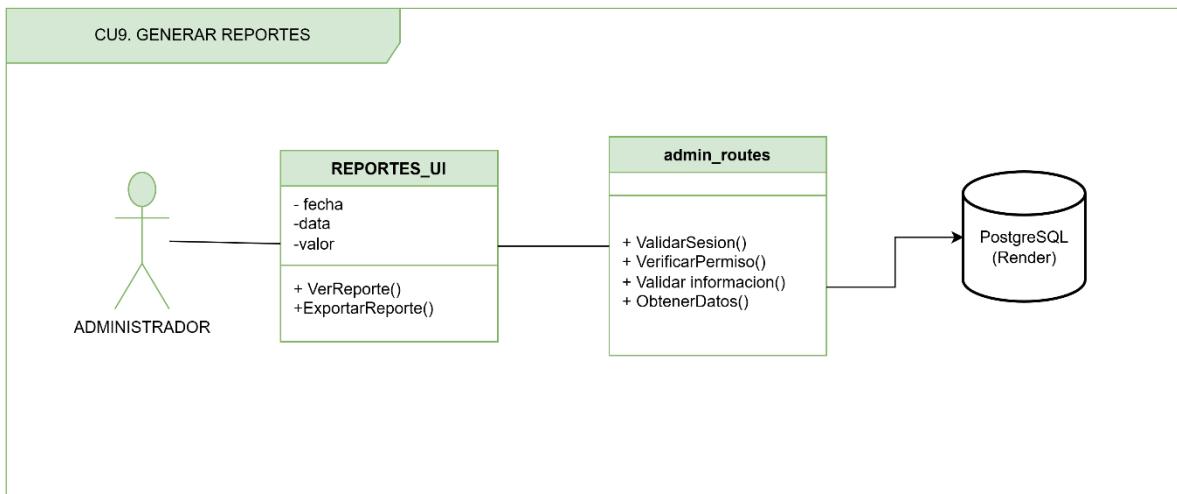
CU8: CONSULTAR HORARIOS SEMANALES



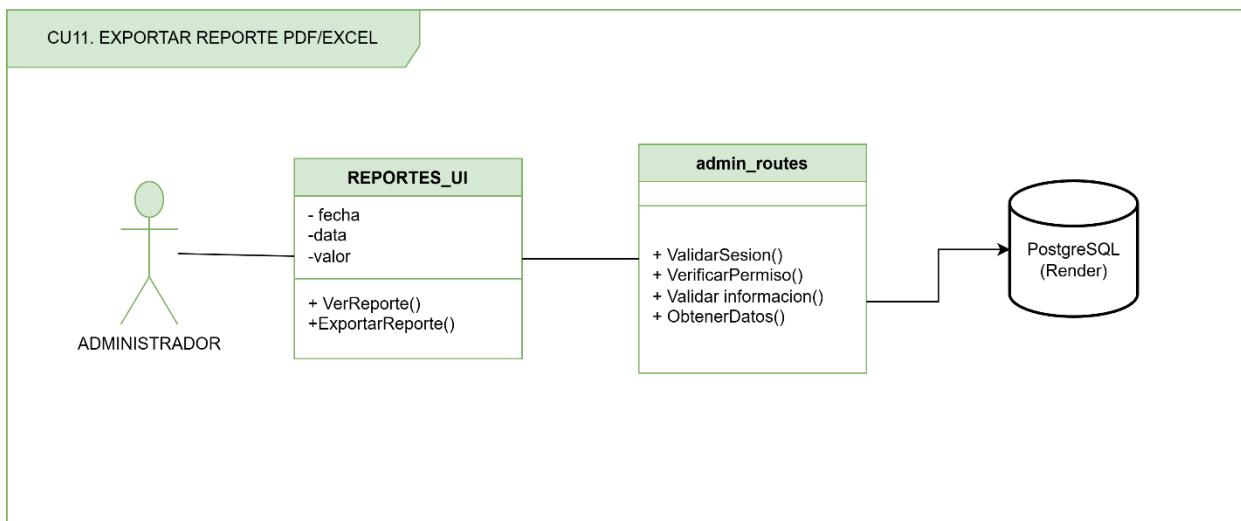
CU21: CONSULTAR HORARIO POR DOCENTE



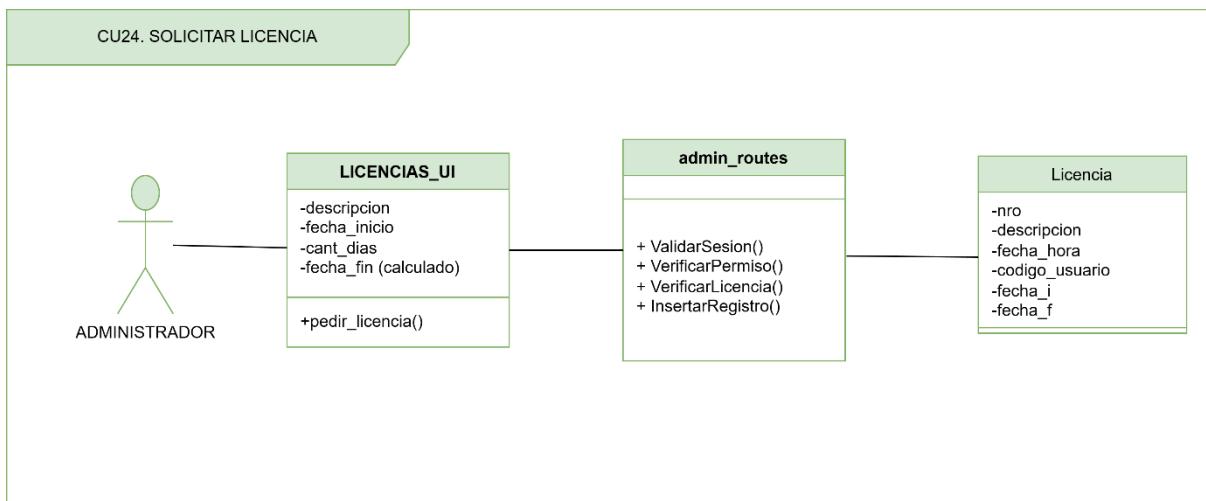
CU9: GENERAR REPORTES



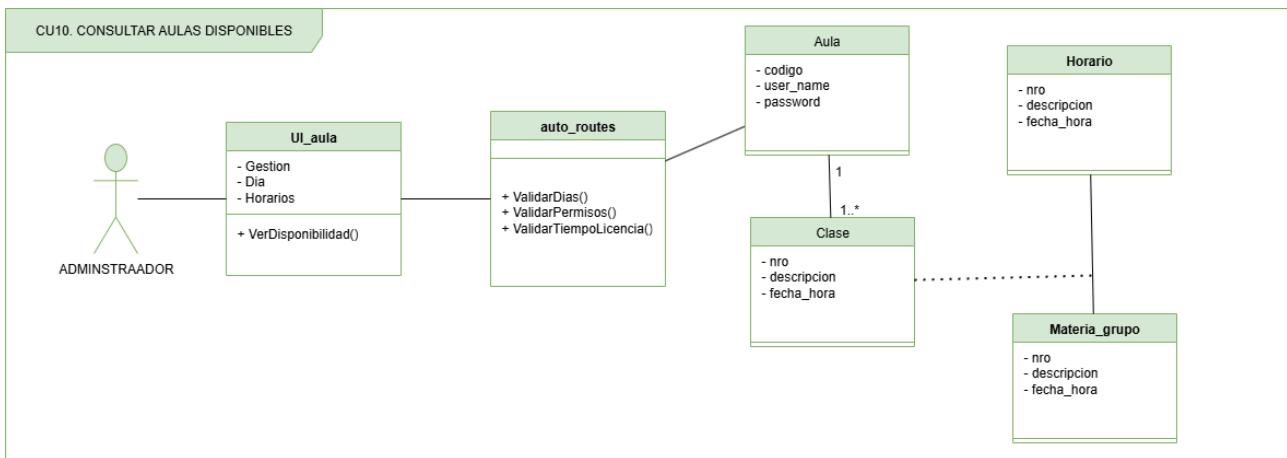
CU11: EXPORTAR REPORTE A PDF/EXCEL



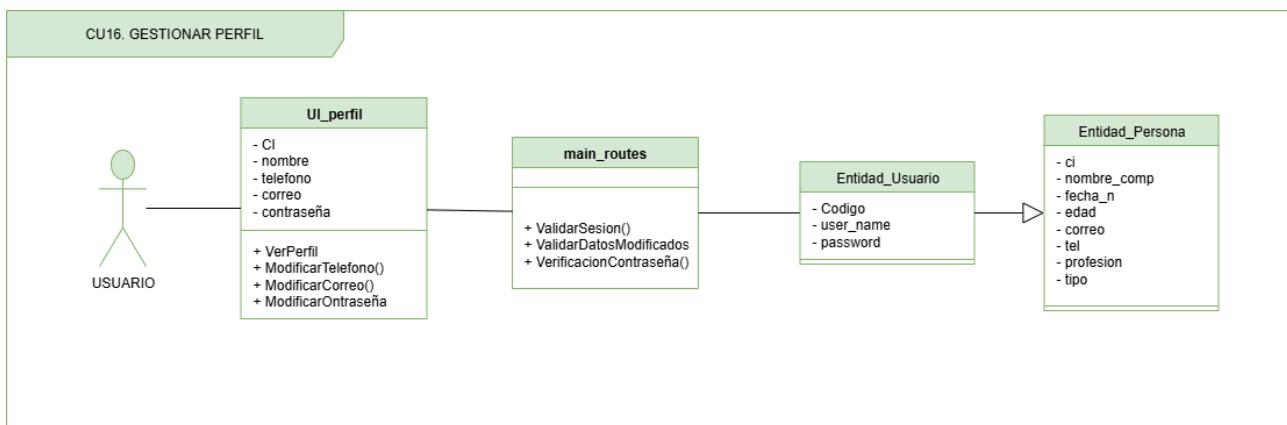
CU24: SOLICITAR LICENCIA



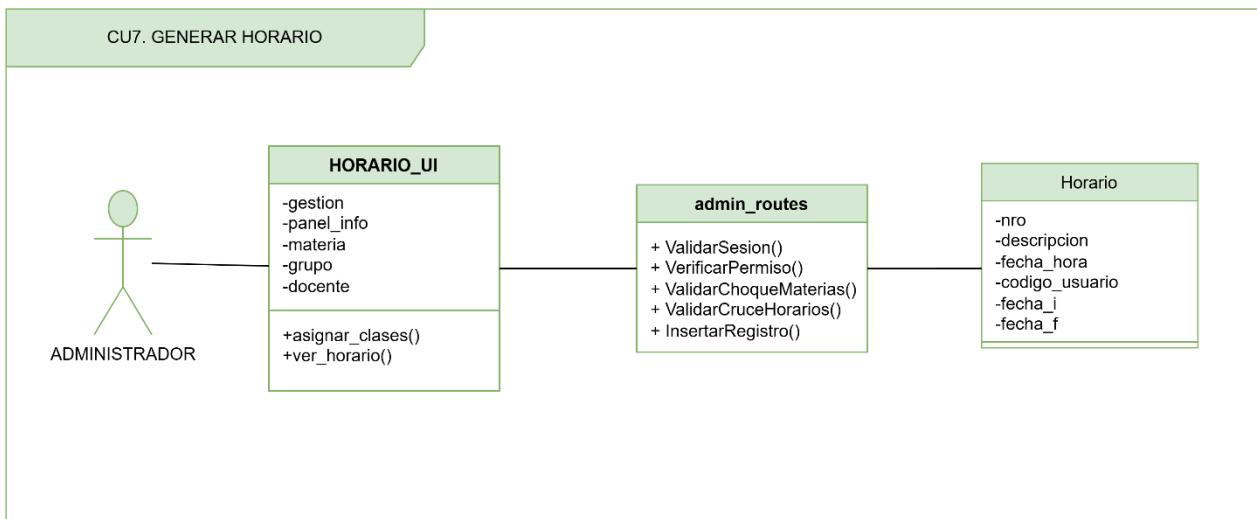
CU10: CONSULTAR AULAS DISPONIBLES



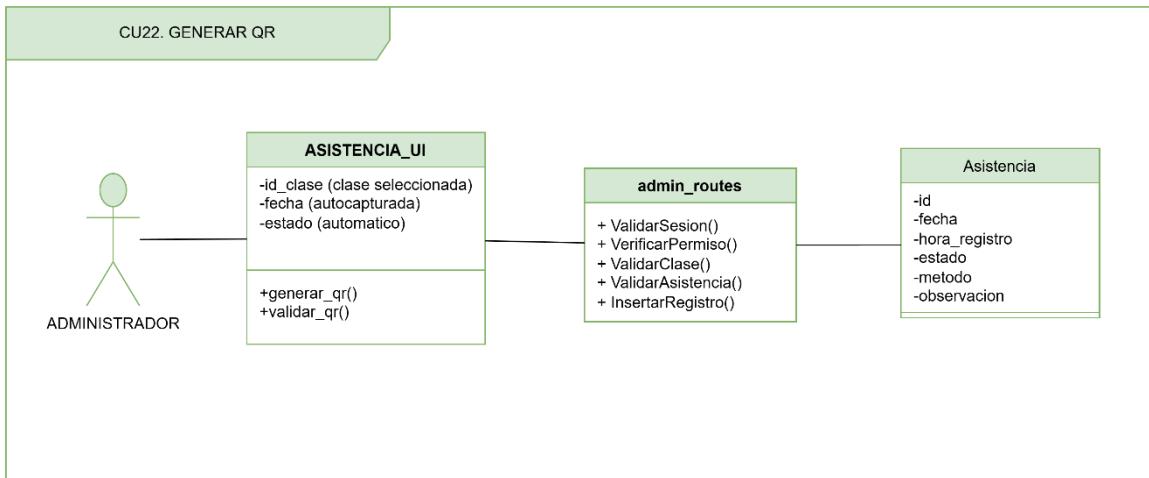
CU16: GESTIONAR PERFIL



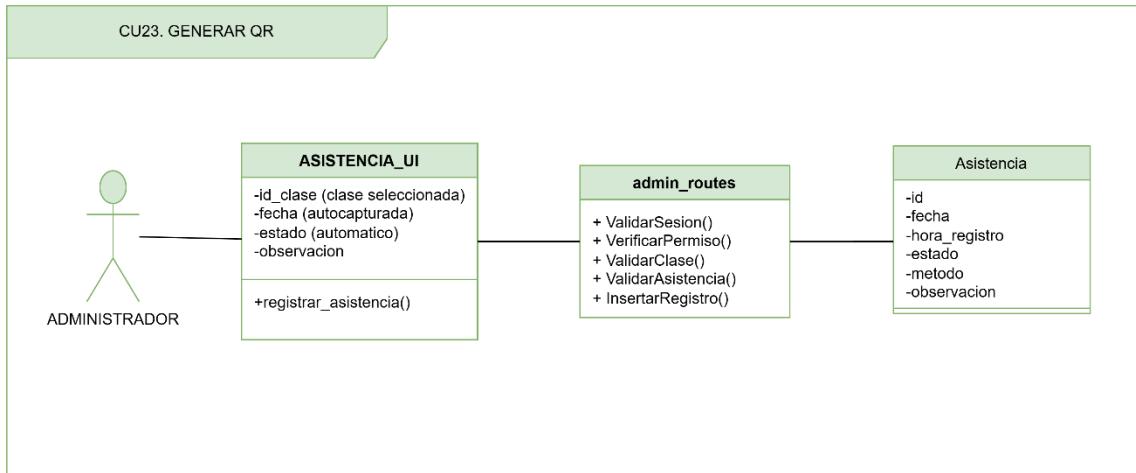
CU7: GENERAR HORARIO



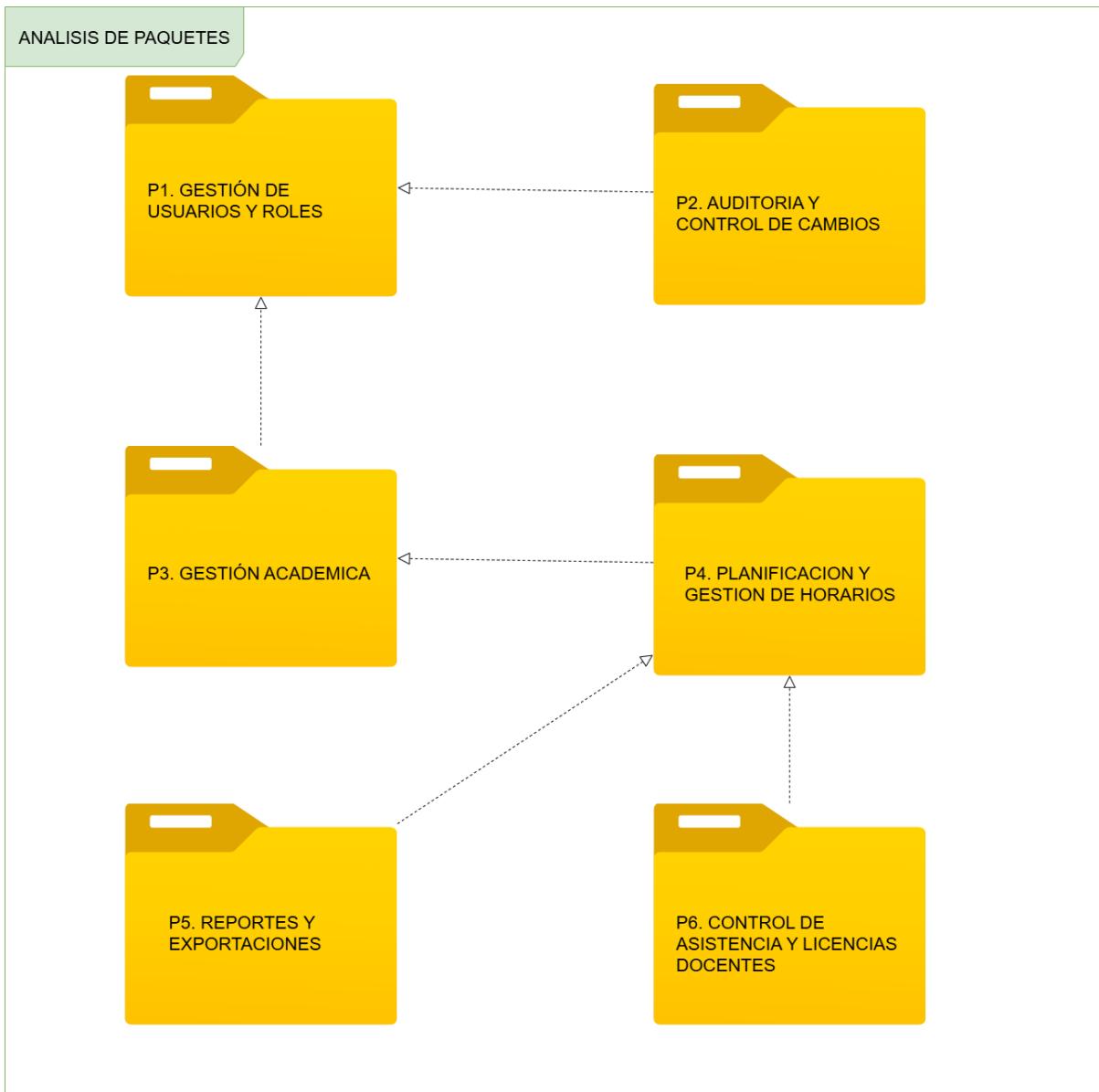
CU22: GENERAR QR



CU23: REGISTRAR ASISTENCIA POR QR O FORMULARIO WEB



5.4 ANALIZAR UN PAQUETE



JUSTIFICACIÓN:

P2 → P1:

La auditoría necesita saber qué usuario hizo una acción para poder registrarla correctamente.

P4 → P3:

El módulo de horarios usa los datos académicos (docentes, materias, aulas, grupos) para generar los horarios.

P5 → P4:

El módulo de reportes obtiene la información de los horarios para generar reportes o exportaciones.

P6 → P4:

El control de asistencia consulta los horarios para saber cuándo y dónde debía estar cada docente.

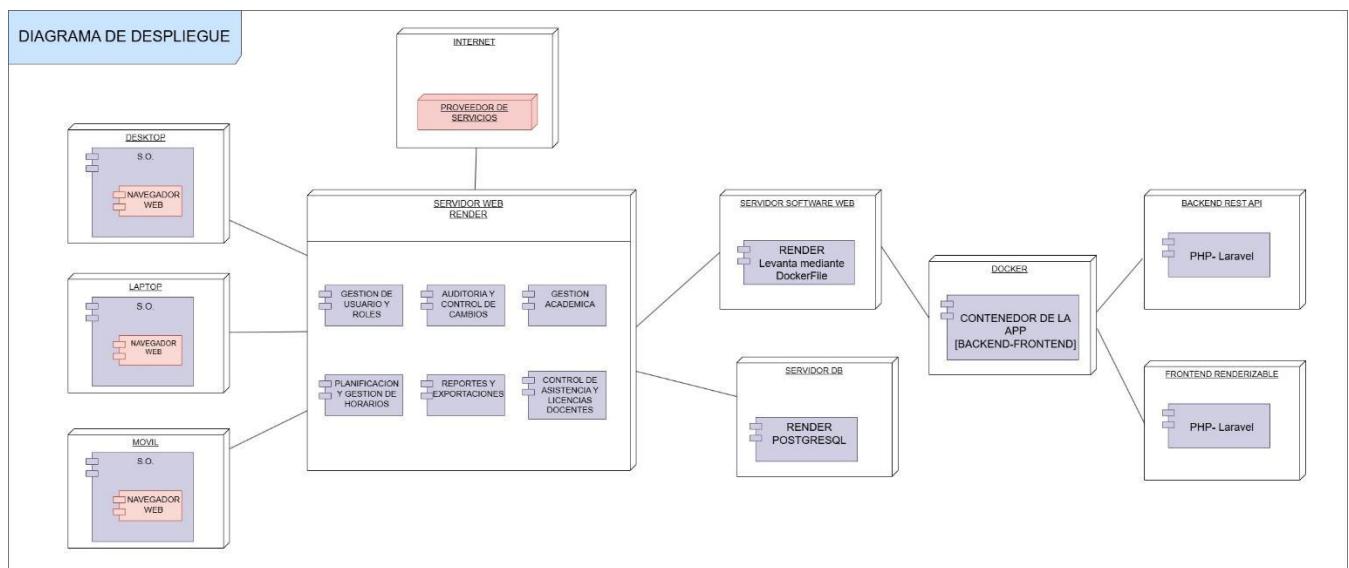
P3 → P1:

El módulo académico valida permisos con el módulo de seguridad antes de permitir cambios.

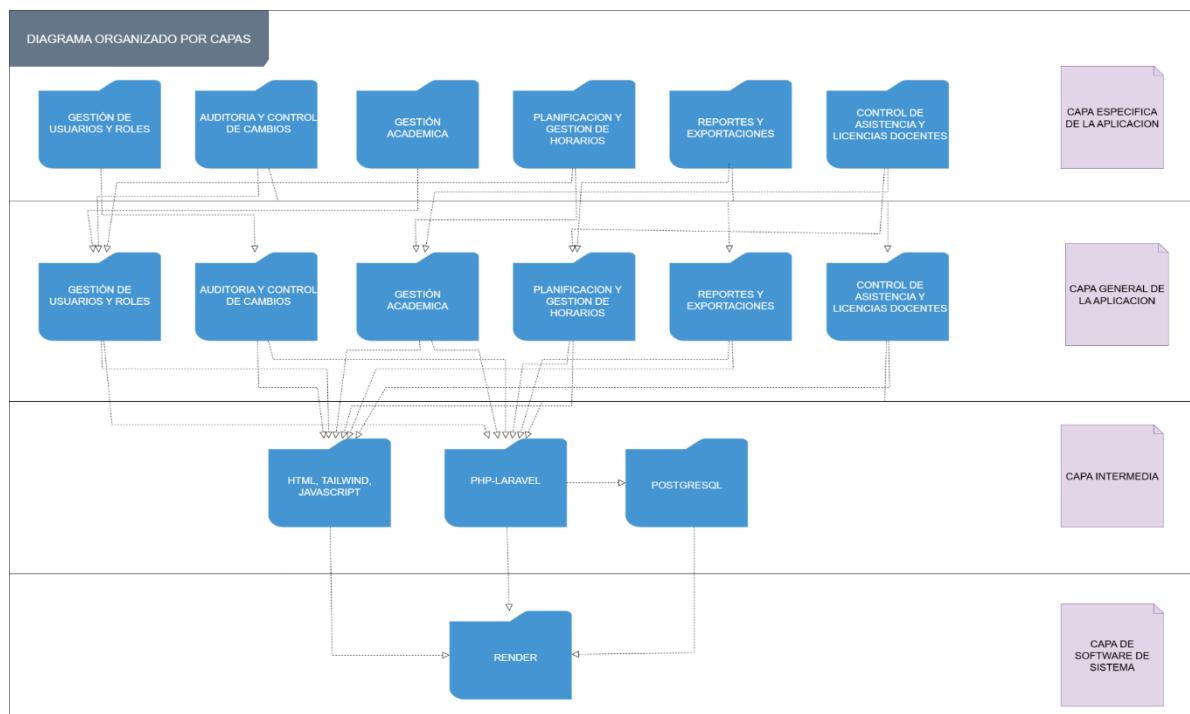
6. FLUJO DE TRABAJO: DISEÑO

6.1 DISEÑO DE ARQUITECTURA

6.1.1 DISEÑO FÍSICO (DIAGRAMA DE DESPLIEGUE)



6.1.2 DISEÑO LÓGICO (DIAGRAMA ORGANIZADO EN CAPAS)

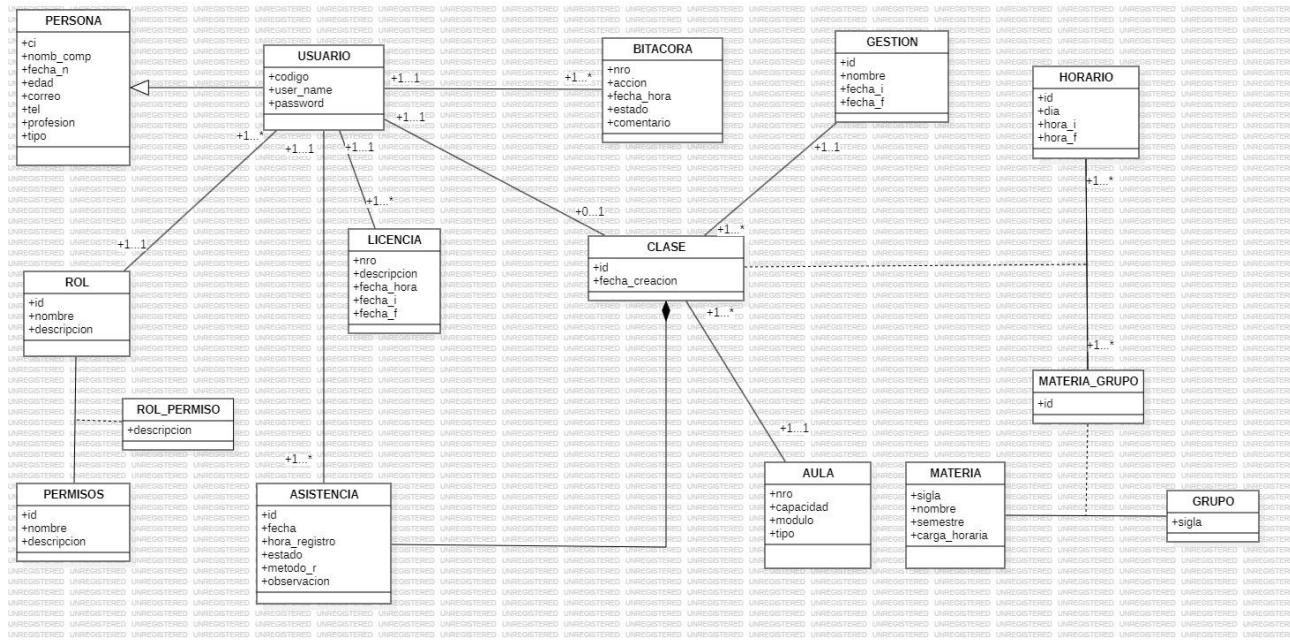


6.2 DISEÑO DE DATOS

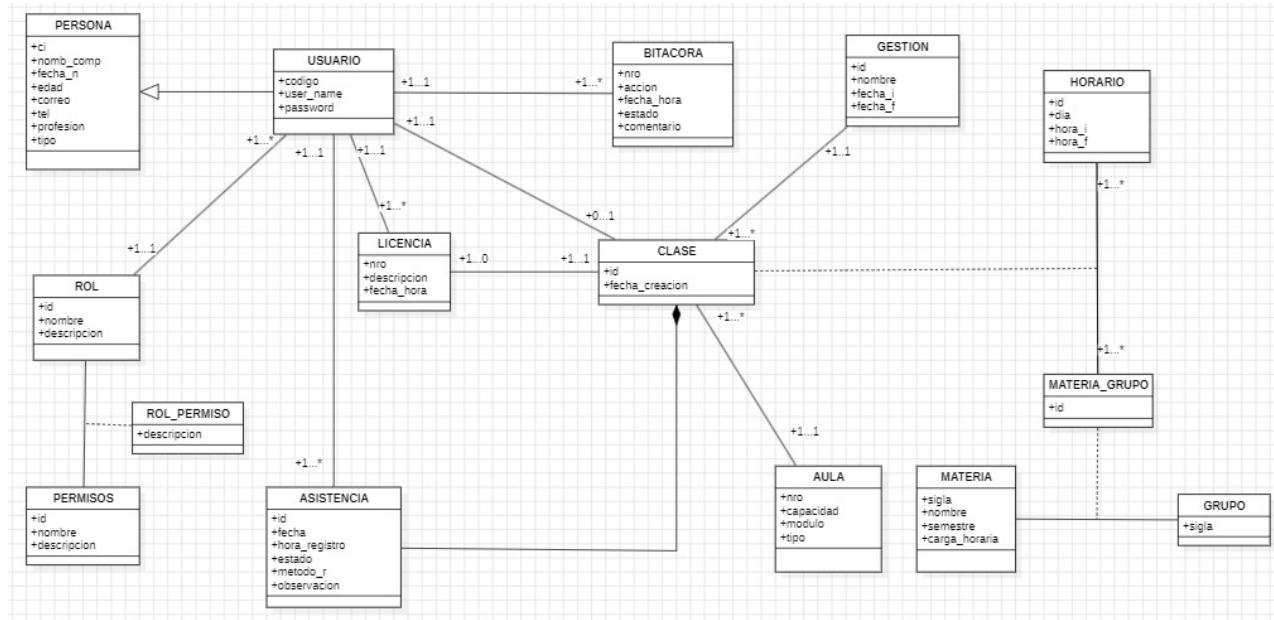
6.2.1 DISEÑO DE DATOS LÓGICO

DIAGRAMA DE CLASE

VERSION FINAL:



VERSION INICIAL:



MAPEO

PERSONA							
PK							
<u>ci</u>	nombr_comp	fecha_n	edad	correo	tel	profesion	tipo
USUARIO							
PK			FK		FK		
codigo	user_name	password	<i>ci</i>		<i>id_rol</i>		
ROL							
PK							
<u>id</u>	nombre	descripcion					
PERMISOS							
PK							
<u>id</u>	nombre	descripcion					
ROL_PERMISO							
FK	FK						
<i>id_rol</i>	<i>id_permiso</i>	descripcion					
BITACORA							
PK							FK
<u>nro</u>	accion	fecha_hora	estado	comentario			<i>usuario_codigo</i>

LICENCIA				
PK			FK	FK
<u>nro</u>	descripcion	fecha_hora	<i>codigo_usuario</i>	<i>id_clase</i>

ASISTENCIA						
PK						FK
<u>id</u>	fecha	hora_registro	estado	<i>metodo_r</i>	<i>observacion</i>	<i>id_clase</i>

GESTION			
PK			
<u>id</u>	nombre	fecha_i	fecha_f

AULA			
PK			
<u>nro</u>	capacidad	modulo	tipo

MATERIA			
PK			
<u>sigla</u>	nombre	semestre	carga_horaria

GRUPO	
PK	
<u>sigla</u>	

MATERIA_GRUPO		
PK	FK	FK
<u>id</u>	<i>sigla_materia</i>	<i>sigla_grupo</i>

NORMALIZACIÓN

El sistema de información ya se encuentra normalizado en 1era, 2da, 3era y 4ta forma normal.

6.2.2 DISEÑO DE DATOS FÍSICO

TABLA DE VOLUMEN

PERSONA						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
ci	Entero	8 bytes	Sí	No	No	Identificador único de la persona
nomb_com_p	Cadena (VARCHAR 100)	-	No	No	No	Nombre completo de la persona
fecha_n	Fecha	3 bytes	No	No	No	Fecha de nacimiento
edad	Entero	4 bytes	No	No	No	Edad calculada o registrada
correo	Cadena (VARCHAR 80)	-	No	No	No	Correo electrónico
tel	Cadena (VARCHAR 20)	-	No	No	Sí	Número de teléfono
profesion	Cadena (VARCHAR 60)	-	No	No	Sí	Profesión de la persona
tipo	Cadena (VARCHAR 20)	-	No	No	No	Tipo de persona (Docente, Autoridad, etc.)

USUARIO						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
codigo	Entero	8 bytes	Sí	No	No	Identificador del usuario
user_name	Cadena (VARCHAR 50)	-	No	No	No	Nombre de usuario
password	Cadena (VARCHAR 255)	-	No	No	No	Contraseña encriptada
ci	Entero	8 bytes	No	Sí	No	Relación con PERSONA
id_rol	Entero	4 bytes	No	Sí	No	Relación con ROL

ROL						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
id	Entero	4 bytes	Sí	No	No	Identificador del rol
nombre	Cadena (VARCHAR 50)	-	No	No	No	Nombre del rol (Administrador, Docente...)
descripcion	Cadena (VARCHAR 100)	-	No	No	Sí	Descripción del rol

PERMISO						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
id	Entero	4 bytes	Sí	No	No	Identificador del permiso
nombre	Cadena (VARCHAR 50)	-	No	No	No	Nombre del permiso
descripcion	Cadena (VARCHAR 100)	-	No	No	Sí	Descripción del permiso

PERMISOS_ROL						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
id_rol	Entero	4 bytes	No	Sí	No	Identificador del rol
id_permiso	Entero	4 bytes	No	Sí	No	Identificador del permiso
descripcion	Cadena (VARCHAR 100)	-	No	No	Sí	Observación de la relación rol-permiso

BITACORA						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
nro	Entero	8 bytes	Sí	No	No	Identificador del registro
accion	Cadena (VARCHAR 100)	-	No	No	No	Acción realizada
fecha_hora	FechaHora	8 bytes	No	No	No	Fecha y hora del evento
estado	Cadena (VARCHAR 20)	-	No	No	Sí	Estado de la acción
comentario	Cadena (VARCHAR 150)	-	No	No	Sí	Descripción o detalle
usuario_codigo	Entero	8 bytes	No	Sí	No	Relación con USUARIO

LICENCIA						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
nro	Entero	8 bytes	Sí	No	No	Identificador de la licencia
descripcion	Cadena (VARCHAR 150)	-	No	No	No	Motivo de la licencia
fecha_hora	FechaHora	8 bytes	No	No	No	Fecha de solicitud
codigo_usuario	Entero	8 bytes	No	Sí	No	Usuario que solicita la licencia
id_clase	Entero	4 bytes	No	Sí	No	Clase afectada por la licencia

ASISTENCIA						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
id	Entero	8 bytes	Sí	No	No	Identificador del registro
fecha	Fecha	3 bytes	No	No	No	Fecha de asistencia
hora_registro	Hora	3 bytes	No	No	No	Hora de marcación
estado	Enum	-	No	No	No	Presente, Ausente, Retraso
metodo_r	Cadena (VARCHAR 30)	-	No	No	No	Método de registro (QR, Manual, etc.)
observacion	Cadena (VARCHAR 100)	-	No	No	Sí	Observaciones
id_clase	Entero	4 bytes	No	Sí	No	Clase a la que corresponde

GESTION						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
id	Entero	4 bytes	Sí	No	No	Identificador de la gestión
nombre	Cadena (VARCHAR 20)	-	No	No	No	Nombre de la gestión (2025-1, 2025-2...)
fecha_i	Fecha	3 bytes	No	No	No	Fecha de inicio
fecha_f	Fecha	3 bytes	No	No	No	Fecha de fin

CLASE						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
id	Entero	8 bytes	Sí	No	No	Identificador de la clase
fecha_creacion	Fecha	3 bytes	No	No	No	Fecha de creación
id_gestion	Entero	4 bytes	No	Sí	No	Gestión a la que pertenece
nro_aula	Entero	4 bytes	No	Sí	No	Aula asignada
id_materia_grupo	Entero	4 bytes	No	Sí	No	Relación materia-grupo
usuario_codigo	Entero	8 bytes	No	Sí	No	Docente responsable

HORARIO						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
id	Entero	8 bytes	Sí	No	No	Identificador del horario
dia	Cadena (VARCHAR 20)	-	No	No	No	Día de la semana
hora_i	Hora	3 bytes	No	No	No	Hora de inicio
hora_f	Hora	3 bytes	No	No	No	Hora de fin

AULA						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
nro	Entero	4 bytes	Sí	No	No	Número del aula
capacidad	Entero	4 bytes	No	No	No	Capacidad máxima
modulo	Cadena (VARCHAR 10)	-	No	No	No	Módulo al que pertenece
tipo	Cadena (VARCHAR 30)	-	No	No	No	Tipo de aula (laboratorio, teórica)

MATERIA						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
sigla	Cadena (VARCHAR 10)	-	Sí	No	No	Identificador corto de la materia
nombre	Cadena (VARCHAR 80)	-	No	No	No	Nombre de la materia
semestre	Cadena (VARCHAR 10)	-	No	No	No	Semestre al que pertenece
carga_horaria	Entero	4 bytes	No	No	No	Horas semanales de clase

GRUPO						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
sigla	Cadena (VARCHAR 10)	-	Sí	No	No	Identificador del grupo (A, B, C, etc.)

MATERIA_GRUPO						
Atributo	Tipo de dato	Amplitud	Llave Primaria	Llave Foránea	NULO	Descripción
id	Entero	8 bytes	Sí	No	No	Identificador de la combinación
sigla_materia	Cadena (VARCHAR 10)	-	No	Sí	No	Materia asociada
sigla_grupo	Cadena (VARCHAR 10)	-	No	Sí	No	Grupo asociado

SCRIPT

```
-- =====  
-- CREACIÓN DE TABLAS - SISTEMA FICCT  
-- Con TIMESTAMPTZ y ON DELETE/UPDATE CASCADE  
-- =====
```

```
CREATE TABLE PERSONA (  
    ci BIGINT PRIMARY KEY,  
    nomb_comp VARCHAR(100) NOT NULL,  
    fecha_n DATE NOT NULL,  
    edad INT,  
    correo VARCHAR(100),  
    tel VARCHAR(20),  
    profesion VARCHAR(50),  
    tipo VARCHAR(30)  
);
```

```
CREATE TABLE ROL (  
    id SERIAL PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    descripcion TEXT  
);
```

```
CREATE TABLE PERMISOS (  
    id SERIAL PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    descripcion TEXT  
);
```

```
CREATE TABLE USUARIO (  
    codigo SERIAL PRIMARY KEY,  
    user_name VARCHAR(50) UNIQUE NOT NULL,  
    password VARCHAR(255) NOT NULL,  
    ci BIGINT ,
```

```
    id_rol INT,  
    FOREIGN KEY (ci) REFERENCES PERSONA(ci)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (id_rol) REFERENCES ROL(id)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE ROL_PERMISO (  
    id_rol INT,  
    id_permiso INT,  
    descripcion TEXT,  
    PRIMARY KEY (id_rol, id_permiso),  
    FOREIGN KEY (id_rol) REFERENCES ROL(id)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (id_permiso) REFERENCES PERMISOS(id)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE BITACORA (  
    nro SERIAL PRIMARY KEY,  
    accion VARCHAR(100) NOT NULL,  
    fecha_hora TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,  
    estado VARCHAR(30),  
    comentario TEXT,  
    usuario_codigo INT,  
    FOREIGN KEY (usuario_codigo) REFERENCES USUARIO(codigo)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE GESTION (  
    id SERIAL PRIMARY KEY,  
    nombre VARCHAR(50),  
    fecha_i DATE,  
    fecha_f DATE
```

);

```
CREATE TABLE AULA (
    nro SERIAL PRIMARY KEY,
    capacidad INT,
    modulo VARCHAR(50),
    tipo VARCHAR(30)
);
```

```
CREATE TABLE MATERIA (
    sigla VARCHAR(20) PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    semestre VARCHAR(20),
    carga_horaria INT
);
```

```
CREATE TABLE GRUPO (
    sigla VARCHAR(20) PRIMARY KEY
);
```

```
CREATE TABLE MATERIA_GRUPO (
    id SERIAL PRIMARY KEY,
    sigla_materia VARCHAR(20),
    sigla_grupo VARCHAR(20),
    FOREIGN KEY (sigla_materia) REFERENCES MATERIA(sigla)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (sigla_grupo) REFERENCES GRUPO(sigla)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE CLASE (
    id SERIAL PRIMARY KEY,
    fecha_creacion TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    id_gestion INT,
    nro_aula INT,
```

```
    id_materia_grupo INT,  
    usuario_codigo INT,  
    FOREIGN KEY (id_gestion) REFERENCES GESTION(id)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (nro_aula) REFERENCES AULA(nro)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (id_materia_grupo) REFERENCES MATERIA_GRUPO(id)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (usuario_codigo) REFERENCES USUARIO(codigo)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE HORARIO (  
    id SERIAL PRIMARY KEY,  
    dia VARCHAR(15),  
    hora_i TIMESTAMPTZ,  
    hora_f TIMESTAMPTZ,  
    id_clase INT,  
    FOREIGN KEY (id_clase) REFERENCES CLASE(id)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE LICENCIA (  
    nro SERIAL PRIMARY KEY,  
    descripcion TEXT NOT NULL,  
    fecha_hora TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,  
    codigo_usuario INT,  
    id_clase INT,  
    FOREIGN KEY (codigo_usuario) REFERENCES USUARIO(codigo)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (id_clase) REFERENCES CLASE(id)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
CREATE TABLE ASISTENCIA (
```

```
    id SERIAL PRIMARY KEY,  
    fecha DATE,  
    hora_registro TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,  
    estado VARCHAR(30),  
    metodo_r VARCHAR(50),  
    observacion TEXT,  
    id_clase INT,  
    FOREIGN KEY (id_clase) REFERENCES CLASE(id)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

```
-- ======  
-- POBLACIÓN DE PRUEBA - SISTEMA FICCT  
-- Basado en el esquema con BIGINT para ci y timestampz  
-- Ejecutar después de haber creado las tablas  
-- ======
```

```
-- ----- ROLES -----  
INSERT INTO rol (id, nombre, descripcion) VALUES  
(1, 'Docente', 'Rol para docentes'),  
(2, 'JefeCarrera', 'Rol para jefes de carrera'),  
(3, 'Autoridad', 'Rol para autoridades de facultad'),  
(4, 'Administrador', 'Rol con todos los permisos');
```

```
-- ----- PERMISOS -----  
INSERT INTO permisos (id, nombre, descripcion) VALUES  
(1, 'VER_GRUPO', 'Ver información de un grupo'),  
(2, 'MODIFICAR_GRUPO', 'Modificar datos de un grupo'),  
(3, 'CREAR_HORARIO', 'Crear horarios'),  
(4, 'MODIFICAR_HORARIO', 'Modificar horarios'),  
(5, 'REGISTRAR_ASISTENCIA', 'Registrar asistencia docente'),  
(6, 'VER_REPORTES', 'Ver reportes administrativos');
```

```
-- ----- ASIGNACIÓN ROL_PERMISO -----  
INSERT INTO rol_permiso (id_rol, id_permiso, descripcion) VALUES
```

(1, 1, 'Docente puede ver grupos'),
(1, 5, 'Docente puede registrar asistencia'),
(2, 1, 'JefeCarrera puede ver grupos'),
(2, 2, 'JefeCarrera puede modificar grupos'),
(2, 3, 'JefeCarrera puede crear horarios'),
(3, 1, 'Autoridad puede ver grupos'),
(4, 1, 'Administrador puede ver grupos'),
(4, 2, 'Administrador puede modificar grupos'),
(4, 3, 'Administrador puede crear horarios'),
(4, 4, 'Administrador puede modificar horarios'),
(4, 5, 'Administrador registra asistencia'),
(4, 6, 'Administrador ve reportes');

-- ----- PERSONAS -----

-- ci como BIGINT (solo números)

```
INSERT INTO persona (ci, nomb_comp, fecha_n, edad, correo, tel, profesion, tipo)
VALUES
(7654321, 'Juan Perez', '1980-04-12', 45, 'juan.perez@ficct.edu.bo', '72812345',
'Licenciado', 'Docente'),
(7890123, 'María Gomez', '1985-09-30', 39, 'maria.gomez@ficct.edu.bo', '71234567',
'Ingeniero de Sistemas', 'Jefe de Carrera'),
(1234567, 'Luis Ortega', '1975-01-20', 50, 'luis.ortega@ficct.edu.bo', '70123456',
'Informático', 'Autoridad'),
(4567890, 'Ana Flores', '1990-07-15', 34, 'ana.flores@ficct.edu.bo', '72234567',
'Economista', 'Docente'),
(1122334, 'Pedro Castillo', '1992-11-11', 32, 'pedro.castillo@ficct.edu.bo', '73011223',
'Licenciado', 'Docente');
```

-- ----- USUARIOS -----

-- codigo SERIAL en el esquema; aquí insertamos ids explícitos para claridad

```
INSERT INTO usuario (codigo, user_name, password, ci, id_rol) VALUES
(1001, 'jperez', 'pass123', 7654321, 1),
(1002, 'mgomez', 'pass123', 7890123, 2),
(1003, 'lortega', 'pass123', 1234567, 3),
(1004, 'aflores', 'pass123', 4567890, 1),
```

```
(1005, 'pcastillo', 'pass123', 1122334, 1),
(9999, 'admin', 'adminpass', 7654321, 4); -- ejemplo de administrador (asociado a persona
Juan Perez)
```

-- ----- AULAS -----

```
-- nro definido explícitamente (son SERIAL en el esquema, pero se pueden insertar
valores)
```

```
INSERT INTO aula (nro, capacidad, modulo, tipo) VALUES
```

```
(10, 40, '236', 'Aula Teórica'),
(12, 30, '236', 'Laboratorio'),
(40, 50, '236', 'Aula Teórica');
```

-- ----- MATERIAS -----

```
INSERT INTO materia (sigla, nombre, semestre, carga_horaria) VALUES
```

```
('INF342', 'Ingeniería de Software Avanzada', '6', 4),
('MAT302', 'Cálculo II', '2', 4),
('PROG101', 'Programación I', '1', 5),
('CALC1', 'Cálculo I', '1', 5),
('BD201', 'Bases de Datos', '3', 4);
```

-- ----- GRUPOS -----

```
INSERT INTO grupo (sigla) VALUES
```

```
('NX'),
('SA'),
('SC');
```

-- ----- MATERIA_GRUPO -----

```
-- combinaciones materia-grupo
```

```
INSERT INTO materia_grupo (id, sigla_materia, sigla_grupo) VALUES
(2001, 'INF342', 'NX'),
(2002, 'INF342', 'SA'),
(2003, 'MAT302', 'NX'),
(2004, 'PROG101', 'SA'),
(2005, 'CALC1', 'SC'),
(2006, 'BD201', 'SA');
```

-- ----- GESTIONES -----

```
INSERT INTO gestion (id, nombre, fecha_i, fecha_f) VALUES
(1, '2025/1', '2025-02-01', '2025-07-31'),
(2, '2025/2', '2025-08-01', '2025-12-31');
```

-- ----- CLASES -----

```
-- relacionan gestion, aula, materia_grupo y usuario (docente responsable)
-- Asigno ids de clase explícitos para poder referenciarlos en horarios/asistencia
INSERT INTO clase (id, fecha_creacion, id_gestion, nro_aula, id_materia_grupo,
usuario_codigo) VALUES
(3001, '2025-02-10 08:00:00-04', 1, 10, 2001, 1001), -- INF342 NX, docente jperez
(3002, '2025-02-10 08:05:00-04', 1, 12, 2002, 1004), -- INF342 SA, docente aflores
(3003, '2025-02-11 09:00:00-04', 1, 10, 2003, 1005), -- MAT302 NX, docente pcastillo
(3004, '2025-02-11 10:00:00-04', 1, 40, 2004, 1001), -- PROG101 SA, docente jperez
(3005, '2025-02-12 11:00:00-04', 1, 12, 2005, 1004), -- CALC1 SC, docente aflores
(3006, '2025-02-12 12:00:00-04', 1, 40, 2006, 1001); -- BD201 SA, docente jperez
```

-- ----- HORARIOS -----

```
-- uso timestamptz (ejemplo: día con horario incluyendo zona -04)
-- Día = 'Lunes' ejemplo, hora_i y hora_f con fecha cualquiera (la zona es -04 para Bolivia)
INSERT INTO horario (id, dia, hora_i, hora_f, id_clase) VALUES
(4001, 'Lunes', '2025-03-03 09:15:00-04'::timestamptz, '2025-03-03 11:30:00-
04'::timestamptz, 3001),
(4002, 'Miércoles', '2025-03-05 09:15:00-04'::timestamptz, '2025-03-05 11:30:00-
04'::timestamptz, 3001),
(4003, 'Martes', '2025-03-03 14:00:00-04'::timestamptz, '2025-03-03 16:00:00-
04'::timestamptz, 3002),
(4004, 'Jueves', '2025-03-04 09:15:00-04'::timestamptz, '2025-03-04 11:30:00-
04'::timestamptz, 3003),
(4005, 'Viernes', '2025-03-06 08:00:00-04'::timestamptz, '2025-03-06 10:00:00-
04'::timestamptz, 3004),
(4006, 'Lunes', '2025-03-03 11:45:00-04'::timestamptz, '2025-03-03 13:30:00-
04'::timestamptz, 3005),
(4007, 'Miércoles', '2025-03-05 12:00:00-04'::timestamptz, '2025-03-05 14:00:00-
```

```
04'::timestampz, 3006);
```

```
-- ----- ASISTENCIA -----
```

```
-- registros de ejemplo (fecha y hora con zona)
```

```
INSERT INTO asistencia (id, fecha, hora_registro, estado, metodo_r, observacion,  
id_clase) VALUES
```

```
(5001, '2025-03-03', '2025-03-03 09:10:00-04'::timestampz, 'Presente', 'QR', NULL, 3001),  
(5002, '2025-03-03', '2025-03-03 09:20:00-04'::timestampz, 'Retraso', 'Manual', 'Llegó  
tarde 5 min', 3001),  
(5003, '2025-03-04', '2025-03-04 09:10:00-04'::timestampz, 'Presente', 'QR', NULL, 3003),  
(5004, '2025-03-06', '2025-03-06 08:05:00-04'::timestampz, 'Presente', 'QR', NULL, 3004);
```

```
-- ----- LICENCIAS -----
```

```
-- algunos ejemplos de licencias (relacionadas a usuario y clase)
```

```
INSERT INTO licencia (nro, descripcion, fecha_hora, codigo_usuario, id_clase) VALUES
```

```
(6001, 'Permiso por enfermedad (2 días)', '2025-03-05 10:00:00-04'::timestampz, 1004,  
3002),  
(6002, 'Permiso por reunión institucional', '2025-03-10 09:00:00-04'::timestampz, 1001,  
3004);
```

DIAGRAMA RELACIONAL



CONSULTAS

-- =====

-- CONSULTAS

-- =====

-- Mostrar todos los docentes registrados

-- Permite ver los docentes y su profesión en la base de datos.

SELECT ci, nomb_comp, profesion, tipo

FROM persona

WHERE tipo = 'Docente';

	ci [PK] bigint	nomb_comp character varying (100)	profesion character varying (50)	tipo character varying (30)
1	7654321	Juan Perez	Licenciado	Docente
2	4567890	Ana Flores	Economista	Docente
3	1122334	Pedro Castillo	Licenciado	Docente

-- Listar todas las materias disponibles

-- Muestra todas las materias que existen con su carga horaria.

SELECT sigla, nombre, semestre, carga_horaria

FROM materia;

	sigla [PK] character varying (20)	nombre character varying (100)	semestre character varying (20)	carga_horaria integer
1	INF342	Ingeniería de Software Avanzada	6	4
2	MAT302	Cálculo II	2	4
3	PROG101	Programación I	1	5
4	CALC1	Cálculo I	1	5
5	BD201	Bases de Datos	3	4

-- Ver los roles existentes en el sistema

-- Lista los diferentes roles (Administrador, Docente, etc.).

SELECT * FROM rol;

	id [PK] integer	nombre character varying (50)	descripcion text
1	1	Docente	Rol para docentes
2	2	JefeCarrera	Rol para jefes de carrera
3	3	Autoridad	Rol para autoridades de facultad
4	4	Administrador	Rol con todos los permisos

-- Mostrar los grupos registrados

-- Devuelve las siglas de los grupos (por ejemplo, SA, SB, SC...).

SELECT sigla AS grupo

FROM grupo;

	grupo character varying (20)
1	NX
2	SA
3	SC

-- Listar todas las aulas y su tipo

-- Muestra cada aula con su capacidad y si es laboratorio o aula normal.

SELECT nro, capacidad, tipo

FROM aula;

	nro [PK] integer	capacidad integer	tipo character varying (30)
1	10	40	Aula Teórica
2	12	30	Laboratorio
3	40	50	Aula Teórica

-- Buscar docentes que sean Ingenieros de Sistemas

-- Encuentra a los docentes cuya profesión contiene la palabra "sistemas".

```
SELECT nombr_comp, profesion
```

```
FROM persona
```

```
WHERE profesion ILIKE '%sistemas%';
```

	nombr_comp character varying (100)	profesion character varying (50)
1	María Gomez	Ingeniero de Sistemas

-- Mostrar usuarios activos con rol “Administrador”

-- Une las tablas usuario y rol para identificar quiénes son administradores.

```
SELECT u.codigo, u.user_name, r.nombre AS rol
```

```
FROM usuario u
```

```
JOIN rol r ON u.id_rol = r.id
```

```
WHERE r.nombre = 'Administrador';
```

	codigo integer	user_name character varying (50)	rol character varying (50)
1	9999	admin	Administrador

-- Ver materias del segundo semestre

-- Filtra solo las materias que pertenecen al semestre 2.

```
SELECT sigla, nombre
```

```
FROM materia
```

```
WHERE semestre = '2';
```

	sigla [PK] character varying (20)	nombre character varying (100)
1	MAT302	Cálculo II

-- Aulas con capacidad mayor a 40 personas

-- Muestra qué aulas pueden albergar más de 40 estudiantes.

```
SELECT nro, capacidad, tipo
```

```
FROM aula
```

```
WHERE capacidad > 40;
```

	nro [PK] integer	capacidad integer	tipo character varying (30)
1	40	50	Aula Teórica

-- Mostrar licencias emitidas en la gestión “2025/1”

-- Relaciona licencia con clase y gestión para ver las licencias activas por gestión.

```
SELECT l.nro, l.descripcion, g.nombre AS gestion
```

```
FROM licencia l
```

```
JOIN clase c ON l.id_clase = c.id
```

```
JOIN gestion g ON c.id_gestion = g.id
```

```
WHERE g.nombre = '2025/1';
```

	nro integer	descripcion text	gestion character varying (50)
1	6001	Permiso por enfermedad (2 días)	2025/1
2	6002	Permiso por reunión institucional	2025/1

-- Ver el nombre del docente asignado a cada clase

-- Muestra qué docente imparte cada materia en una gestión específica.

```
SELECT c.id AS clase_id, p.nomb_comp AS docente, m.nombre AS materia, g.nombre  
AS gestion
```

```
FROM clase c
```

```
JOIN usuario u ON c.usuario_codigo = u.codigo
```

```
JOIN persona p ON u.ci = p.ci
```

```
JOIN materia_grupo mg ON c.id_materia_grupo = mg.id
```

```
JOIN materia m ON mg.sigla_materia = m.sigla
```

```
JOIN gestion g ON c.id_gestion = g.id;
```

	clase_id integer	docente character varying (100)	materia character varying (100)	gestion character varying (50)
1	3001	Juan Perez	Ingeniería de Software Avanzada	2025/1
2	3002	Ana Flores	Ingeniería de Software Avanzada	2025/1
3	3003	Pedro Castillo	Cálculo II	2025/1
4	3004	Juan Perez	Programación I	2025/1
5	3005	Ana Flores	Cálculo I	2025/1
6	3006	Juan Perez	Bases de Datos	2025/1

-- Mostrar las materias con sus respectivos grupos

-- Permite conocer qué grupos pertenecen a cada materia.

```
SELECT m.nombre AS materia, g.sigla AS grupo
```

```

FROM materia_grupo mg
JOIN materia m ON mg.sigla_materia = m.sigla
JOIN grupo g ON mg.sigla_grupo = g.sigla;

```

	materia character varying (100)	grupo character varying (20)
1	Ingeniería de Software Avanzada	NX
2	Ingeniería de Software Avanzada	SA
3	Cálculo II	NX
4	Programación I	SA
5	Cálculo I	SC
6	Bases de Datos	SA

-- Ver qué permisos tiene cada rol
-- Muestra las relaciones entre roles y permisos asignados en el sistema.

```

SELECT r.nombre AS rol, p.nombre AS permiso
FROM rol_permiso rp
JOIN rol r ON rp.id_rol = r.id
JOIN permisos p ON rp.id_permiso = p.id;

```

	rol character varying (50)	permiso character varying (50)
1	Docente	VER_GRUPO
2	Docente	REGISTRAR_ASISTENCIA
3	JefeCarrera	VER_GRUPO
4	JefeCarrera	MODIFICAR_GRUPO
5	JefeCarrera	CREAR_HORARIO
6	Autoridad	VER_GRUPO
7	Administrador	VER_GRUPO
8	Administrador	MODIFICAR_GRUPO
9	Administrador	CREAR_HORARIO
10	Administrador	MODIFICAR_HORARIO
11	Administrador	REGISTRAR_ASISTENCIA
12	Administrador	VER_REPORTES

-- Mostrar asistencias con el nombre del docente y materia
-- Indica qué docentes marcaron asistencia y en qué materia.

```

SELECT a.id, p.nomb_comp AS docente, m.nombre AS materia, a.fecha, a.estado
FROM asistencia a
JOIN clase c ON a.id_clase = c.id
JOIN usuario u ON c.usuario_codigo = u.codigo
JOIN persona p ON u.ci = p.ci
JOIN materia_grupo mg ON c.id_materia_grupo = mg.id
JOIN materia m ON mg.sigla_materia = m.sigla;

```

	id integer	docente character varying (100)	materia character varying (100)	fecha date	estado character varying (30)
1	5001	Juan Perez	Ingeniería de Software Avanzada	2025-03-03	Presente
2	5002	Juan Perez	Ingeniería de Software Avanzada	2025-03-03	Retraso
3	5003	Pedro Castillo	Cálculo II	2025-03-04	Presente
4	5004	Juan Perez	Programación I	2025-03-06	Presente

-- Contar cuántos docentes hay por tipo de profesión
-- Agrupa los docentes por profesión para analizar la distribución.

```

SELECT profesion, COUNT(*) AS cantidad_docentes
FROM persona
WHERE tipo = 'Docente'
GROUP BY profesion
ORDER BY cantidad_docentes DESC;

```

	profesion character varying (50)	cantidad_docentes bigint
1	Licenciado	2
2	Economista	1

PROCEDIMIENTOS ALMACENADOS

-- 1) Crear persona + usuario (si se desea crear ambos a la vez)

```

CREATE OR REPLACE FUNCTION sp_create_persona_usuario(
    p_ci BIGINT,

```

```

    p_nomb_comp VARCHAR,
    p_fecha_n DATE,
    p_correo VARCHAR,
    p_tel VARCHAR,
    p_profesion VARCHAR,
    p_tipo VARCHAR,
    p_user_name VARCHAR,
    p_password VARCHAR,
    p_id_rol INT
) RETURNS INT AS $$

DECLARE
    v_codigo_usuario INT;
BEGIN
    -- Inserta o actualiza persona (si ya existe, actualiza datos básicos)
    INSERT INTO PERSONA(ci, nomb_comp, fecha_n, correo, tel, profesion, tipo)
    VALUES (p_ci, p_nomb_comp, p_fecha_n, p_correo, p_tel, p_profesion, p_tipo)
    ON CONFLICT (ci) DO UPDATE
        SET nomb_comp = EXCLUDED.nomb_comp,
            fecha_n = EXCLUDED.fecha_n,
            correo = EXCLUDED.correo,
            tel = EXCLUDED.tel,
            profesion = EXCLUDED.profesion,
            tipo = EXCLUDED.tipo;

    -- Crea usuario vinculado a la persona
    INSERT INTO USUARIO(user_name, password, ci, id_rol)
    VALUES (p_user_name, p_password, p_ci, p_id_rol)
    RETURNING codigo INTO v_codigo_usuario;

    -- Registrar en bitacora (llamamos a la tabla directamente)
    INSERT INTO BITACORA(accion, estado, comentario, usuario_codigo)
    VALUES ('CREAR_PERSONA_USUARIO', 'OK', 'Creada persona y usuario con ci=' || p_ci || '
user=' || p_user_name, v_codigo_usuario);

    RETURN v_codigo_usuario;
END;
$$ LANGUAGE plpgsql;

-- 2) Asignar permiso a rol (crea si no existe)
CREATE OR REPLACE FUNCTION sp_asignar_permiso_rol(
    p_id_rol INT,
    p_id_permiso INT,
    p_descripcion TEXT DEFAULT NULL
) RETURNS VOID AS $$

BEGIN
    INSERT INTO ROL_PERMISO(id_rol, id_permiso, descripcion)
    VALUES (p_id_rol, p_id_permiso, p_descripcion)
    ON CONFLICT (id_rol, id_permiso) DO UPDATE
        SET descripcion = EXCLUDED.descripcion;

```

```

INSERT INTO BITACORA(accion, estado, comentario, usuario_codigo)
VALUES ('ASIGNAR_PERMISO', 'OK', 'Permiso ' || p_id_permiso || ' asignado a rol ' || p_id_rol,
NULL);
END;
$$ LANGUAGE plpgsql;

```

-- 3) Crear clase + horario (valida conflictos de aula)

```
CREATE OR REPLACE FUNCTION sp_crear_clase_con_horario(

```

```

    p_id_gestion INT,
    p_nro_aula INT,
    p_id_materia_grupo INT,
    p_usuario_codigo INT,
    p_dia VARCHAR,
    p_hora_i TIMESTAMPTZ,
    p_hora_f TIMESTAMPTZ
)
```

) RETURNS INT AS \$\$

DECLARE

```

    v_id_clase INT;
    v_conflicto INT;

```

BEGIN

```

    IF p_hora_i >= p_hora_f THEN
        RAISE EXCEPTION 'hora_i debe ser menor que hora_f';
    END IF;

```

```

    SELECT 1 INTO v_conflicto
    FROM HORARIO h
    JOIN CLASE c ON h.id_clase = c.id
    WHERE c.nro_aula = p_nro_aula
    AND h.dia = p_dia
    AND tstzrange(h.hora_i, h.hora_f, '[]') && tstzrange(p_hora_i, p_hora_f, '[]')
    LIMIT 1;

```

IF FOUND THEN

```

        RAISE EXCEPTION 'Conflictivo de horario en aula % en % entre % y %', p_nro_aula, p_dia,
    p_hora_i, p_hora_f;
    END IF;

```

```

    INSERT INTO CLASE(id_gestion, nro_aula, id_materia_grupo, usuario_codigo)
    VALUES (p_id_gestion, p_nro_aula, p_id_materia_grupo, p_usuario_codigo)
    RETURNING id INTO v_id_clase;

```

```

    INSERT INTO HORARIO(dia, hora_i, hora_f, id_clase)
    VALUES (p_dia, p_hora_i, p_hora_f, v_id_clase);

```

```

    INSERT INTO BITACORA(accion, estado, comentario, usuario_codigo)

```

```

    VALUES ('CREAR_CLASE_HORARIO', 'OK', 'Clase ' || v_id_clase || ' aula ' || p_nro_aula || ' dia '
    || p_dia, p_usuario_codigo);

```

```

    RETURN v_id_clase;

```

END;

```

$$ LANGUAGE plpgsql;

-- 4) Registrar asistencia
CREATE OR REPLACE FUNCTION sp_registrar_asistencia(
    p_id_clase INT,
    p_fecha DATE,
    p_estado VARCHAR,
    p_metodo_r VARCHAR,
    p_observacion TEXT
) RETURNS INT AS $$

DECLARE
    v_id_asistencia INT;
    v_gestion_id INT;
    v_fecha_i DATE;
    v_fecha_f DATE;

BEGIN
    IF NOT EXISTS (SELECT 1 FROM CLASE WHERE id = p_id_clase) THEN
        RAISE EXCEPTION 'La clase % no existe', p_id_clase;
    END IF;

    SELECT g.id, g.fecha_i, g.fecha_f INTO v_gestion_id, v_fecha_i, v_fecha_f
    FROM CLASE c JOIN GESTION g ON c.id_gestion = g.id
    WHERE c.id = p_id_clase;

    IF v_gestion_id IS NOT NULL THEN
        IF p_fecha < v_fecha_i OR p_fecha > v_fecha_f THEN
            RAISE EXCEPTION 'La fecha % no está dentro de la gestión (% a %)', p_fecha, v_fecha_i, v_fecha_f;
        END IF;
    END IF;

    INSERT INTO ASISTENCIA(fecha, hora_registro, estado, metodo_r, observacion, id_clase)
    VALUES (p_fecha, CURRENT_TIMESTAMP, p_estado, p_metodo_r, p_observacion,
    p_id_clase)
    RETURNING id INTO v_id_asistencia;

    INSERT INTO BITACORA(accion, estado, comentario, usuario_codigo)
    VALUES ('REGISTRAR_ASISTENCIA', 'OK', 'Asistencia ' || v_id_asistencia || ' clase ' ||
    p_id_clase || ' estado ' || p_estado, NULL);

    RETURN v_id_asistencia;
END;
$$ LANGUAGE plpgsql;

```

TRIGGERS

```
-- =====
```

-- A) Trigger para calcular/actualizar la edad en PERSONA basado en fecha_n

```
CREATE OR REPLACE FUNCTION fnc_set_edad_persona() RETURNS TRIGGER AS  
$$
```

```
BEGIN
```

```
    IF NEW.fecha_n IS NOT NULL THEN
```

```
        NEW.edad := FLOOR(EXTRACT(YEAR FROM AGE(NEW.fecha_n)));
```

```
    ELSE
```

```
        NEW.edad := NULL;
```

```
    END IF;
```

```
    RETURN NEW;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_set_edad_persona
```

```
BEFORE INSERT OR UPDATE ON PERSONA
```

```
FOR EACH ROW
```

```
EXECUTE FUNCTION fnc_set_edad_persona();
```

-- B) Trigger para chequear solapamiento de horarios al insertar en HORARIO

```
CREATE OR REPLACE FUNCTION fnc_check_horario_conflict() RETURNS TRIGGER
```

AS \$\$

DECLARE

v_nro_aula INT;

v_conflicto INT;

BEGIN

-- Obtener aula de la clase a insertar/modificar

SELECT nro_aula INTO v_nro_aula FROM CLASE WHERE id = NEW.id_clase;

IF v_nro_aula IS NULL THEN

RAISE EXCEPTION 'La clase % no existe o no tiene aula asignada',
NEW.id_clase;

END IF;

-- Comprobar conflictos: misma aula, mismo dia, solapamiento de horas

SELECT 1 INTO v_conflicto

FROM HORARIO h

JOIN CLASE c ON h.id_clase = c.id

WHERE c.nro_aula = v_nro_aula

AND h.dia = NEW.dia

AND h.id_clase <> NEW.id_clase

AND tstzrange(h.hora_i, h.hora_f, '[') && tstzrange(NEW.hora_i, NEW.hora_f, '[')

LIMIT 1;

IF FOUND THEN

```
        RAISE EXCEPTION 'Conflict de horario detectado en aula % para % entre % y
%', v_nro_aula, NEW.dia, NEW.hora_i, NEW.hora_f;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_check_horario_conflict
BEFORE INSERT OR UPDATE ON HORARIO
FOR EACH ROW
EXECUTE FUNCTION fnc_check_horario_conflict();
```

-- C) Trigger genérico para insertar en BITACORA cada vez que se
inserte/actualice/elimine en tablas críticas

```
CREATE OR REPLACE FUNCTION fnc_bitacora_registrar() RETURNS TRIGGER AS $$

DECLARE
    v_accion TEXT;
    v_usuario_codigo INT := NULL;

BEGIN
    IF TG_OP = 'INSERT' THEN
        v_accion := 'INSERT_' || TG_TABLE_NAME;
```

```
-- intentar obtener usuario asociado dentro del registro si existe

BEGIN

v_usuario_codigo := NEW.usuario_codigo;

EXCEPTION WHEN OTHERS THEN

v_usuario_codigo := NULL;

END;

ELSIF TG_OP = 'UPDATE' THEN

v_accion := 'UPDATE_' || TG_TABLE_NAME;

BEGIN

v_usuario_codigo := COALESCE(NEW.usuario_codigo, OLD.usuario_codigo);

EXCEPTION WHEN OTHERS THEN

v_usuario_codigo := NULL;

END;

ELSIF TG_OP = 'DELETE' THEN

v_accion := 'DELETE_' || TG_TABLE_NAME;

BEGIN

v_usuario_codigo := OLD.usuario_codigo;

EXCEPTION WHEN OTHERS THEN

v_usuario_codigo := NULL;

END;

ELSE

v_accion := TG_OP || '_' || TG_TABLE_NAME;
```

END IF;

```
INSERT INTO BITACORA(accion, estado, comentario, usuario_codigo)
VALUES (v_accion, 'OK', 'Trigger: ' || v_accion || ' on ' || TG_TABLE_NAME,
v_usuario_codigo);
```

```
RETURN NULL; -- triggers AFTER ... FOR EACH ROW: return value ignored
END;
```

```
$$ LANGUAGE plpgsql;
```

-- Asignamos este trigger AFTER a varias tablas críticas

```
CREATE TRIGGER trg_bitacora_usuario
AFTER INSERT OR UPDATE OR DELETE ON USUARIO
FOR EACH ROW EXECUTE FUNCTION fnc_bitacora_registrar();
```

```
CREATE TRIGGER trg_bitacora_clase
```

```
AFTER INSERT OR UPDATE OR DELETE ON CLASE
FOR EACH ROW EXECUTE FUNCTION fnc_bitacora_registrar();
```

```
CREATE TRIGGER trg_bitacora_asistencia
```

```
AFTER INSERT OR UPDATE OR DELETE ON ASISTENCIA
FOR EACH ROW EXECUTE FUNCTION fnc_bitacora_registrar();
```

```
CREATE TRIGGER trg_bitacora_horario
```

```
AFTER INSERT OR UPDATE OR DELETE ON HORARIO
```

```
FOR EACH ROW EXECUTE FUNCTION fnc_bitacora_registrar();
```

-- D) Trigger para validar que la fecha de ASISTENCIA esté dentro del período de la GESTIÓN

```
CREATE OR REPLACE FUNCTION fnc_validar_asistencia_en_gestion() RETURNS
TRIGGER AS $$
```

```
DECLARE
```

```
    v_fecha_i DATE;
```

```
    v_fecha_f DATE;
```

```
BEGIN
```

```
    -- Obtener gestion asociada a la clase
```

```
    SELECT g.fecha_i, g.fecha_f INTO v_fecha_i, v_fecha_f
```

```
    FROM CLASE c JOIN GESTION g ON c.id_gestion = g.id
```

```
    WHERE c.id = NEW.id_clase;
```

```
    IF v_fecha_i IS NOT NULL AND v_fecha_f IS NOT NULL THEN
```

```
        IF NEW.fecha < v_fecha_i OR NEW.fecha > v_fecha_f THEN
```

```
            RAISE EXCEPTION 'La fecha de asistencia % no está dentro de la gestión (% - %)', NEW.fecha, v_fecha_i, v_fecha_f;
```

```
        END IF;
```

```
END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

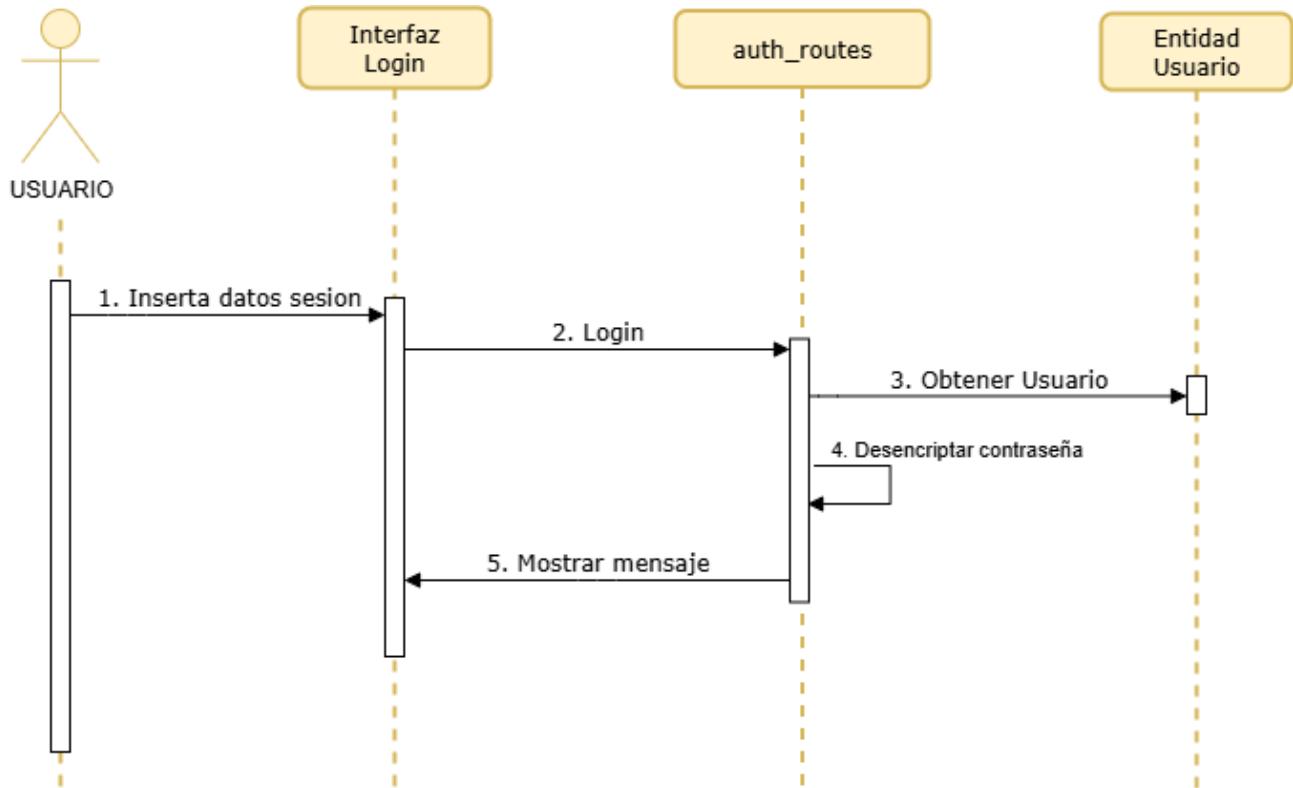
CREATE TRIGGER trg_validar_asistencia_en_gestion
BEFORE INSERT OR UPDATE ON ASISTENCIA
FOR EACH ROW
EXECUTE FUNCTION fnc_validar_asistencia_en_gestion();
```

6.2. DISEÑAR CASOS DE USO

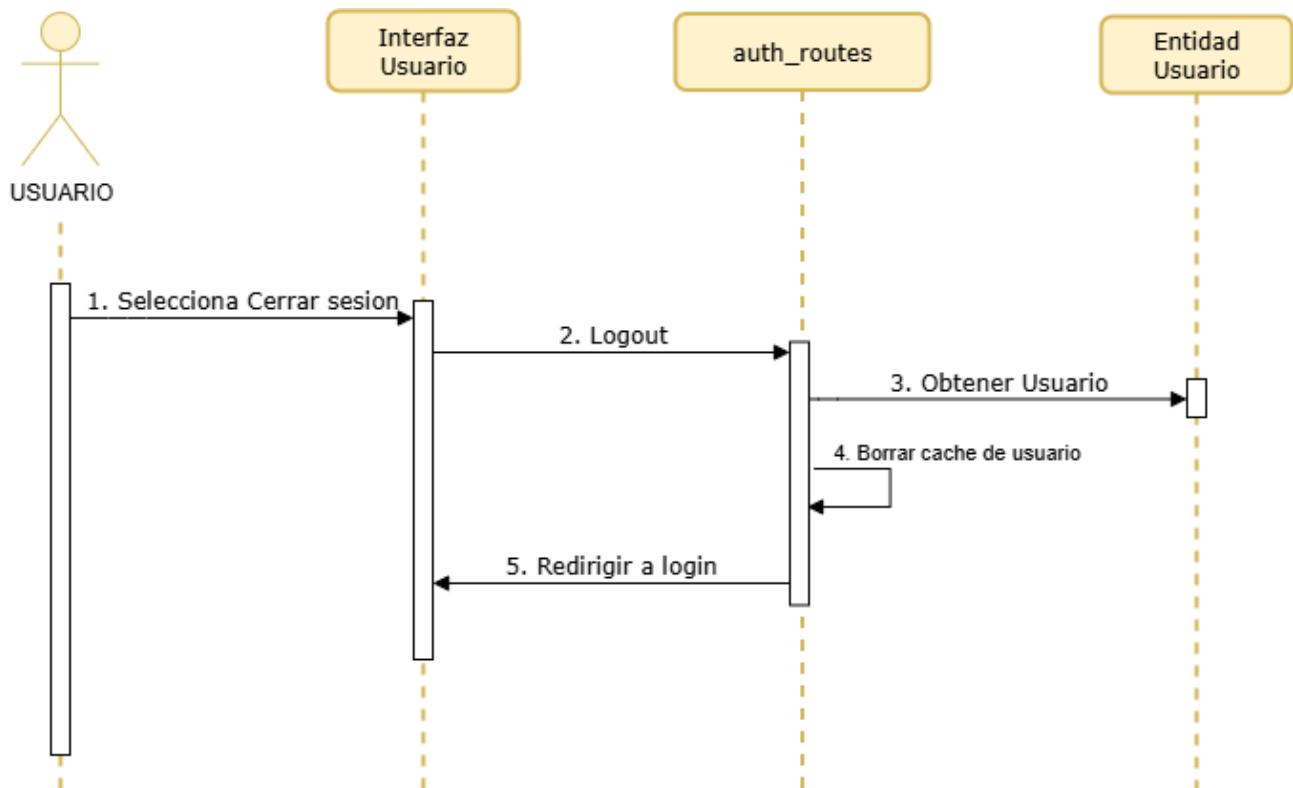
DIAGRAMA DE SECUENCIA

CICLO#1

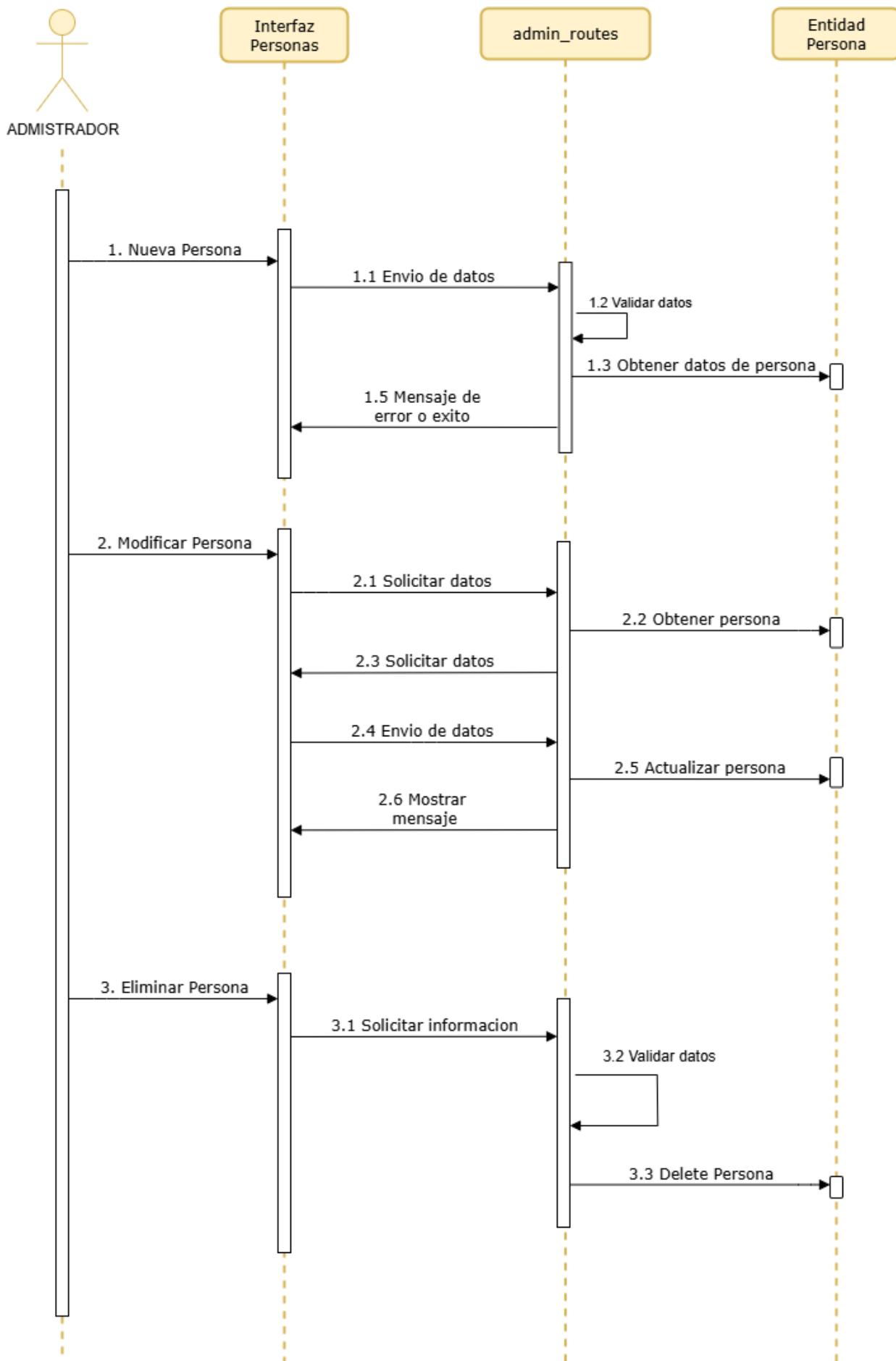
CU13. INICIAR SESIÓN



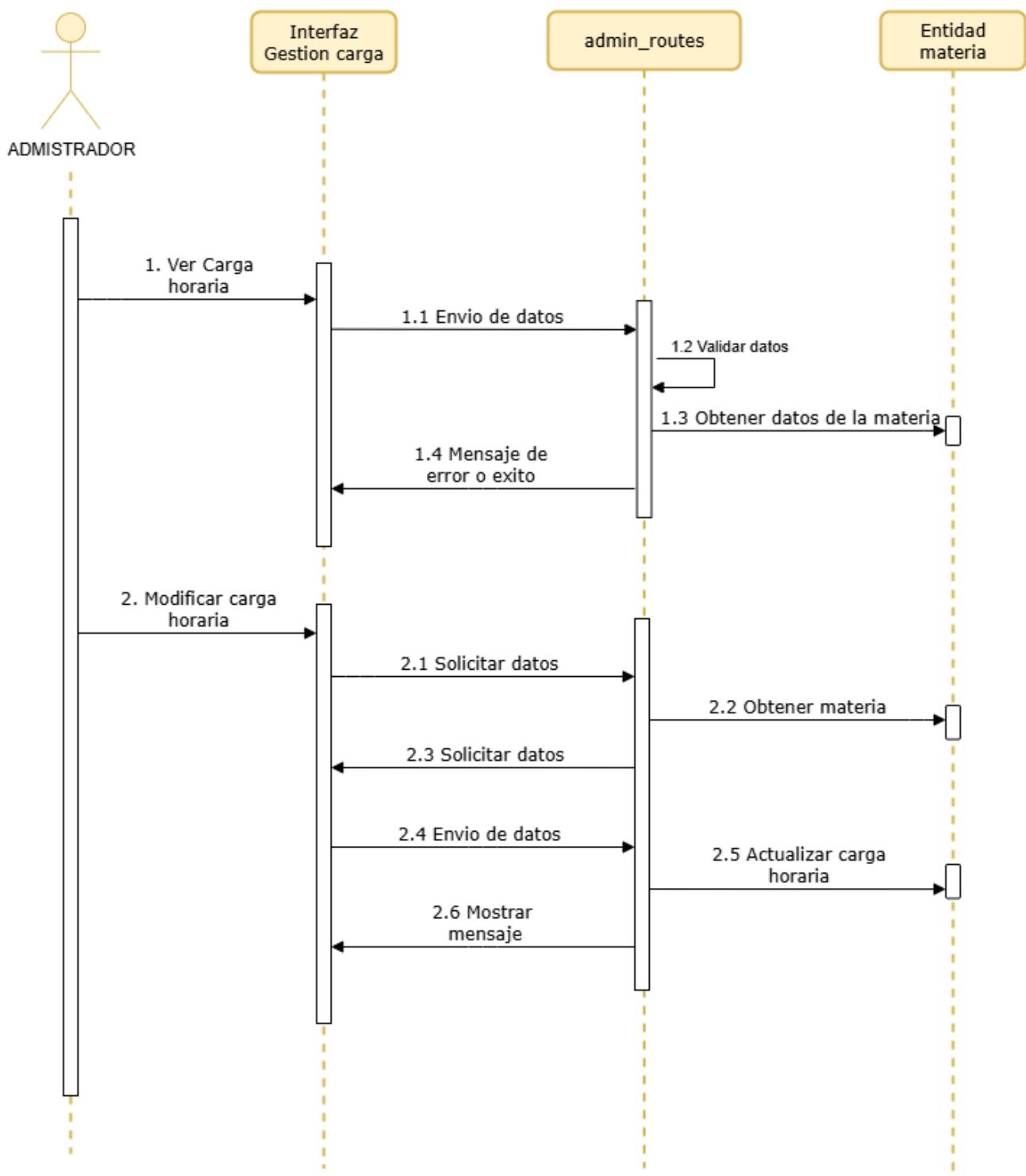
CU14. CERRAR SESIÓN



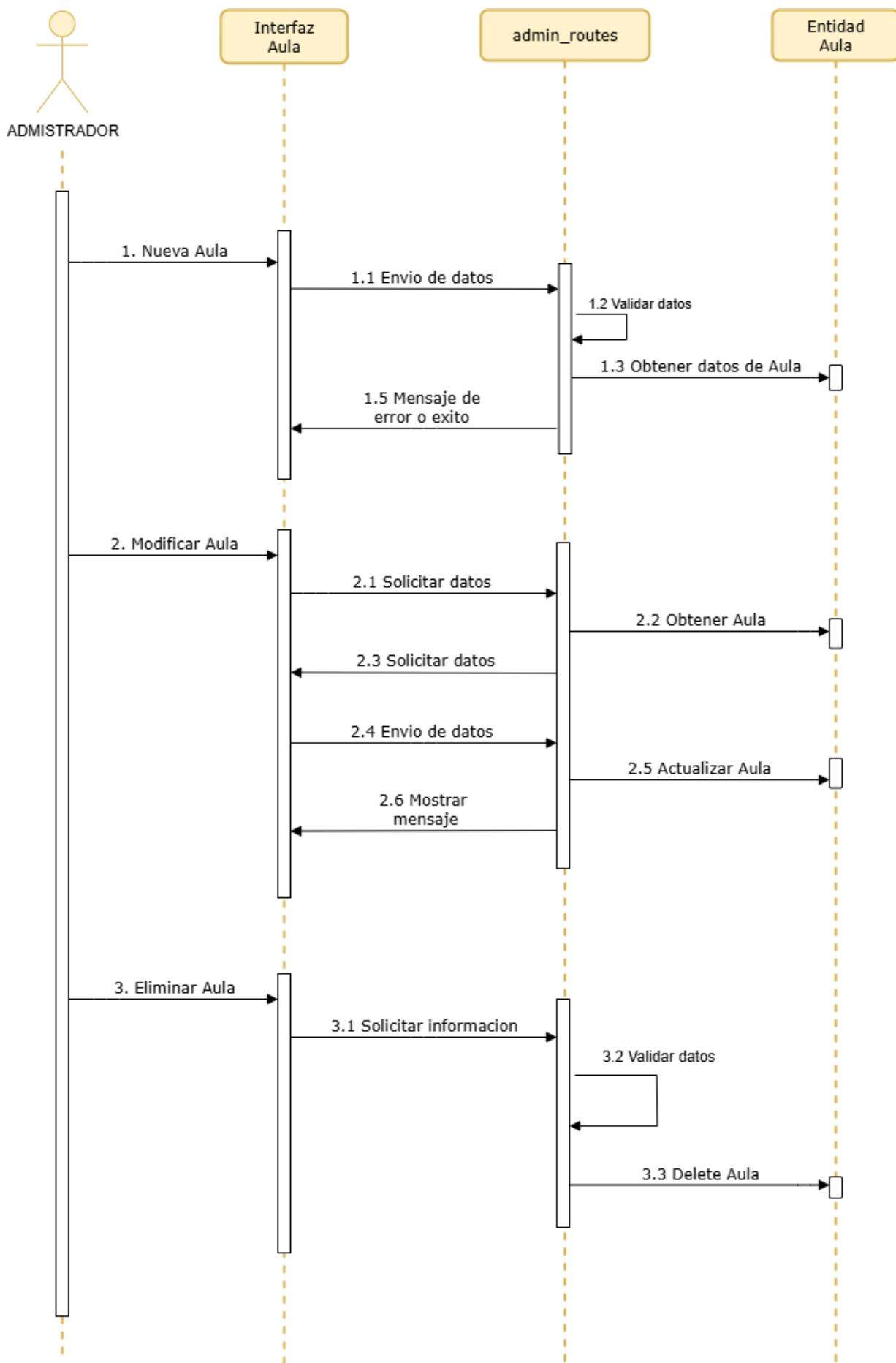
CU3. GESTIONAR DOCENTE



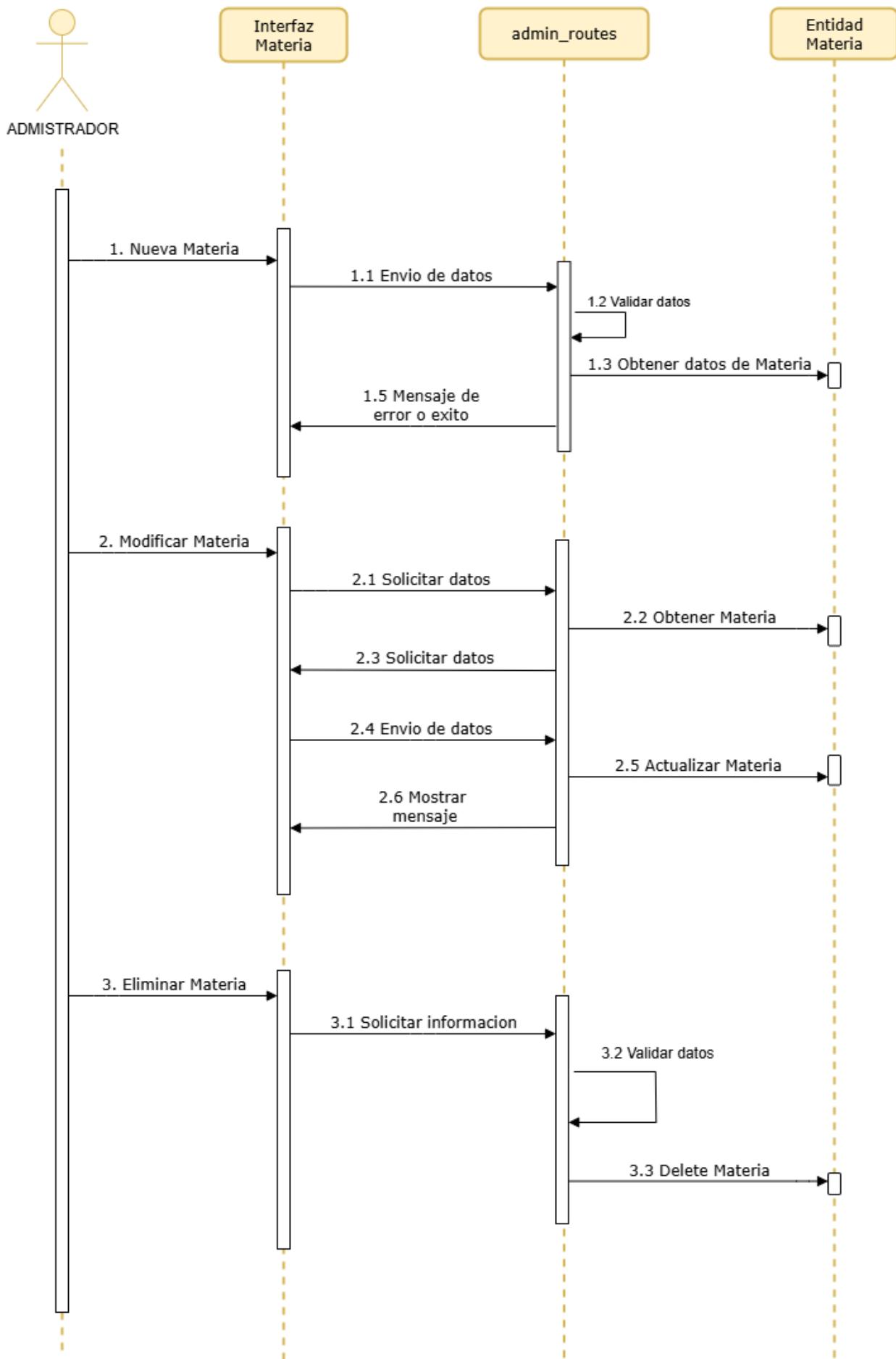
CU4. GESTIONAR CARGA HORARIA DE DOCENTE



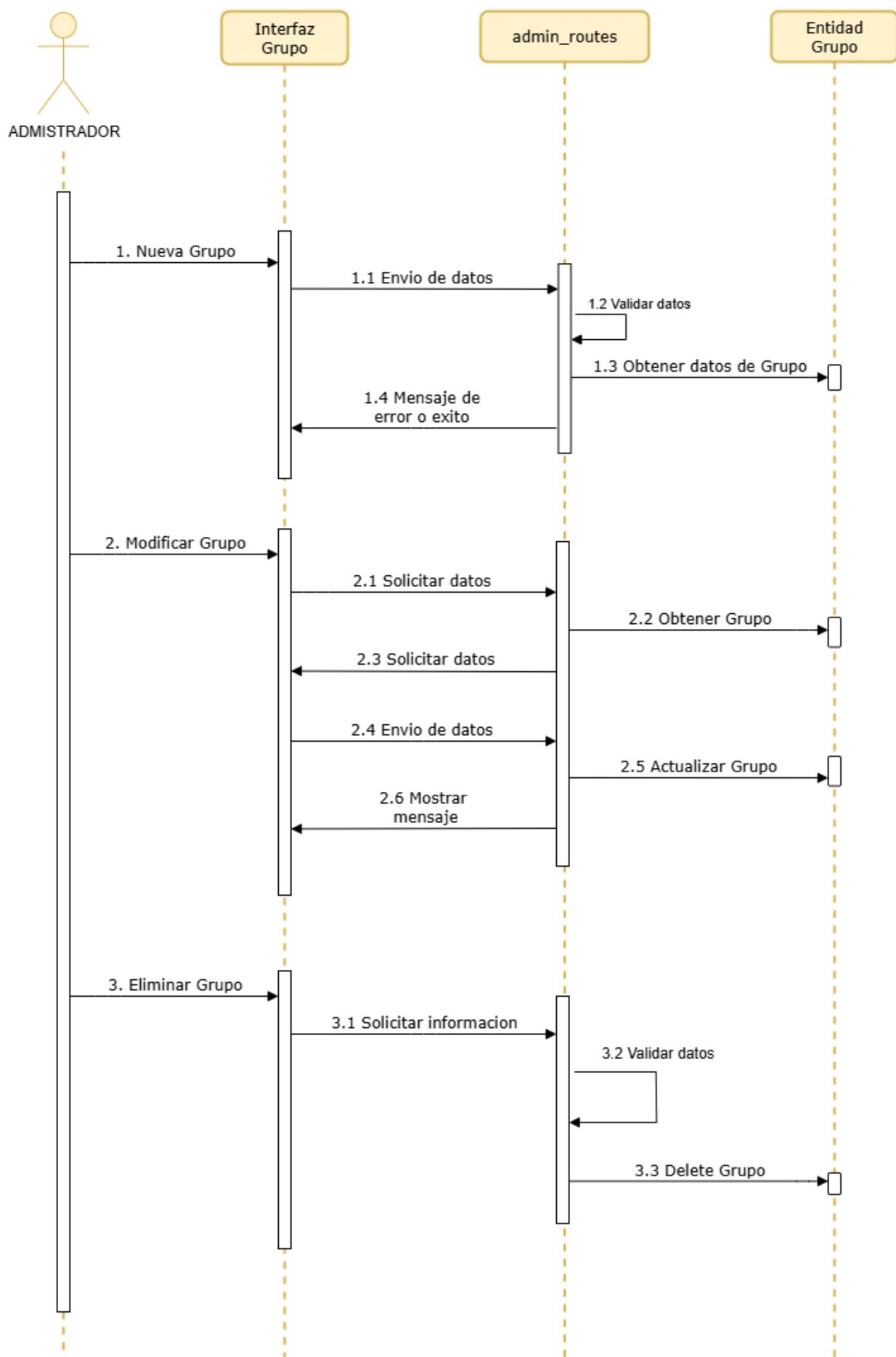
CU20. GESTIONAR AULA



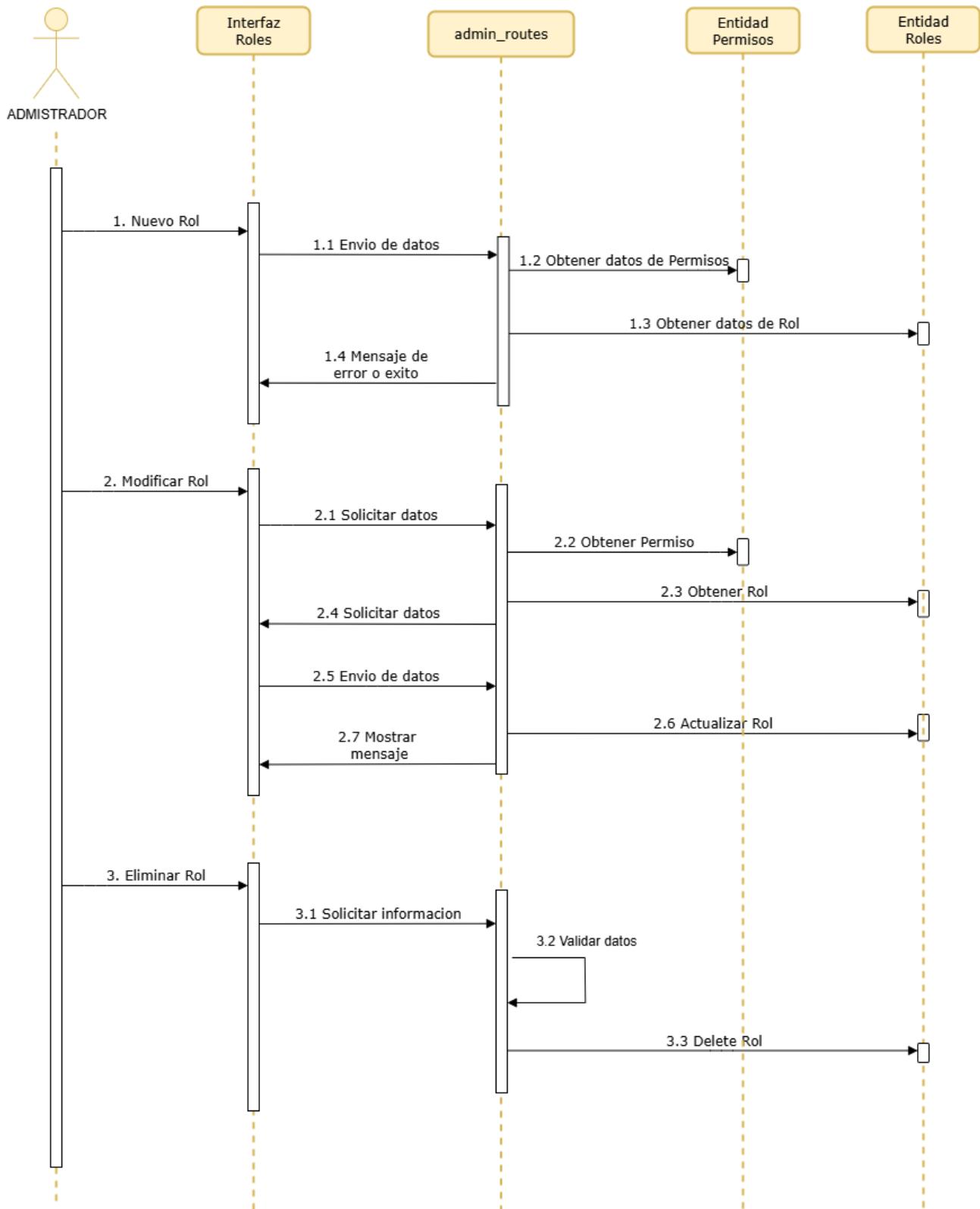
CU5. GESTIONAR MATERIA



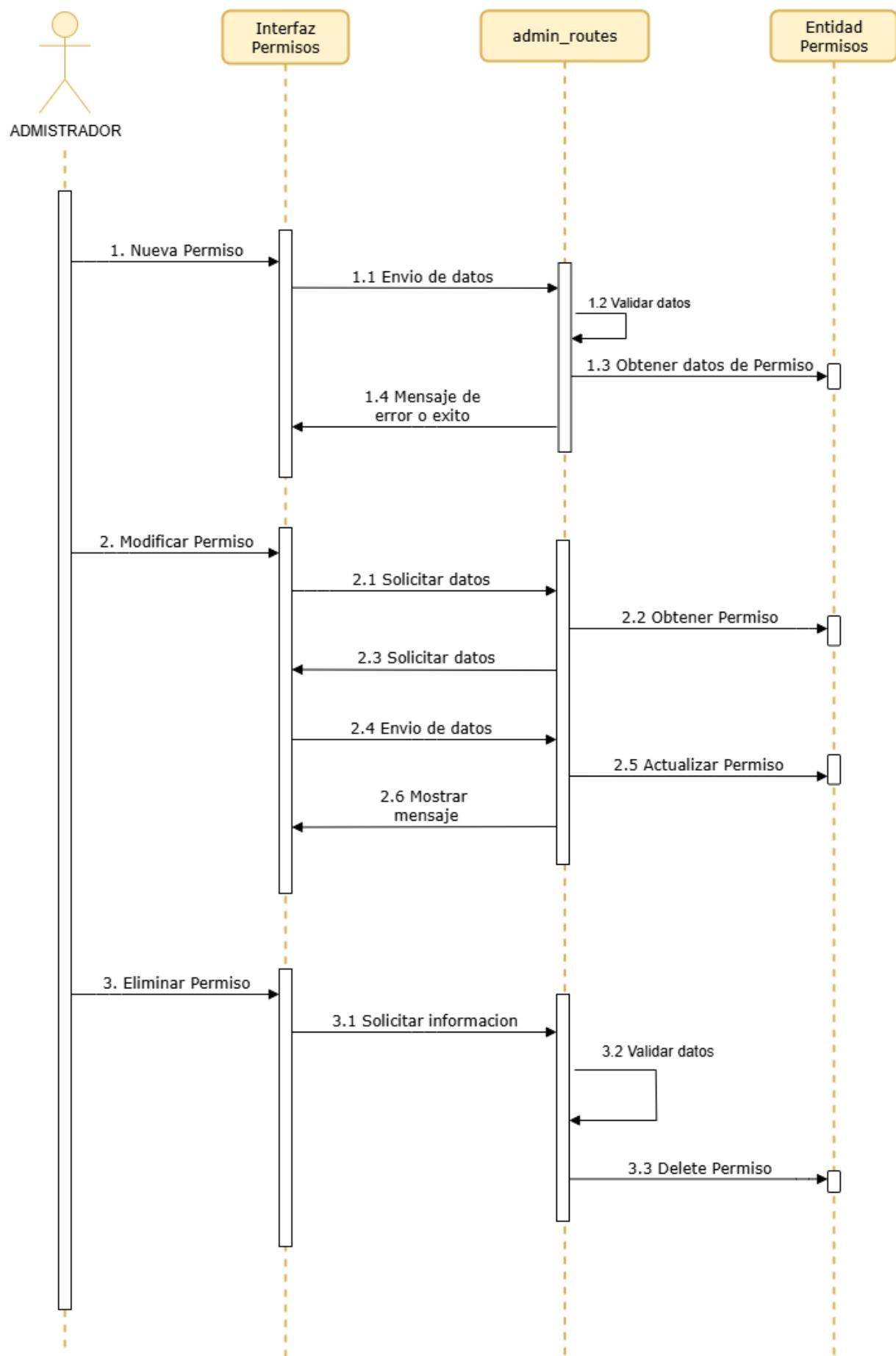
CU6. GESTIONAR GRUPO



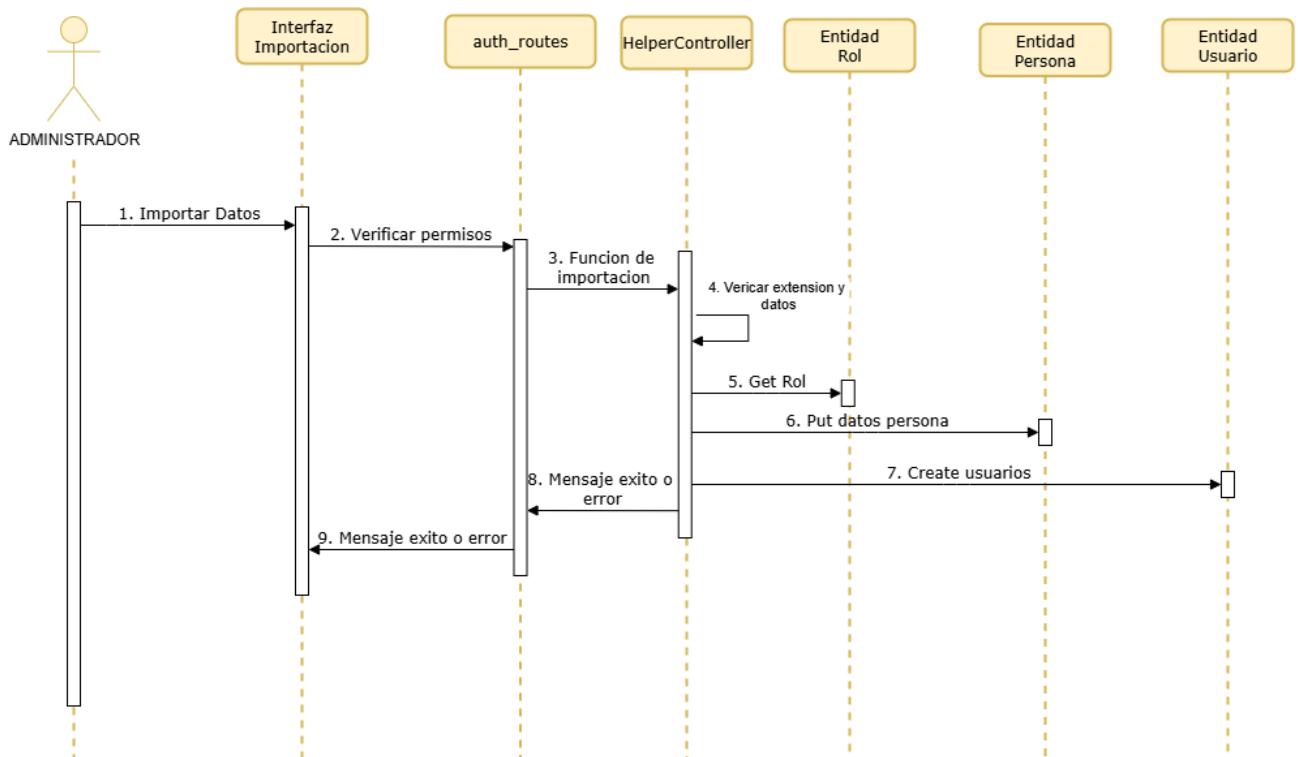
CU17. GESTIONAR ROL



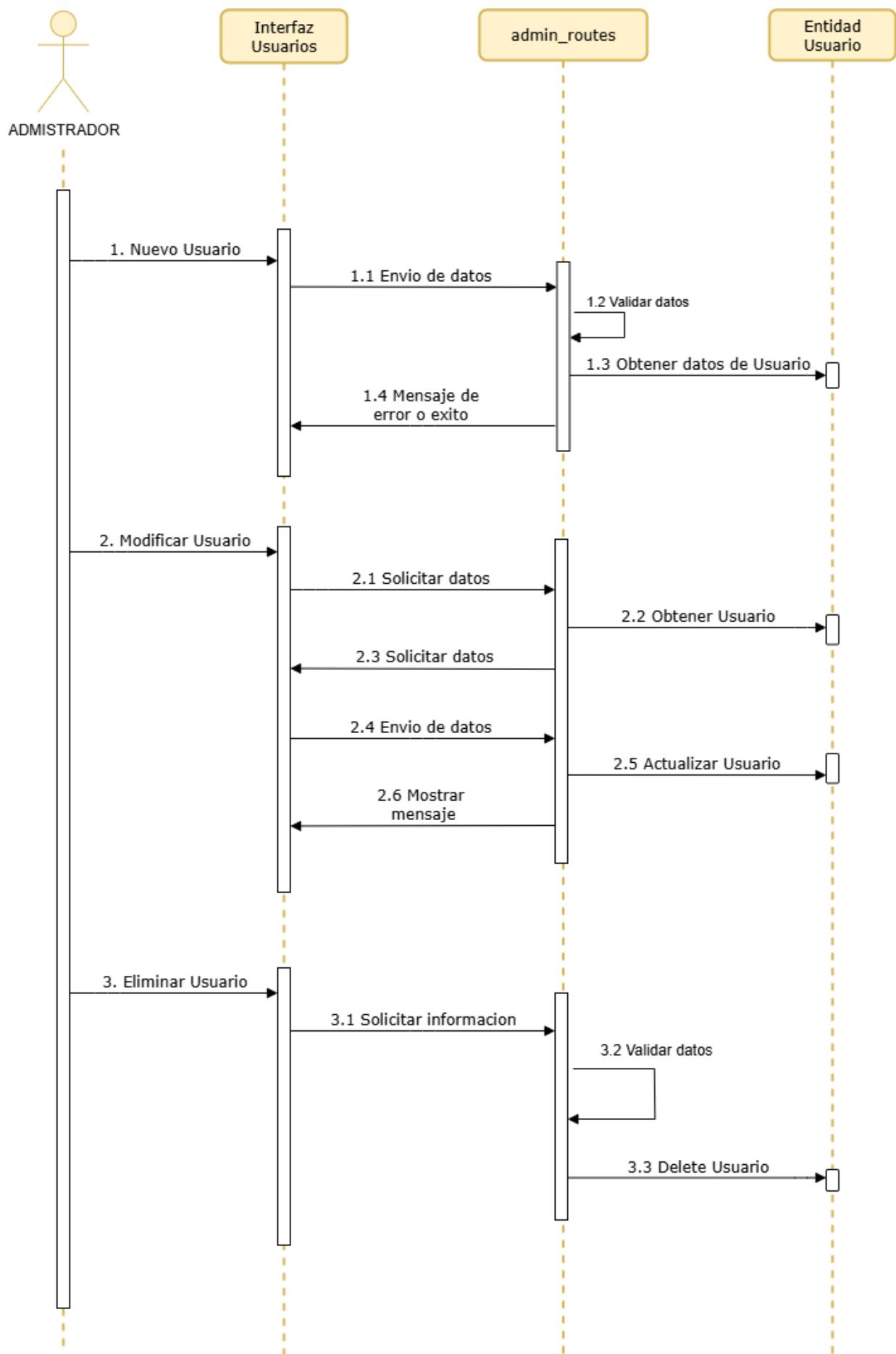
CU18. GESTIONAR PERMISO



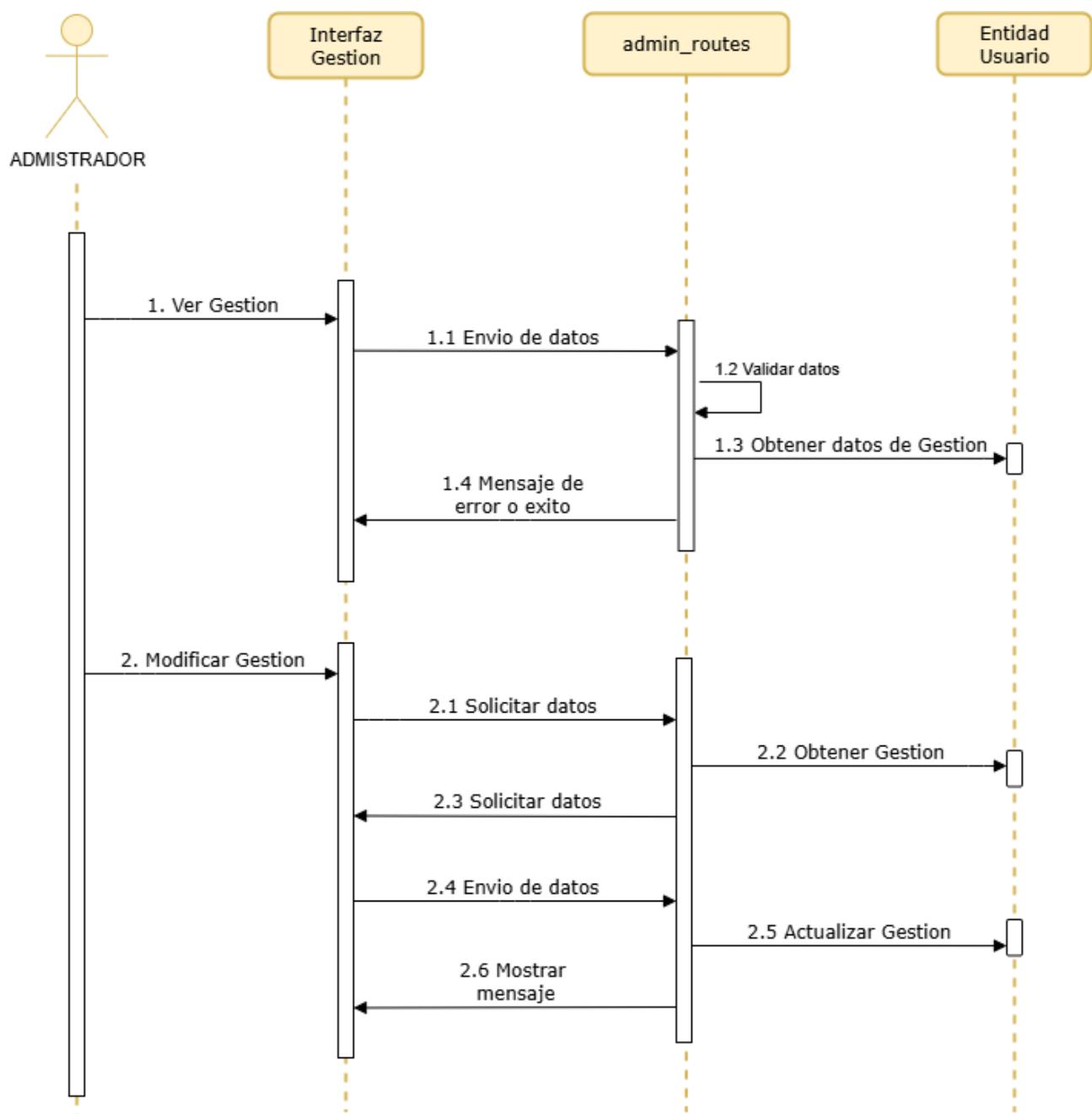
CU12. IMPORTAR USUARIOS



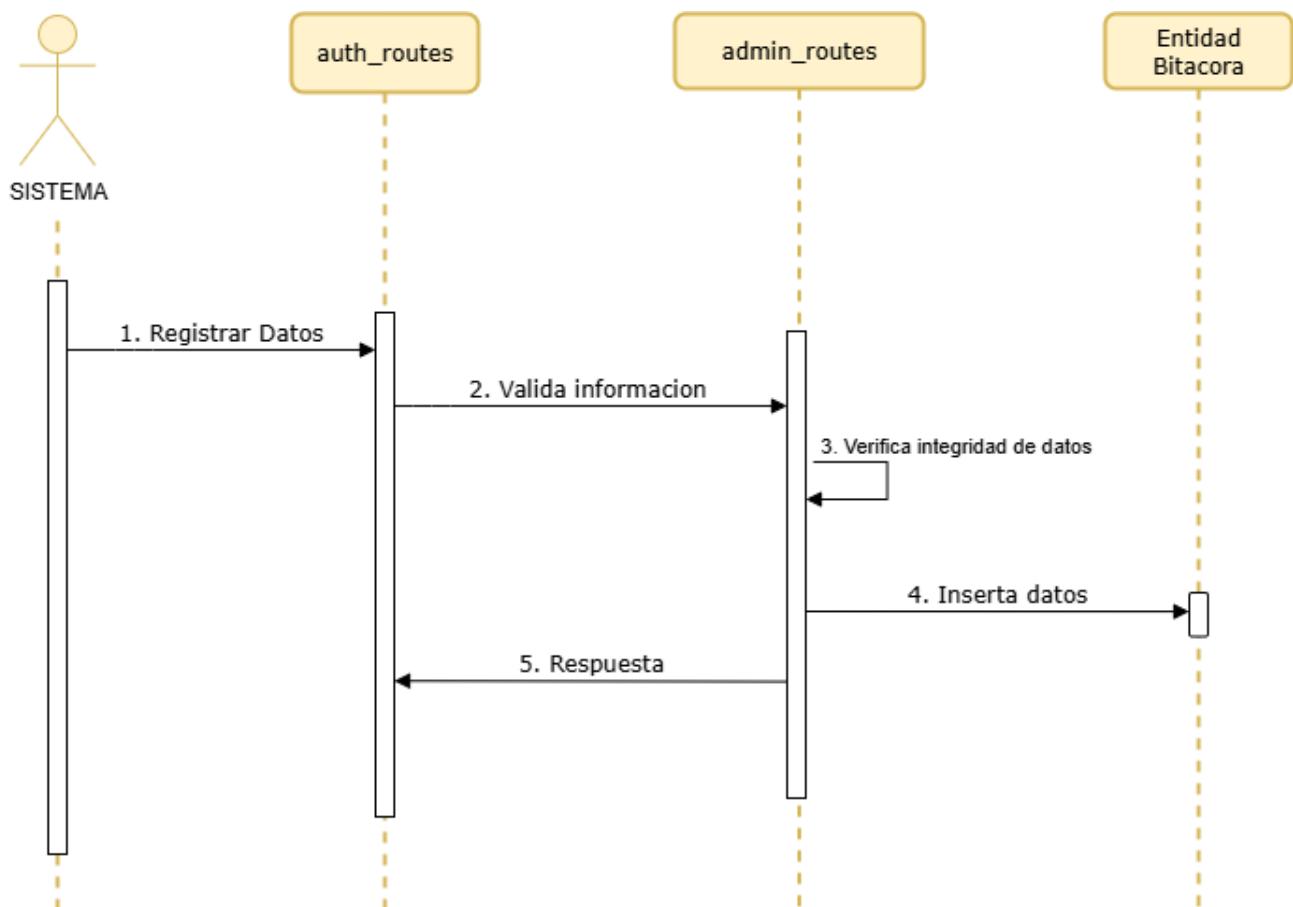
CU15. GESTIONAR USUARIO



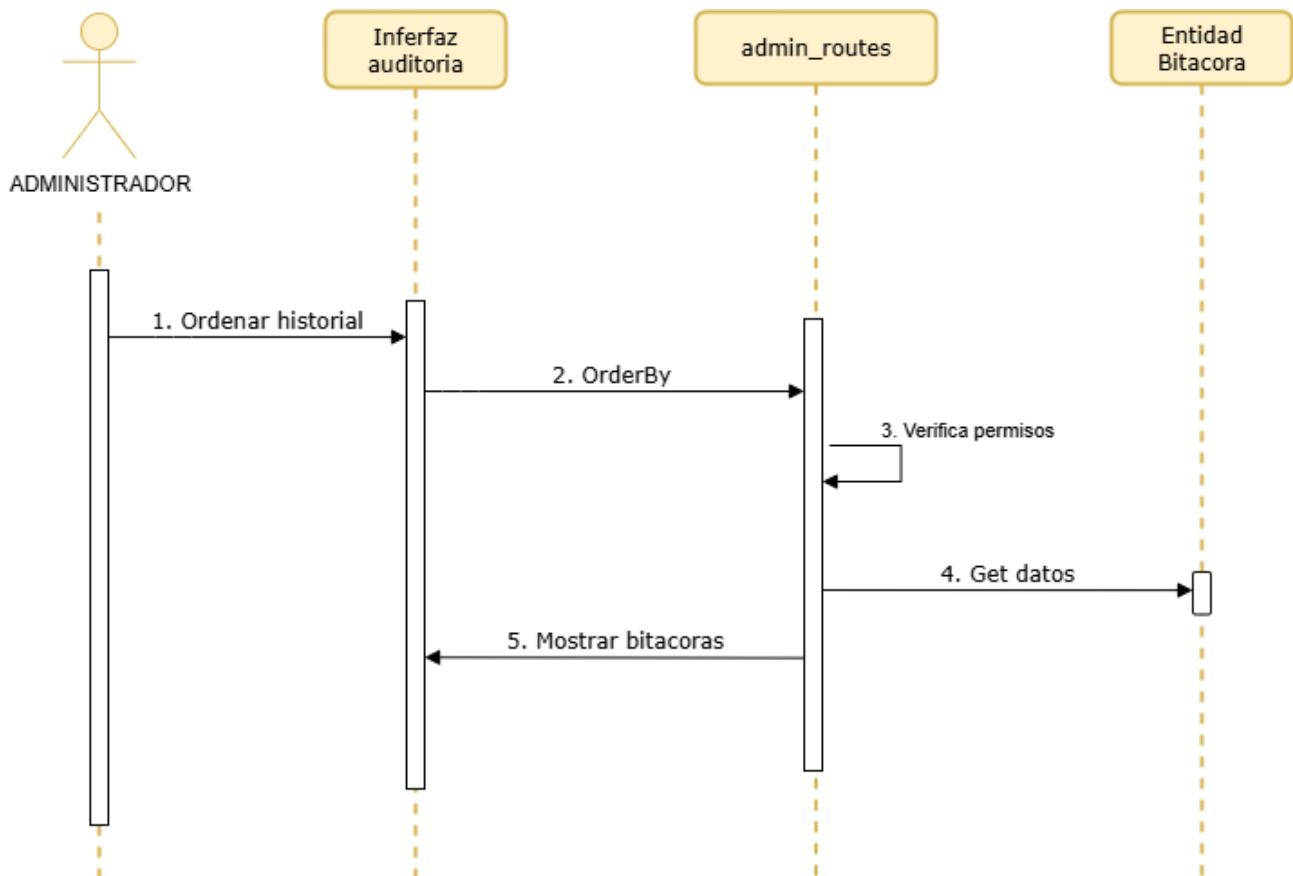
CU19. GESTIONAR PERIODOS ACADÉMICOS



CU1. REGISTRAR HISTORIAL DE CAMBIOS

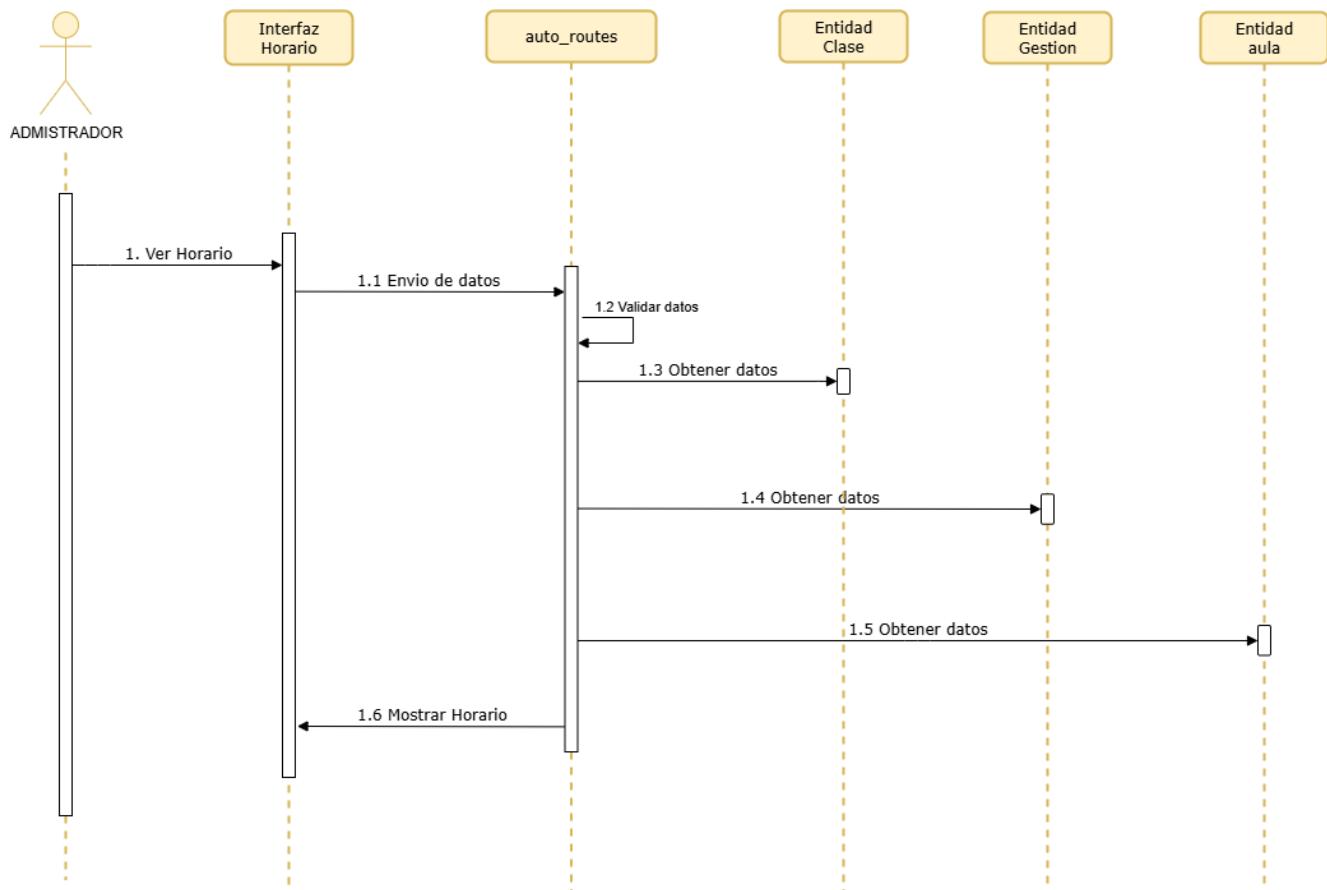


CU2. CONSULTAR HISTORIAL DE ACCIONES

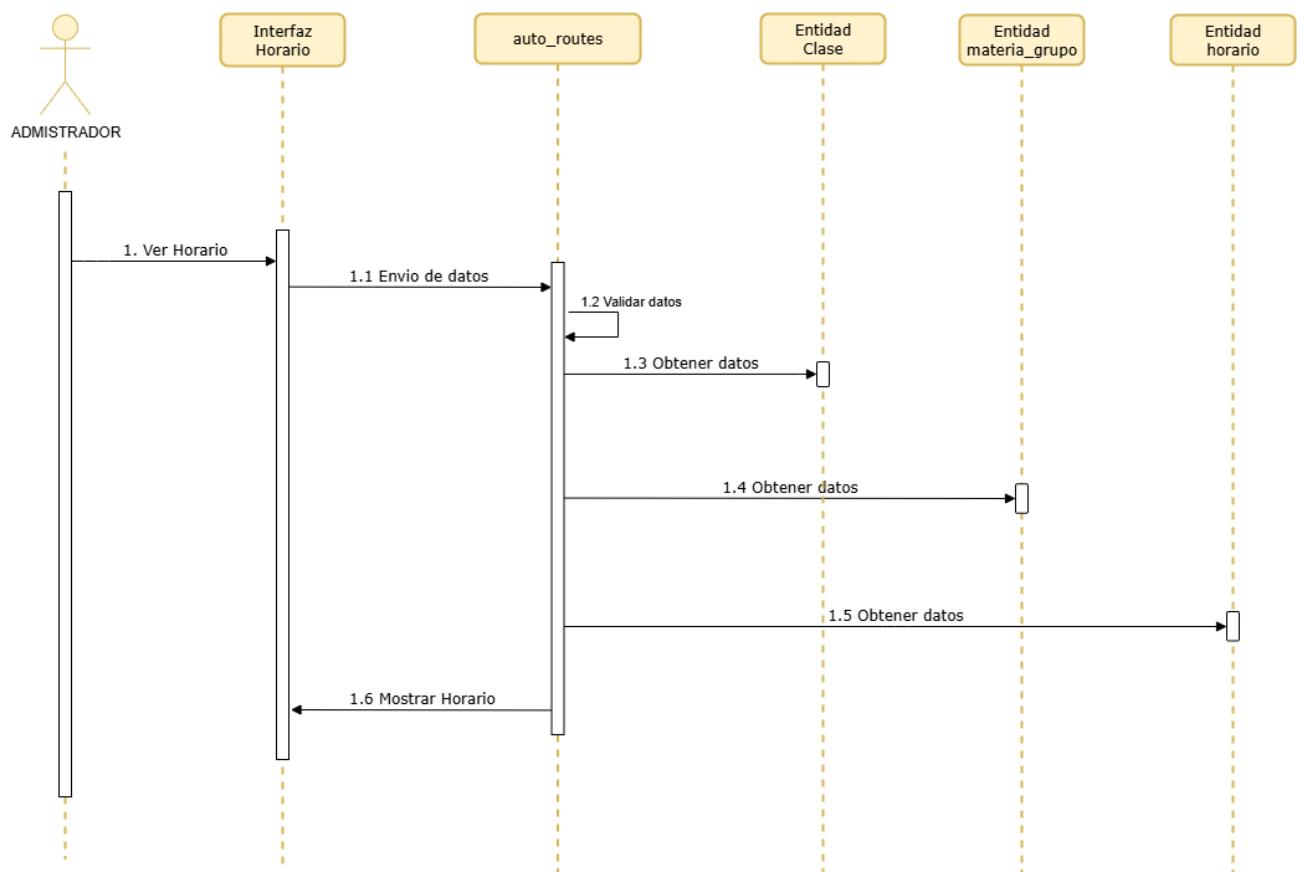


CICLO #2

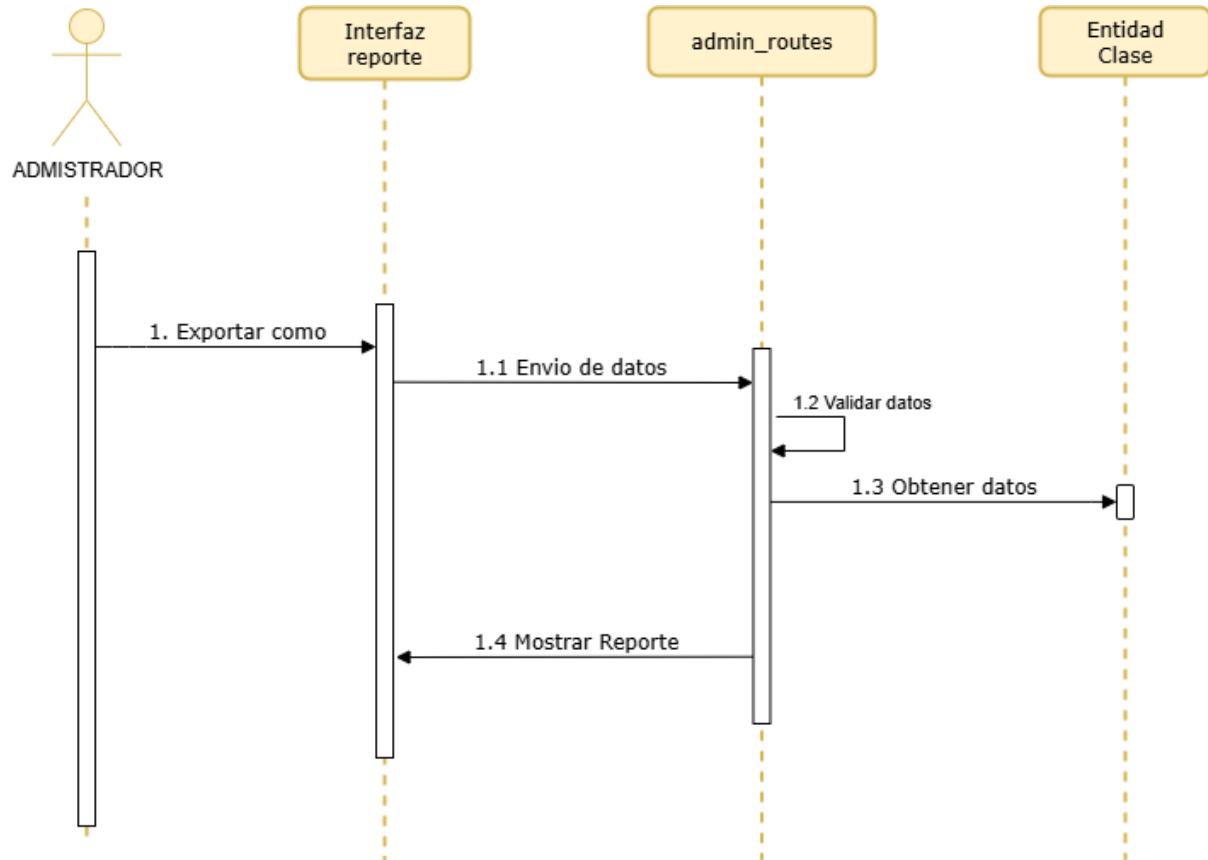
CU8: CONSULTAR HORARIOS SEMANALES



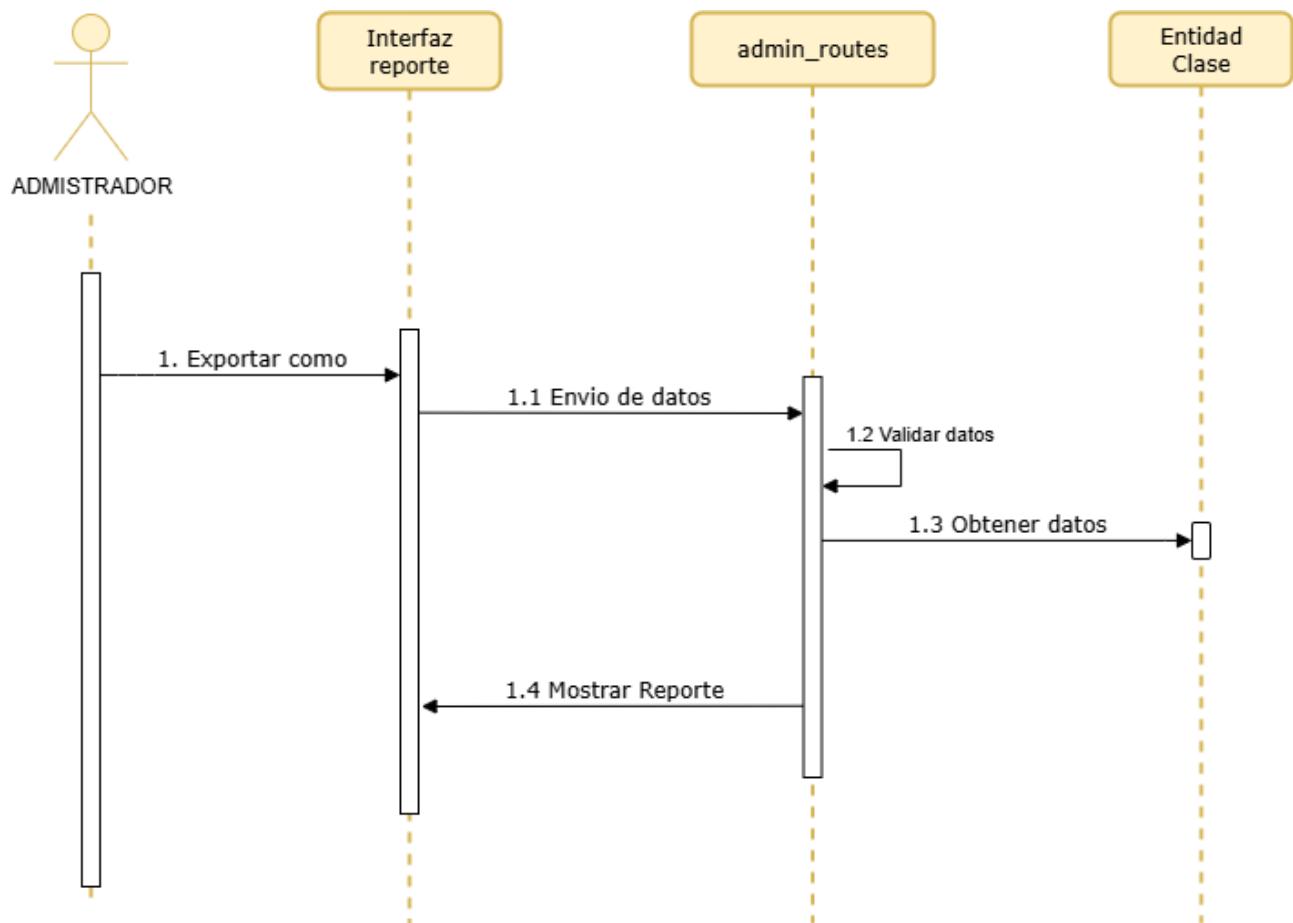
CU21: CONSULTAR HORARIO POR DOCENTE



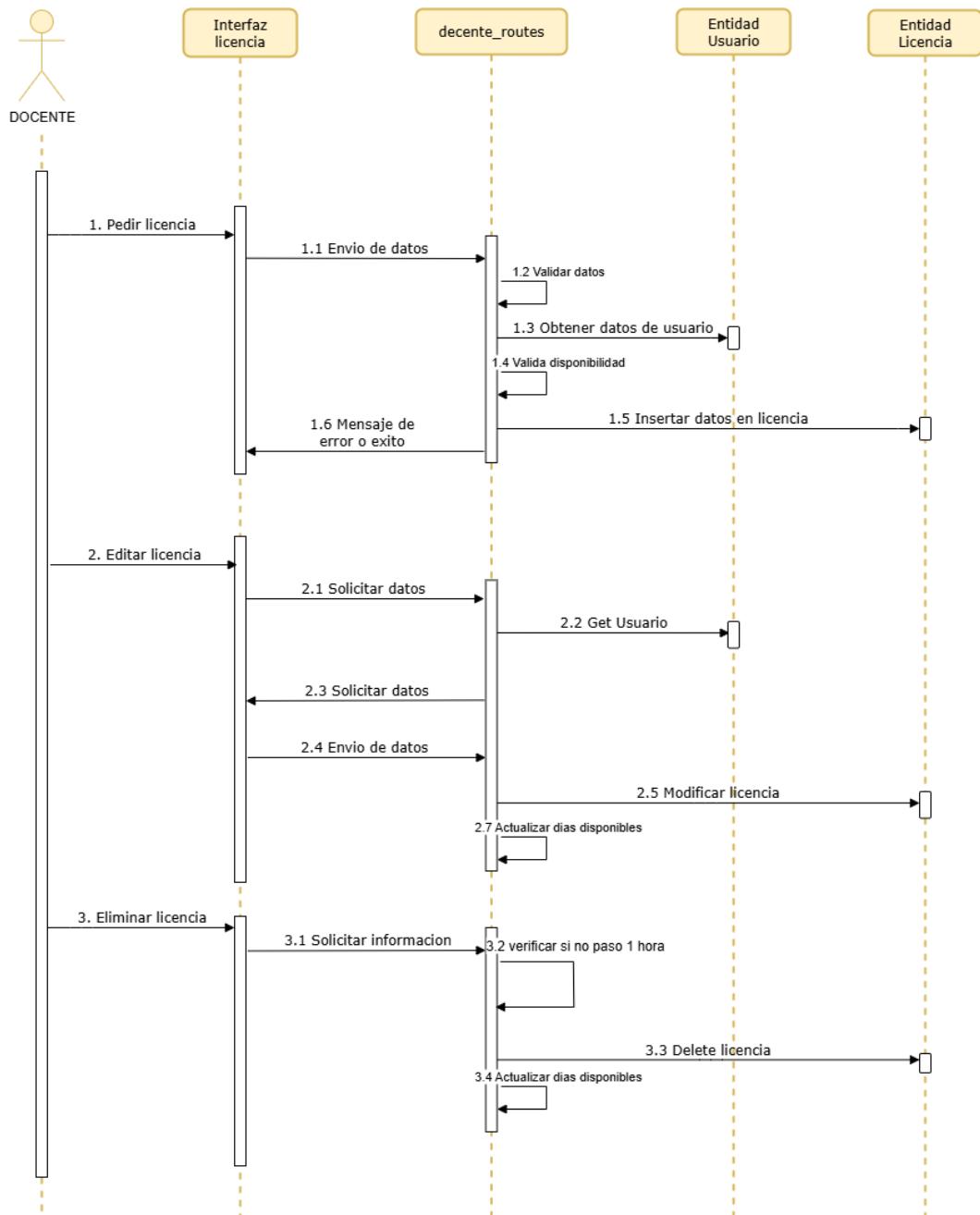
CU9: GENERAR REPORTES



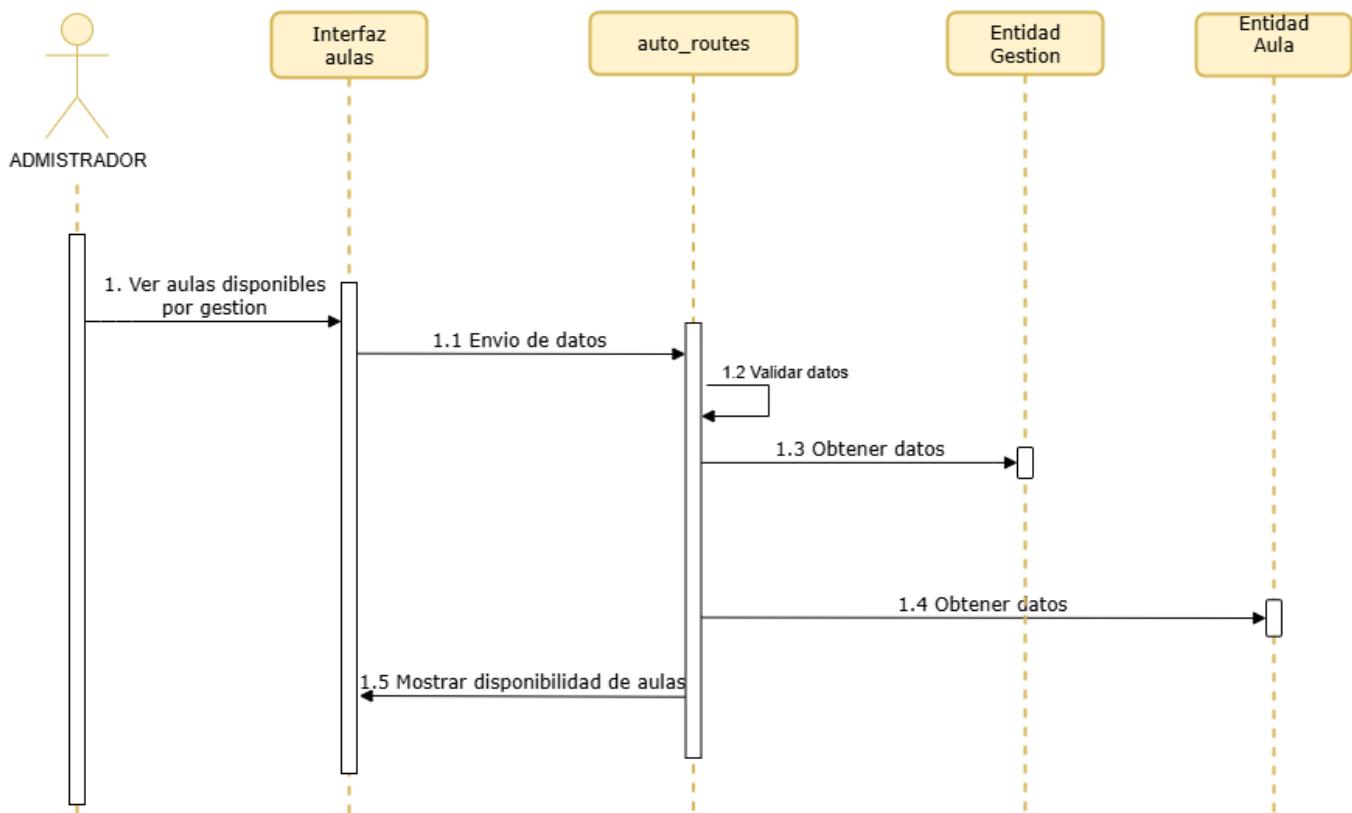
CU11: EXPORTAR REPORTE A PDF/EXCEL



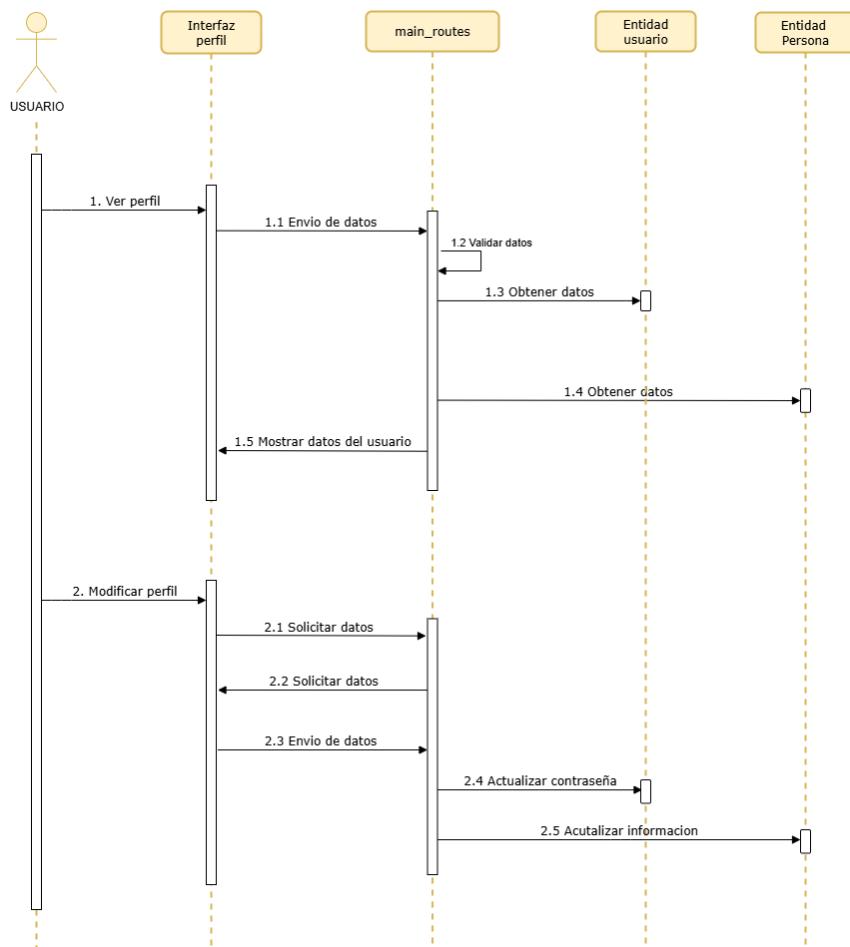
CU24: SOLICITAR LICENCIA



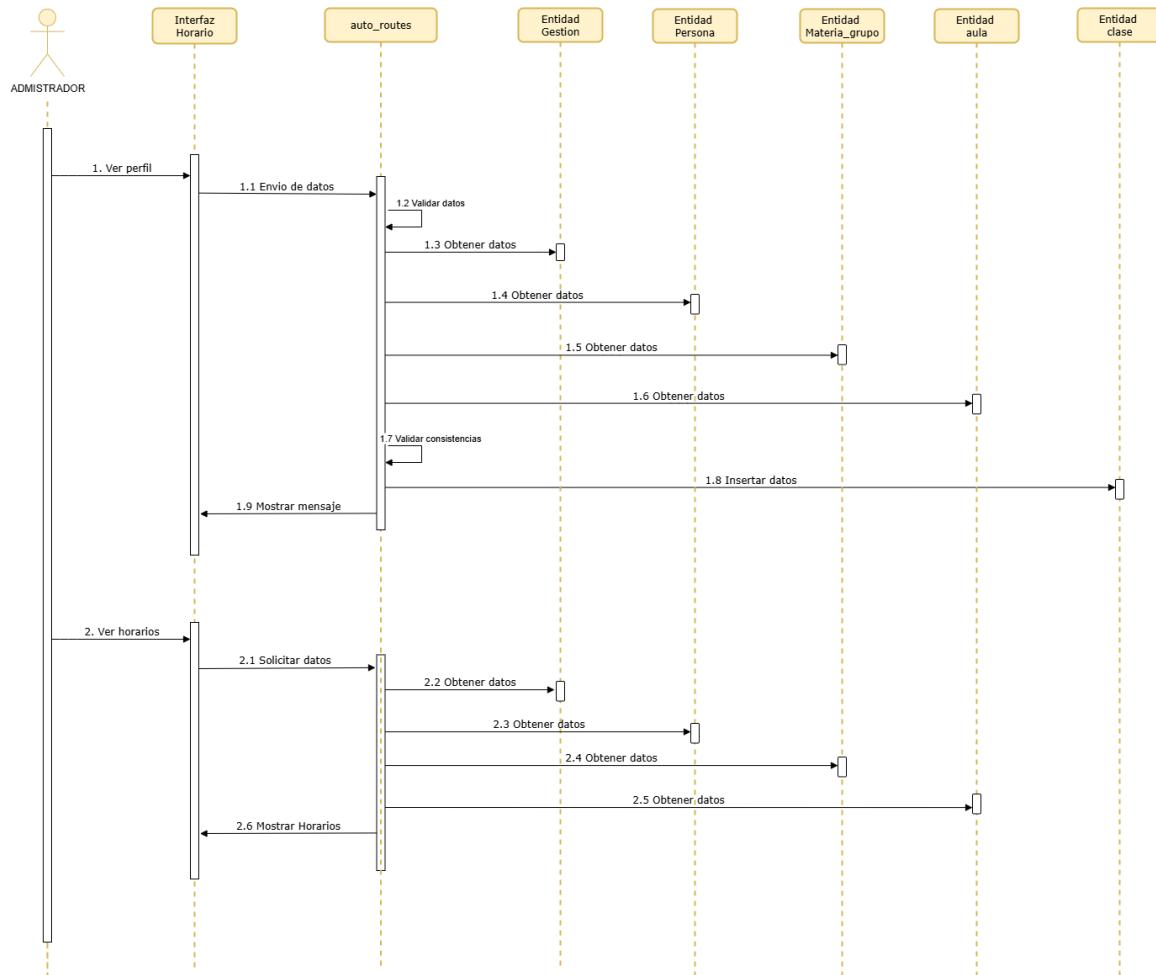
CU10: CONSULTAR AULAS DISPONIBLES



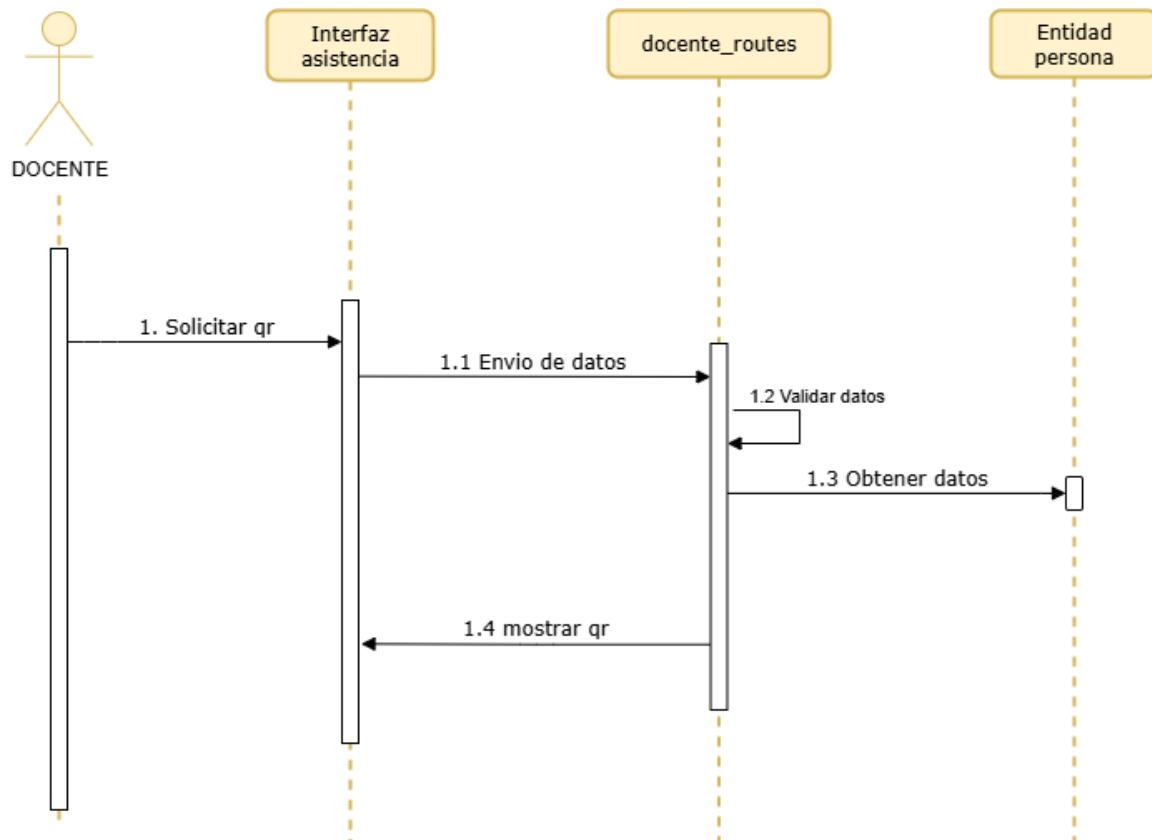
CU16: GESTIONAR PERFIL



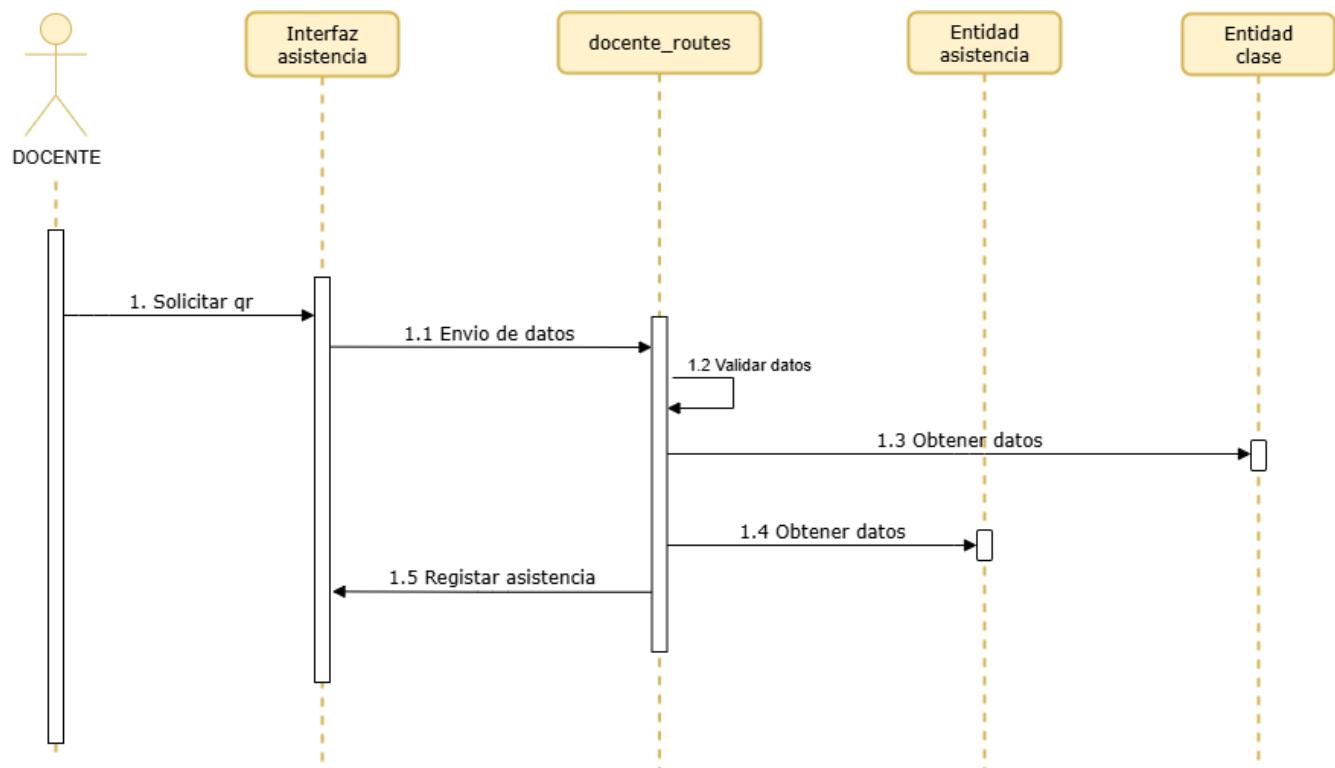
CU7: GENERAR HORARIO



CU22: GENERAR QR



CU23: REGISTRAR ASISTENCIA POR QR



7. FLUJO DE TRABAJO: IMPLEMENTACIÓN

7.1) Herramientas de desarrollo de la aplicación WEB

La selección de herramientas y tecnologías para el presente proyecto se diseñó para construir una aplicación web robusta, mantenible y eficiente, dentro de un enfoque de arquitectura monolítica. Siguiendo los lineamientos de la cátedra, se estableció un stack de desarrollo específico basado en PHP.

7.1.1 Lenguaje de Programación y Framework Principal

El núcleo de la aplicación se desarrolla utilizando **PHP**, un lenguaje de scripting del lado del servidor ampliamente adoptado y con un vasto ecosistema. Para estructurar y agilizar el desarrollo, se seleccionó **Laravel**, uno de los frameworks de PHP más modernos y potentes. Laravel proporciona una arquitectura subyacente limpia (basada en el patrón Modelo-Vista-Controlador) y un conjunto de herramientas integradas que facilitan tareas comunes como el enrutamiento, la autenticación, la gestión de sesiones y la interacción con la base de datos.

A diferencia de un stack con un frontend desacoplado (como React o Vue), este proyecto opera como un **monolito tradicional**:

- **Backend y Frontend Unificados:** Laravel maneja tanto la lógica de negocio (consultas a la base de datos, autenticación, servicios) como el renderizado de la interfaz de usuario.
- **Motor de Plantillas:** La interfaz de usuario se construye utilizando el motor de plantillas **Blade** de Laravel. Esto permite combinar HTML estándar con directivas de PHP para mostrar datos dinámicos, crear componentes reutilizables (layouts, partials) e integrar lógica de control directamente en las vistas (archivos .blade.php).
- **Estilos y JavaScript:** El diseño visual se implementa con **TailwindCSS**, un framework de CSS "utility-first" que permite crear diseños modernos y responsivos rápidamente sin salir del HTML. Se complementa con archivos CSS personalizados para estilos globales. El dinamismo del lado del cliente (como validaciones de formularios o pequeñas interacciones) se maneja con **JavaScript puro** embebido directamente en los archivos Blade, evitando la complejidad de un framework de JavaScript pesado.

7.1.2 Base de Datos

La elección para el sistema de gestión de base de datos fue **PostgreSQL**. Se seleccionó por su robustez, fiabilidad, cumplimiento de estándares SQL y sus capacidades avanzadas para manejar consultas complejas e integridad de datos, siendo una opción de nivel empresarial superior a otras alternativas como MySQL en ciertos aspectos de escalabilidad.

La comunicación con la base de datos desde la aplicación se gestiona a través de **Eloquent**, el ORM (Object-Relational Mapping) integrado en Laravel. Eloquent simplifica drásticamente las operaciones CRUD (Crear, Leer, Actualizar, Borrar) al permitir a los desarrolladores interactuar con las tablas de la base de datos como si fueran objetos de PHP, abstrayendo la necesidad de escribir consultas SQL manualmente y mejorando la seguridad y mantenibilidad del código.

7.1.3 Plataforma de Despliegue y Sistema Operativo

Tanto la aplicación web monolítica como la base de datos PostgreSQL se encuentran desplegadas en **Render**, una plataforma de nube unificada (PaaS - Platform as a Service). Render simplifica el proceso de despliegue al conectarse directamente con el repositorio de GitHub y gestionar automáticamente la compilación, el despliegue y el aprovisionamiento de la infraestructura necesaria (servidores web, balanceadores de carga, y la instancia de la base de datos).

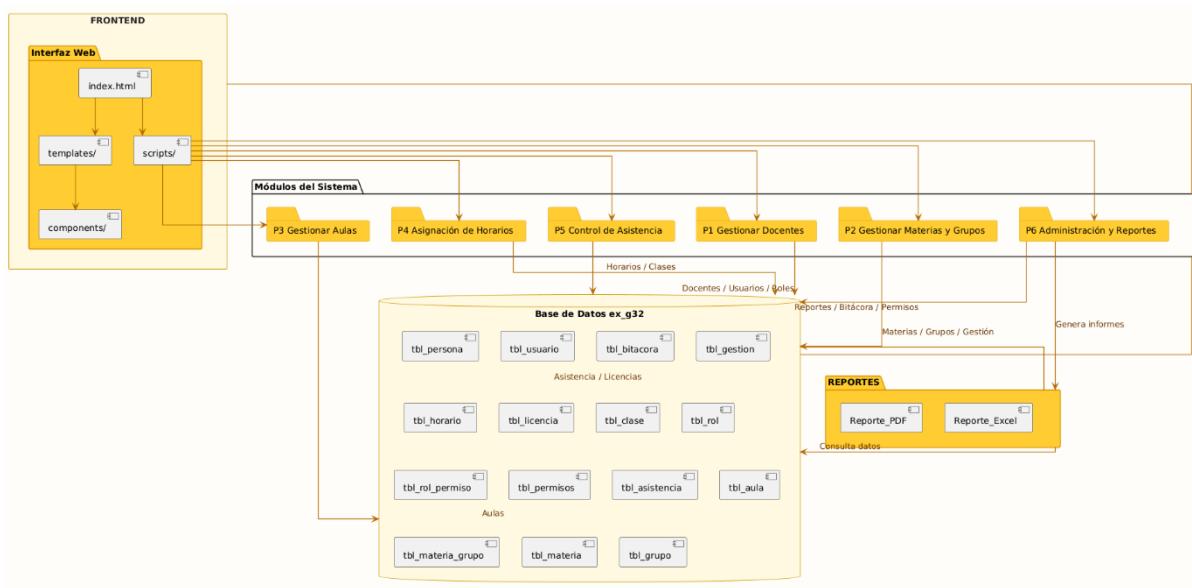
Al utilizar un PaaS, la aplicación se abstrae del sistema operativo subyacente, ya que Render gestiona el entorno de ejecución (generalmente contenedores basados en Linux). Esto permite al equipo de desarrollo centrarse en el código y no en la administración de servidores.

La aplicación es accesible desde cualquier **sistema operativo** que soporte un navegador web moderno, incluyendo Windows, macOS, Linux, Android e iOS, garantizando una experiencia de usuario consistente en múltiples dispositivos.

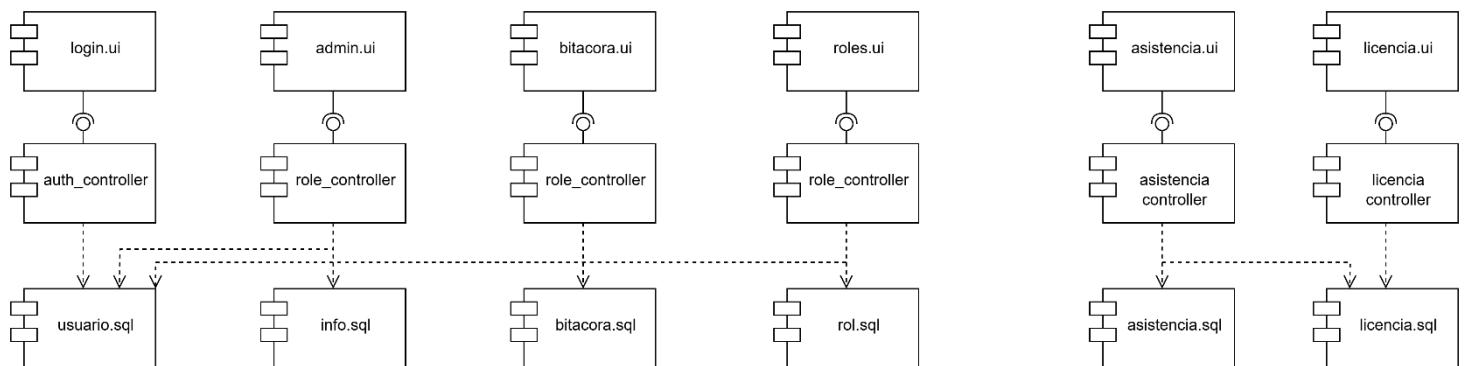
7.1.4 Otras Herramientas y Control de Versiones

- **Gestión de Dependencias (PHP):** Se utiliza **Composer** como el gestor de paquetes estándar para PHP. Composer maneja la instalación y actualización de todas las librerías del framework de Laravel y otras dependencias del proyecto, asegurando un entorno de desarrollo consistente.
- **Control de Versiones:** El proyecto utiliza **Git** para el control de versiones. El código fuente se aloja en **GitHub**, que actúa como el repositorio centralizado, facilitando la colaboración, el seguimiento de cambios y la integración con Render para los despliegues automáticos.
- **Herramientas de CLI:** Se hace uso extensivo de **Artisan**, la interfaz de línea de comandos (CLI) de Laravel. Artisan se utiliza para tareas esenciales como la creación de modelos, controladores, migraciones de base de datos, y la ejecución de tareas personalizadas.
- **Validación y Seguridad:** Se utilizan los componentes integrados de Laravel para la **validación de datos** entrantes (reemplazando la necesidad de herramientas externas como Zod en otros stacks) y para la gestión de la **autenticación y autorización** de usuarios, asegurando que solo los usuarios con permisos adecuados puedan acceder a las distintas secciones del sistema.

7.2 IMPLEMENTACIÓN DE LA ARQUITECTURA DEL SISTEMA



7.3 IMPLEMENTACIÓN DE LA ARQUITECTURA DEL SUBSISTEMAS



CONCLUSIÓN

El desarrollo del Sistema Web para la Asignación de Horarios, Aulas, Grupos y Asistencia Docente representa una solución tecnológica orientada a optimizar la planificación académica y administrativa dentro de la Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones (FICCT).

A través del análisis realizado, se identificaron los principales problemas que afectan la gestión actual —como los conflictos de horarios, la sobreasignación de docentes y la falta de trazabilidad en la asistencia—, proponiendo una plataforma integral que automatiza procesos y mejora la transparencia institucional.

La arquitectura propuesta garantiza una administración centralizada y segura de la información, con un control de acceso basado en roles, lo cual facilita la gestión diferenciada entre administradores, coordinadores y docentes.

Además, la incorporación de herramientas como la asignación automática de horarios, el registro digital de asistencia mediante códigos QR, y la generación de reportes estadísticos, asegura una toma de decisiones más ágil, precisa y respaldada por datos confiables.

Finalmente, este proyecto sienta las bases para futuras ampliaciones, como la integración del control de evaluaciones o la gestión de estudiantes, consolidando una infraestructura tecnológica escalable y adaptable a las necesidades académicas de la universidad.

8. RECOMENDACIÓN

Implementación gradual:

Se recomienda desplegar el sistema de forma progresiva, iniciando con los módulos de *Gestión de Docentes, Aulas y Asignación de Horarios*, para asegurar una adopción ordenada y corregir posibles incidencias antes de habilitar todas las funciones.

Capacitación de usuarios:

Capacitar al personal administrativo y docente en el uso de la plataforma, especialmente en los módulos de asignación y control de asistencia, garantizando un uso correcto y eficiente del sistema.

Mantenimiento y mejora continua:

Establecer un plan de mantenimiento técnico y de actualización periódica, incorporando nuevas funcionalidades (por ejemplo, gestión de estudiantes o comunicación directa docente-administración) según las necesidades institucionales.

Seguridad y respaldo de datos:

Implementar copias de seguridad automáticas y controles de auditoría para resguardar la integridad de la información, evitando pérdidas de datos y garantizando trazabilidad de todas las operaciones.

Evaluación de desempeño:

Realizar evaluaciones periódicas del rendimiento del sistema y la satisfacción de los usuarios, con el fin de optimizar la usabilidad, tiempos de respuesta y estabilidad general de la aplicación.

9. BIBLIOGRAFÍA

El Proceso Unificado de Desarrollo de Software

Autor :*Jacobson I., Booch G., Rumbaugh J. (1999). -Introducción a la Teoría General de Sistemas*

Autor: *Oscar Johansen Bertoglio, Universidad de Chile*

Manual de Uso de Repositorios en Github para repositorios publicos o privados

<https://github.com/features#documentation>

ENLACES DEL PROYECTO:

Enlace del Proyecto en GitHub

https://github.com/MigueAuad2002/exa2_inf342

Código QR del Proyecto en GitHub



Enlace a la Aplicación Web – Software Web

Realizamos el despliegue de la aplicación web con el servicio de despliegue gratuito Render, dado a que el plan es gratuito si la aplicación permanece inactiva durante 15 minutos entra en estado de suspensión, puede acceder a ella presionando sobre el siguiente enlace:

<https://exa2-inf342.onrender.com/>

Código QR de la Aplicación Web – Software Web

