Informe Final
G07
GeoAda

Laboratorio de Ingeniería del Software

Grado de Ingeniería Informática

Universidad de Zaragoza

Curso 2025/26

28/05/2026

Porroche Llorén, Ariana - 874055

Vayad Díaz, Iván - 874762

ÍNDICE

1. INTRODUCCIÓN

El proyecto GeoAda consiste en el desarrollo de un sistema de gestión y reserva de espacios para el edificio Ada Byron de la Universidad de Zaragoza. El sistema permite a los usuarios consultar, filtrar y reservar los distintos espacios disponibles en el edificio, así como gestionar su perfil y visualizar un mapa interactivo de las instalaciones.

El equipo de desarrollo, perteneciente al Grupo G07, está compuesto por Ariana Porroche Llorén e Iván Vayad Díaz.

El código fuente y el progreso del proyecto se encuentran alojados en la organización de GitHub bajo las siguientes URLs:

- Organización: https://github.com/UNIZAR-30249-2026-G07
- Backend: https://github.com/UNIZAR-30249-2026-G07/backend
- Frontend: https://github.com/UNIZAR-30249-2026-G07/frontend

Este documento recoge los resultados obtenidos a lo largo del proyecto, estructurándose en tres bloques principales. En primer lugar, se definen los requisitos funcionales y no funcionales del sistema, junto con el diccionario de datos. En segundo lugar, se describe el diseño del modelo de dominio, la arquitectura del sistema y el estado actual de la implementación. Finalmente, se presentan las conclusiones del trabajo realizado, incluyendo los bocetos de la interfaz de usuario en el anexo.

2. REQUISITOS

2.1. STAKEHOLDERS

- **Usuario:** usuario del sistema que se identifica mediante login y que se corresponde con una persona (DD2) previamente registrada.

2.2. DICCIONARIO DE DATOS

- **(DD1) Mapa:** mapa urbano interactivo de los espacios (DD5) del edificio (DD12) Ada Byron por plantas.
- **(DD2) Persona:** nombre, e-mail, roles (DD3), departamento (DD4).
- **(DD3) Rol:** “estudiante”, “investigador contratado”, “docente-investigador”, “conserje”, “técnico de laboratorio”, “gerente”.
- **(DD4) Departamento:** “informática e ingeniería de sistemas”, “ingeniería electrónica y comunicaciones”.
- **(DD5) Espacio:** tamaño, identificador, tipo (DD6), número máximo de ocupantes, reservable, categoría de reserva (DD7), hora de apertura, hora de cierre, asignado a, porcentaje de uso máximo, planta, edificio (DD12).
- **(DD6) Tipo:** “aula”, “seminario”, “laboratorio”, “despacho”, “sala común”.
- **(DD7) Categoría de reserva:** “aula”, “seminario”, “laboratorio”, “despacho”, “sala común”.
- **(DD8) Reserva:** espacios (DD5) reservados, tipo de uso (DD9), número máximo de personas asistentes, fecha, hora de inicio, hora de fin, explicación, validez (DD11).
- **(DD9) Tipo de uso:** “docencia”, “investigación”, “gestión”, “otros”.
- **(DD10) Reserva viva:** reserva (DD8) cuya hora y fecha de inicio son posteriores al momento actual.
- **(DD11) Validez:** “válida”, “inválida”, “potencialmente inválida”, fecha de actualización.
- **(DD12) Edificio:** identificador, calendario de apertura, horario de apertura (hora de apertura, hora de cierre), porcentaje de ocupación máxima.

2.3. REQUISITOS FUNCIONALES

RF_U1	El usuario deberá ser capaz de iniciar sesión. **Criterios de aceptación:** ● El inicio de sesión se realizará mediante e-mail y contraseña. **GUI:** [Pantalla Iniciar sesión]
RF_U2_A	El usuario deberá ser capaz de visualizar un mapa (DD1). **Criterios de aceptación:** ● El mapa (DD1) deberá representar los espacios (DD5) del tipo (DD6) “aula”, “seminario”, “laboratorio”, “despacho” y “sala común” con un color diferente. [Pantalla Mapa del edificio]

<table>
<tr><td>RF_U3_B1</td><td>El usuario deberá ser capaz de modificar su rol (DD3).

Criterios de aceptación:
Una persona (DD2) deberá tener un único rol (DD3).
Como excepción, una persona (DD2) podrá tener los dos roles (DD3) de “gerente” y “docente-investigador”.
La modificación de un rol (DD3) deberá realizarse mediante la API.
El rol (DD3) actual de la persona (DD2) deberá tener permitido cambiar al nuevo rol (DD3) según esta tabla.</td></tr>
<tr><td>RF_U4</td><td>El usuario deberá ser capaz de consultar su rol (DD3).

GUI:
[Pantalla Perfil del usuario]</td></tr>
<tr><td>RF_U5_B3</td><td>El usuario deberá ser capaz de adscribirse a un departamento (DD4).

Criterios de aceptación:
La adscripción a un departamento (DD4) deberá realizarse mediante la API.
El número de departamentos (DD4) a los que podrá estar adscrita una persona (DD2) según su rol (DD3) será:
<table><tr><td>ROL</td><td>Nº DEPARTAMENTOS ADSCRITOS</td></tr><tr><td>estudiante</td><td>0</td></tr><tr><td>investigador contratado</td><td>0-1</td></tr><tr><td>docente-investigador</td><td>0-1</td></tr><tr><td>conserje</td><td>0</td></tr><tr><td>técnico de laboratorio</td><td>0-1</td></tr><tr><td>gerente</td><td>0</td></tr></table></td></tr>
<tr><td>RF_U6_B3</td><td>El usuario deberá ser capaz de eliminar su adscripción a un departamento (DD4).

Criterios de aceptación:
La eliminación de la adscripción a un departamento (DD4) deberá realizarse mediante la API.
El sistema comprobará automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho usuario, modificando la validez (DD11) a “potencialmente inválida” en aquellas que no cumplan al menos una de las reglas.</td></tr>
<tr><td>RF_U7</td><td>El usuario deberá ser capaz de consultar el departamento (DD4) al que está adscrito.

GUI:
[Pantalla Perfil del usuario]</td></tr>
</table>

<table>
<tr><td>RF_U8</td><td>El usuario deberá ser capaz de crear una persona (DD2).
Criterios de aceptación:
La creación de una persona (DD2) deberá realizarse mediante la API.</td></tr>
<tr><td>RF_U9</td><td>El usuario deberá ser capaz de consultar los espacios (DD5) del edificio (DD12).

GUI:
[Pantalla Listado de espacios con filtros]</td></tr>
<tr><td>RF_U10_C1</td><td>El usuario deberá ser capaz de modificar si un espacio (DD5) es reservable.

Criterios de aceptación:
La modificación de un espacio (DD5) podrá realizarse únicamente por personas (DD2) con el rol (DD3) “gerente”.
El sistema deberá comprobar automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho espacio (DD5), modificando la validez (DD11) a “potencialmente inválida” en aquellas que no cumplan al menos una de las reglas.
GUI:
[Pantalla Modificar espacio]</td></tr>
<tr><td>RF_U11_C2</td><td>El usuario deberá ser capaz de modificar la categoría de reserva (DD7) de un espacio (DD5).

Criterios de aceptación:
La modificación de un espacio (DD5) podrá realizarse únicamente por personas (DD2) con el rol (DD3) “gerente”.
La modificación de la categoría de reserva (DD7) de un espacio (DD5) podrá realizarse únicamente si este es reservable.
La modificación de la categoría de reserva (DD7) de un espacio (DD5) deberá estar permitida en función del tipo (DD6) del espacio (DD5) según esta tabla:
<table>
<tr><th>TIPO / CATEGORÍA DE RESERVA</th><th>aula</th><th>seminario</th><th>laboratorio</th><th>despacho</th><th>sala común</th></tr>
<tr><td>aula</td><td>SI</td><td>SI</td><td>SI</td><td>-</td><td>-</td></tr>
<tr><td>seminario</td><td>SI</td><td>SI</td><td>SI</td><td>-</td><td>-</td></tr>
<tr><td>laboratorio</td><td>SI</td><td>SI</td><td>SI</td><td>-</td><td>-</td></tr>
<tr><td>despacho</td><td>-</td><td>-</td><td>-</td><td>SI</td><td>-</td></tr>
<tr><td>sala común</td><td>-</td><td>-</td><td>-</td><td>-</td><td>SI</td></tr>
</table>
El sistema deberá comprobar automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho espacio (DD5), modificando la validez (DD11) a “potencialmente inválida” en aquellas que no cumplan al menos una de las reglas.
GUI:
[Pantalla Modificar espacio]</td></tr>
</table>

RF_U12_C3	El usuario deberá ser capaz de modificar los horarios disponibles de un espacio (DD5). **Criterios de aceptación:** ● La modificación de un espacio (DD5) podrá realizarse únicamente por personas (DD2) con el rol (DD3) “gerente”. ● Los horarios reservables de un espacio (DD5) deberán ser por defecto los del edificio (DD12) en el que se encuentra. ● El sistema deberá comprobar automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho espacio (DD5), modificando la validez (DD11) a “potencialmente inválida” en aquellas que no cumplan al menos una de las reglas. **GUI:** [Pantalla Modificar espacio]
RF_U13_C4	El usuario deberá ser capaz de modificar la asignación de un espacio (DD5). **Criterios de aceptación:** ● La modificación de un espacio (DD5) podrá realizarse únicamente por personas (DD2) con el rol (DD3) “gerente”. ● El sistema deberá comprobar automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho espacio (DD5), modificando la validez (DD11) a “potencialmente inválida” en aquellas que no cumplan al menos una de las reglas. **GUI:** [Pantalla Modificar espacio]
RF_U14_C5_O1	El usuario deberá ser capaz de modificar el porcentaje de uso máximo de un espacio (DD5). **Criterios de aceptación:** ● La modificación de un espacio (DD5) podrá realizarse únicamente por personas (DD2) con el rol (DD3) “gerente”. ● El porcentaje de uso máximo de un espacio (DD5) deberá ser por defecto el porcentaje de uso máximo del edificio (DD12) en el que se encuentre. ● El sistema deberá comprobar automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho espacio (DD5), modificando la validez (DD11) a “potencialmente inválida” en aquellas que no cumplan al menos una de las reglas. **GUI:** [Pantalla Modificar espacio]
RF_U15_D	El usuario deberá ser capaz de buscar espacios (DD5). **Criterios de aceptación:** ● La búsqueda de espacios (DD5) deberá ser por identificador, categoría reserva (DD7), número de ocupantes máximo o planta. **GUI:** [Pantalla Listado de espacios con filtros]
RF_U16_E_F_O3	El usuario deberá ser capaz de hacer una reserva (DD8).

<table>
<tr><td></td><td>
Criterios de aceptación:

La hora de inicio de la reserva (DD8) deberá ser anterior a la hora de fin.
El número máximo de personas asistentes de una reserva (DD8) deberá ser inferior o igual al producto del número máximo de ocupantes por el porcentaje de uso máximo de cada espacio (DD5) a reservar.
Una reserva (DD8) de un espacio (DD5) no podrá solaparse total o parcialmente el tiempo de reserva con el de otra reserva (DD8) del mismo espacio (DD5).
El rol (DD3) de la persona (DD2) deberá tener permitido reservar los espacios (DD5) dada su categoría de reserva (DD7) según la siguiente tabla:

<table>
<tr><th>ROL / CATEGORÍA DE RESERVA</th><th>aula</th><th>seminario</th><th>laboratorio</th><th>despacho</th><th>sala común</th></tr>
<tr><td>estudiante</td><td>-</td><td>-</td><td>-</td><td>-</td><td>SI</td></tr>
<tr><td>investigador contratado</td><td>SI</td><td>SI</td><td>SI si pertenecen al mismo departamento</td><td>SI si pertenecen al mismo departamento</td><td>SI</td></tr>
<tr><td>docente-investigador</td><td>SI</td><td>SI</td><td>SI si pertenecen al mismo departamento</td><td>SI si pertenecen al mismo departamento</td><td>SI</td></tr>
<tr><td>conserje</td><td>SI</td><td>SI</td><td>SI</td><td>-</td><td>SI</td></tr>
<tr><td>técnico de laboratorio</td><td>-</td><td>SI</td><td>SI si pertenecen al mismo departamento</td><td>-</td><td>SI</td></tr>
<tr><td>gerente</td><td>SI</td><td>SI</td><td>SI</td><td>SI</td><td>SI</td></tr>
</table>

El sistema deberá comprobar automáticamente la validez (DD11) de la reserva (DD8).
Cuando una reserva (DD8) es válida, el sistema deberá registrar automáticamente la reserva (DD8).
Cuando una reserva (DD8) no es válida, el sistema deberá informar automáticamente al usuario de las condiciones incumplidas.

GUI:
[Pantalla Hacer una reserva (Paso 1), Pantalla Hacer una reserva (Paso 2)]
</td></tr>
<tr><td>RF_U17_H</td><td>
El usuario deberá ser capaz de consultar las reservas vivas (DD10).

Criterios de aceptación:

Las reservas vivas (DD10) podrán ser consultadas únicamente por las personas (DD2) con rol (DD3) “gerente”.

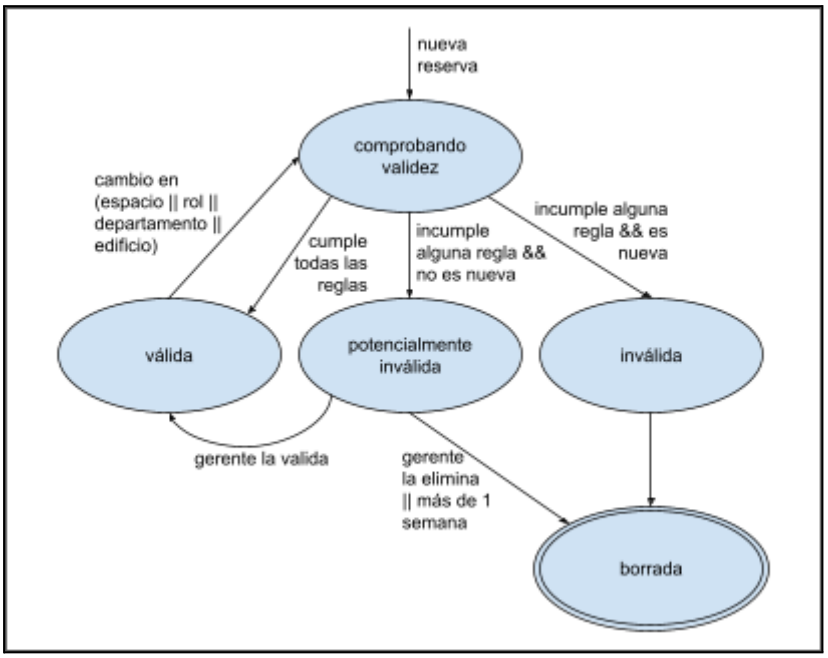
GUI:
</td></tr>
</table>

	[Pantalla Listado de reservas]
RF_U18_H	El usuario deberá ser capaz de eliminar una reserva (DD8). **Criterios de aceptación:** ● Las reservas (DD8) podrán ser eliminadas únicamente por las personas (DD2) con rol (DD3) "gerente". ● El sistema deberá informar automáticamente al usuario de que su reserva (DD8) ha sido eliminada. **GUI:** [Pantalla Listado de reservas]
RF_U19_H_O4	El usuario deberá ser capaz de modificar la validez (DD11) de una reserva (DD8) de "potencialmente inválida" a "válida". **Criterios de aceptación:** ● Las reservas (DD8) con validez (DD11) "potencialmente inválida" podrán ser modificadas únicamente por las personas (DD2) con rol (DD3) "gerente". ● Si el espacio (DD5) no es reservable, no se podrá modificar la validez (DD11) de la reserva (DD8), sólo se podrá eliminar la reserva (DD8). **GUI:** [Pantalla Listado de reservas]
RF_U20	El usuario deberá ser capaz de modificar el calendario de apertura del edificio (DD12). **Criterios de aceptación:** ● La modificación del calendario de apertura del edificio (DD12) deberá realizarse mediante la API. ● El sistema deberá comprobar automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho edificio (DD12), modificando la validez (DD11) a "potencialmente inválida" en aquellas que no cumplan al menos una de las reglas.
RF_U21	El usuario deberá ser capaz de modificar el horario de apertura del edificio (DD12). **Criterios de aceptación:** ● La modificación del horario de apertura del edificio (DD12) deberá realizarse mediante la API. ● El sistema deberá comprobar automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho edificio (DD12), modificando la validez (DD11) a "potencialmente inválida" en aquellas que no cumplan al menos una de las reglas.
RF_U22	El usuario deberá ser capaz de modificar el porcentaje de ocupación máxima del edificio (DD12). **Criterios de aceptación:** ● La modificación del porcentaje de ocupación máxima del edificio (DD12) deberá realizarse mediante la API. ● El sistema deberá comprobar automáticamente la validez (DD11) de todas las reservas vivas (DD10) asociadas a dicho edificio (DD12), modificando la validez (DD11) a "potencialmente inválida" en aquellas que no cumplan al menos una de las reglas.

Tabla de las modificaciones de rol (DD3) permitidas

rol actual / rol siguiente	estudiante	investigador contratado	docente-investigador	conserje	técnico de laboratorio	gerente
estudiante	SI	SI	SI	-	SI	-
investigador contratado	SI	SI	SI	-	SI	-
docente-investigador	SI	SI	SI	-	SI	-
conserje	-	-	-	SI	-	SI
técnico de laboratorio	SI	SI	SI	SI	SI	-
gerente	-	-	-	SI	-	SI

Ciclo de vida de las reservas



2.4. REQUISITOS NO FUNCIONALES

RNF_1_C4	Los espacios (DD5) del tipo (DD6) “aula” y “sala” deberán estar asignados a la “EINA”.
RNF_2_C4	Los espacios (DD5) del tipo (DD6) “despacho” deberán estar asignados a un personal (DD2) con los roles “investigador contratado” o “docente-investigador”, o a un departamento (DD4).
RNF_3_C4	Los espacios (DD5) del tipo (DD6) “seminario” y “laboratorio” deberán estar asignados a un departamento (DD4) o a la “EINA”.

RNF_4_I_O5	Cuando una reserva (DD8) es “potencialmente inválida” y el tiempo transcurrido entre la fecha actual y la fecha de actualización es de más de una semana, el sistema eliminará automáticamente la reserva (DD8).
RNF_5	El mapa (DD1) deberá ser un servicio tipo WMS (Web Map Service).
RNF_6	La base de datos deberá ser PostgreSQL con PostGIS.
RNF_7	El servidor geográfico deberá utilizar PyGeoAPI.
RNF_8	El sistema deberá ser capaz de atender las peticiones de distintos usuarios concurrentes.

3. DISEÑO E IMPLEMENTACIÓN

3.1. MODELO DE DOMINIO

El siguiente diagrama de clases representa el modelo de dominio del sistema, en el que se identifican las principales entidades, objetos de valor y agregados, así como las relaciones existentes entre ellos. A través de este diagrama se refleja la estructura del dominio, y la forma en que se organizan las responsabilidades dentro del sistema.

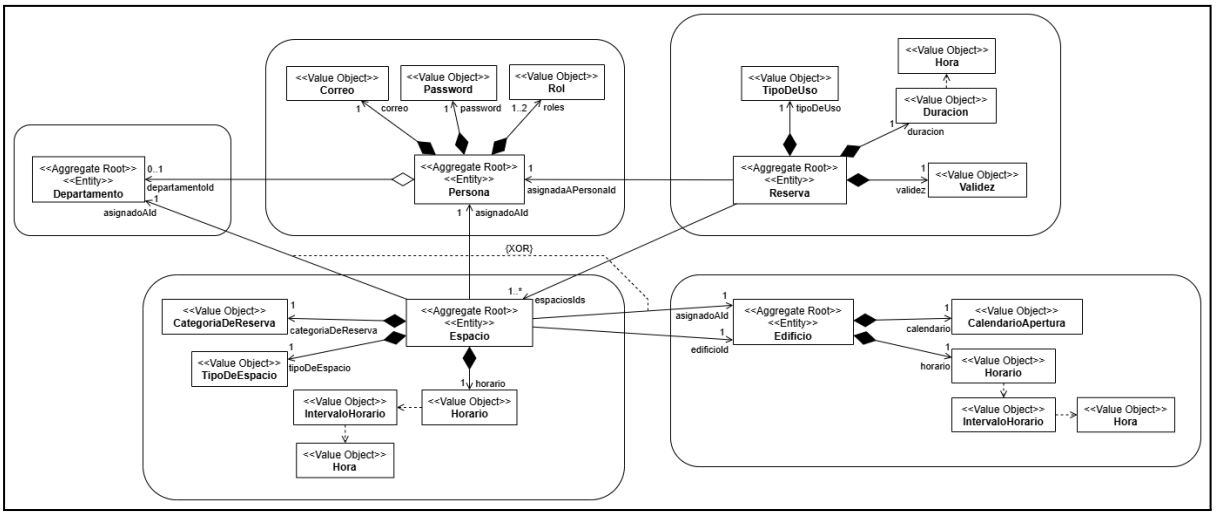


Figura 1: Diagrama de clases del dominio

El modelo de dominio definido se compone de un total de **5 entidades**: *Persona*, *Departamento*, *Edificio*, *Espacio* y *Reserva*. Cada una de estas entidades actúa como raíz de su propio agregado, encapsulando tanto su estado como la lógica asociada y garantizando el cumplimiento de las invariantes del sistema.

Asimismo, el modelo incluye **12 objetos de valor**, los cuales representan conceptos del dominio sin identidad propia y permiten expresar de forma más precisa y segura las restricciones y características de los datos manejados.

En cuanto a la lógica de negocio, se han identificado en total **11 políticas** que regulan el comportamiento del sistema: actualmente se han definido 2 políticas en la entidad Persona, 6 en Espacio y 3 en Reserva. Estas políticas reflejan reglas e invariantes del dominio que aseguran la coherencia del modelo y guían la evolución de las operaciones implementadas.

3.1.1. AGREGADO DEPARTAMENTO

El **agregado *Departamento*** está compuesto de **1 entidad** *Departamento* y **1 tipo enumerado:** *Departamentos*.

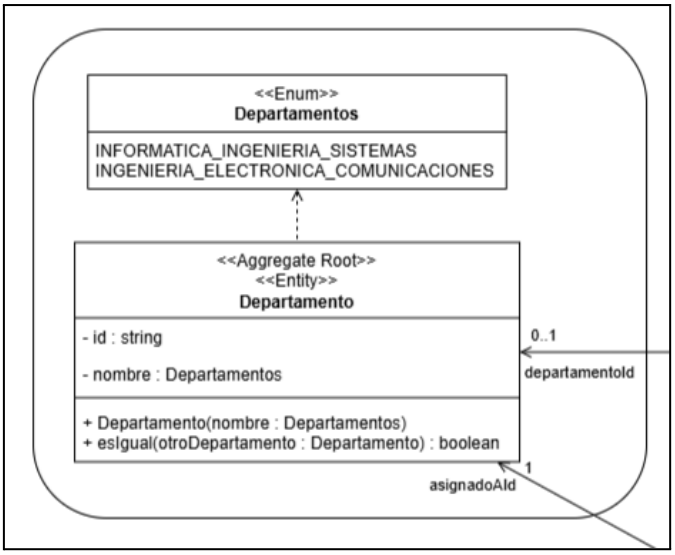


Figura 2: Diagrama de clases del agregado *Departamento*

3.1.2. AGREGADO PERSONA

El **agregado *Persona*** está compuesto de **1 entidad** *Persona* y **3 objetos valor**: *Correo*, *Password* y *Rol*. Asimismo, la entidad cuenta con un atributo *departamentoId*.

La entidad *Persona* define un **invariante** que establece que una persona puede tener un único rol, o bien dos roles únicamente en el caso de que estos sean “gerente” y “docente-investigador”. Este invariante se valida tanto en el constructor de la entidad como en aquellos métodos encargados de modificar su estado, garantizando así la consistencia del agregado en todo momento.

El objeto de valor *Rol* encapsula un atributo *rol* de tipo enumerado *Roles*, restringiendo sus posibles valores a “gerente”, “conserje”, “estudiante”, “técnico de laboratorio”, “investigador contratado” y “docente-investigador”. El atributo *departamentoId* guarda el identificador del departamento al que está adscrita la persona, si es que está adscrita a alguno.

En cuanto a la lógica de negocio, se han definido **2 políticas** dentro de este agregado. La primera, *PolíticaAdscribirDepartamento*, verifica que la persona posea alguno de los roles “investigador contratado”, “docente-investigador” o “técnico de laboratorio” para poder adscribirse a un departamento. La segunda, *PolíticaCambiarRol*, determina si una persona puede modificar sus roles actuales a unos nuevos, de acuerdo con las reglas establecidas en esta tabla.

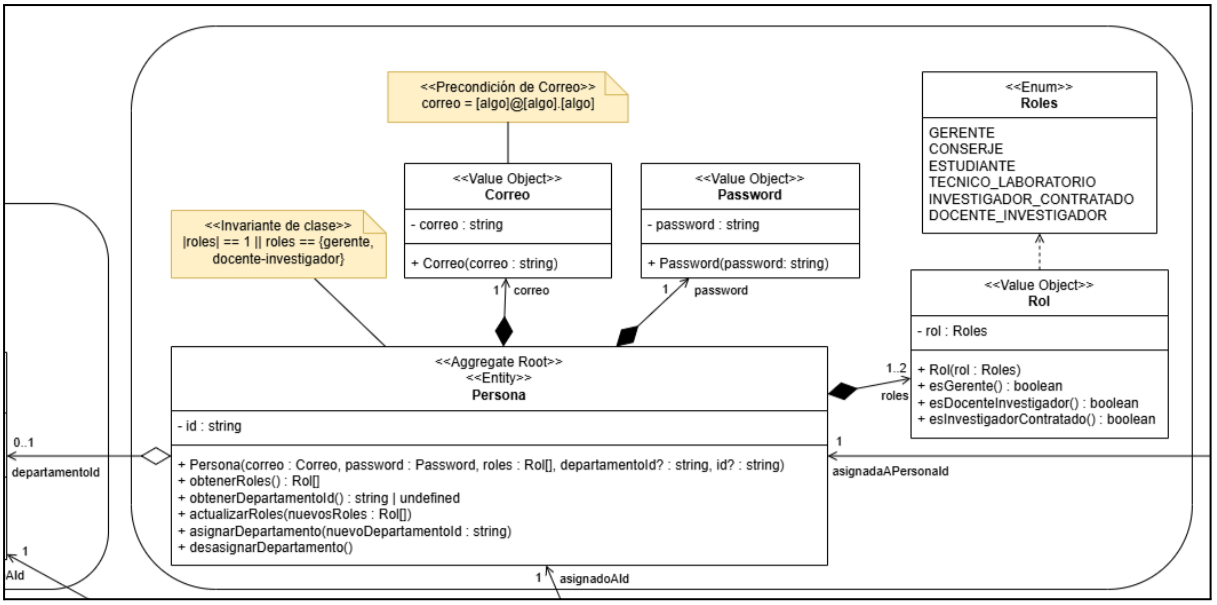


Figura 3: Diagrama de clases del agregado *Persona*

3.1.3. AGREGADO EDIFICIO

El **agregado *Edificio*** está compuesto de **1 entidad** *Edificio* y **2 objetos valor**: *CalendarioApertura* y *Horario.* Asimismo, la entidad cuenta con 3 atributos adicionales: el número de plantas, *plantas*, el nombre del edificio, *nombre*, y el porcentaje de ocupación máxima permitido, *porcentajeOcupacionMax*.

La entidad *Edificio* define un **invariante** que establece que el porcentaje de ocupación máxima debe estar entre 0 y 100. Además, en el constructor de la entidad *Edificio* se comprueba que el número de plantas sea estrictamente mayor que cero, garantizando así la validez de su estado.

El objeto de valor *CalendarioApertura* incluye un atributo *diasAbiertos*, que recoge los días de la semana en los que el edificio permanece abierto mediante el uso del tipo enumerado *DiasSemana*. Además, dispone de un atributo *festivos*, de tipo *Date*, que indica los días concretos en los que el edificio permanece cerrado por motivo de festividad.

Por su parte, el objeto de valor *Horario* contiene un único atributo, *horarioPorDia*, que establece una correspondencia entre cada día de la semana y el intervalo horario de apertura del edificio. Dicho intervalo, representado mediante el objeto *IntervaloHorario*, se compone de una hora de apertura y una hora de cierre, siendo necesario que la primera sea siempre anterior a la segunda.

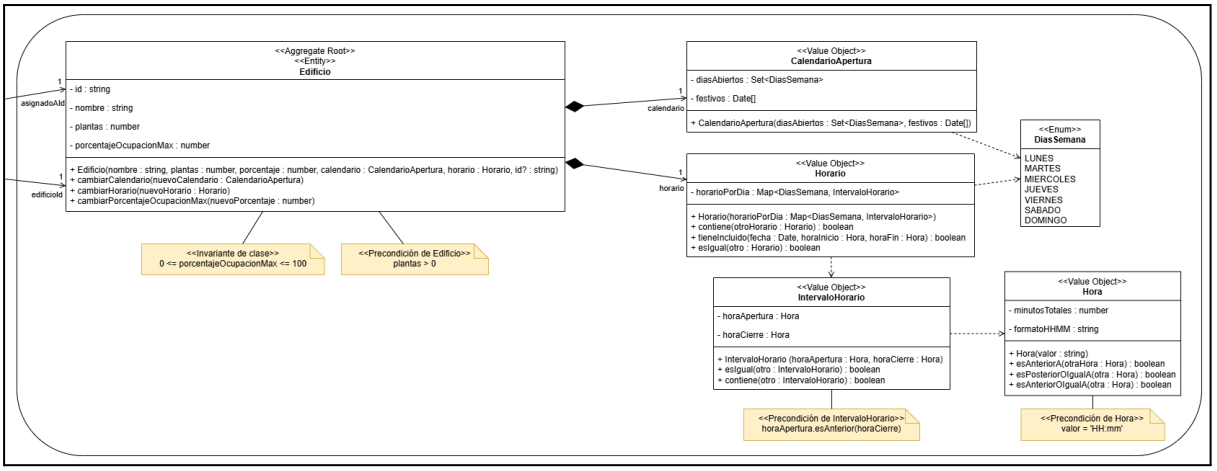


Figura 4: Diagrama de clases del agregado *Edificio*

3.1.4. AGREGADO ESPACIO

El **agregado *Espacio*** está compuesto de **1 entidad** *Espacio* y **3 objetos valor**: *CategoríaDeReserva, TipoDeEspacio* y *Horario.* Además, la entidad cuenta con 7 atributos adicionales: el nombre del espacio (*nombre*), el tamaño del espacio (*tamaño*), la planta en la que se encuentra el espacio (*planta*), si es reservable (*reservable*), el número de ocupantes máximo permitido (*numOcupantesMax*), el porcentaje de uso máximo permitido (*porcentajeUsoMax*), el id del edificio en el que se encuentra (*edificioId*), el tipo de recurso al que está asignado (*asignadoATipo*) y el id del recurso al que está asignado (*asignadoAId*).

La entidad *Espacio* define un **invariante** que establece que el porcentaje de uso máximo, *porcentajeUsoMax*, debe estar entre 0 y 100, el número de ocupantes máximo, *numOcupantesMax*, debe ser mayor que 0, y que el *tipoDeEspacio* es compatible con la *categoriaDeReserva.*

El objeto de valor *Horario* se modela de forma análoga al definido en el agregado *Edificio*, mediante una estructura que asocia cada día de la semana (*DiasSemana*) con un intervalo horario (*IntervaloHorario*).

Cabe destacar la relación de asignación existente entre la entidad *Espacio* y las demás entidades (*Persona*, *Departamento* o *Edificio*). Esta relación se modela con una restricción XOR, restringiendo la asignación a una única entidad de las mencionadas. Esto se guarda en los atributos *asignadoAId* y *asignadoATipo*.

En cuanto a la lógica de negocio, se han definido **6 políticas** dentro de este agregado. Las políticas PolíticaCambioAsignaciónEspacio, *PolíticaCambioReservabilidadEspacio*, *PolíticaCambioHorarioEspacio* y *PolíticaCambioPorcentajeUsoMaxEspacio* verifican que la persona que realiza la modificación posea el rol “gerente”. Por otra parte, la política *PolíticaCambioCategoríaDeReservaEspacio*, además de requerir dicho rol, valida que, dado el tipo de espacio, el cambio a la nueva categoría de reserva sea permitido conforme a las reglas establecidas en esta tabla. Finalmente, la política *PolíticaAsignaciónEspacio* verifica que el espacio dado su tipo, *tipoDeEspacio*, pueda ser asignado al recurso *asignadoAId* de tipo *asignadoATipo* según estos requisitos (RNF_1_C4, RNF_2_C4, RNF_3_C4).

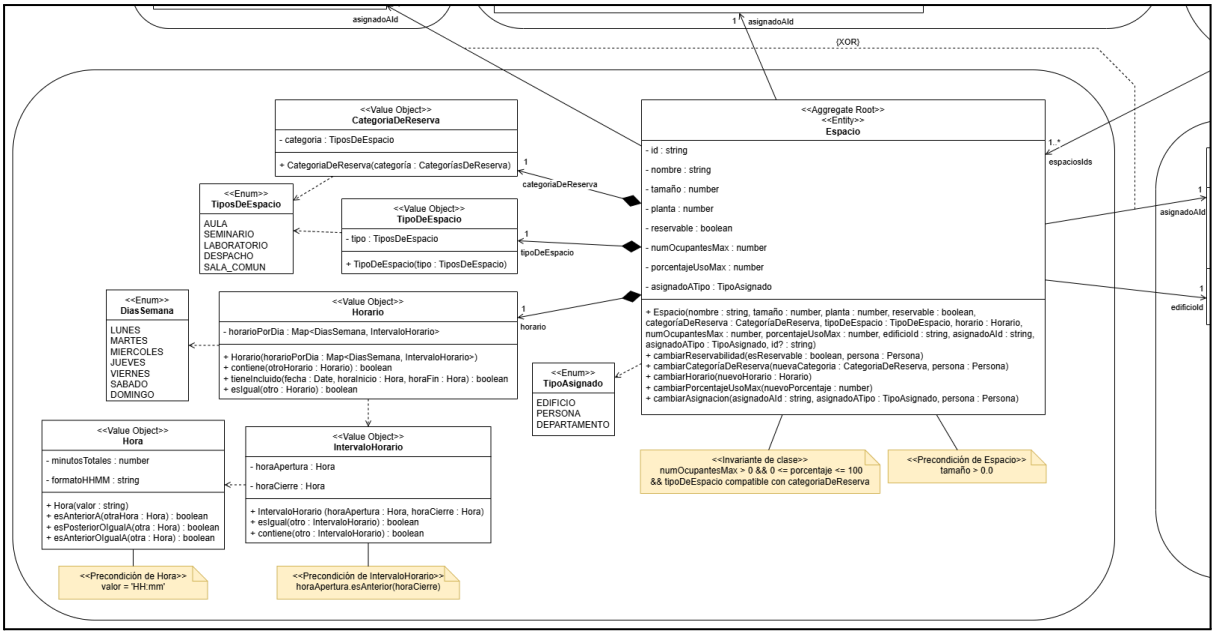


Figura 5: Diagrama de clases del agregado *Espacio*

3.1.5. AGREGADO RESERVA

El **agregado *Reserva*** está compuesto de **1 entidad** *Reserva* y **3 objetos valor**: *TipoDeUso, Duración* y *Validez.* Asimismo, la entidad dispone de 4 atributos adicionales: *numMaxAsistentes*, *explicación, fechaÚltimaActualización* y *asignadaAPersonaId.*

La entidad *Reserva* define un **invariante** que establece varias condiciones fundamentales. En primer lugar, comprueba que la duración sea válida y de mínimo 30 minutos. También comprueba que el número de asistentes sea mayor que 0, que se reserve mínimo 1 espacio y que la explicación no esté vacía.

El objeto de valor *TipoDeUso* encapsula un atributo *tipo* de tipo enumerado *TiposDeUso*, cuyos valores posibles son “docencia”, “investigación”, “gestión” y “otros”. De manera análoga, el objeto de valor *Validez* contiene un atributo *validez* de tipo enumerado *EstadosReserva*, que limita sus valores a “válida”, “inválida” y “potencialmente inválida”.

En cuanto a la lógica de negocio, se han definido **3 políticas** dentro de este agregado. Las políticas *PolíticaCambioValidezReserva* y *PolíticaEliminaciónReserva* verifican que la persona que realiza la modificación posea el rol “gerente”, requisito necesario para poder efectuar dicho cambio. Por su parte, la política *PolíticaPermisosReserva*, verifica que el rol de la persona que realiza la reserva debe estar autorizado para reservar espacios de acuerdo con su categoría de reserva, según esta tabla.

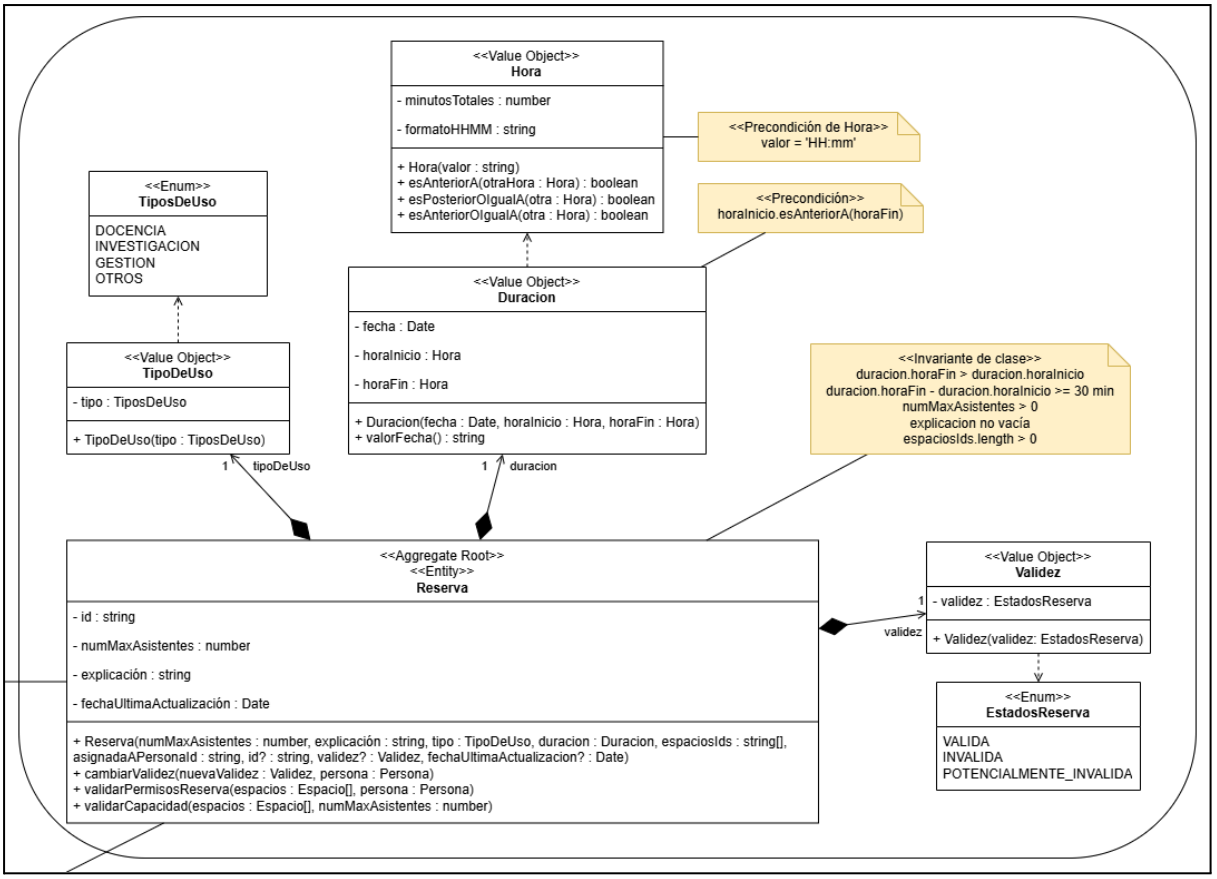


Figura 6: Diagrama de clases del agregado *Reserva*

3.1.6. PERSISTENCIA

Para dotar de persistencia a los objetos del modelo de dominio, se ha optado por el uso de un **ORM**, concretamente **Prisma**, con el objetivo de mapear cada agregado a su correspondiente representación en la **base de datos PostgreSQL**. En este enfoque, cada agregado se materializa en una tabla, manteniendo la coherencia con las reglas del dominio.

La arquitectura de persistencia se ha diseñado siguiendo el **patrón Repository** junto con el **patrón Data Mapper**. De este modo, para cada agregado se define una interfaz de repositorio en la capa de dominio, completamente independiente de la tecnología de persistencia, y una implementación concreta en la capa de infraestructura. Además, cada agregado dispone de un mapper encargado de transformar las entidades del dominio a estructuras compatibles con la base de datos y viceversa.

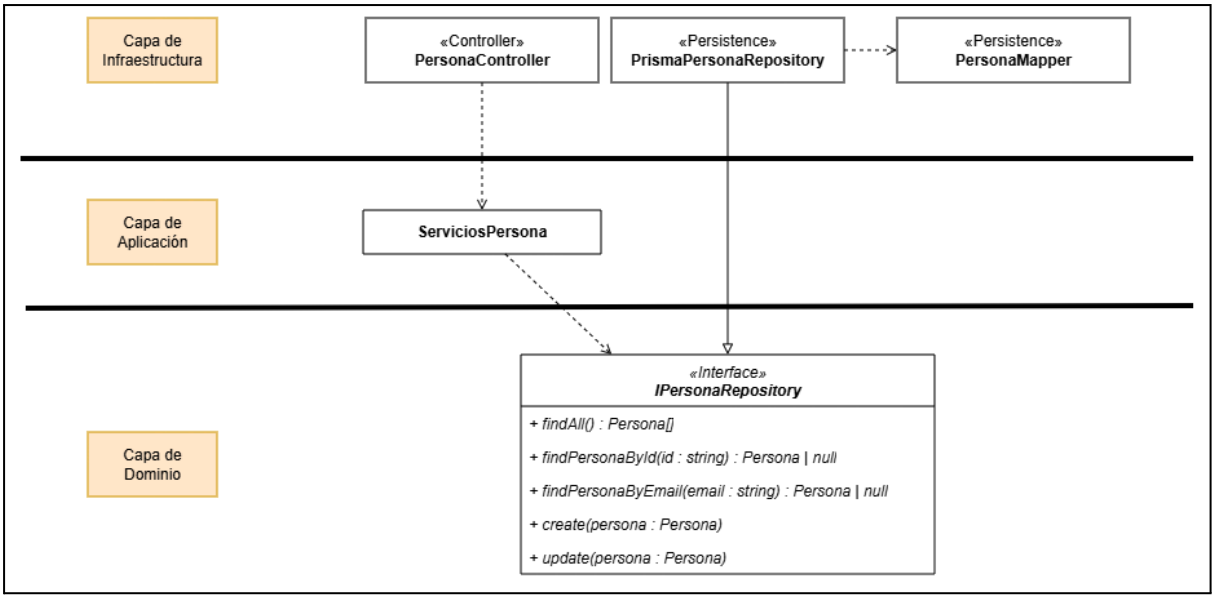


Figura 7: Diagrama de módulos de la persistencia del agregado *Persona*

En el caso del agregado *Persona*, se define la interfaz *IPersonaRepository*, la cual expone los métodos *findAll(), findPersonaById(id: string), findPersonaByEmail(email: string), create(persona: Persona)* y *update(persona: Persona).* Esta interfaz es implementada por la clase *PrismaPersonaRepository*, que utiliza Prisma ORM para la interacción con la base de datos y delega la transformación de datos al *PersonaMapper*, encargado de convertir entre el modelo de dominio y el modelo de persistencia.

Sobre esta base, se expone un controlador *PersonaController* en la capa de infraestructura, el cual define los endpoints de la API relacionados con el agregado Persona. Cada endpoint delega su lógica en la capa de aplicación, a través de servicios que encapsulan los casos de uso del sistema y coordinan las operaciones del dominio. En la capa de aplicación tenemos ***ServiciosPersona*** que expone un total de **7 servicios**: *ServicioCrearPersona*, para registrar una nueva persona en el sistema, *ServicioBuscarPersonas*, encargado de obtener todas las personas del sistema; *ServicioConsultarRolPersona*, que permite consultar el rol de una persona específica; *ServicioCambiarRolPersona*, que gestiona la modificación de roles; *ServicioConsultarDepartamentoPersona*, que obtiene el departamento asociado a una persona, si existe; *ServicioAsignarDepartamentoPersona*, que asigna una persona a un departamento; y *ServicioDesasignarDepartamentoPersona*, que elimina dicha asignación.

Para el resto de agregados se sigue la misma estructura de ficheros y comunicación entre las capas.

3.2. ARQUITECTURA

3.2.1. VISTA DE MÓDULOS

La vista de módulos del sistema se ha estructurado siguiendo un estilo arquitectural por capas, basado en el patrón de **arquitectura hexagonal**. Esta organización divide el software en **3 capas** diferenciadas, con una jerarquía unidireccional hacia el centro:

- **Infraestructura** (controladores, implementaciones de persistencia)
- **Aplicación** (servicios)
- **Dominio** (modelos, políticas, interfaces del repositorio)

Más concretamente, el módulo de Persona gestiona la entidad central del sistema y la seguridad. El controlador *PersonaController* interactúa con el núcleo a través de los *ServiciosPersona*, donde se aplican políticas. La persistencia se desacopla mediante la interfaz *IPersonaRepository*, la cual se implementa en la capa de infraestructura mediante la clase *PrismaPersonaRepository*.

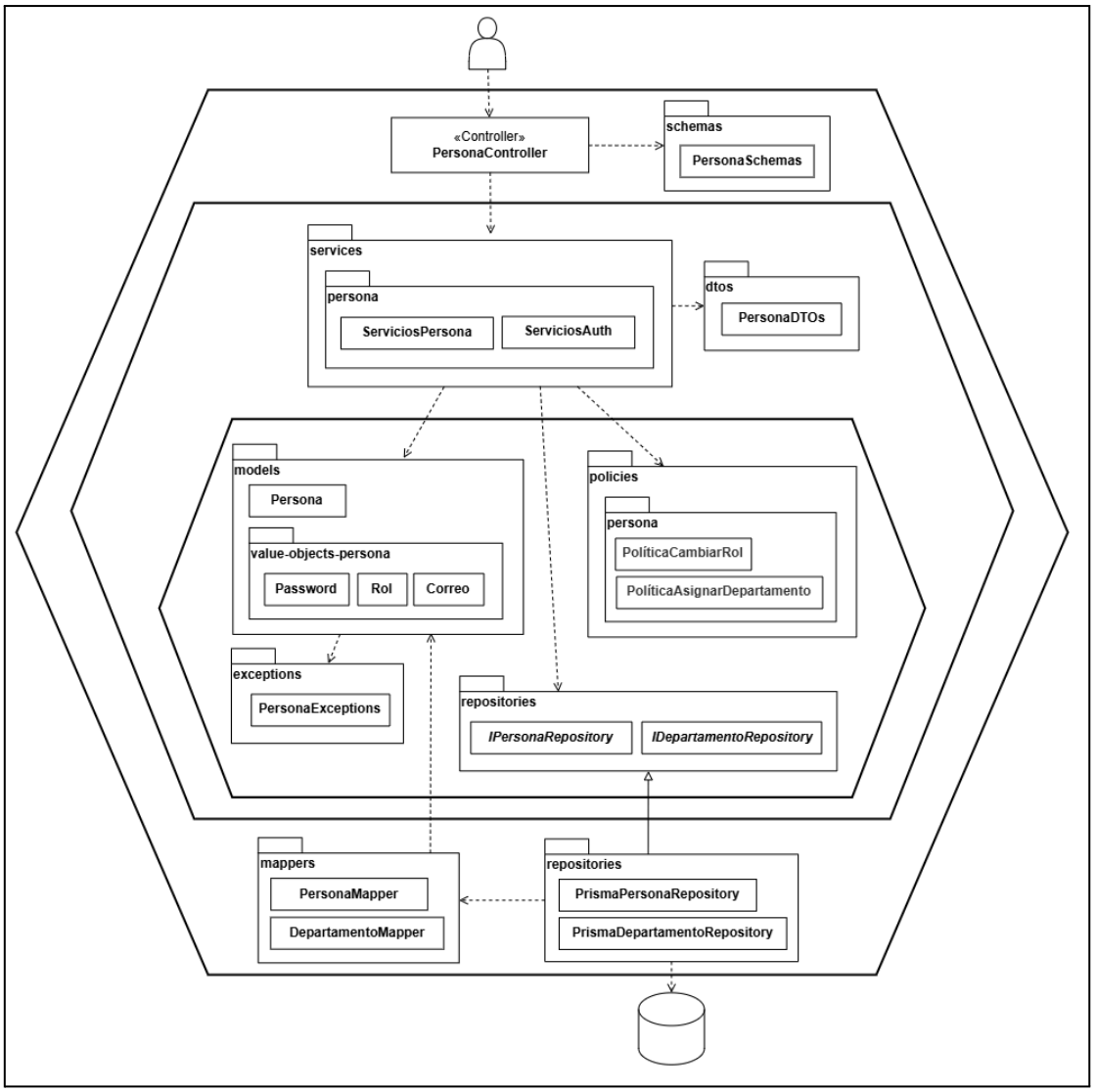


Figura 8: Vista de módulos del agregado Persona

El módulo *Edificio* se centra en la gestión de los edificios del sistema. Su lógica de negocio reside en *ServiciosEdificio*, con la cual interactúa el controlador *EdificioController*. La persistencia se desacopla mediante la interfaz *IEdificioRepository*, la cual se implementa en la capa de infraestructura mediante la clase *PrismaEdificioRepository*.

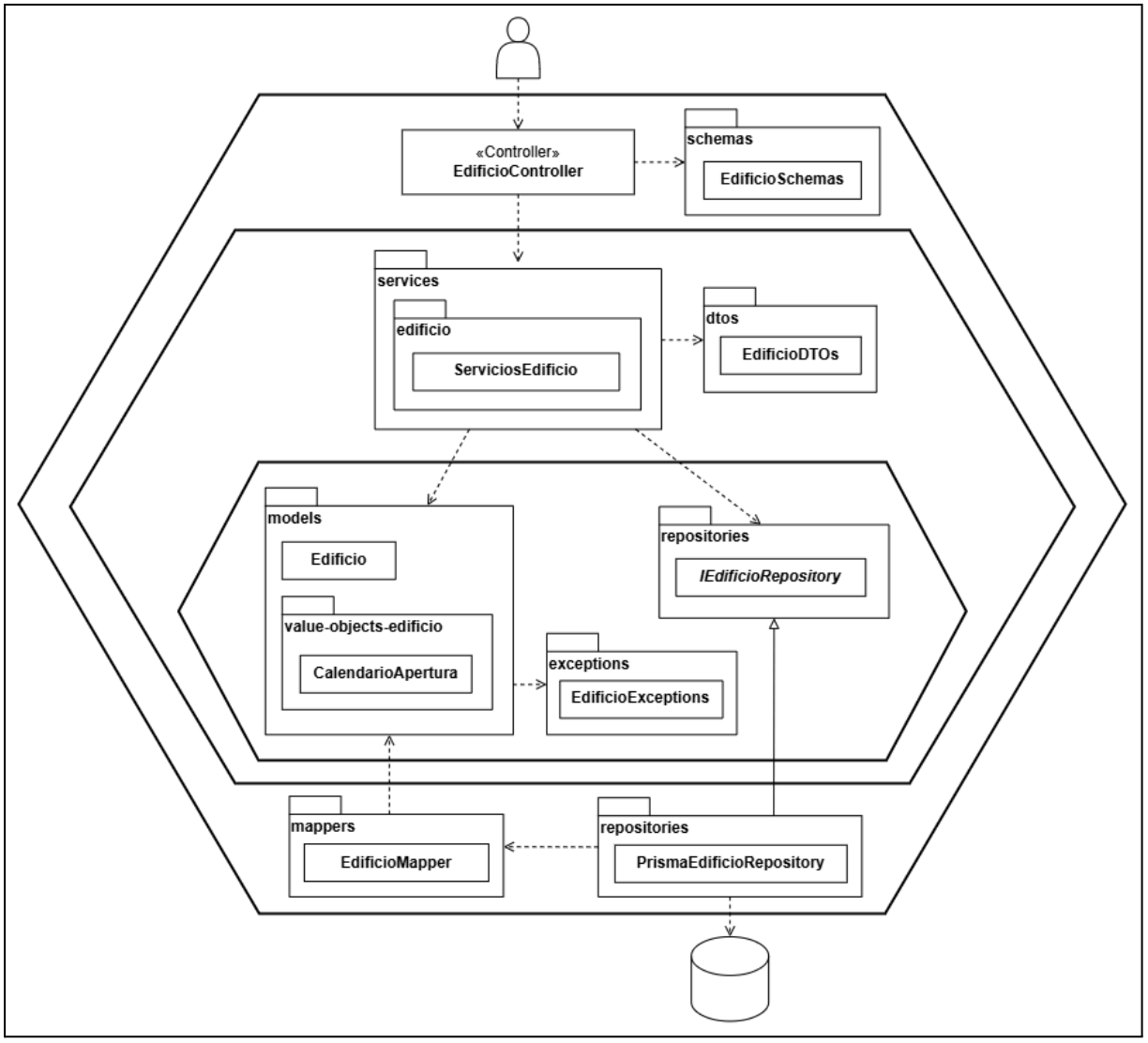


Figura 9: Vista de módulos del agregado Edificio

El módulo *Espacio* se centra en la gestión de los distintos espacios que hay en los edificios. Su lógica de negocio reside en *ServiciosEspacio*, con la cual interactúa el controlador *EspacioController*. La persistencia se desacopla mediante la interfaz *IEspacioRepository*, la cual se implementa en la capa de infraestructura mediante la clase *PrismaEspacioRepository*.

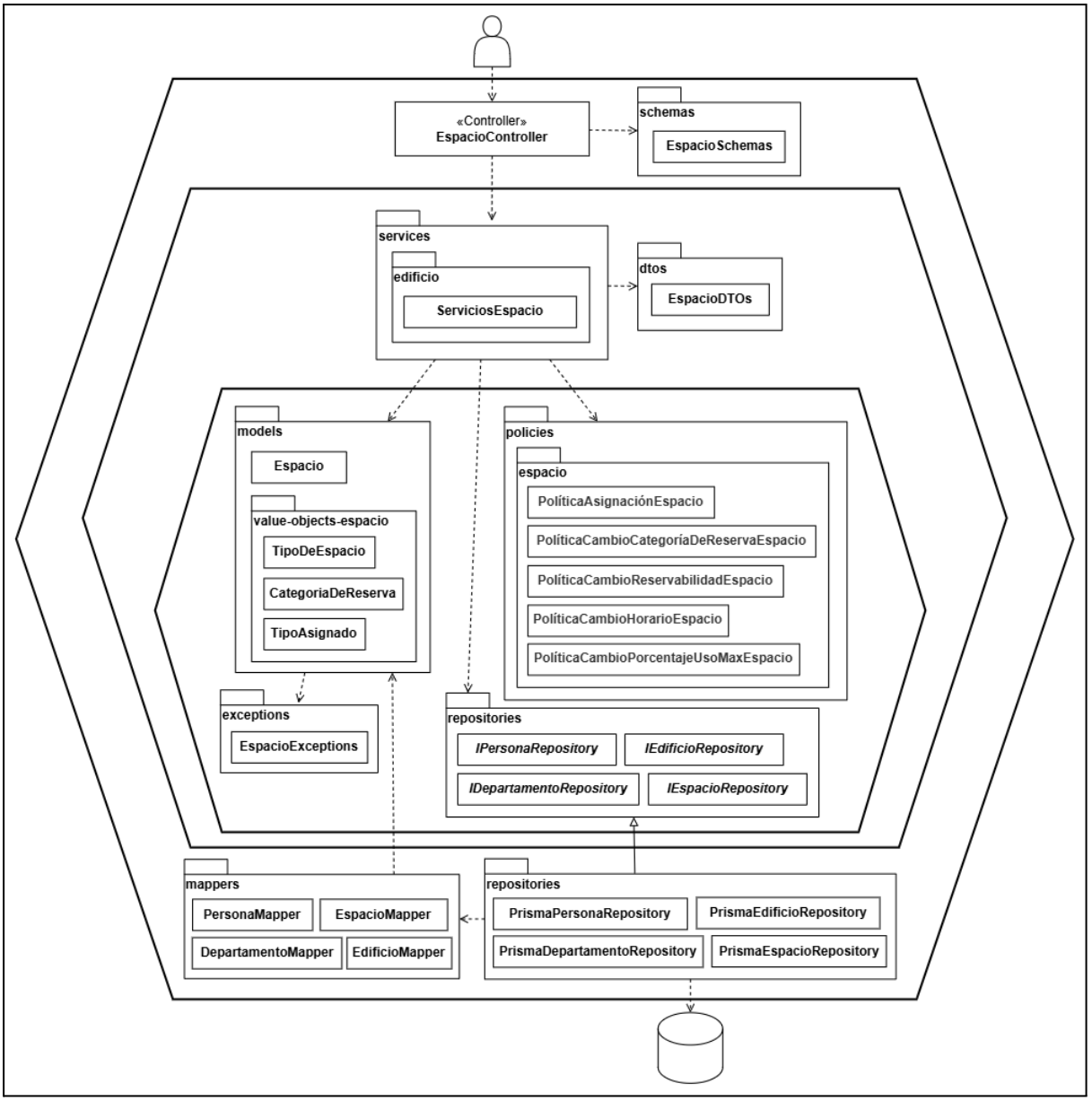


Figura 10: Vista de módulos del agregado Espacio

Finalmente, el módulo *Reserva* se centra en la gestión de las reservas de espacios. Su lógica de negocio reside en *ServiciosReserva*, con la cual interactúa el controlador *ReservaController*. La persistencia se desacopla mediante la interfaz *IReservaRepository*, la cual se implementa en la capa de infraestructura mediante la clase *PrismaReservaRepository*.

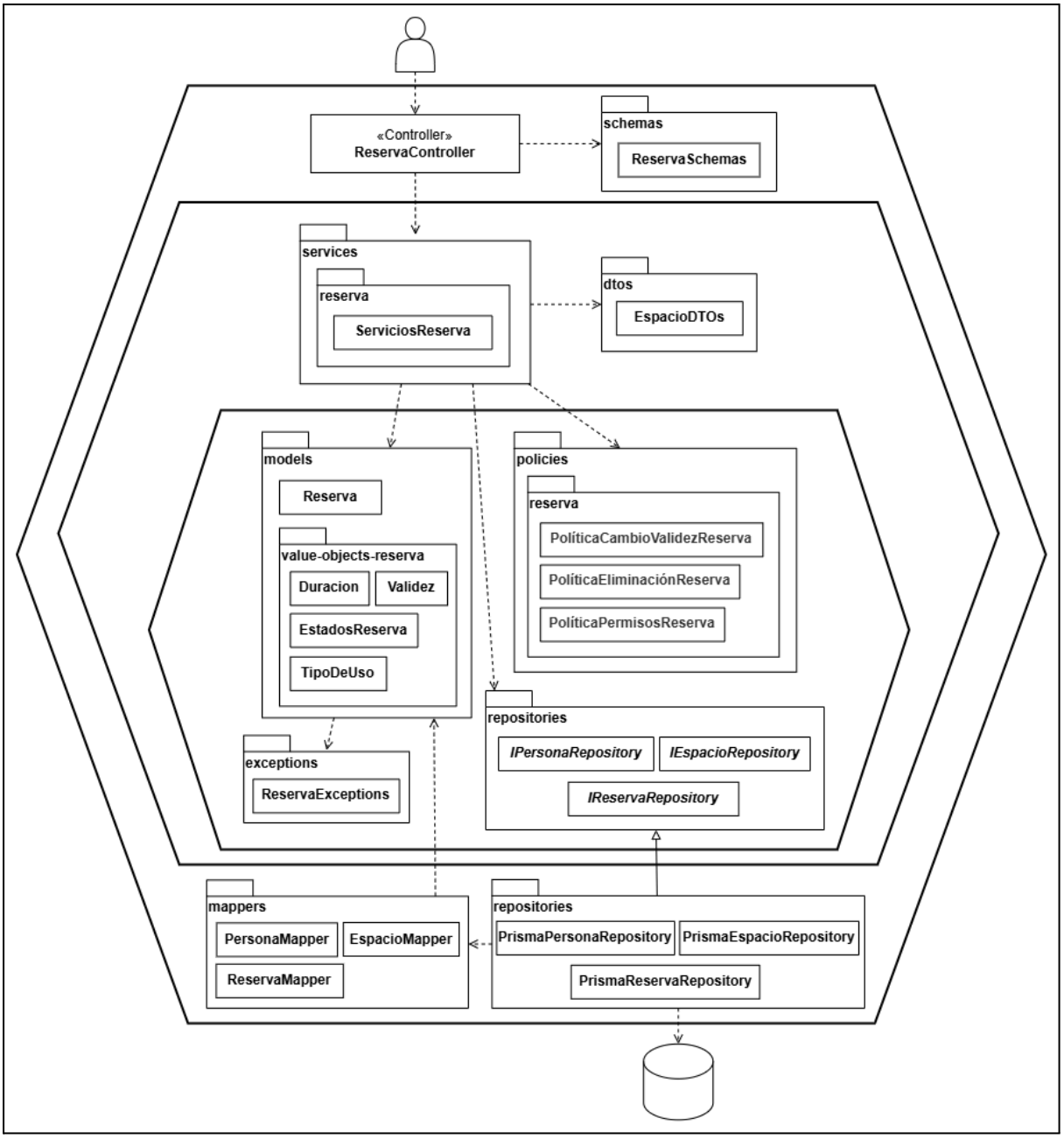


Figura 11: Vista de módulos del agregado Reserva

3.2.2. VISTA DE COMPONENTES Y CONECTORES

La vista de componentes y conectores del sistema sigue un estilo N-Tier organizado en **4 tiers** claramente diferenciados:

- **Nivel de Cliente:** está representado por el cliente web, que actúa como la interfaz de usuario y consume los servicios del sistema mediante el protocolo HTTP.
- **Nivel de Web:** está compuesto por el servidor web y el servidor geográfico, los cuales actúan como puerta de entrada, gestionando las peticiones de los controladores y delegando el procesamiento a la capa de aplicación en el caso del servidor web.

- **Nivel de Aplicación:** en el servidor de aplicación reside toda la lógica de negocio. La conexión con el nivel de web se realiza de forma asíncrona y desacoplada mediante RabbitMQ.
- **Nivel de Datos:** constituido por la base de datos, que centraliza la persistencia de los datos de la aplicación y los datos geográficos. Se comunica con los niveles superiores a través de conectores de tipo SQL/TCP.

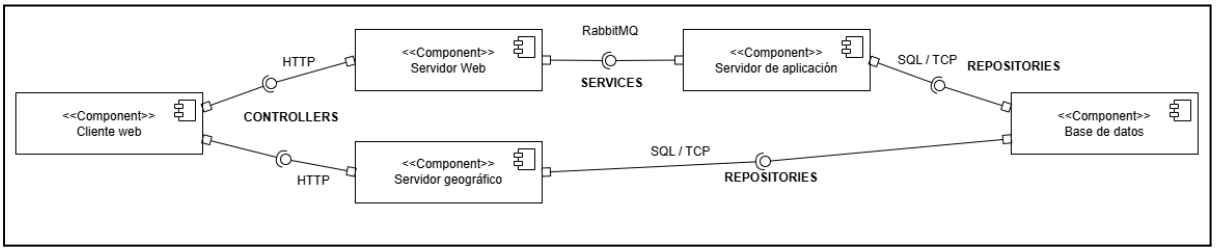


Figura 12: Vista de componentes y conectores del sistema

3.2.3. VISTA DE DESPLIEGUE

La vista de despliegue ilustra la distribución física de los componentes del sistema en sus respectivos nodos de ejecución. El sistema se distribuye principalmente en 5 nodos interconectados:

- **Cliente:** el componente cliente web implementado en React se ejecuta en el navegador del usuario, por defecto en el puerto 5173.
- **Servidor web:** implementado en NestJS sobre Node.js constituye el punto de entrada para la lógica de la aplicación, a través del puerto 3000.
- **Servidor de aplicación:** también implementado en NestJS se comunica con el servidor web mediante un bus de mensajes RabbitMQ, en el puerto 5672.
- **Servidor geográfico:** implementado con pygeoapi, sirve datos geográficos por el puerto 5000.
- **Servidor de base de datos:** se despliega una instancia de PostgreSQL 14 con la extensión PostGIS en el puerto 5432.

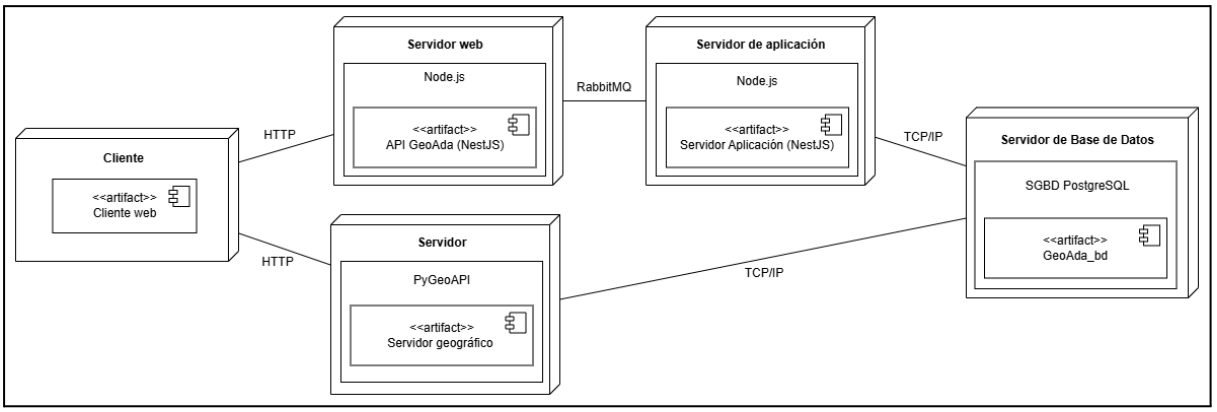


Figura 13: Vista de despliegue del sistema

3.2.4. DIAGRAMAS DE SECUENCIA

A continuación, detallamos los requisitos implementados con el diagrama de secuencia que refleja cómo ha sido implementado.

RF_U3_B1	El usuario deberá ser capaz de modificar su rol (DD3).

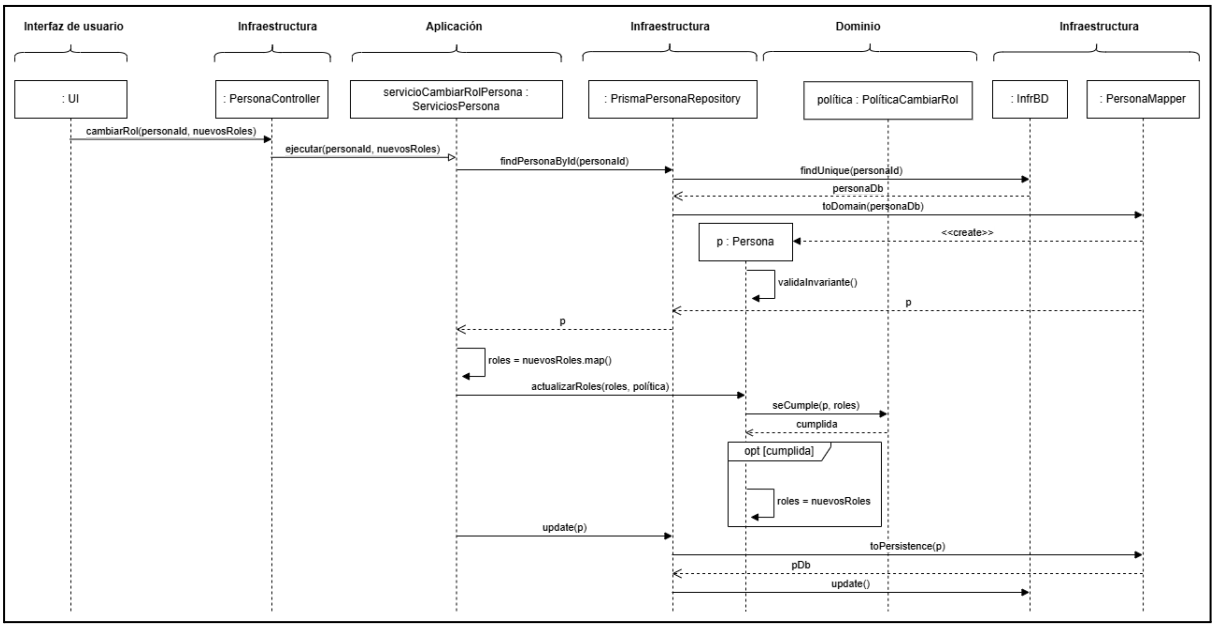


Figura 14: Diagrama de secuencia (RF_U3_B1)

RF_U4	El usuario deberá ser capaz de consultar su rol (DD3).

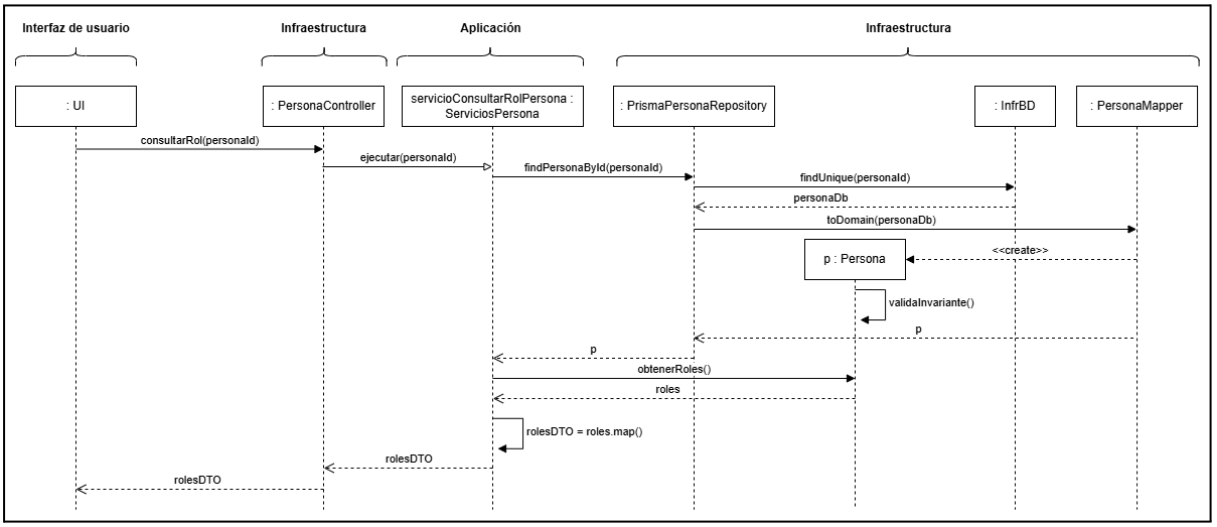


Figura 15: Diagrama de secuencia (RF_U4)

RF_U5_B3	El usuario deberá ser capaz de adscribirse a un departamento (DD4).

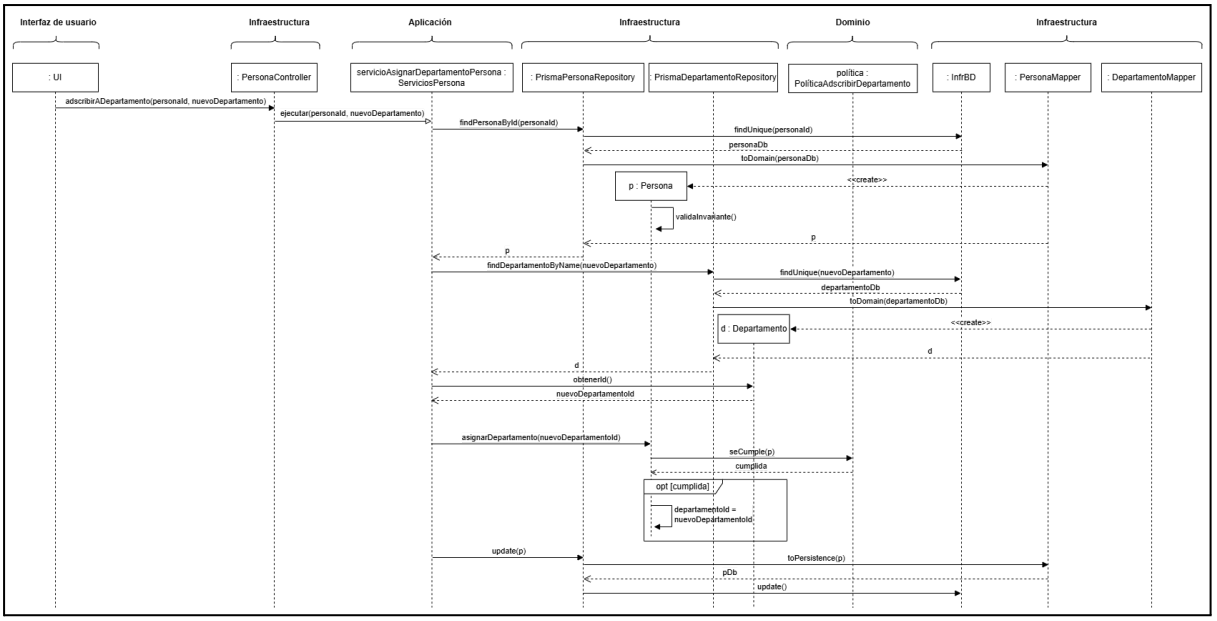


Figura 16: Diagrama de secuencia (RF_U5_B3)

RF_U6_B3	El usuario deberá ser capaz de eliminar su adscripción a un departamento (DD4).

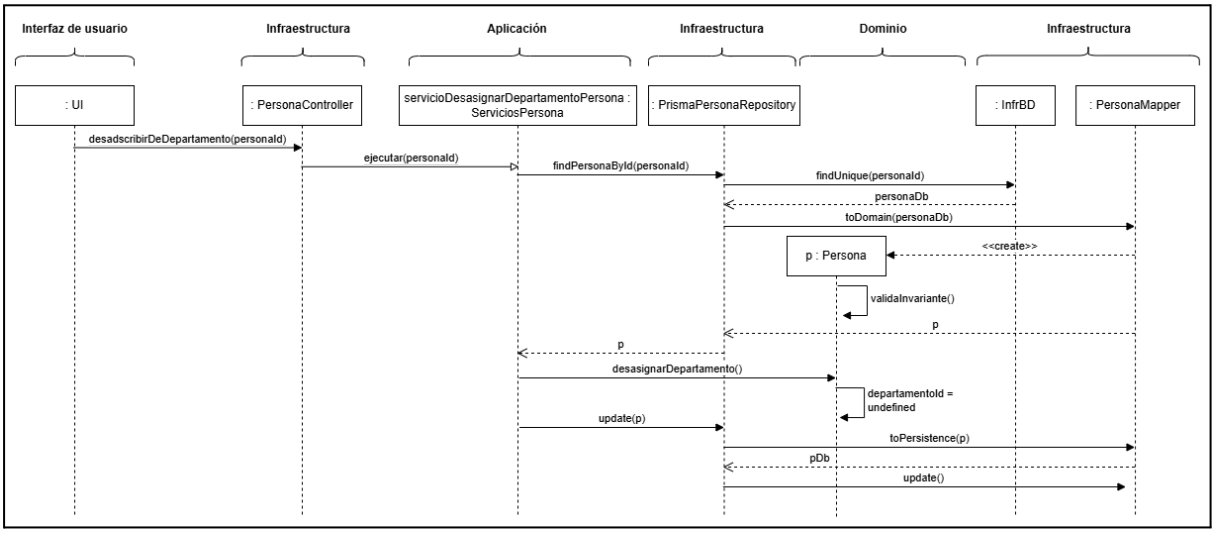


Figura 17: Diagrama de secuencia (RF_U6_B3)

RF_U7	El usuario deberá ser capaz de consultar el departamento (DD4) al que está adscrito.

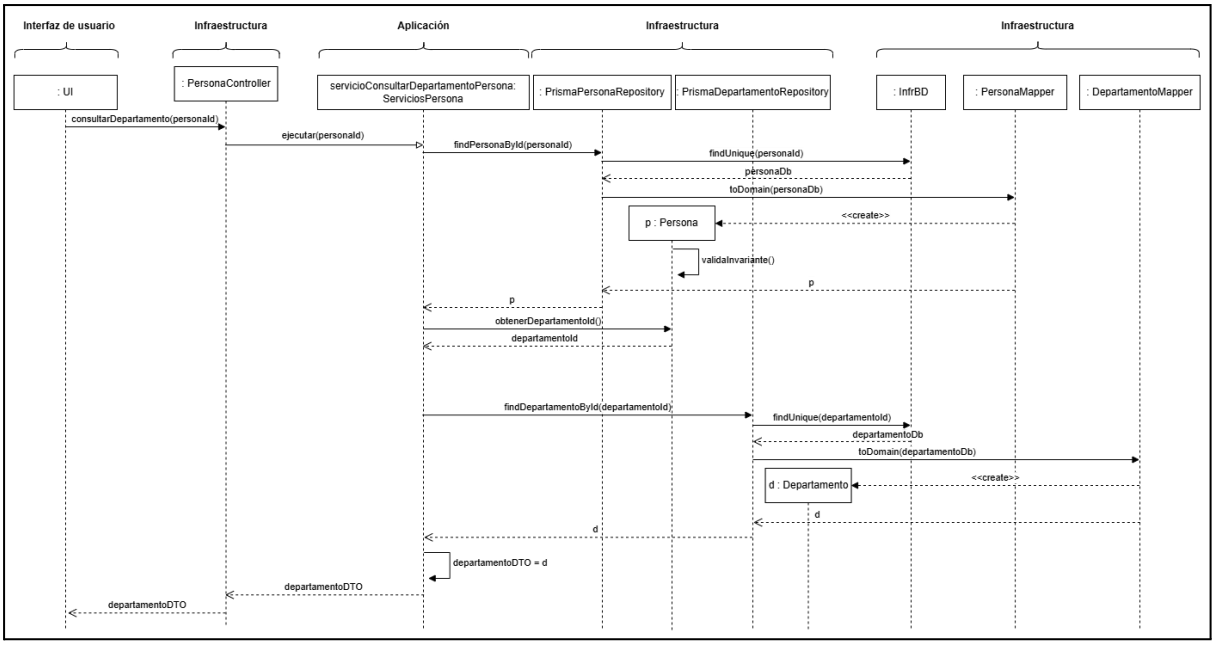


Figura 18: Diagrama de secuencia (RF_U7)

RF_U8	El usuario deberá ser capaz de crear una persona (DD2).

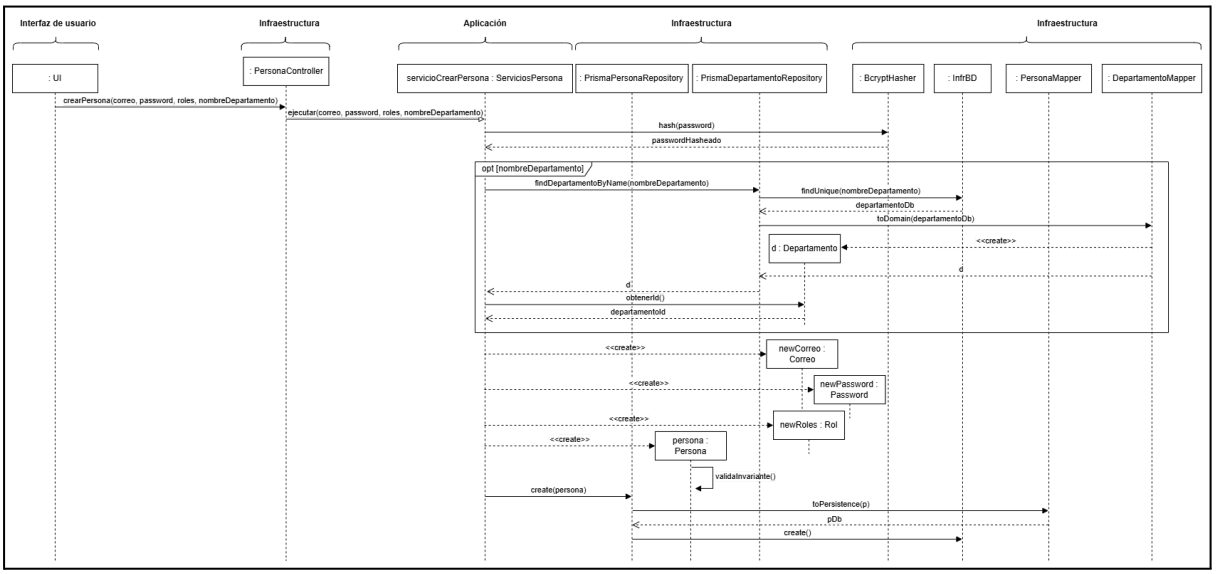


Figura 19: Diagrama de secuencia (RF_U8)

RF_U9	El usuario deberá ser capaz de consultar los espacios (DD5) del edificio (DD12).

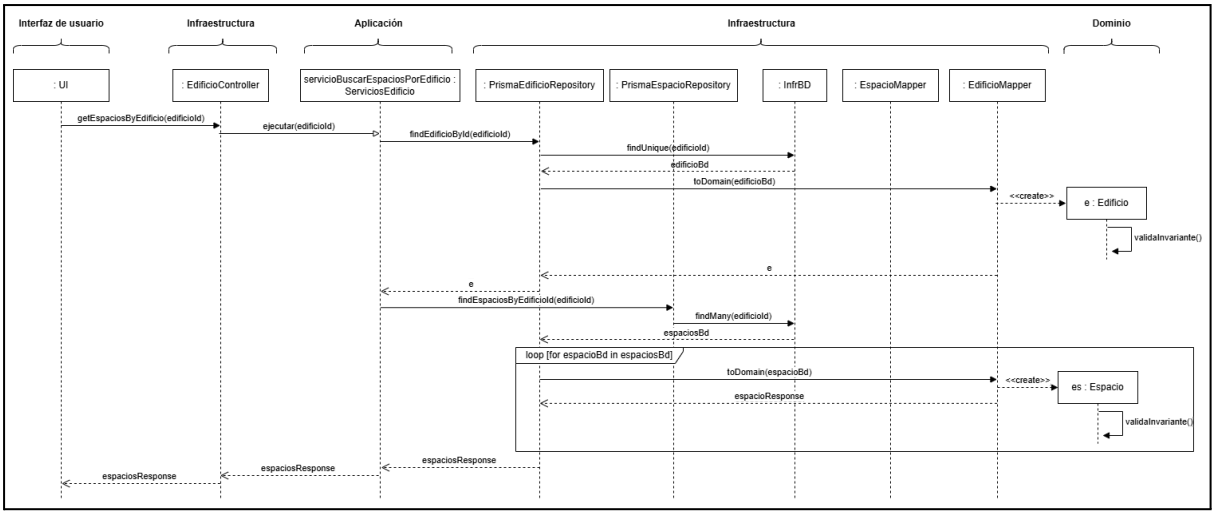


Figura 20: Diagrama de secuencia (RF_U9)

RF_U10_C1	El usuario deberá ser capaz de modificar si un espacio (DD5) es reservable.

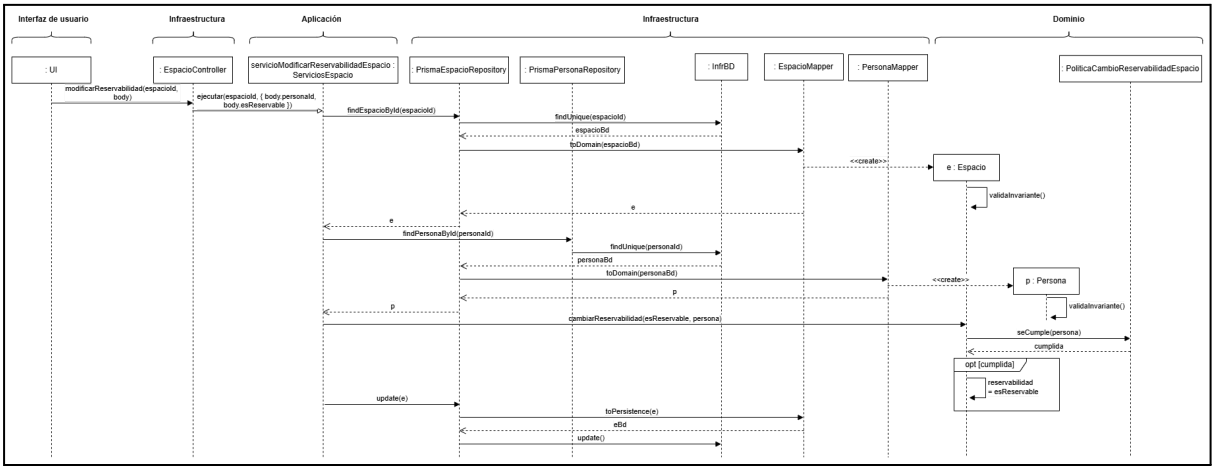


Figura 21: Diagrama de secuencia (RF_U10_C1)

RF_U11_C2	El usuario deberá ser capaz de modificar la categoría de reserva (DD7) de un espacio (DD5).

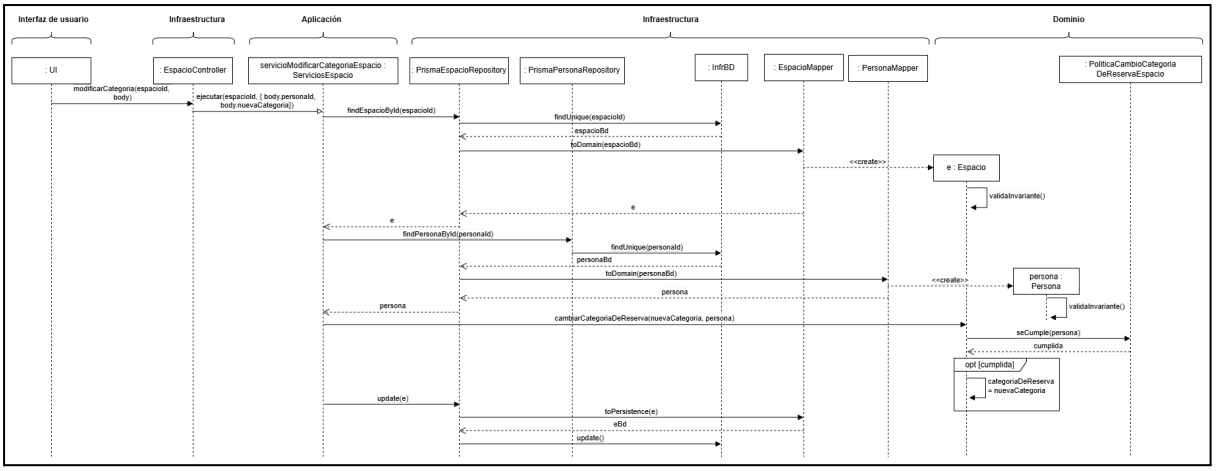


Figura 22: Diagrama de secuencia (RF_U11_C2)

RF_U12_C3	El usuario deberá ser capaz de modificar los horarios disponibles de un espacio (DD5).

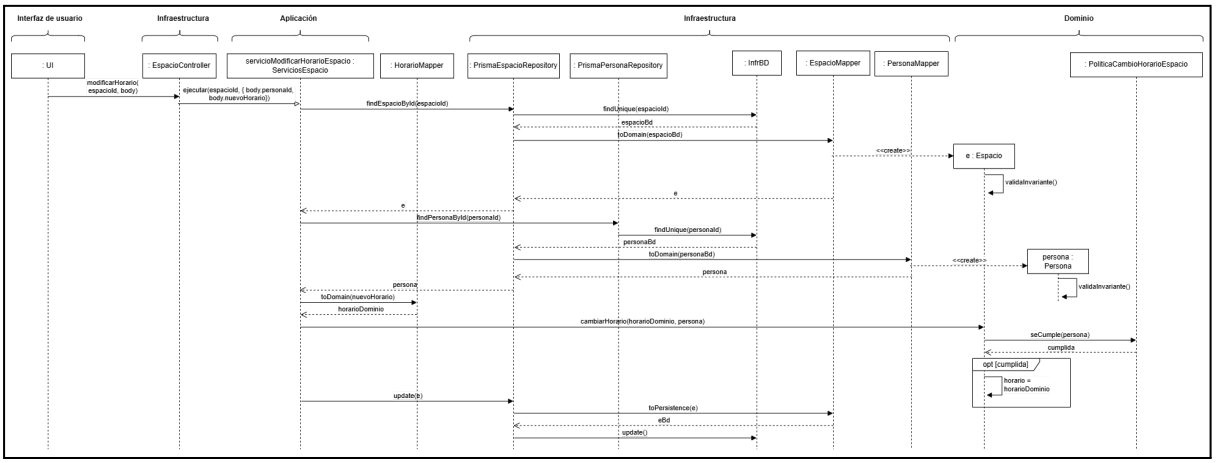


Figura 23: Diagrama de secuencia (RF_U12_C3)

RF_U13_C4	El usuario deberá ser capaz de modificar la asignación de un espacio (DD5).

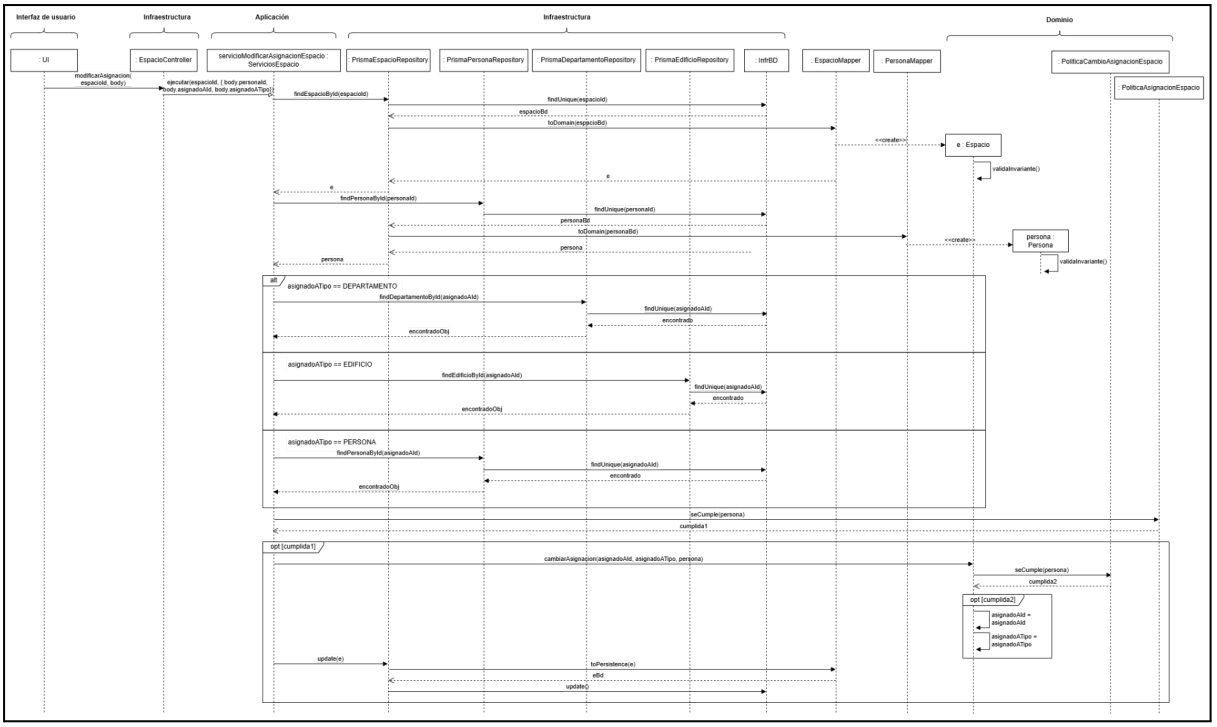


Figura 24: Diagrama de secuencia (RF_U13_C4)

RF_U14_C5_O1	El usuario deberá ser capaz de modificar el porcentaje de uso máximo de un espacio (DD5).

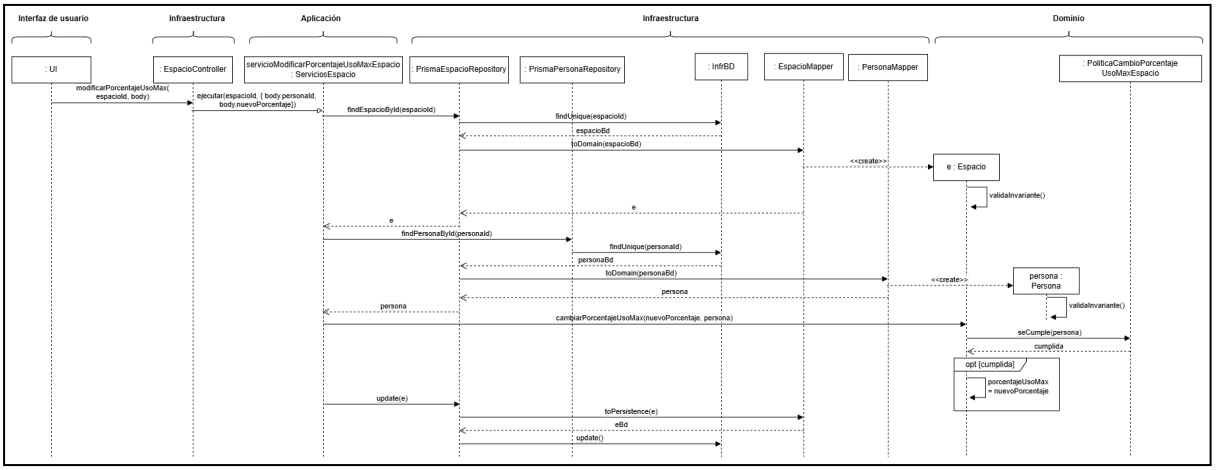


Figura 25: Diagrama de secuencia (RF_U14_C5_O1)

RF_U15_D	El usuario deberá ser capaz de buscar espacios (DD5).

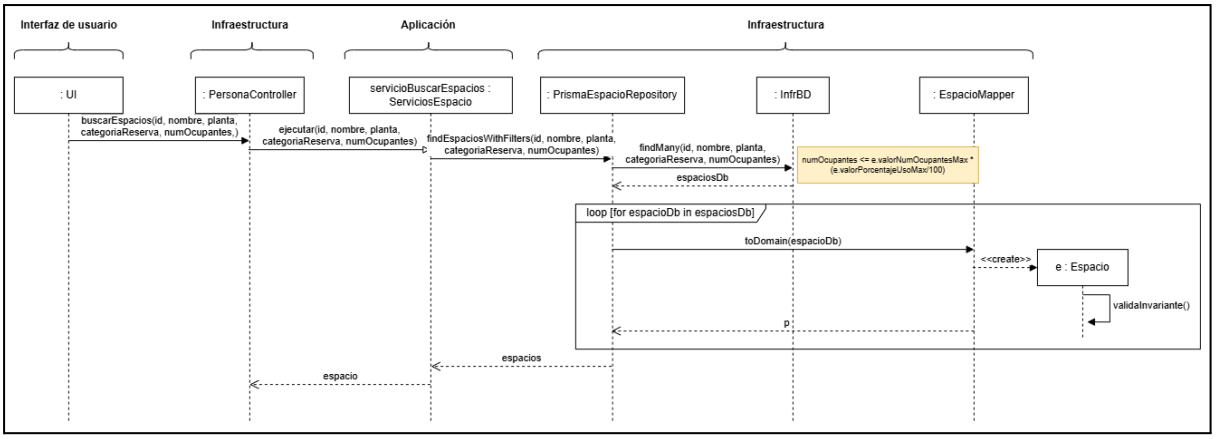


Figura 26: Diagrama de secuencia (RF_U15_D)

RF_U16_E_F_O3	El usuario deberá ser capaz de hacer una reserva (DD8).

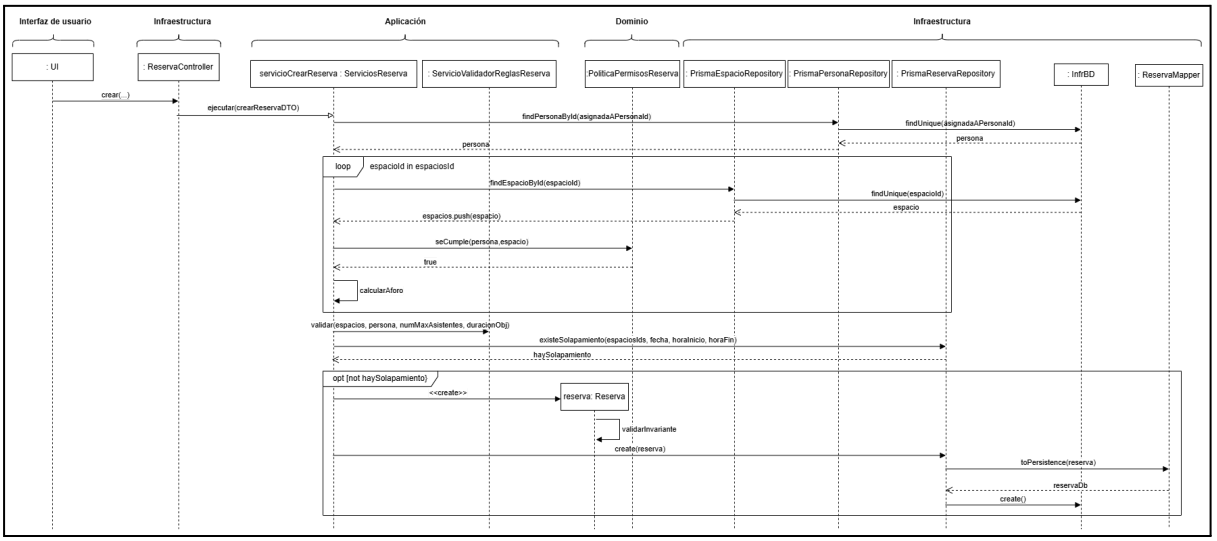


Figura 27: Diagrama de secuencia (RF_U16_E_F_O3)

RF_U17_H	El usuario deberá ser capaz de consultar las reservas vivas (DD10).

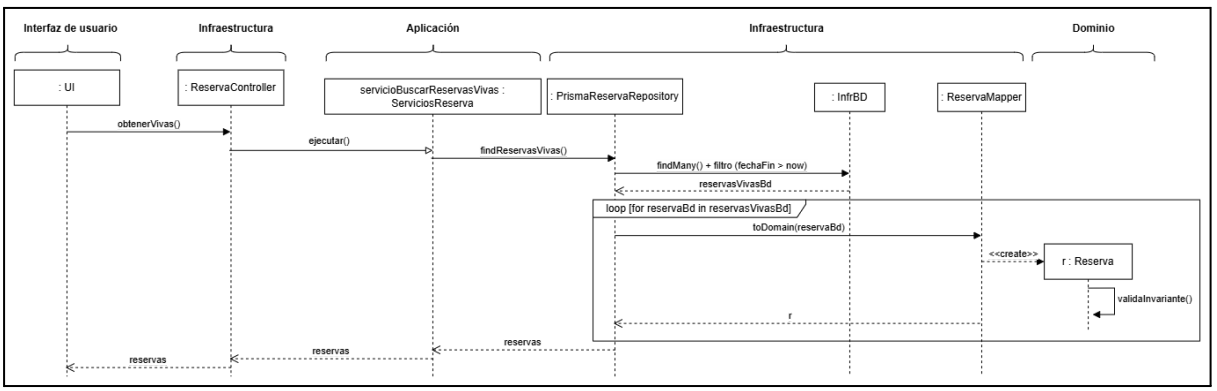


Figura 28: Diagrama de secuencia (RF_U17_H)

RF_U18_H	El usuario deberá ser capaz de eliminar una reserva (DD8).

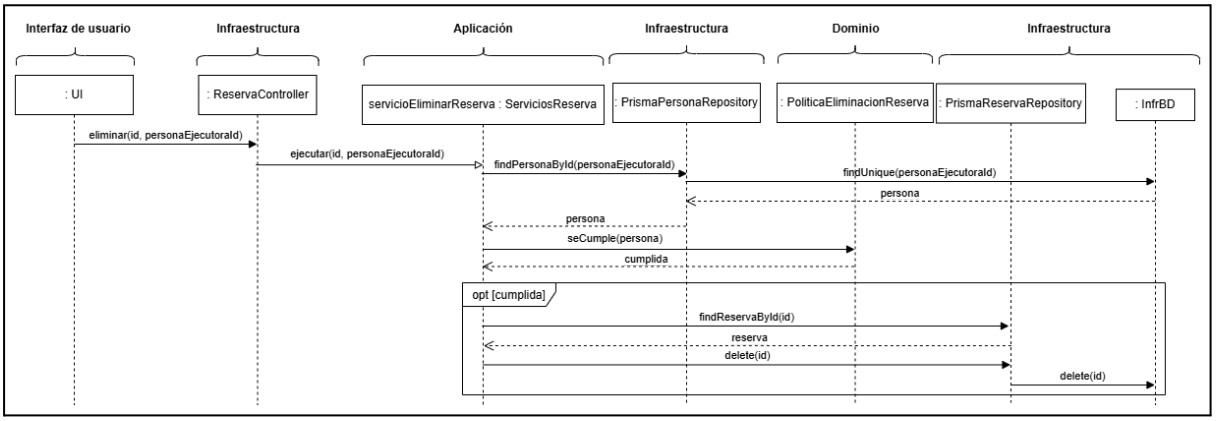


Figura 29: Diagrama de secuencia (RF_U18_H)

RF_U19_H_O4	El usuario deberá ser capaz de modificar la validez (DD11) de una reserva (DD8) de “potencialmente inválida” a “válida”.

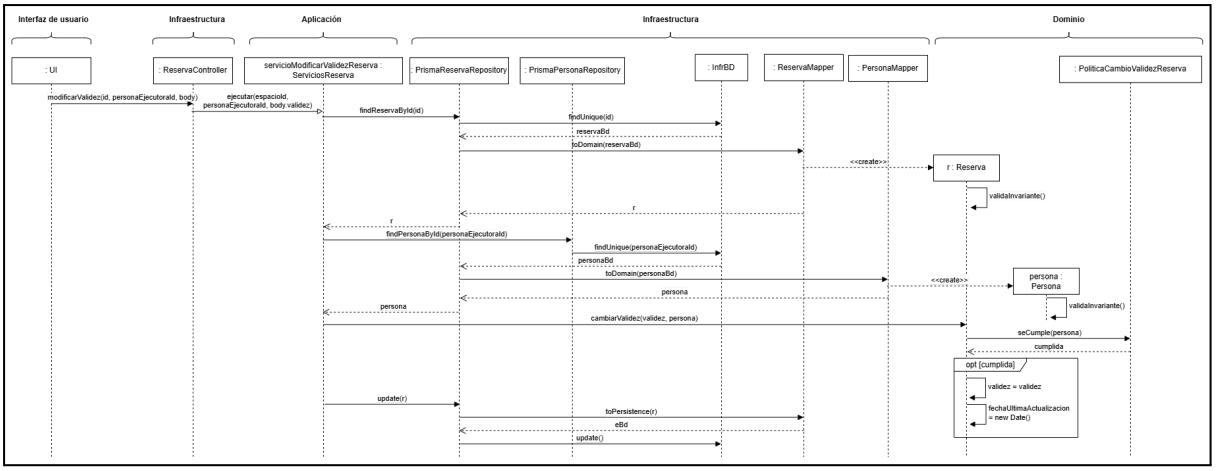


Figura 30: Diagrama de secuencia (RF_U19_H_O4)

RF_U20	El usuario deberá ser capaz de modificar el calendario de apertura del edificio (DD12).

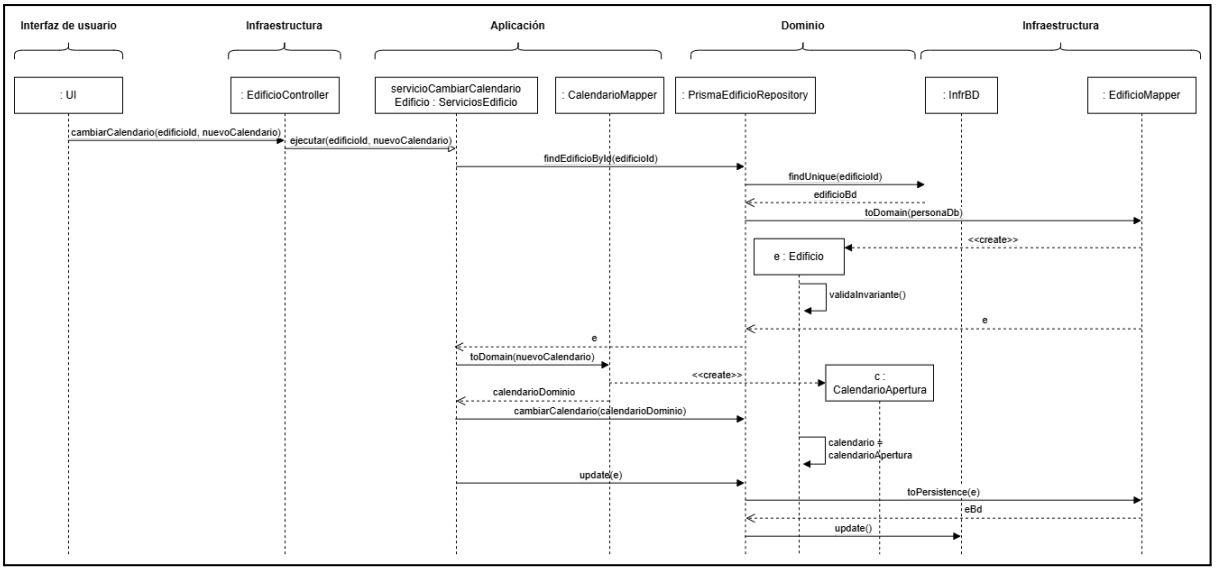


Figura 31: Diagrama de secuencia (RF_U20)

RF_U21	El usuario deberá ser capaz de modificar el horario de apertura del edificio (DD12).

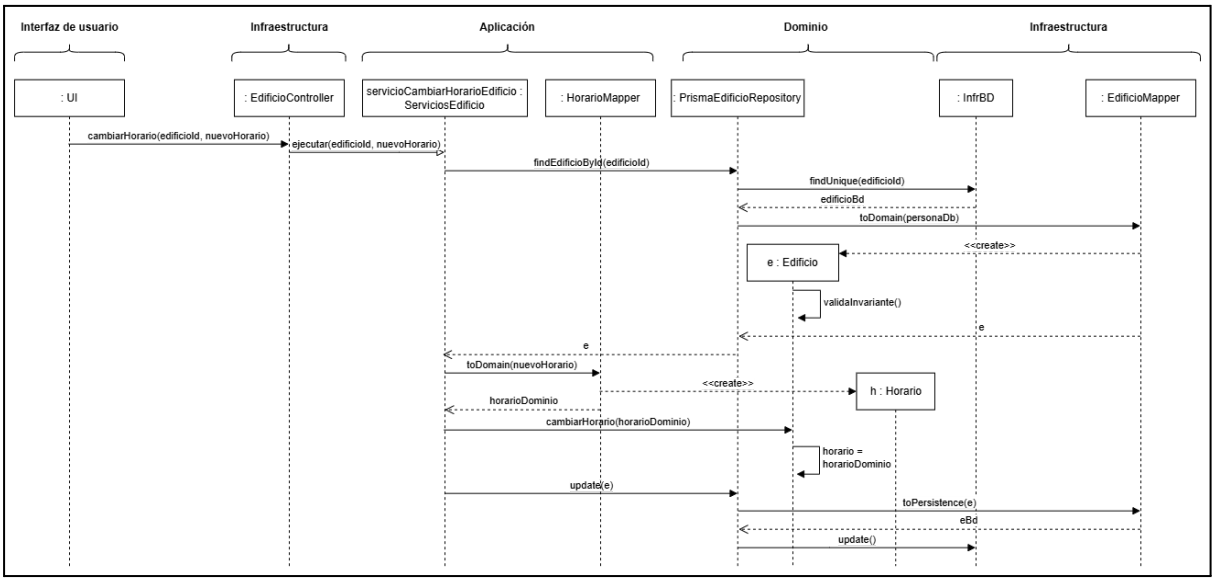


Figura 32: Diagrama de secuencia (RF_U21)

RF_U22	El usuario deberá ser capaz de modificar el porcentaje de ocupación máxima del edificio (DD12).

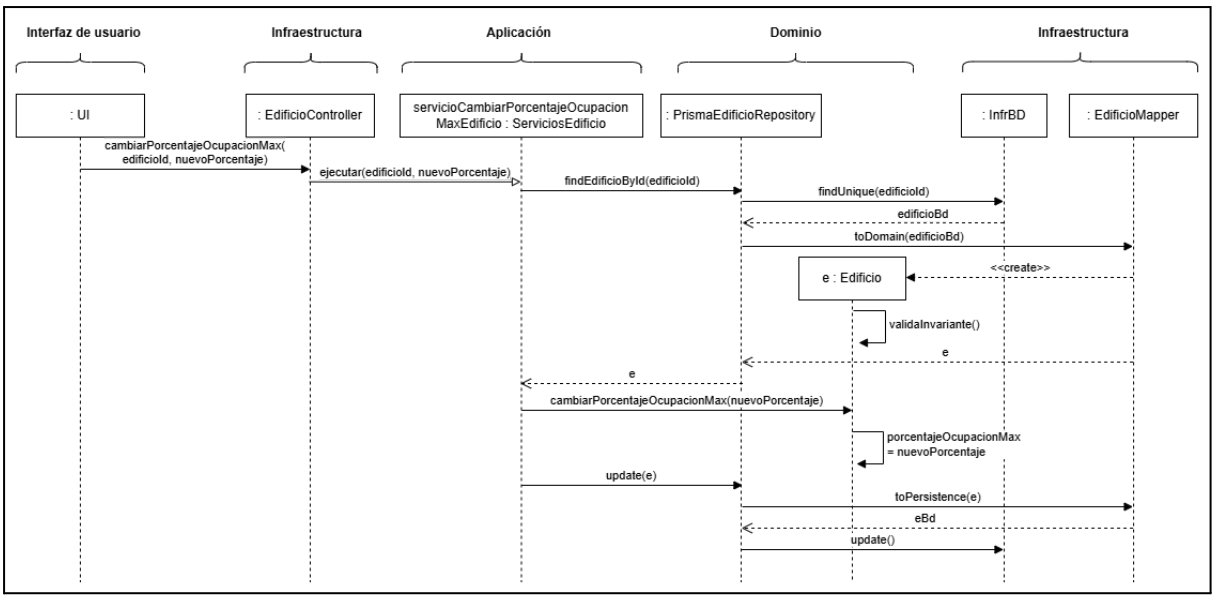
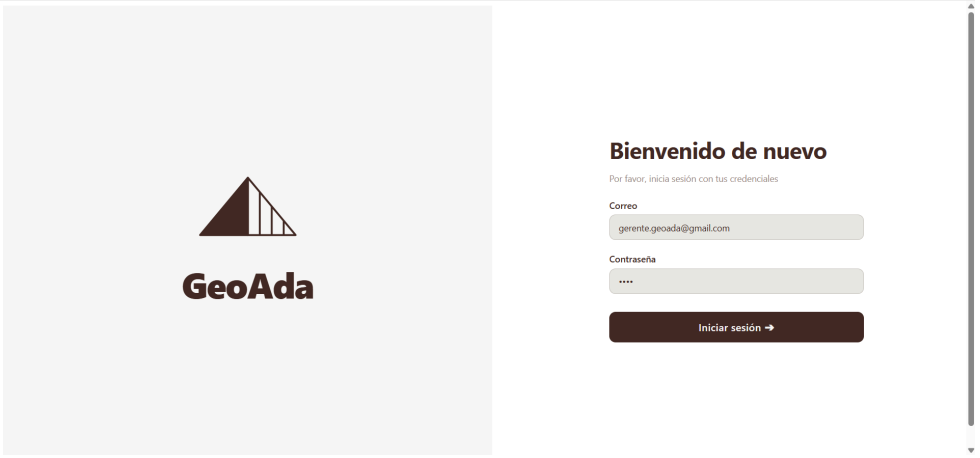
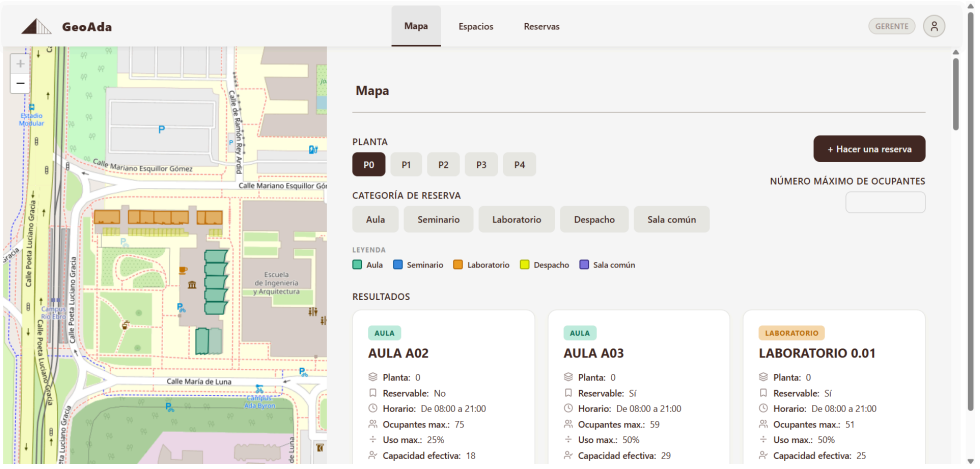
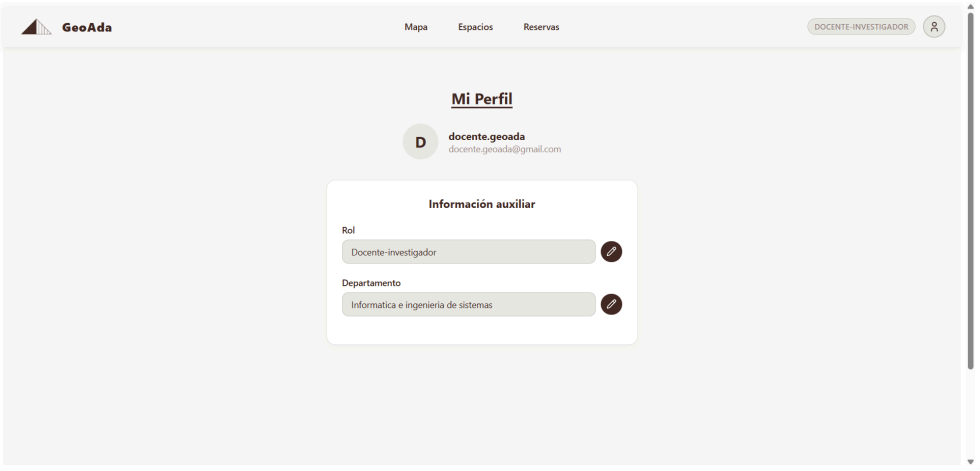
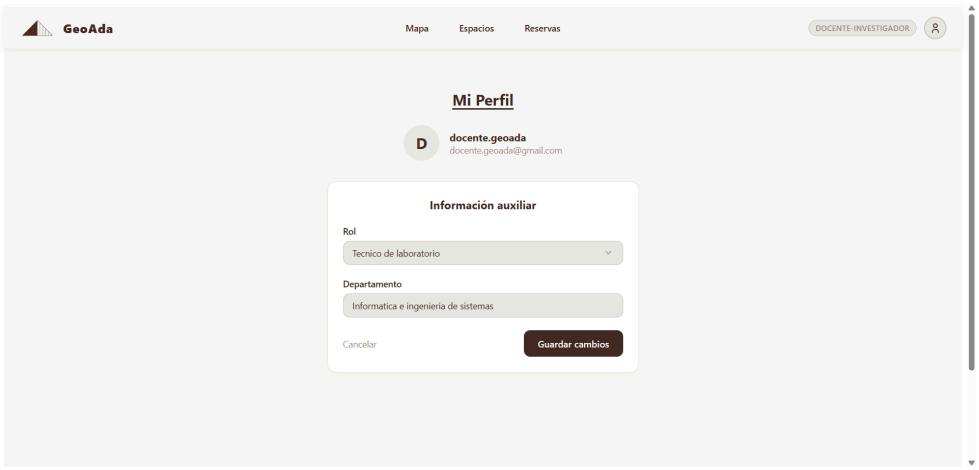
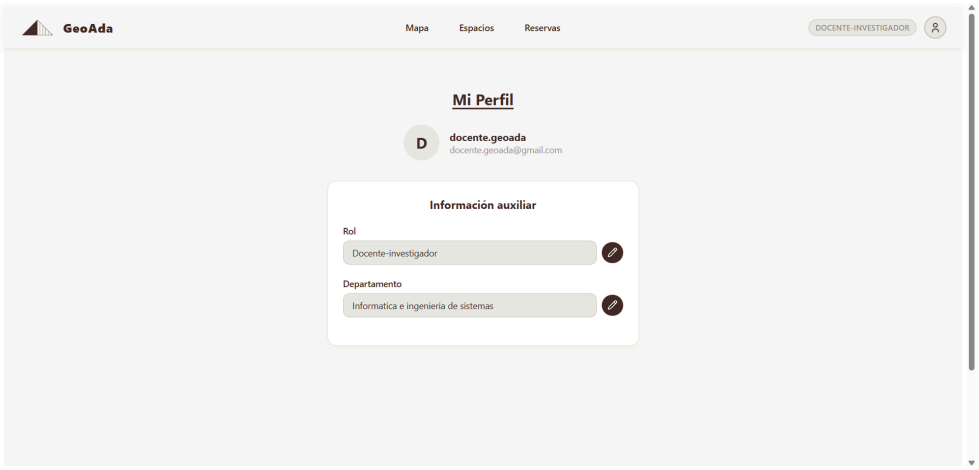
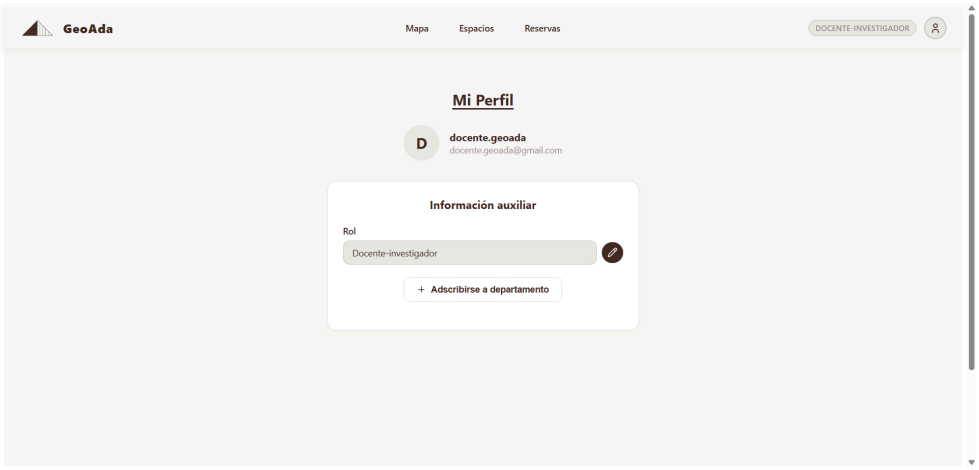


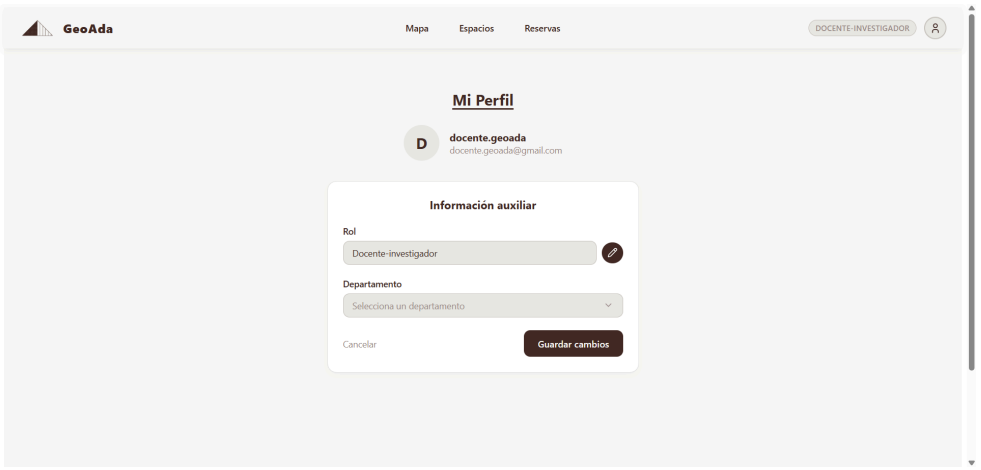
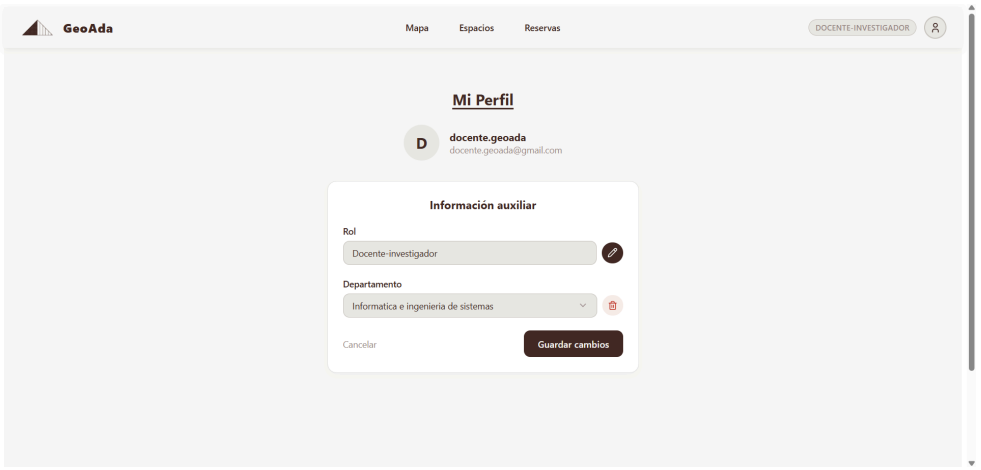
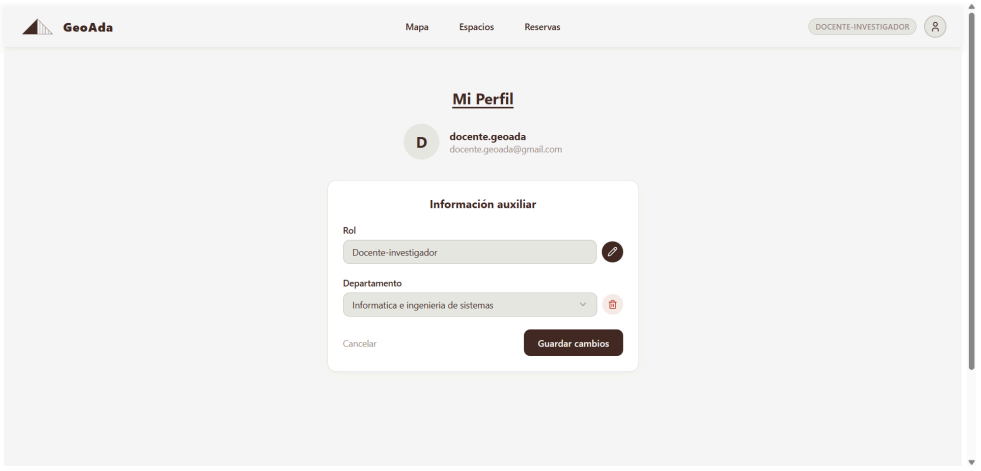
Figura 33: Diagrama de secuencia (RF_U22)

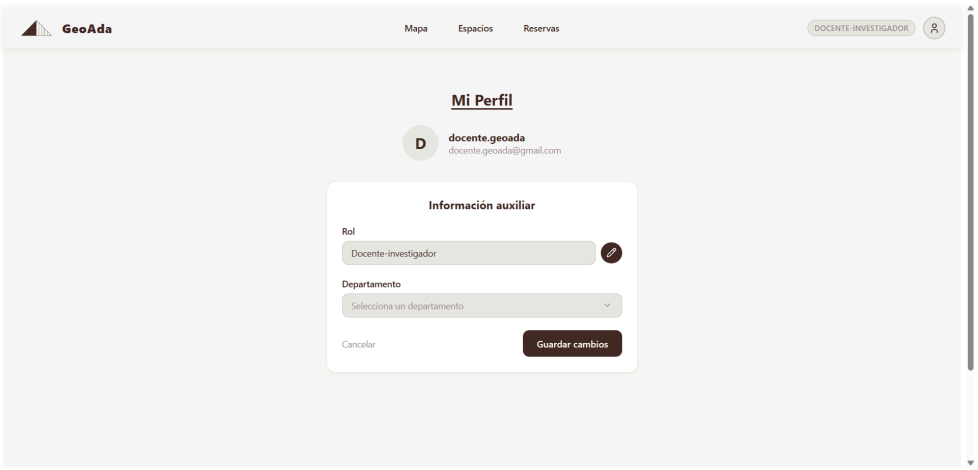
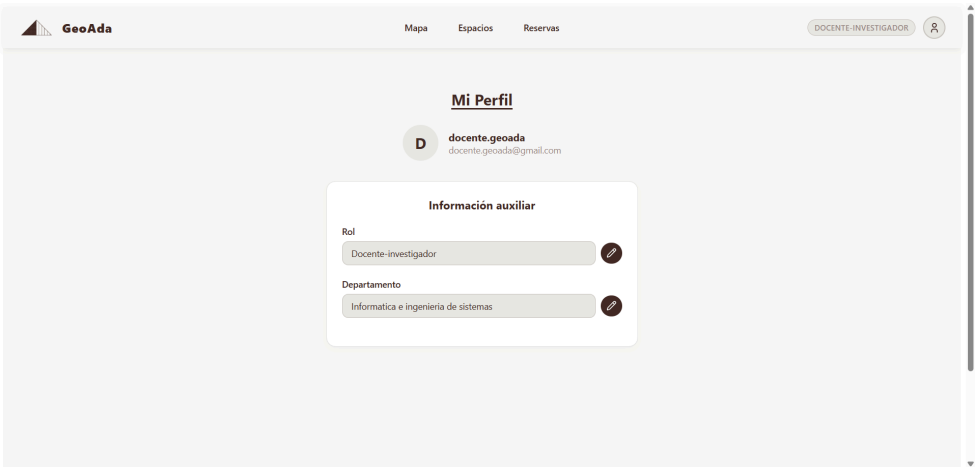
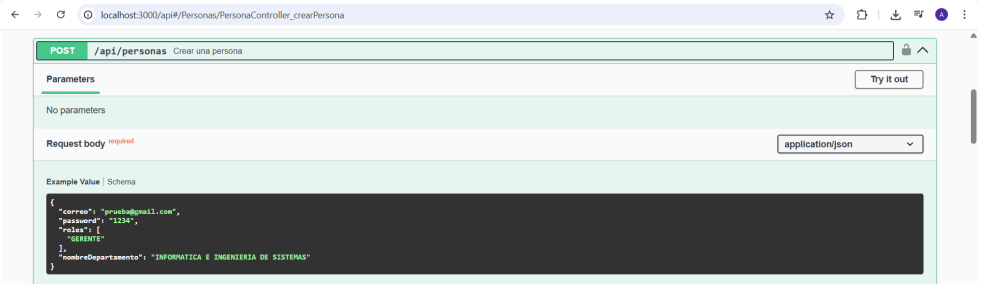
3.3. ESTADO ACTUAL DE LA APLICACIÓN

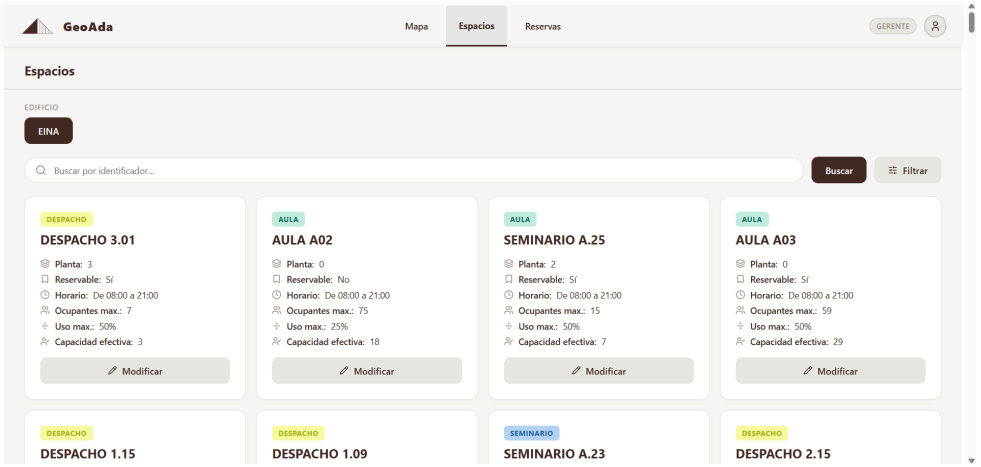
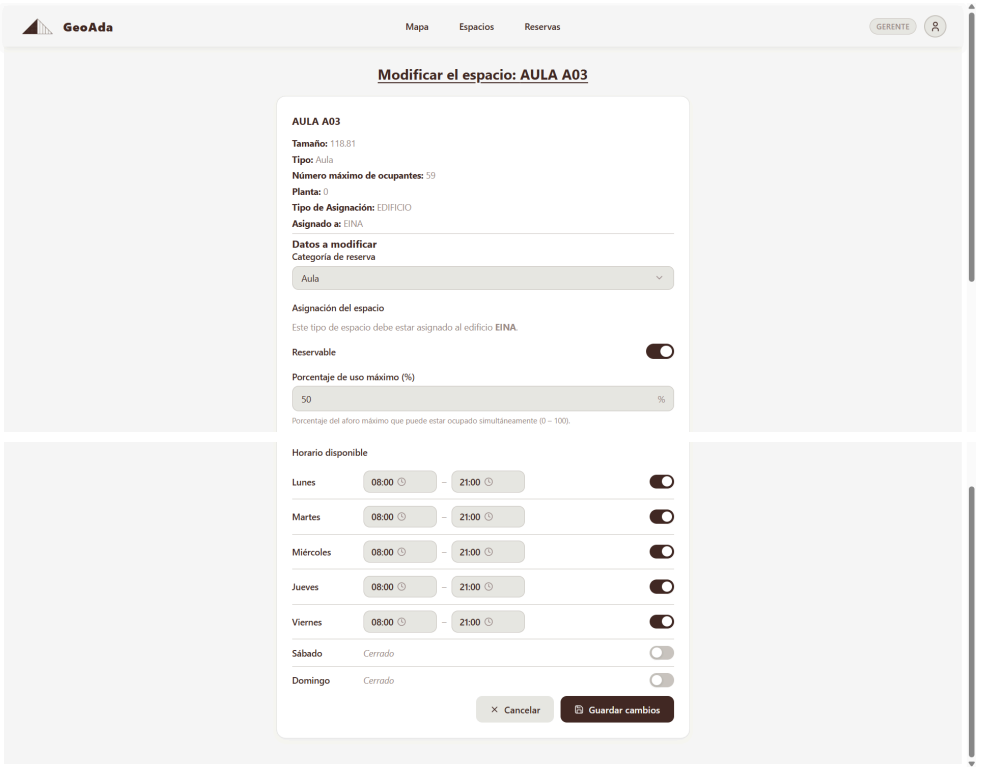
3.3.1. GUI

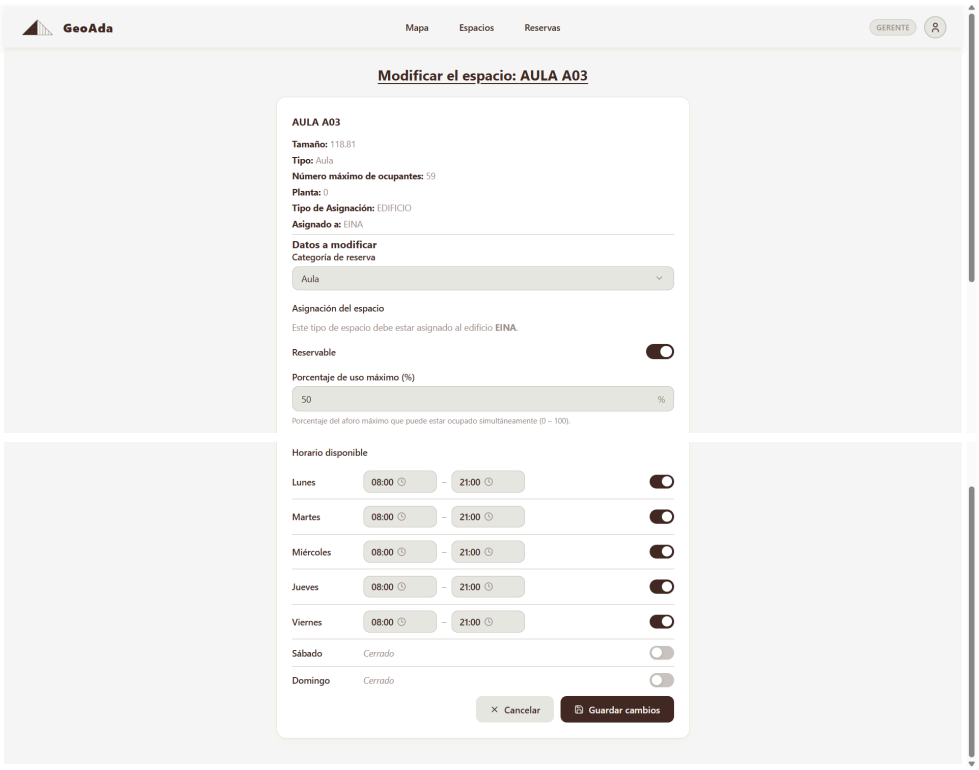
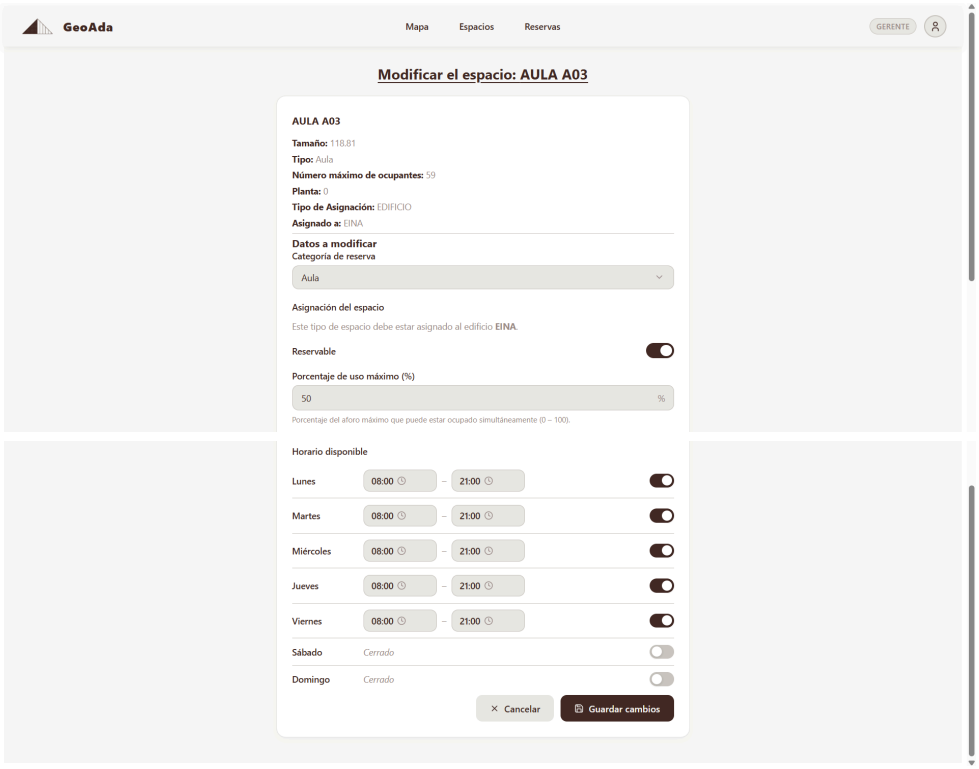
RF_U1	El usuario deberá ser capaz de iniciar sesión. 
RF_U2_A	El usuario deberá ser capaz de visualizar un mapa (DD1). 
RF_U3_B1	El usuario deberá ser capaz de modificar su rol (DD3). 

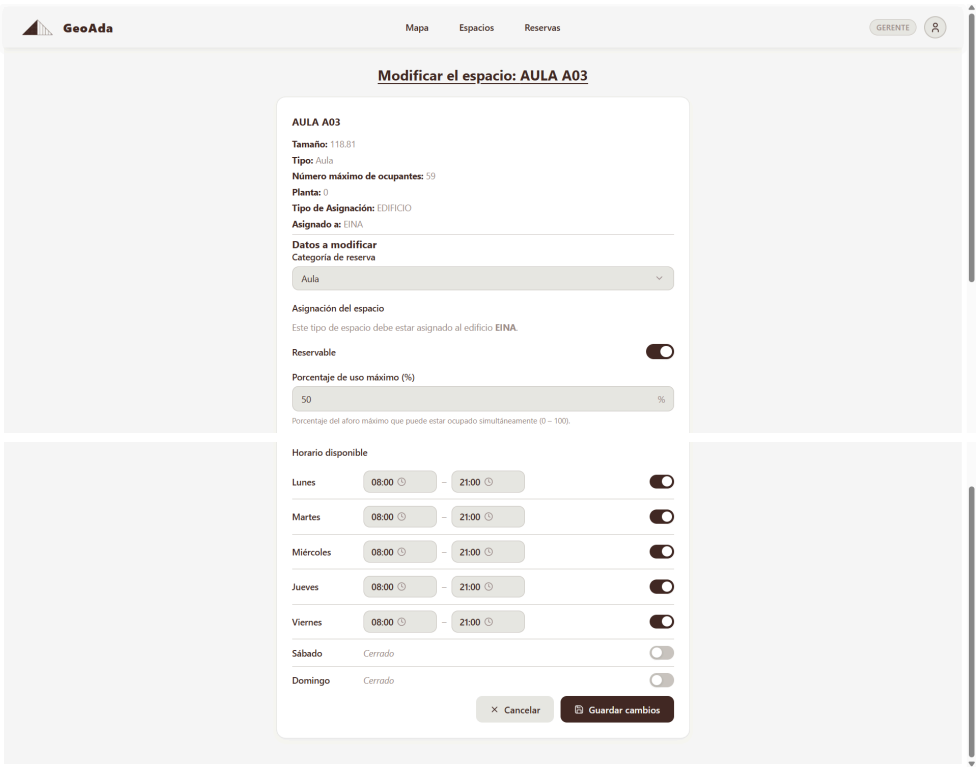
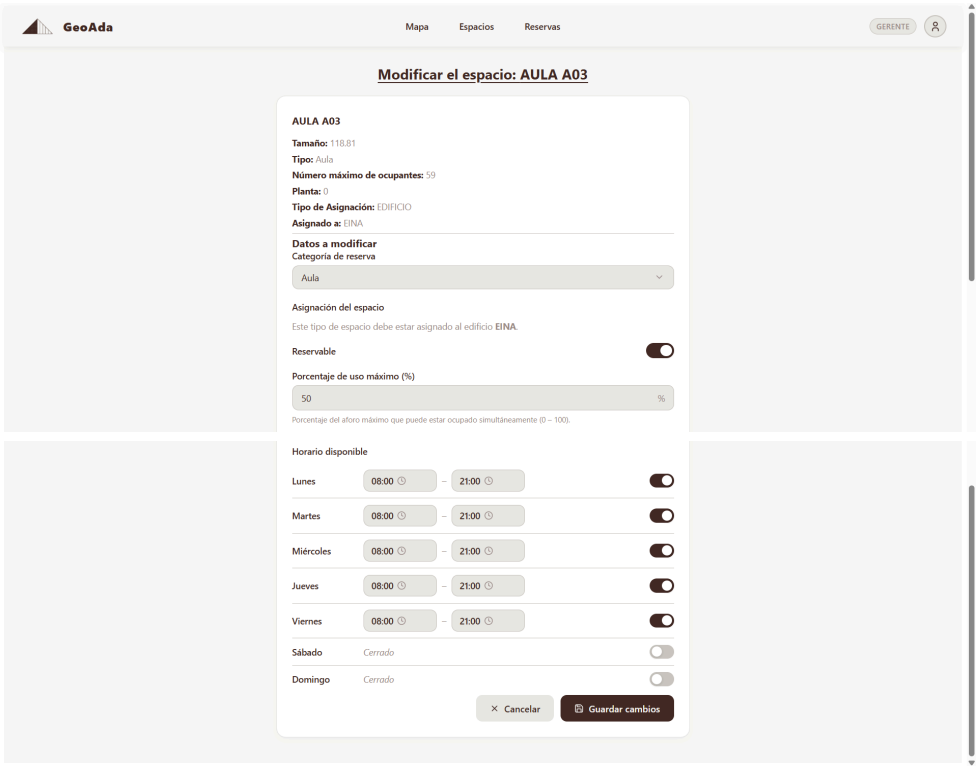
	
RF_U4	El usuario deberá ser capaz de consultar su rol (DD3). 
RF_U5_B3	El usuario deberá ser capaz de adscribirse a un departamento (DD4). 

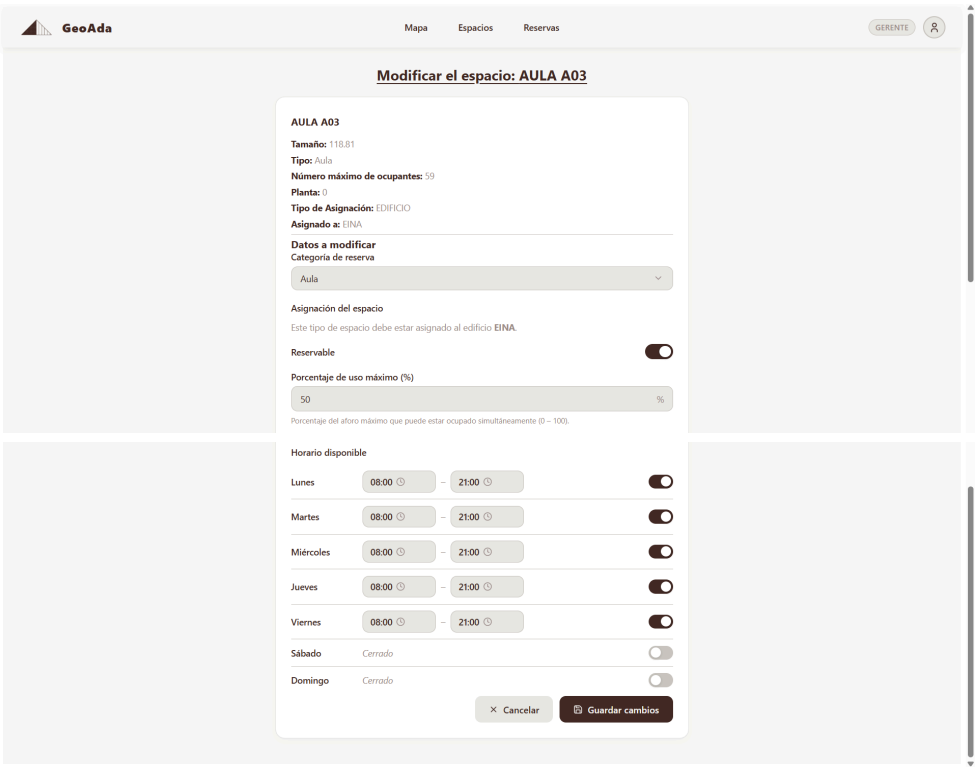
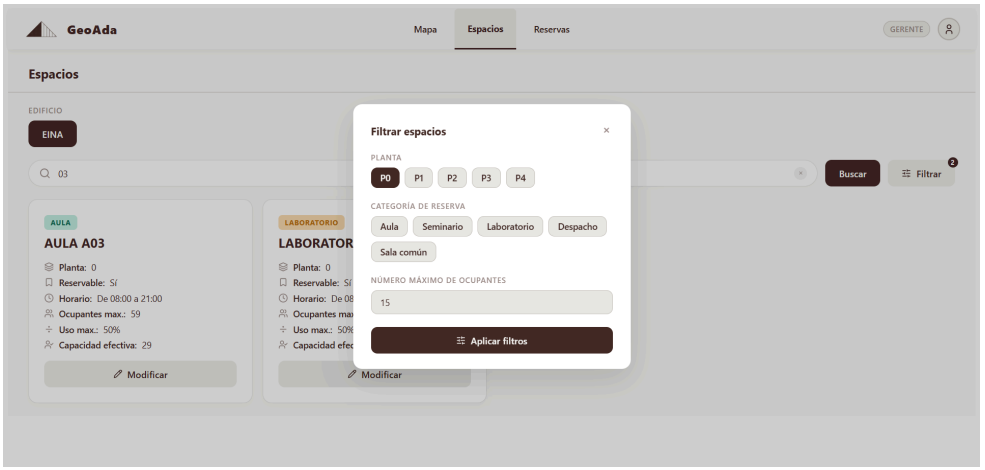
	 
RF_U6_B3	El usuario deberá ser capaz de eliminar su adscripción a un departamento (DD4). 

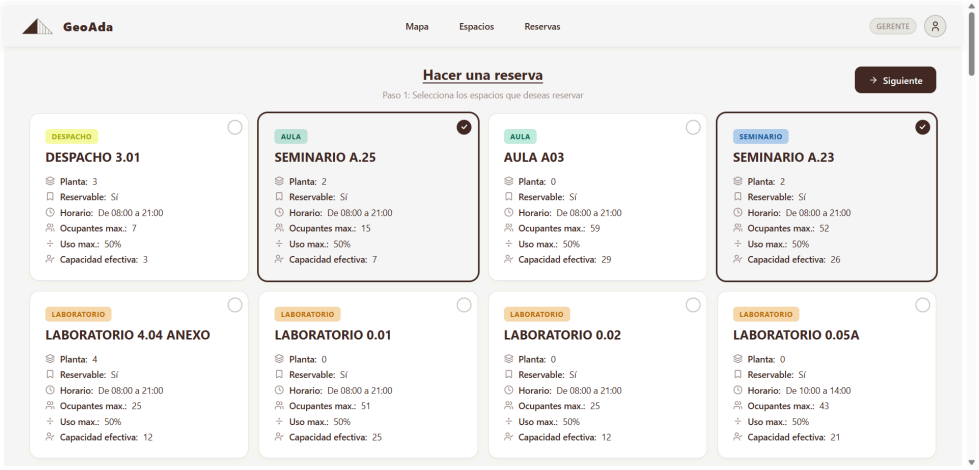
	
RF_U7	El usuario deberá ser capaz de consultar el departamento (DD4) al que está adscrito. 
RF_U8	El usuario deberá ser capaz de crear una persona (DD2). 
RF_U9	El usuario deberá ser capaz de consultar los espacios (DD5) del edificio (DD12).

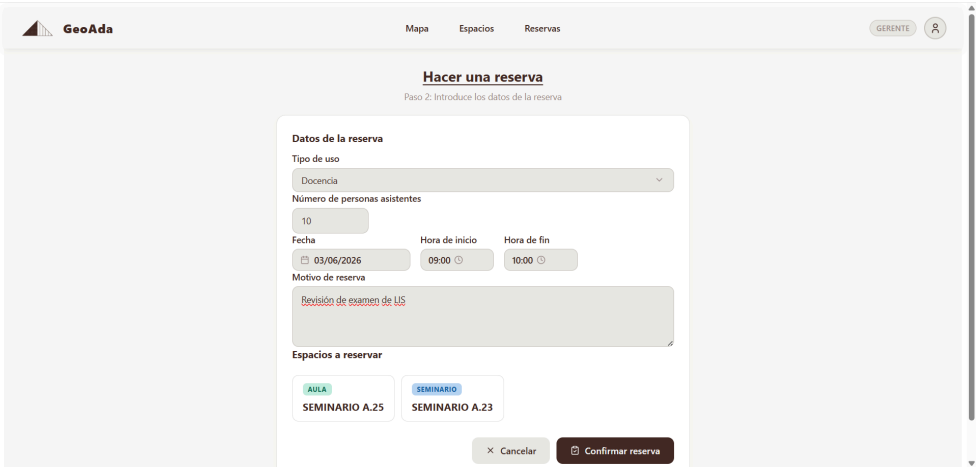
	
RF_U10_C1	El usuario deberá ser capaz de modificar si un espacio (DD5) es reservable. 
RF_U11_C2	El usuario deberá ser capaz de modificar la categoría de reserva (DD7) de un espacio (DD5).

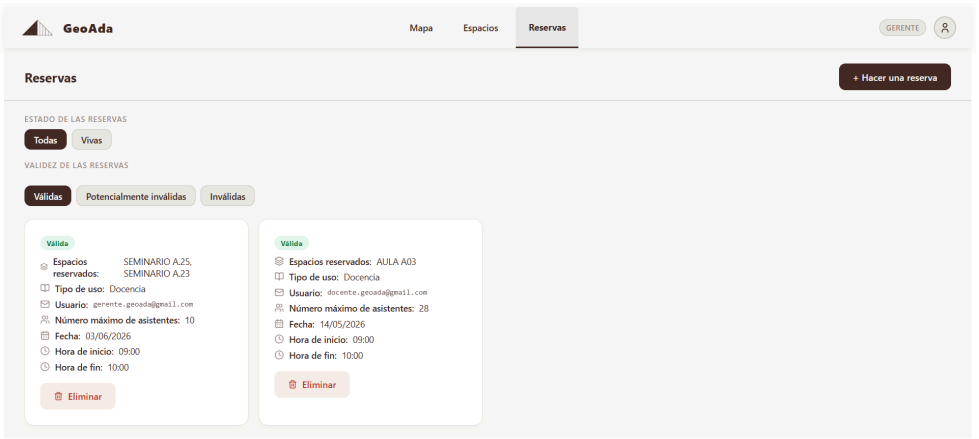
	
RF_U12_C3	El usuario deberá ser capaz de modificar los horarios disponibles de un espacio (DD5). 
RF_U13_C4	El usuario deberá ser capaz de modificar la asignación de un espacio (DD5).

	
RF_U14_C5_O1	El usuario deberá ser capaz de modificar el porcentaje de uso máximo de un espacio (DD5). 

	
RF_U15_D	El usuario deberá ser capaz de buscar espacios (DD5). 
RF_U16_E_F_O3	El usuario deberá ser capaz de hacer una reserva (DD8).

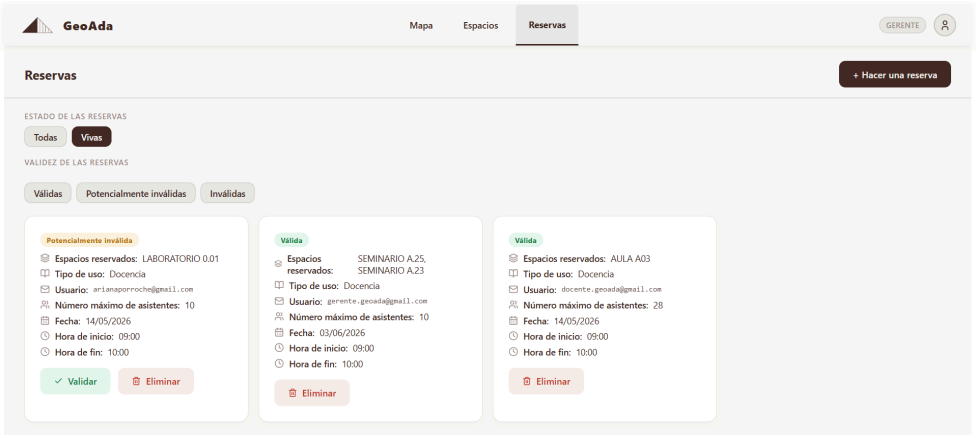
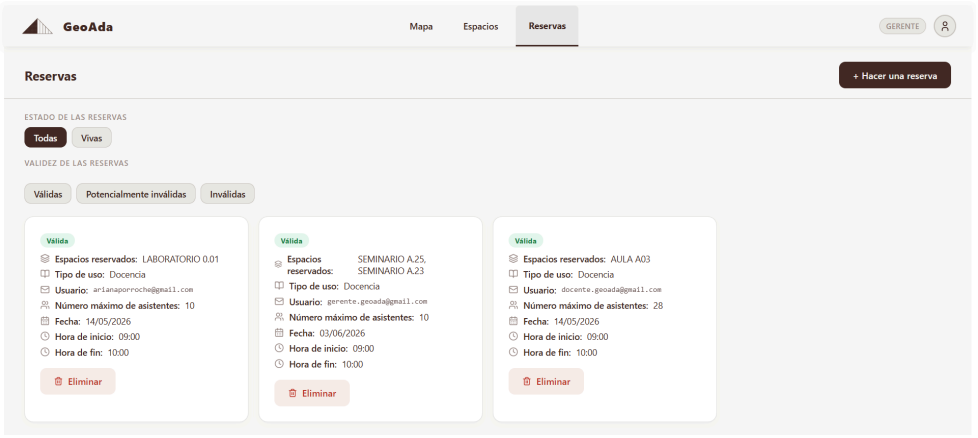
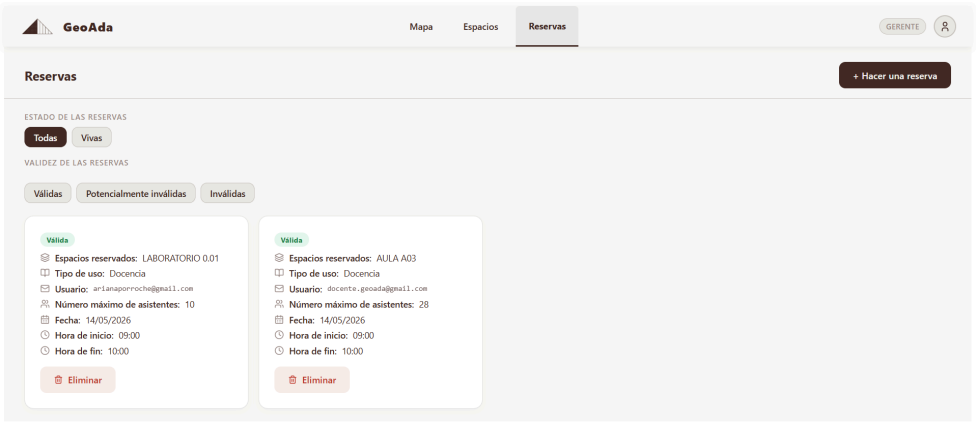
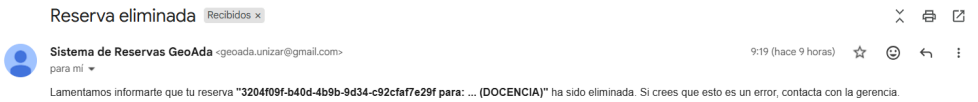


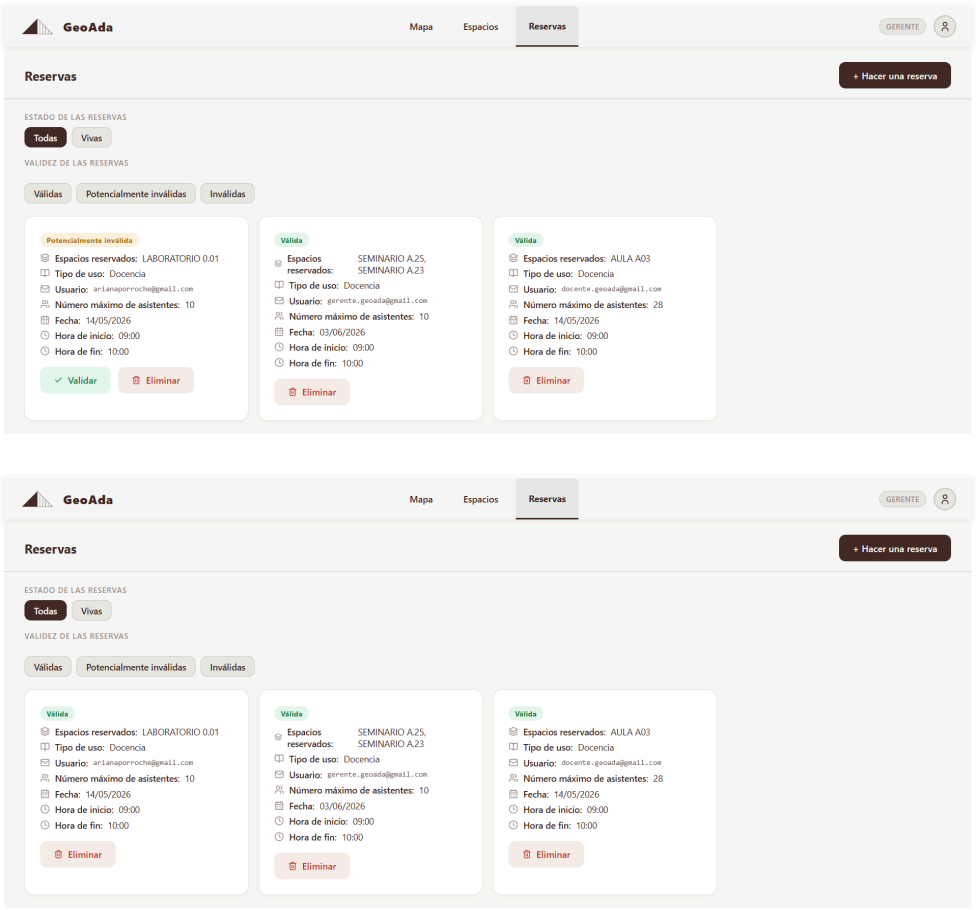
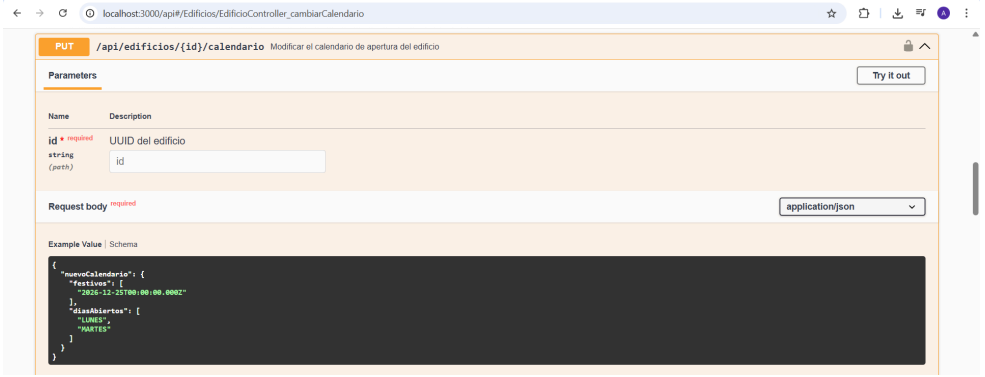


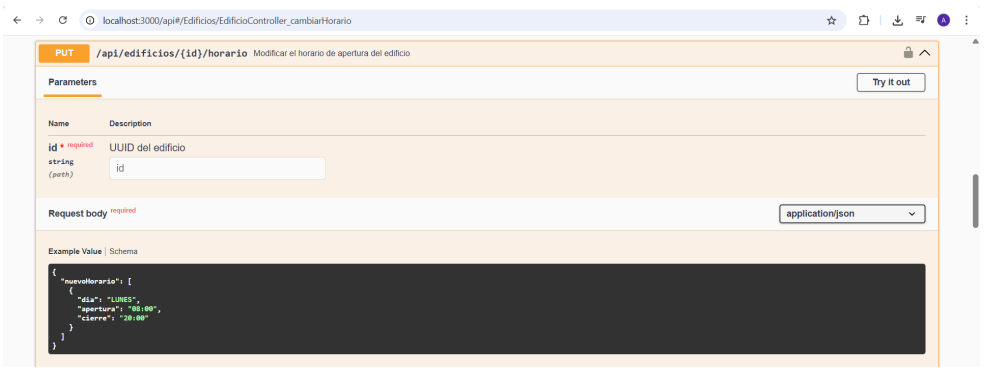
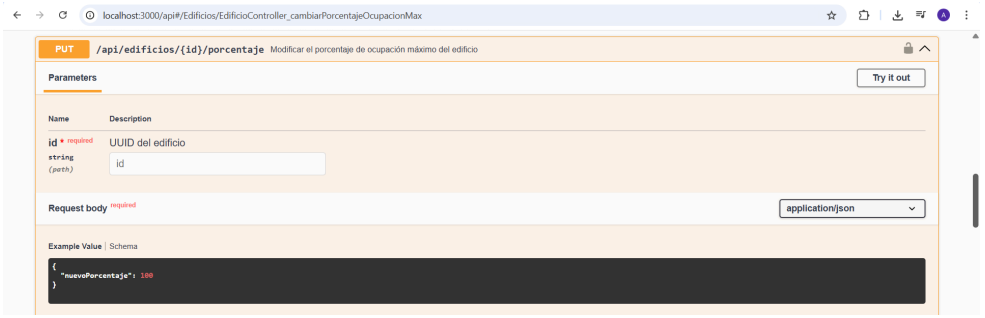


Si la reserva creada fuese potencialmente inválida, al usuario le llegaría esta notificación:

RF_U17_H	El usuario deberá ser capaz de consultar las reservas vivas (DD10).

	
RF_U18_H	El usuario deberá ser capaz de eliminar una reserva (DD8).   Cuando la reserva de un usuario es eliminada, le llega esta notificación: 
RF_U19_H_O4	El usuario deberá ser capaz de modificar la validez (DD11) de una reserva (DD8) de “potencialmente inválida” a “válida”.

	
RF_U20	El usuario deberá ser capaz de modificar el calendario de apertura del edificio (DD12). 
RF_U21	El usuario deberá ser capaz de modificar el horario de apertura del edificio (DD12).

	
RF_U22	El usuario deberá ser capaz de modificar el porcentaje de ocupación máxima del edificio (DD12). 

3.3.2. API

La documentación completa de la API se encuentra disponible mediante Swagger en la dirección http://localhost:3000/api, donde se pueden consultar todos los endpoints expuestos por el sistema, así como sus respectivos esquemas de entrada y salida.

La API del sistema se organiza en seis módulos diferenciados, cada uno correspondiente a un agregado o funcionalidad del dominio: autenticación, departamentos, personas, espacios, edificios y reservas. Todos los endpoints, a excepción de los de autenticación, están protegidos mediante autenticación por token, por lo que es necesario iniciar sesión previamente para poder operar con ellos.

A continuación se describe cada uno de los módulos, detallando las operaciones disponibles y su propósito dentro del sistema.

3.3.2.1. Gestión de departamentos

Figura 34: Documentación de /api/departamentos

El agregado Departamento expone una API con dos operaciones básicas que permiten gestionar los departamentos del sistema. Dado que los departamentos son un conjunto cerrado y predefinido en el dominio, estas operaciones se limitan a la creación y consulta de los mismos.

- *POST /api/departamentos*: permite crear un nuevo departamento en el sistema, recibiendo como cuerpo de la petición el nombre del departamento.
- *GET /api/departamentos*: permite obtener el listado completo de departamentos registrados en el sistema.

A continuación, se muestran los ejemplos de respuesta correspondientes a la ejecución de cada endpoint, tanto respuestas de éxito como de error.

Figura 35: Respuestas del endpoint POST /api/departamentos

Figura 36: Respuestas del endpoint GET /api/departamentos

3.3.2.2. Gestión de personas

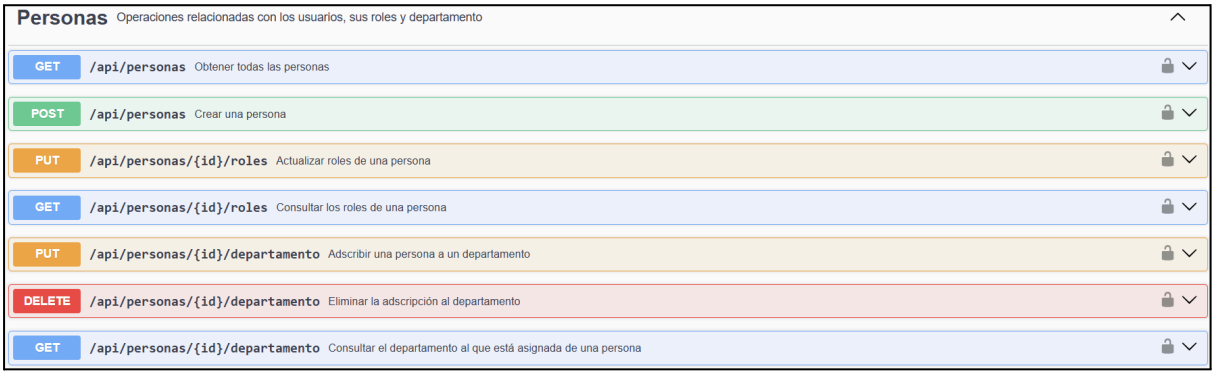


Figura 37: Documentación de /api/personas

El agregado Persona es el núcleo central del sistema, ya que gestiona los usuarios, sus roles y su adscripción a departamentos. Su API expone siete operaciones que cubren todos los casos de uso relacionados con la gestión de personas.

- *GET /api/personas*: permite obtener el listado completo de personas registradas en el sistema.
- *POST /api/personas*: permite crear una nueva persona en el sistema, proporcionando su correo, contraseña, rol y, opcionalmente, el departamento al que pertenece.
- *PUT /api/personas/{id}/roles*: permite actualizar los roles asignados a una persona específica, aplicando las reglas de transición de roles definidas en el dominio.
- *GET /api/personas/{id}/roles*: permite consultar los roles actualmente asignados a una persona.
- *PUT /api/personas/{id}/departamento*: permite adscribir una persona a un departamento, siempre que su rol lo permita según las reglas del sistema.
- *DELETE /api/personas/{id}/departamento*: permite eliminar la adscripción de una persona a su departamento actual.
- *GET /api/personas/{id}/departamento*: permite consultar el departamento al que está adscrita una persona, en caso de existir.

Estos endpoints están documentados y probados mediante Swagger, facilitando su exploración, validación y prueba durante el desarrollo.

A continuación, se muestran los ejemplos de respuesta correspondientes a la ejecución de cada endpoint, tanto respuestas de éxito como de error.

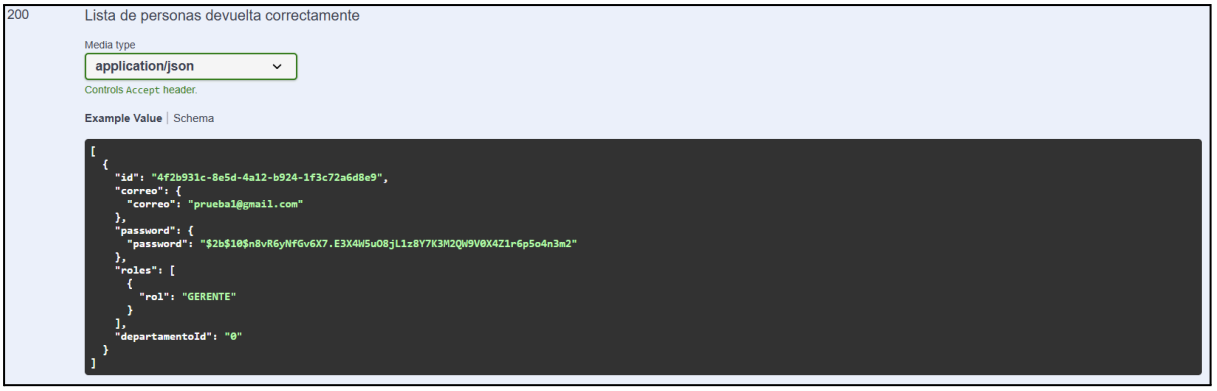


Figura 38: Respuestas del endpoint GET /personas

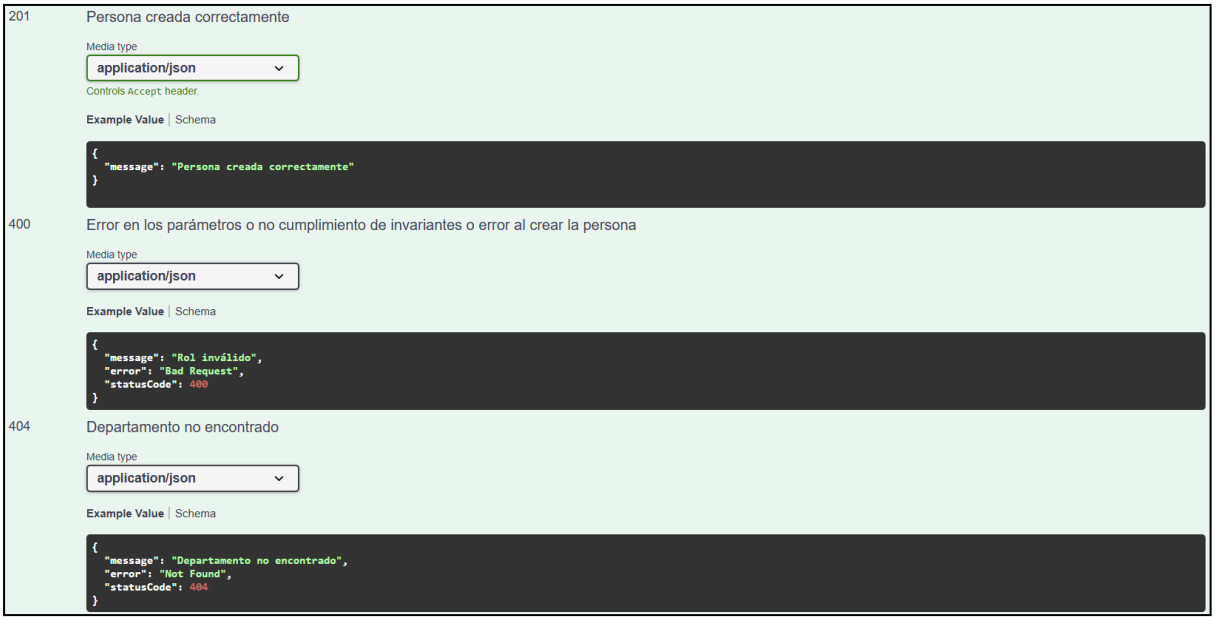


Figura 39: Respuestas del endpoint POST /personas

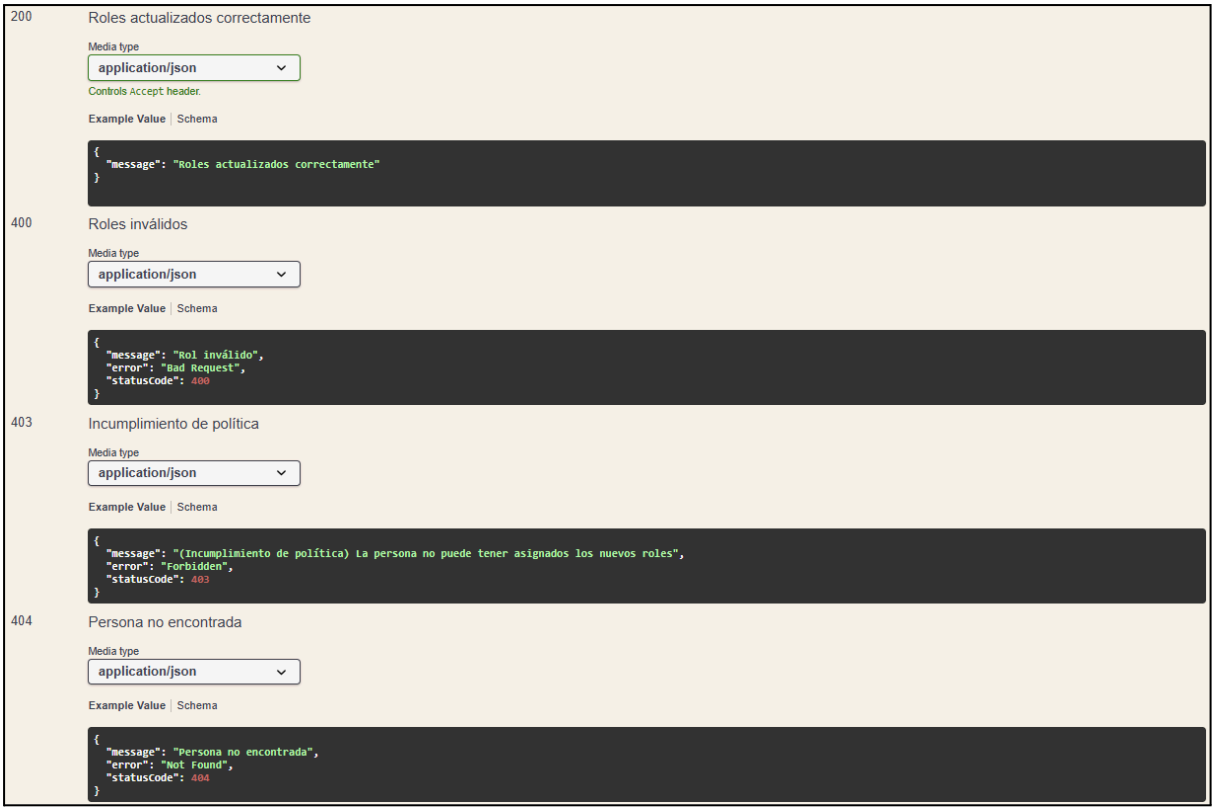


Figura 40: Respuestas del endpoint PUT /personas/{id}/roles

Figura 41: Respuestas del endpoint GET /personas/{id}/roles

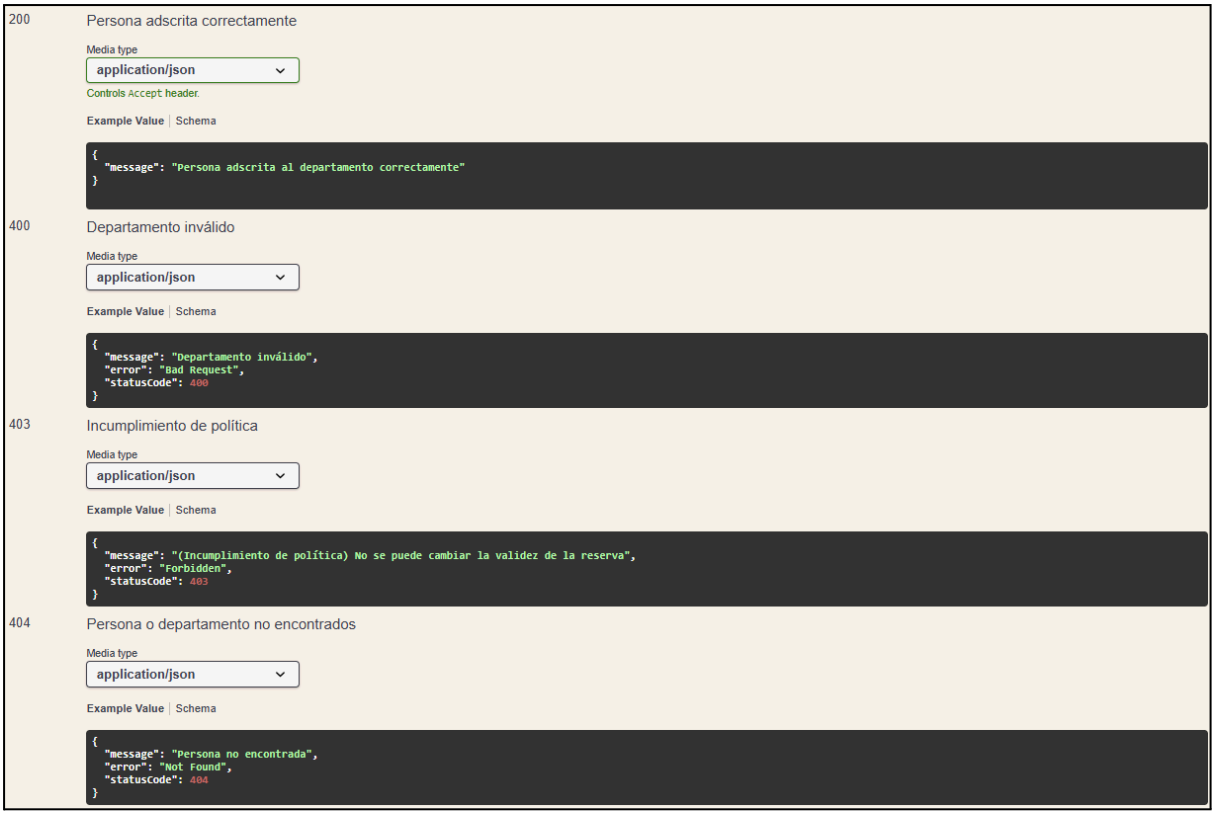


Figura 42: Respuestas del endpoint PUT /personas/{id}/departamento

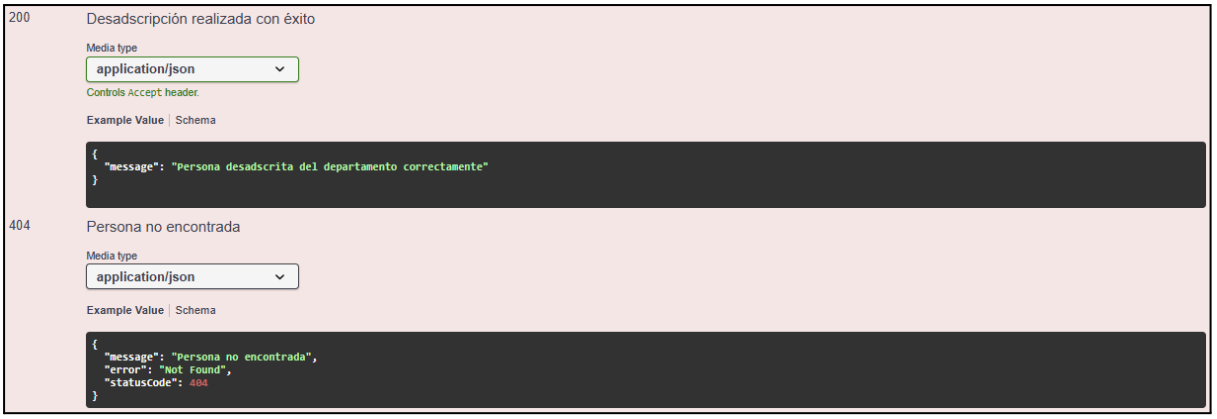


Figura 43: Respuestas del endpoint DELETE /personas/{id}/departamento

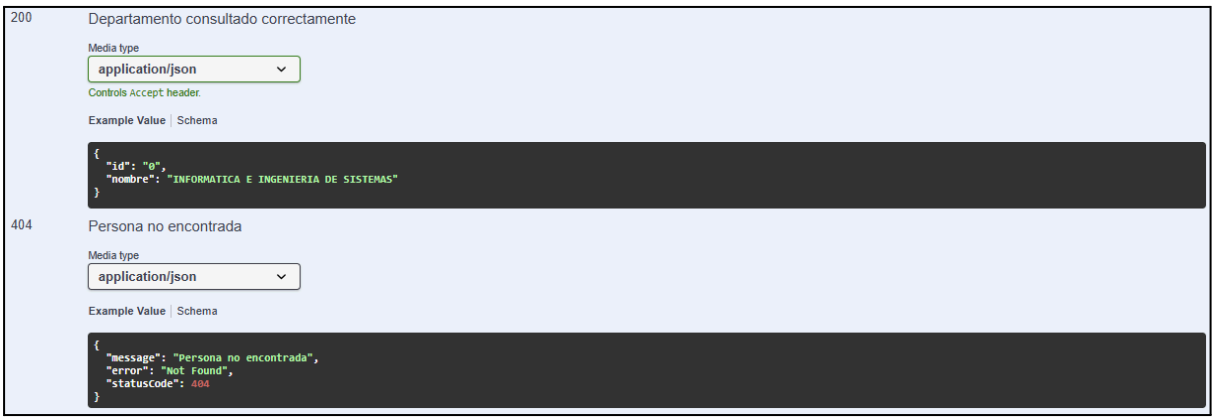


Figura 44: Respuestas del endpoint GET /personas/{id}/departamento

3.3.2.3. Gestión de espacios

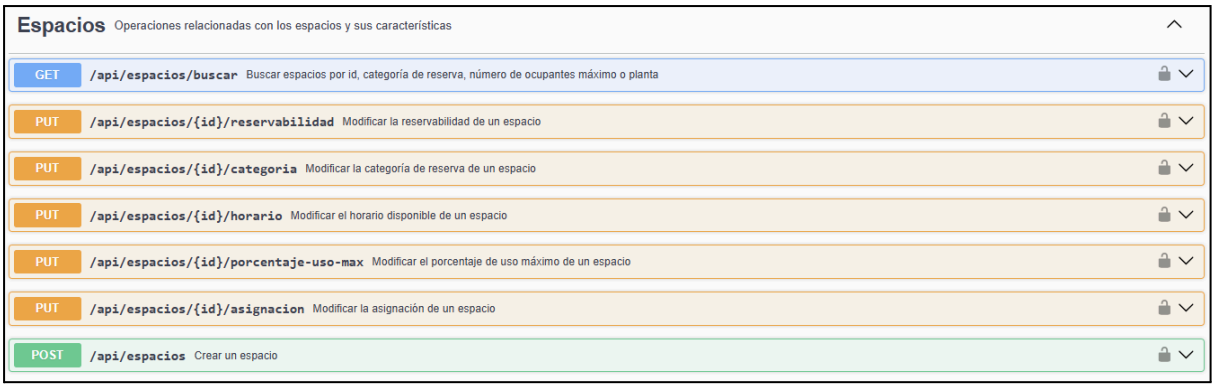


Figura 45: Documentación de /api/espacios

El agregado Espacio expone una API que permite gestionar los distintos espacios del edificio y sus características. Sus operaciones cubren tanto la búsqueda y creación de espacios como la modificación de sus propiedades, todas ellas reservadas al rol "gerente" a excepción de la búsqueda.

- *GET /api/espacios/buscar*: permite buscar espacios filtrando por identificador, categoría de reserva, número máximo de ocupantes o planta.
- *PUT /api/espacios/{id}/reservabilidad:* permite modificar si un espacio es reservable o no.
- *PUT /api/espacios/{id}/categoria*: permite modificar la categoría de reserva de un espacio, respetando las restricciones de compatibilidad con su tipo.
- *PUT /api/espacios/{id}/horario*: permite modificar el horario disponible de un espacio, pudiendo diferir del horario general del edificio.
- *PUT /api/espacios/{id}/porcentaje-uso-max*: permite modificar el porcentaje máximo de uso de un espacio, que determina la capacidad efectiva disponible para reservas.
- *PUT /api/espacios/{id}/asignacion*: permite modificar la entidad a la que está asignado un espacio, ya sea un departamento, una persona o el propio edificio.
- *POST /api/espacios*: permite crear un nuevo espacio en el sistema con todas sus características iniciales.

Estos endpoints están documentados y probados mediante Swagger, facilitando su exploración, validación y prueba durante el desarrollo.

A continuación, se muestran los ejemplos de respuesta correspondientes a la ejecución de cada endpoint, tanto respuestas de éxito como de error.

Figura 46: Respuestas del endpoint GET /api/espacios/buscar

Figura 47: Respuestas del endpoint PUT /api/espacios/{id}/reservabilidad

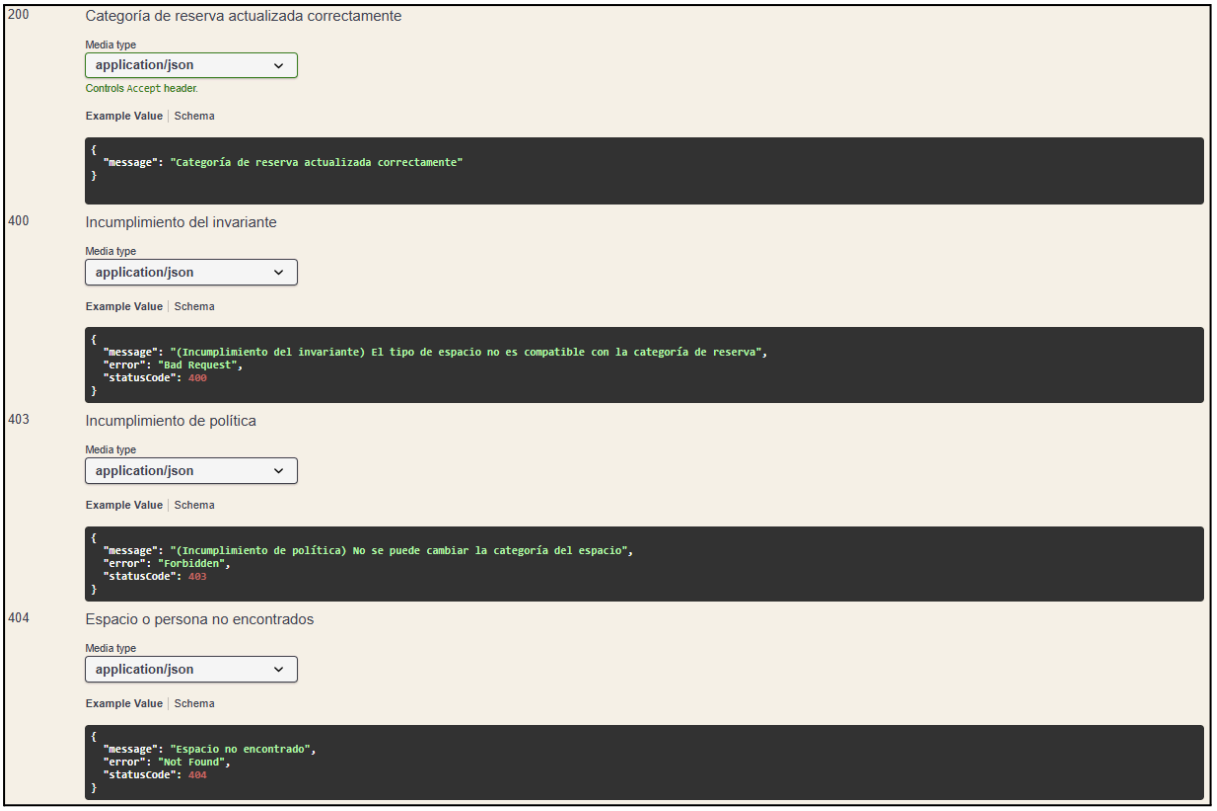


Figura 48: Respuestas del endpoint PUT /api/espacios/{id}/categoria

Figura 49: Respuestas del endpoint PUT /api/espacios/{id}/horario

Figura 50: Respuestas del endpoint PUT /api/espacios/{id}/porcentaje-uso-max

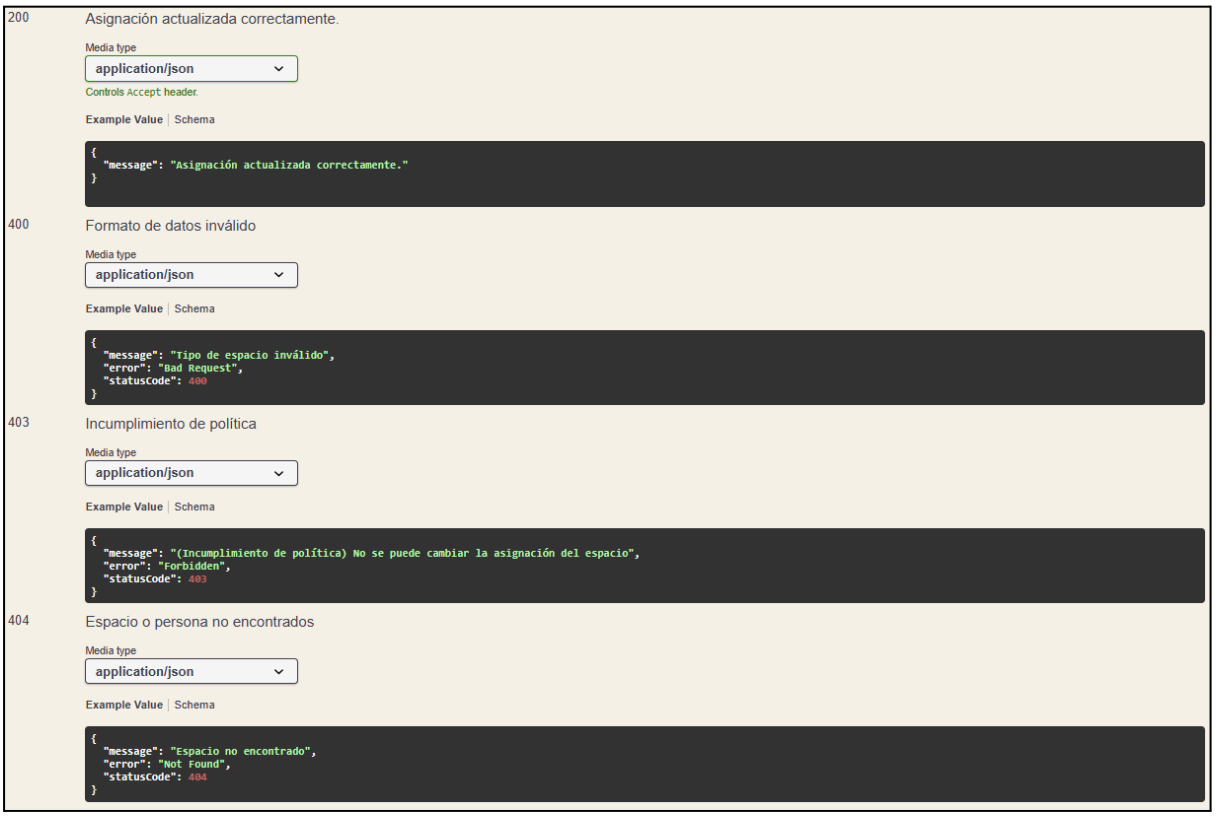


Figura 51: Respuestas del endpoint PUT /api/espacios/{id}/asignacion

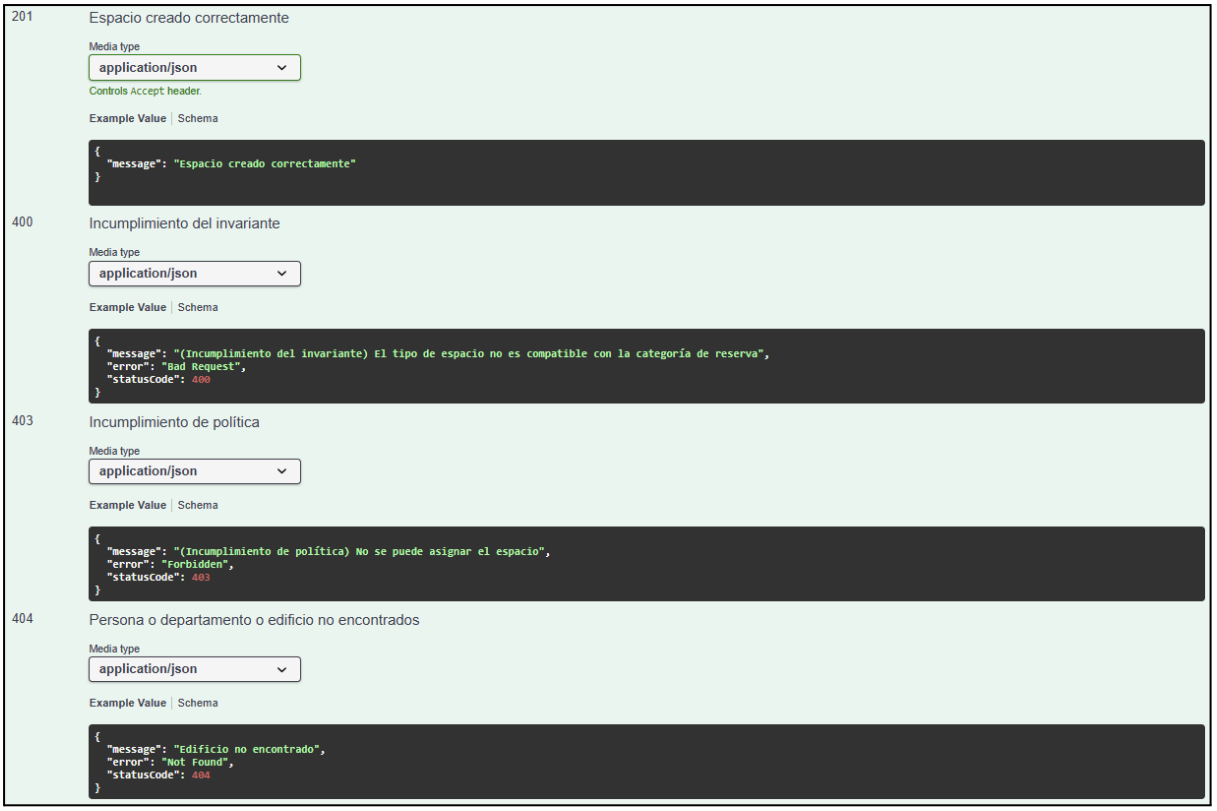


Figura 52: Respuestas del endpoint POST /api/espacios

3.3.2.4. Gestión de edificios

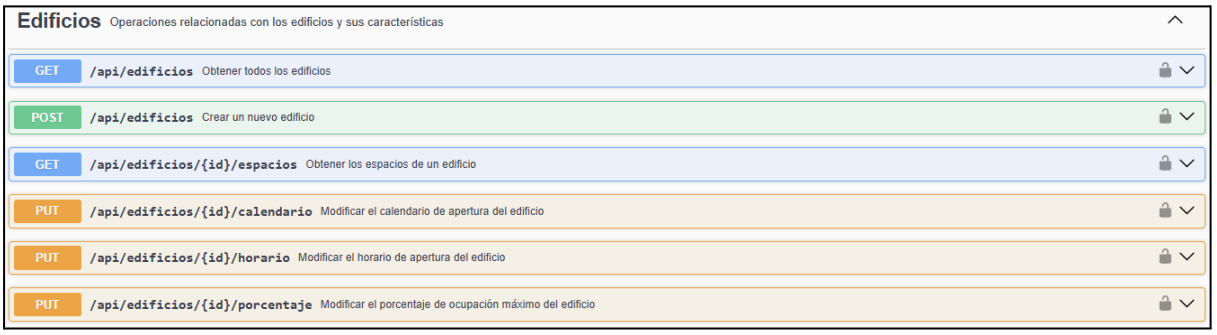


Figura 53: Documentación de /api/edificios

El agregado Edificio expone una API que permite gestionar los edificios del sistema y sus propiedades operativas. Sus operaciones abarcan desde la consulta y creación de edificios hasta la modificación de su configuración de apertura y ocupación, siendo estas últimas accesibles únicamente mediante la API.

- *GET /api/edificios*: permite obtener el listado completo de edificios registrados en el sistema.
- *POST /api/edificios*: permite crear un nuevo edificio en el sistema con sus características iniciales.
- *GET /api/edificios/{id}/espacios*: permite obtener todos los espacios pertenecientes a un edificio concreto.

- *PUT /api/edificios/{id}/calendario*: permite modificar el calendario de apertura del edificio, indicando los días de la semana en que permanece abierto y los festivos en que permanece cerrado.
- *PUT /api/edificios/{id}/horario*: permite modificar el horario de apertura del edificio, estableciendo la hora de apertura y cierre para cada día de la semana.
- *PUT /api/edificios/{id}/porcentaje*: permite modificar el porcentaje de ocupación máxima del edificio, que actúa como valor por defecto para los espacios que no tengan uno propio definido.

Estos endpoints están documentados y probados mediante Swagger, facilitando su exploración, validación y prueba durante el desarrollo.

A continuación, se muestran los ejemplos de respuesta correspondientes a la ejecución de cada endpoint, tanto respuestas de éxito como de error.

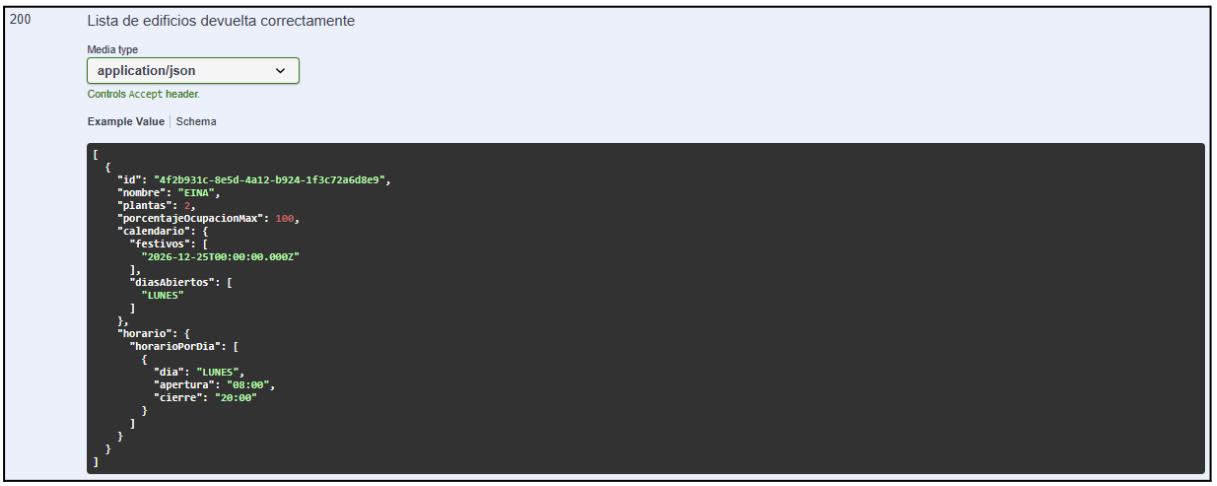


Figura 54: Respuestas del endpoint GET /api/edificios

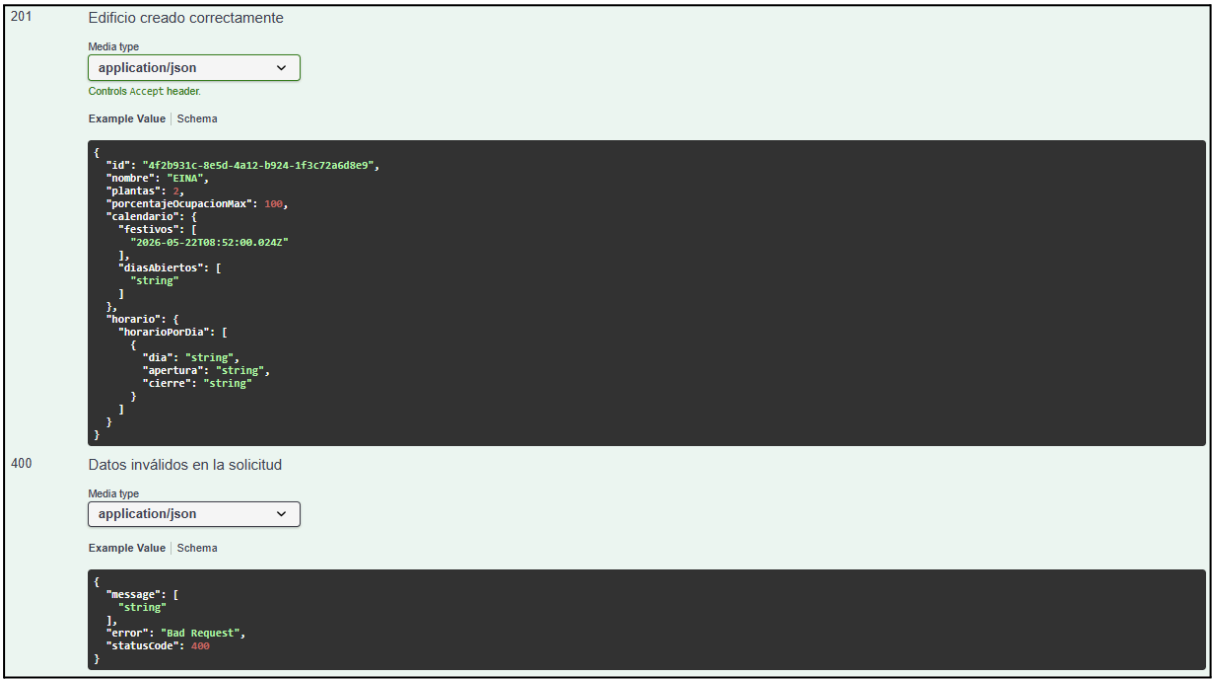


Figura 55: Respuestas del endpoint POST /api/edificios

Figura 56: Respuestas del endpoint GET /api/edificios/{id}/espacios

Figura 57: Respuestas del endpoint PUT /api/edificios/{id}/calendario

Figura 58: Respuestas del endpoint PUT /api/edificios/{id}/horario

Figura 59: Respuestas del endpoint PUT /api/edificios/{id}/porcentaje

3.3.2.5. Gestión de reservas

Figura 60: Documentación de /api/reservas

El agregado Reserva expone una API que gestiona el ciclo de vida completo de las reservas del sistema. Sus operaciones permiten crear, consultar, eliminar y modificar reservas, con ciertas operaciones restringidas exclusivamente al rol "gerente".

- *GET /api/reservas*: permite obtener el listado completo de reservas registradas en el sistema.
- *POST /api/reservas*: permite crear una nueva reserva, validando automáticamente que se cumplan todas las reglas del dominio antes de registrarla.
- *GET /api/reservas/vivas*: permite obtener todas las reservas vivas, es decir, aquellas cuya fecha y hora de inicio son posteriores al momento actual. Esta operación está restringida al rol "gerente".
- *GET /api/reservas/persona/{id}*: permite obtener todas las reservas asociadas a una persona concreta.
- *DELETE /api/reservas/{id}*: permite eliminar una reserva existente. Esta operación está restringida al rol "gerente", quien además deberá notificar automáticamente al usuario afectado.

- *PUT /api/reservas/{id}/validez*: permite modificar la validez de una reserva, concretamente para cambiar su estado de "potencialmente inválida" a "válida". Esta operación está restringida al rol "gerente".

Estos endpoints están documentados y probados mediante Swagger, facilitando su exploración, validación y prueba durante el desarrollo.

A continuación, se muestran los ejemplos de respuesta correspondientes a la ejecución de cada endpoint, tanto respuestas de éxito como de error.

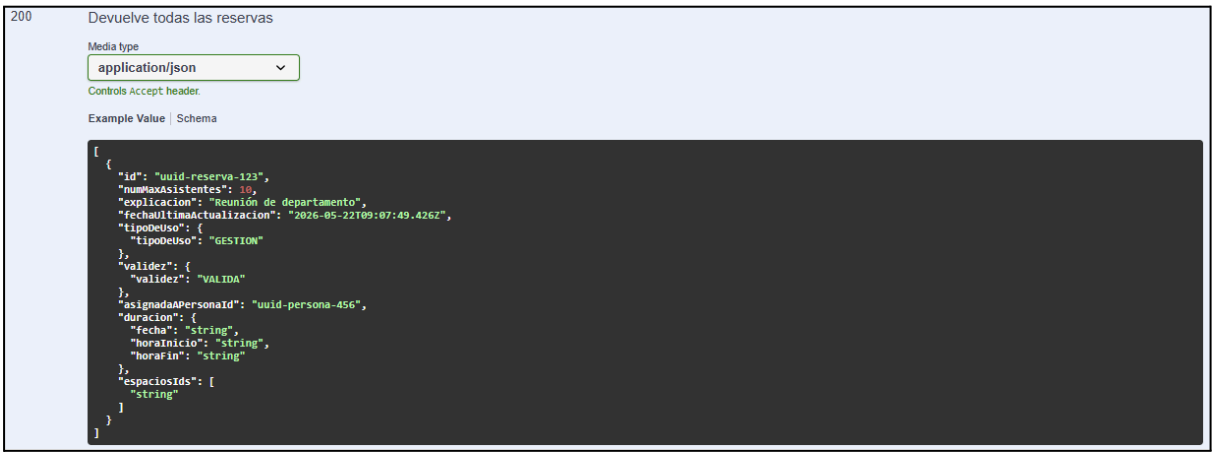


Figura 61: Respuestas del endpoint GET /api/reservas

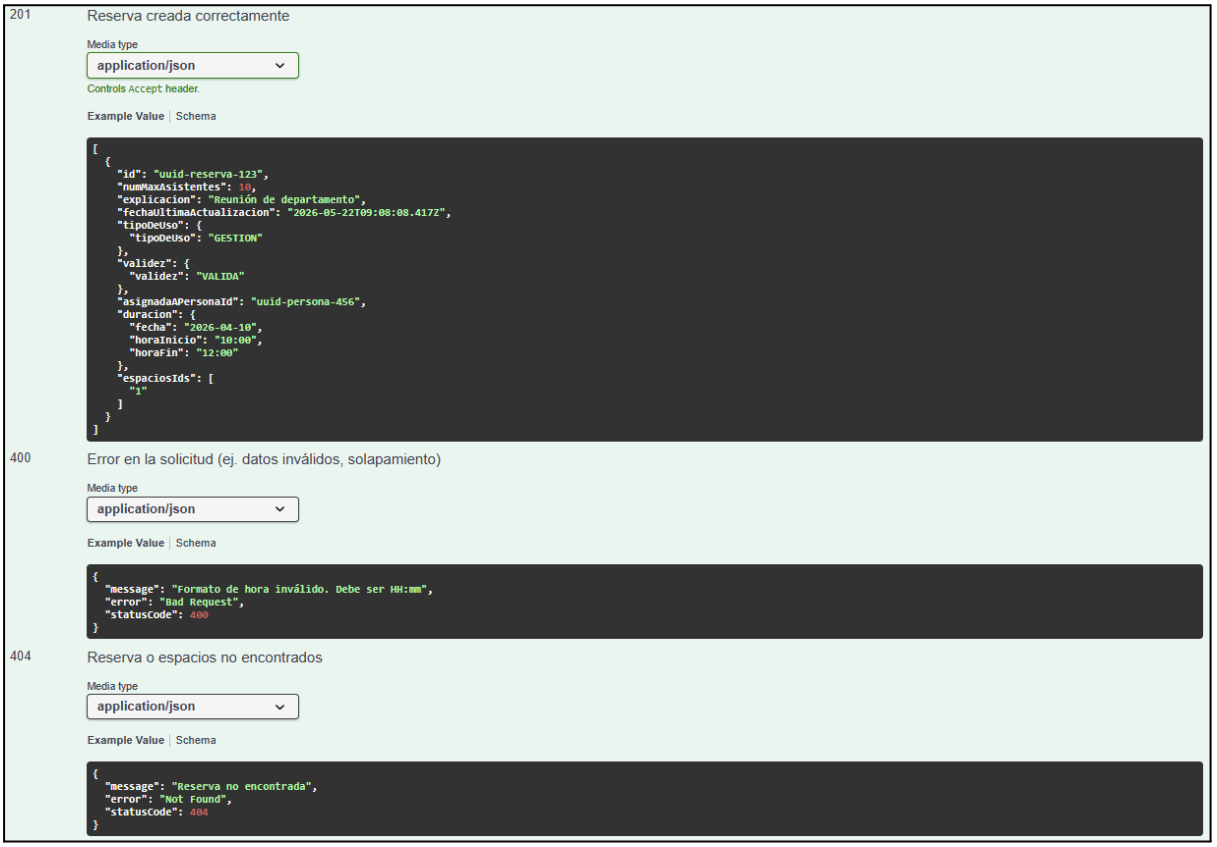


Figura 62: Respuestas del endpoint POST /api/reservas

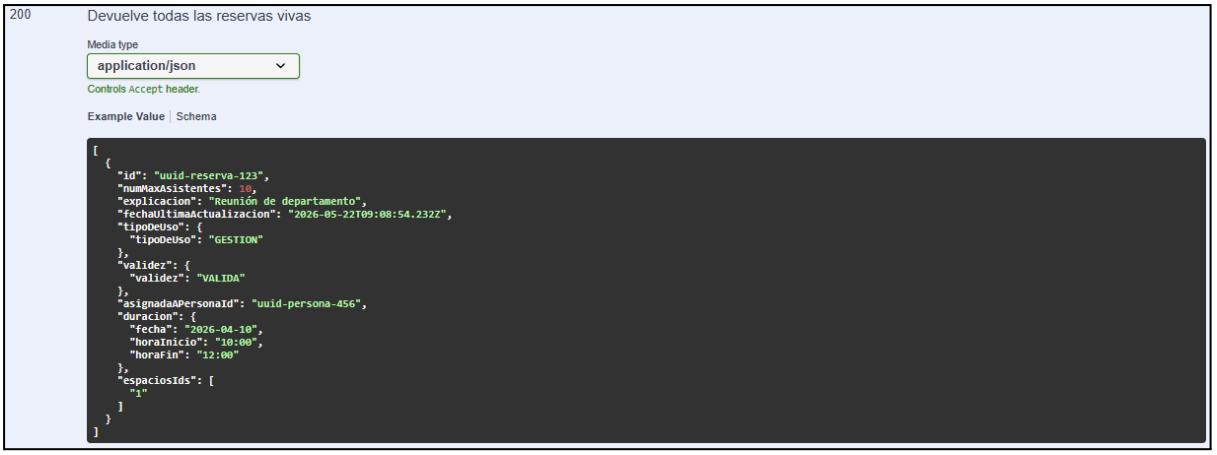


Figura 63: Respuestas del endpoint GET /api/reservas/vivas

Figura 64: Respuestas del endpoint GET /api/reservas/persona/{id}

Figura 65: Respuestas del endpoint DELETE /api/reservas/{id}

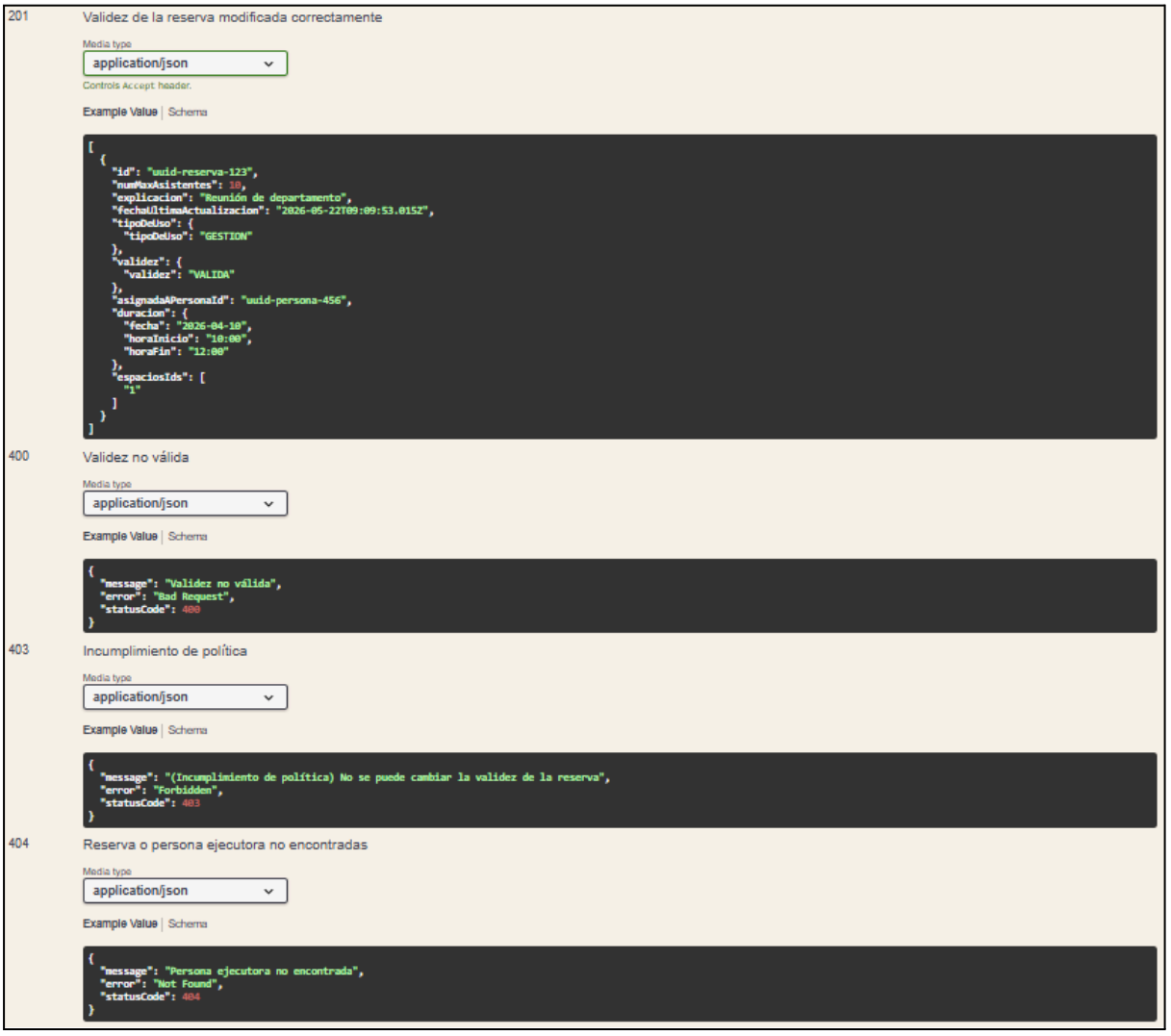


Figura 66: Respuestas del endpoint PUT /api/reservas/{id}/validez

3.3.2.6. Gestión de autenticación

Figura 67: Documentación de /api/auth

El módulo de autenticación expone una API que gestiona el acceso al sistema. Sus operaciones permiten iniciar y cerrar sesión, así como consultar la información del usuario autenticado en cada momento. La autenticación se realiza mediante correo electrónico y contraseña, y el sistema utiliza tokens para mantener la sesión activa.

- *POST /api/auth/login*: permite iniciar sesión en el sistema proporcionando el correo electrónico y la contraseña del usuario. Si las credenciales son correctas, el sistema devuelve un token de autenticación.
- *GET /api/auth/me*: permite obtener la información del usuario actualmente autenticado a partir de su token de sesión.
- *POST /api/auth/logout*: permite cerrar la sesión del usuario actual, invalidando el token de autenticación.

Estos endpoints están documentados y probados mediante Swagger, facilitando su exploración, validación y prueba durante el desarrollo.

A continuación, se muestran los ejemplos de respuesta correspondientes a la ejecución de cada endpoint, tanto respuestas de éxito como de error.

Figura 68: Respuestas del endpoint POST /api/auth/login

Figura 69: Respuestas del endpoint GET /api/auth/me

Figura 70: Respuestas del endpoint POST /api/auth/logout

3.4. TESTING

Para la validación del sistema se han implementado tests automatizados utilizando Jest como framework de testing. La cobertura global obtenida es del 63.99% de sentencias, 52.88% de ramas, 67.5% de funciones y 63.84% de líneas.

File	% Stmts	% Branch	% Funcs	% Lines
All files	63.99	52.88	67.5	63.84

Figura 71: Cobertura global de tests

Los esfuerzos de testing se han concentrado en las capas de dominio y aplicación, que son las que contienen la lógica de negocio del sistema.

En la capa de dominio se ha alcanzado la cobertura más alta. Los modelos principales (Departamento, Edificio, Persona y Reserva) obtienen un 100% en todas las métricas, mientras que Espacio se sitúa muy cerca con un 98.16% de sentencias. Los objetos de valor alcanzan igualmente el 100% en su totalidad, lo que garantiza la correcta validación de precondiciones e invariantes. Las políticas de negocio, tanto las del agregado Espacio como las de Persona y Reserva, presentan también una cobertura del 100% en sentencias y funciones, asegurando que todas las reglas de dominio se comportan correctamente.

En la capa de aplicación, los servicios principales presentan una cobertura media superior al 95% en sentencias. Destacan ServiciosAuth y ServiciosDepartamento, que alcanzan el 100%, seguidos de cerca por ServiciosPersona con un 98.49%, ServiciosEdificio con un 99.05% y ServiciosEspacio con un 96.84%. Estos resultados reflejan que prácticamente la totalidad de los casos de uso del sistema han sido validados mediante tests.

Por último, los controladores del servidor web también presentan una cobertura muy elevada, con una media cercana al 98% en sentencias, lo que confirma que el comportamiento de la API ante distintas situaciones ha sido correctamente verificado.

4. CONCLUSIONES

El proyecto GeoAda ha concluido satisfactoriamente con el desarrollo completo de un sistema de gestión y reserva de espacios para el edificio Ada Byron de la Universidad de Zaragoza. A lo largo del proyecto se han abordado todas las fases del ciclo de desarrollo software, desde la especificación de requisitos hasta la implementación, pruebas y despliegue.

En cuanto al modelo de dominio, se han identificado e implementado 5 entidades: Persona, Departamento, Edificio, Espacio y Reserva, cada una actuando como raíz de su propio agregado. El modelo incorpora 12 objetos de valor y 11 políticas de negocio que garantizan la consistencia y coherencia del sistema en todo momento.

Respecto a la arquitectura, el sistema sigue un estilo de arquitectura hexagonal organizado en tres capas (infraestructura, aplicación y dominio), con una vista de componentes y conectores de tipo N-Tier con cuatro niveles. La comunicación asíncrona entre el servidor web y el servidor de aplicación se realiza mediante RabbitMQ, y la persistencia se gestiona con PostgreSQL y PostGIS a través del ORM Prisma.

En lo referente a la implementación, se han desarrollado todos los requisitos funcionales establecidos, tanto obligatorios como opcionales, cubriendo la gestión de personas, roles y departamentos, gestión de espacios y edificios, y el ciclo de vida completo de las reservas. La interfaz en React permite usar todas las funcionalidades del sistema, incluido el mapa interactivo del edificio. La API REST, documentada con Swagger, expone seis módulos diferenciados y está protegida mediante autenticación por token.

En cuanto al testing, se han implementado tests automatizados con Jest, alcanzando una cobertura global del 63.99% de sentencias. Las capas de dominio y aplicación presentan los niveles más altos de cobertura, con un 100% en los modelos principales, objetos de valor y políticas de negocio, y una cobertura media superior al 95% en los servicios de aplicación.

Como valoración global, el equipo considera que se han cumplido los objetivos planteados, logrando un sistema funcional, bien estructurado y correctamente documentado, que da respuesta a las necesidades de gestión de espacios del edificio Ada Byron.

A continuación, se detalla el **desglose de horas por persona y tarea**:

#	Descripción	Tiempo Ariana	Tiempo Iván	Tiempo Total
1	Especificación de los requisitos	8:00	8:05	16:05
2	Modelo de dominio	3:55	1:40	5:35
3	Implementación Backend	79:25	36:22	115:47
4	Implementación Frontend	4:13	33:20	37:33
5	Documentación	18:30	9:03	27:33
TOTAL		**114:03**	**88:30**	**202:33**

ANEXOS

ANEXO I: BOCETOS GUI

Figura 72: Pantalla Iniciar sesión

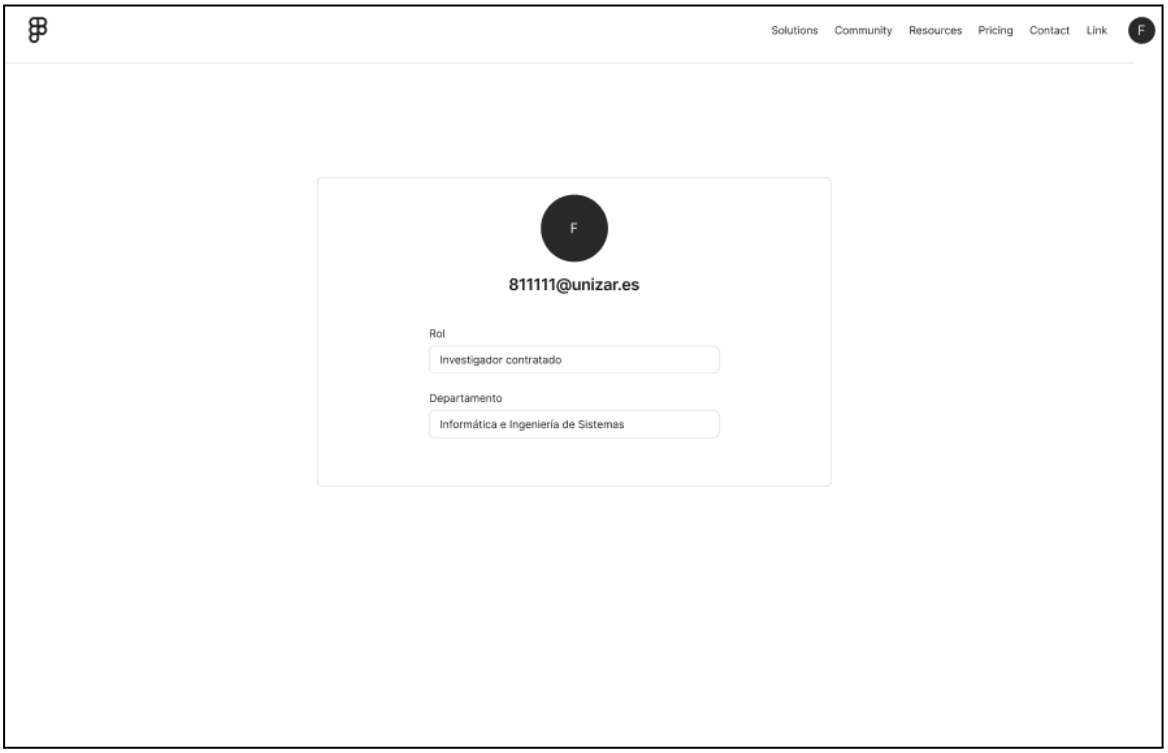


Figura 73: Pantalla Perfil del usuario

Figura 74: Pantalla Mapa del edificio

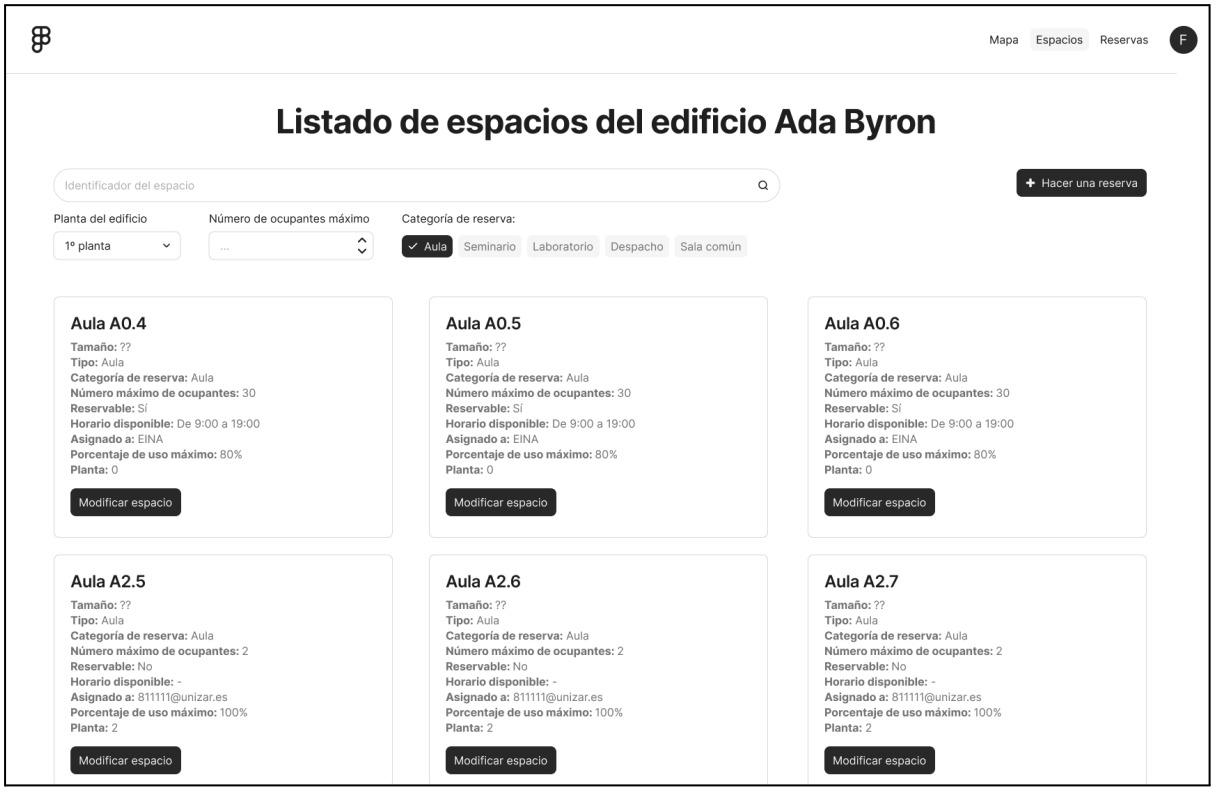


Figura 75: Pantalla Listado de espacios con filtros

Figura 76: Pantalla Modificar espacio

Figura 77: Pantalla Hacer una reserva (Paso 1)

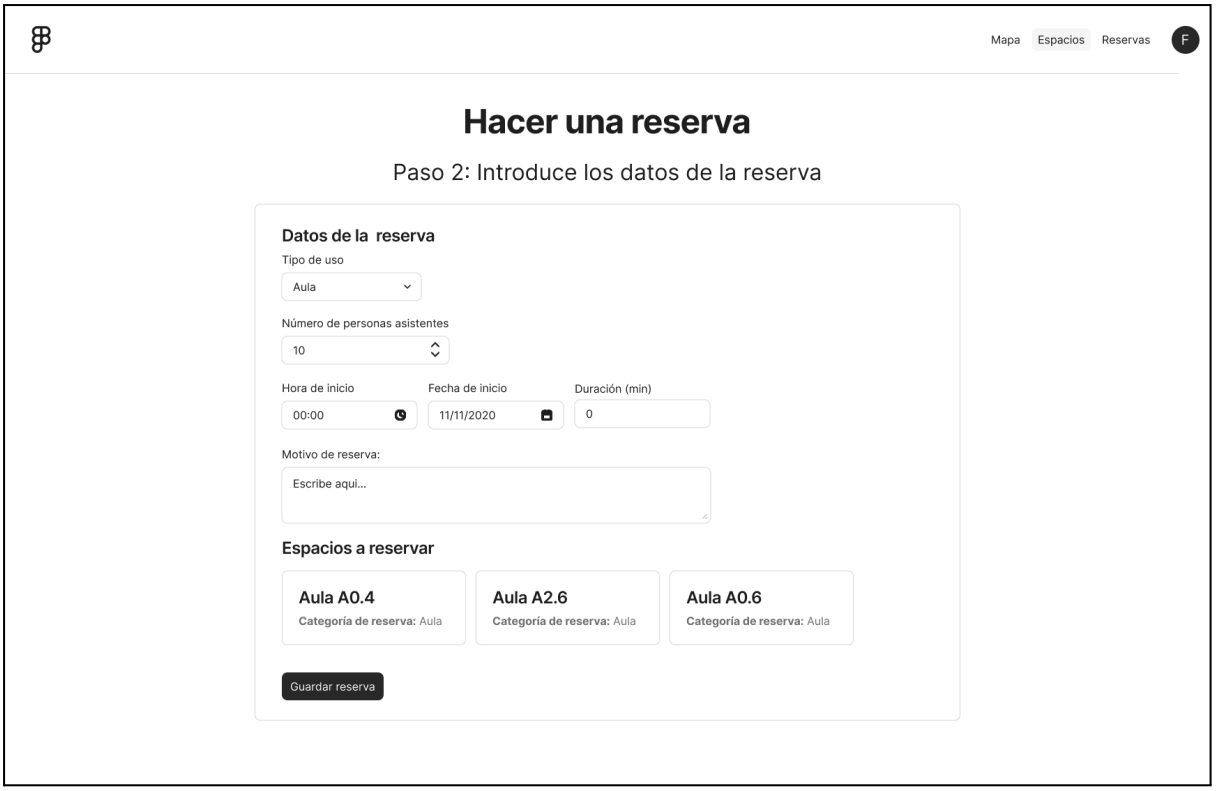


Figura 78: Pantalla Hacer una reserva (Paso 2)

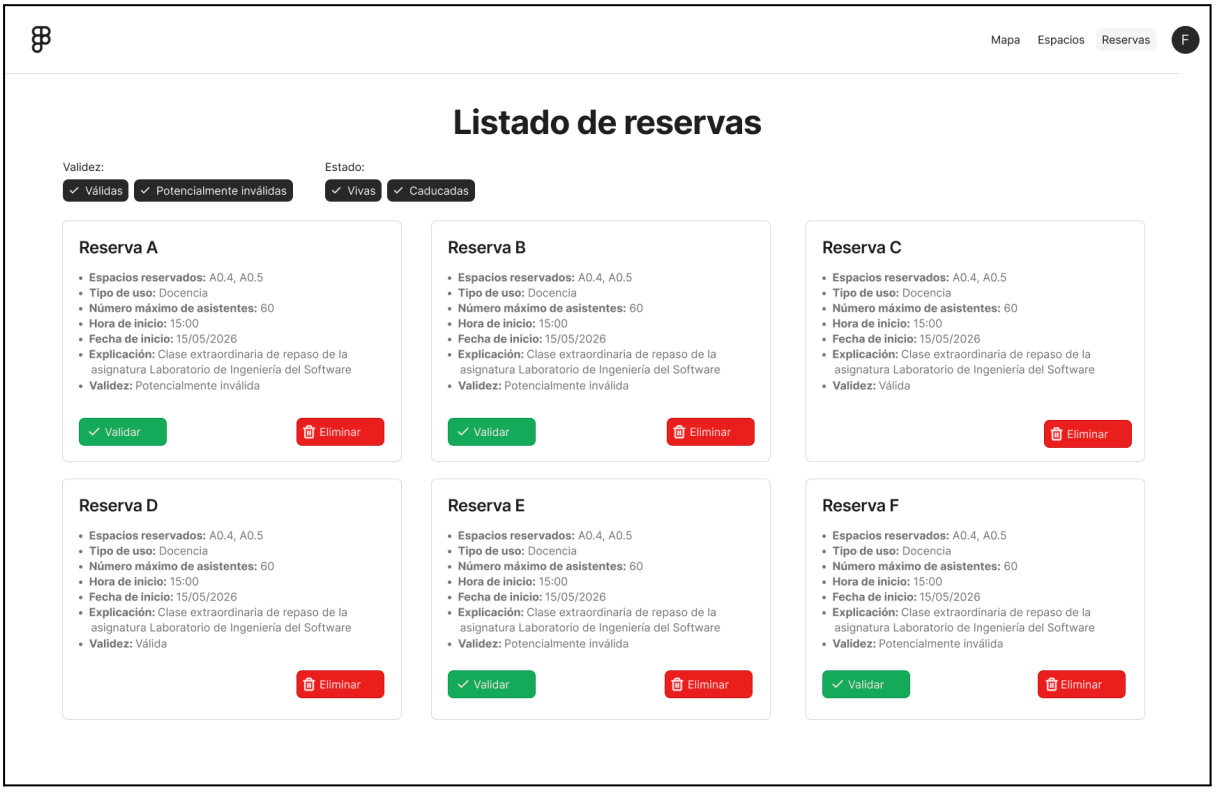


Figura 79: Pantalla Listado de reservas