INFORME – Sistema de Gestión de Reservas de Salas de Estudio

Base de Datos 1 – Universidad Católica del Uruguay (UCU)
Segundo Semestre 2025
Trabajo Obligatorio – Gestión de Salas de Estudio
Integrantes: Esteban Durán, Sebastián Martony y Joaquín Viola

1. Introducción

El presente informe documenta el desarrollo completo del sistema solicitado por la consigna del trabajo obligatorio “Sistema para Gestión de Reserva de Salas de Estudio” para la UCU. Este proyecto abarcó:

- Diseño y construcción de la base de datos MySQL.
- Implementación de un backend en Python (FastAPI) sin ORM.
- Creación de un frontend en React + Vite + Tailwind.
- Dockerización de la aplicación.
- Implementación de todas las reglas de negocio descritas en la consigna.
- Validaciones y controles de seguridad.

El objetivo fue desarrollar un sistema que permita administrar salas de estudio de manera centralizada, registrando reservas, participantes, asistencia y sanciones, y mantener trazabilidad completa del uso de las salas.

2. Descripción General del Sistema

El sistema implementa:

Gestión completa (ABM) de:

- Participantes
- Salas
- Turnos
- Reservas

- Sanciones

Procesos automáticos:

- Registro de asistencia
- Generación automática de sanciones cuando una reserva finaliza sin asistencia
- Validación de restricciones (horas permitidas, tipos de salas, límite de horas diarias, límite semanal, etc.)

Seguridad:

- Contraseñas hasheadas con bcrypt
- Validación de datos en las distintas capas
- Restricciones en la base de datos
- Dockerización evitando exponer puertos innecesarios

Consultas para reportes (BI):

- Salas más reservadas
- Turnos más usados
- Asistencias
- Reservas por facultad/programa
- Entre otras

Estas funcionalidades abarcan todos los puntos obligatorios definidos en la consigna del curso.

3. Arquitectura General

La solución se estructuró en **tres capas**:

Frontend (React + Vite + Tailwind)
|
v
Backend (FastAPI – Python – REST)
|

v
Base de Datos (MySQL 8 – Docker)

3.1. Tecnologías utilizadas

Frontend

- React 18
- Vite
- TailwindCSS
- Lucide Icons
- TypeScript
- Next (para compatibilidad de estructura)
- Vercel Analytics (removible)

Se incluyeron componentes reutilizables, formularios y validaciones complementarias antes de enviar los datos al backend.

Backend

- Python 3.12
- FastAPI
- bcrypt
- mysql-connector-python
- Modularización por routers, models y services
- Validadores propios para todas las reglas de negocio
- Responses estandarizadas (200, 400, 404, 422)

Base de Datos

- MySQL 8.0
- Estructura exacta a la solicitada en la letra:
    - participante
    - login
    - sala
    - reserva

- turno
- sancion_participante
- etc.

Infraestructura

- Docker
- Docker Compose (servicios: backend, base de datos)

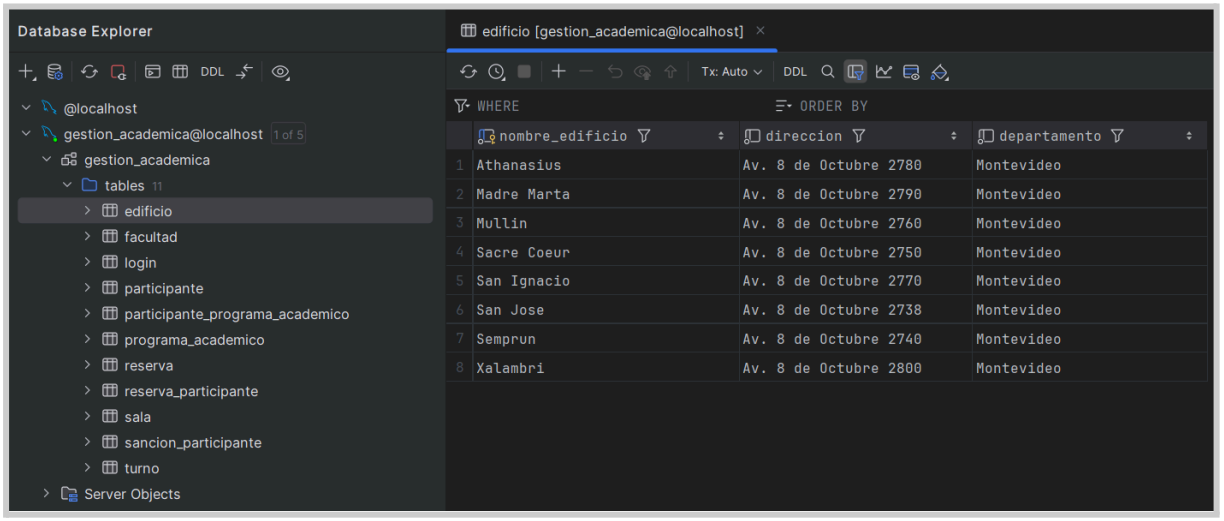
4. Base de Datos (MySQL)

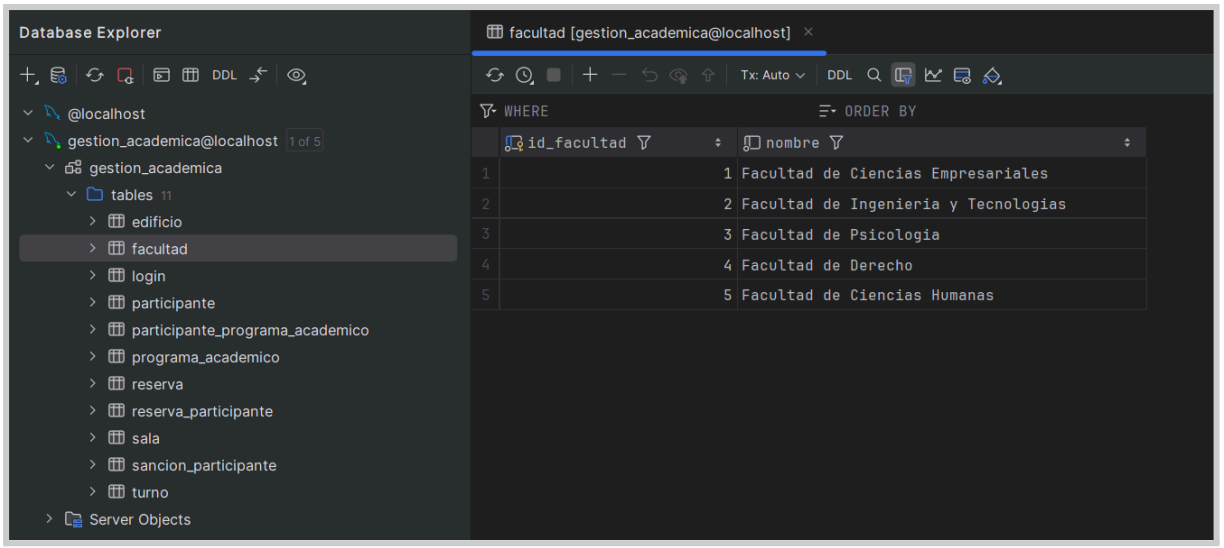
La estructura respeta estrictamente los lineamientos:

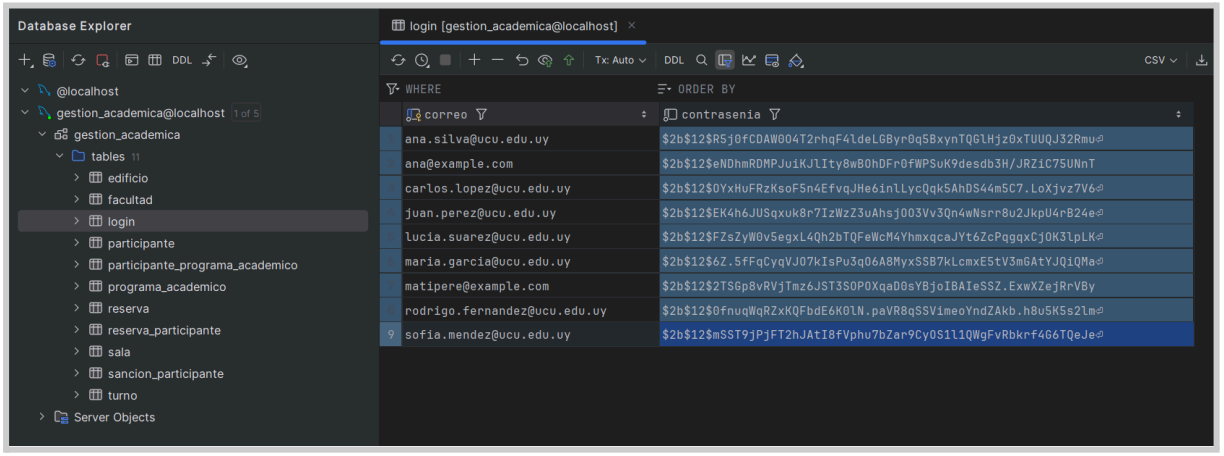
- Tablas y claves primarias
- Relaciones con claves foráneas y ON DELETE CASCADE cuando corresponde
- Validaciones con CHECK (por ejemplo: duración del turno = 1 hora exacta)
- Integridad referencial total

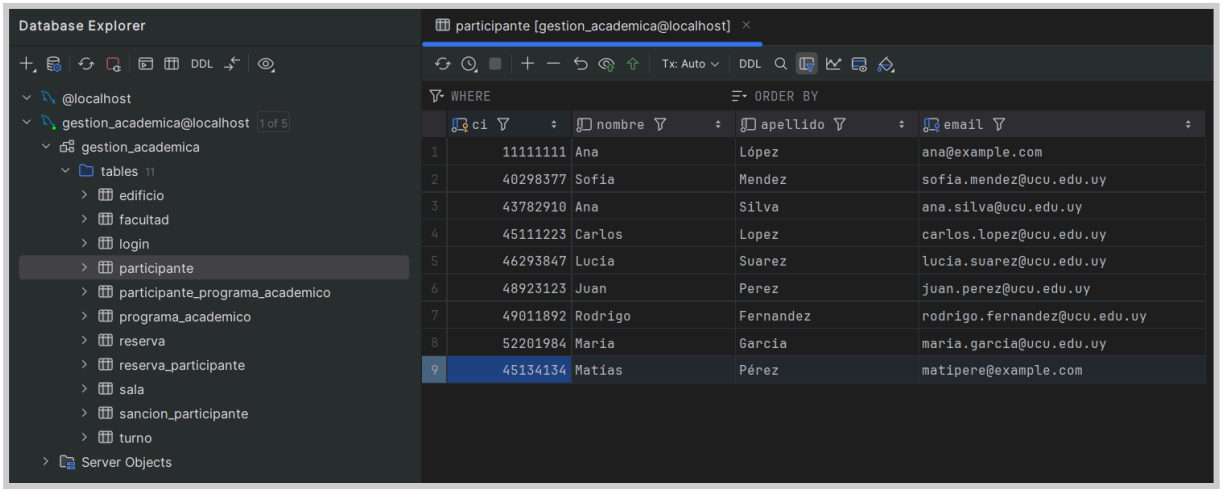
Incluye:

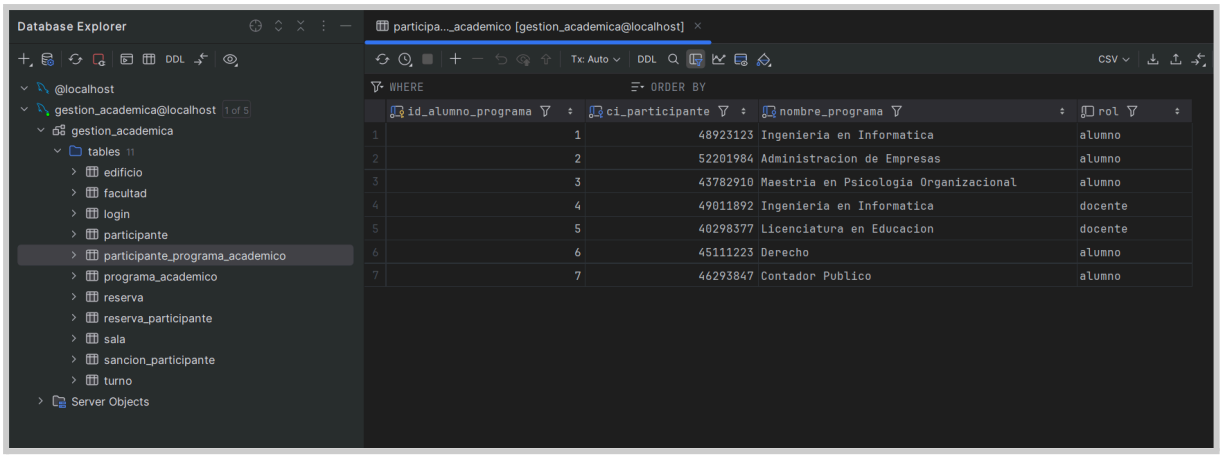
- Script completo SQL, que se ejecuta automáticamente al levantar el contendor, en /database/docker-entry
- Datos iniciales (salas, facultades, programas académicos, turnos, participantes)

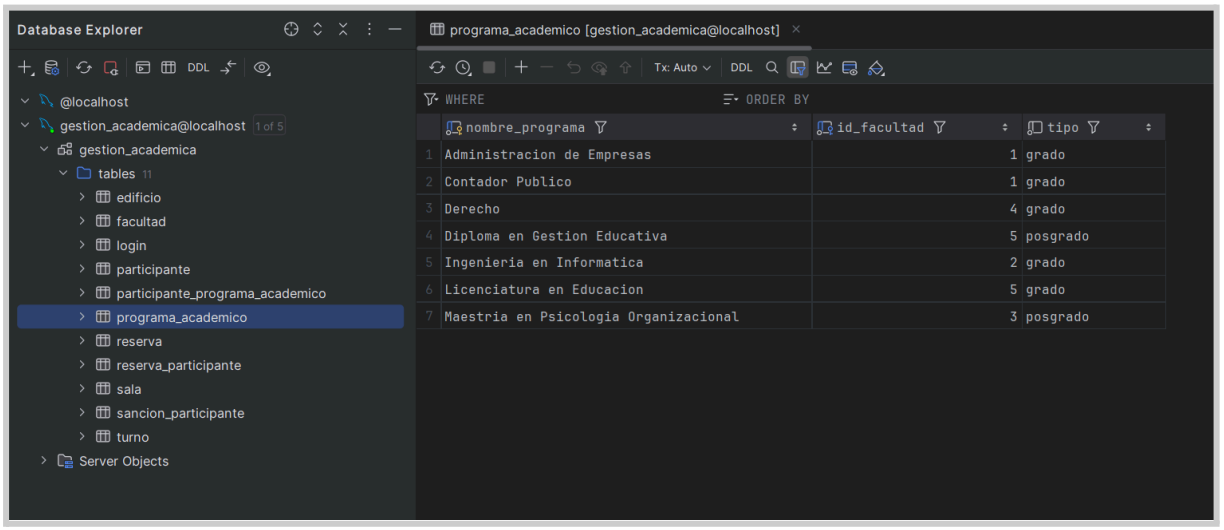


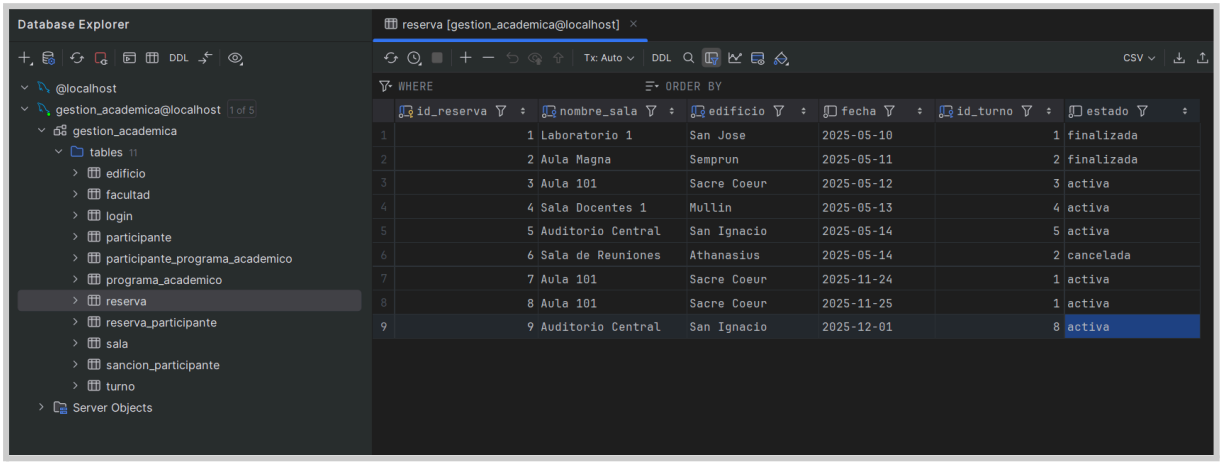

Database Explorer
facultad [gestion_academica@localhost]
@localhost
gestion_academica@localhost 1 of 5
gestion_academica
tables 11
edificio
facultad
login
participante
participante_programa_academico
programa_academico
reserva
reserva_participante
sala
sancion_participante
turno
Server Objects
WHERE
ORDER BY
id_facultad
nombre
1 Facultad de Ciencias Empresariales
2 Facultad de Ingenieria y Tecnologias
3 Facultad de Psicologia
4 Facultad de Derecho
5 Facultad de Ciencias Humanas

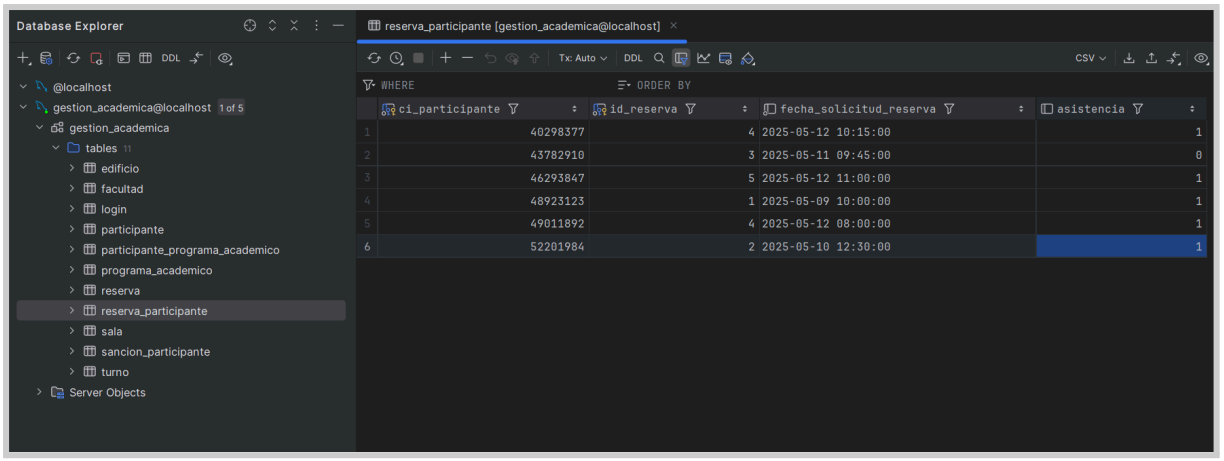

Database Explorer
login [gestion_academica@localhost]
WHERE
ORDER BY
correo
contrasenia
ana.silva@ucu.edu.uy
ana@example.com
carlos.lopez@ucu.edu.uy
juan.perez@ucu.edu.uy
lucia.suarez@ucu.edu.uy
maria.garcia@ucu.edu.uy
matipere@example.com
rodrigo.fernandez@ucu.edu.uy
sofia.mendez@ucu.edu.uy

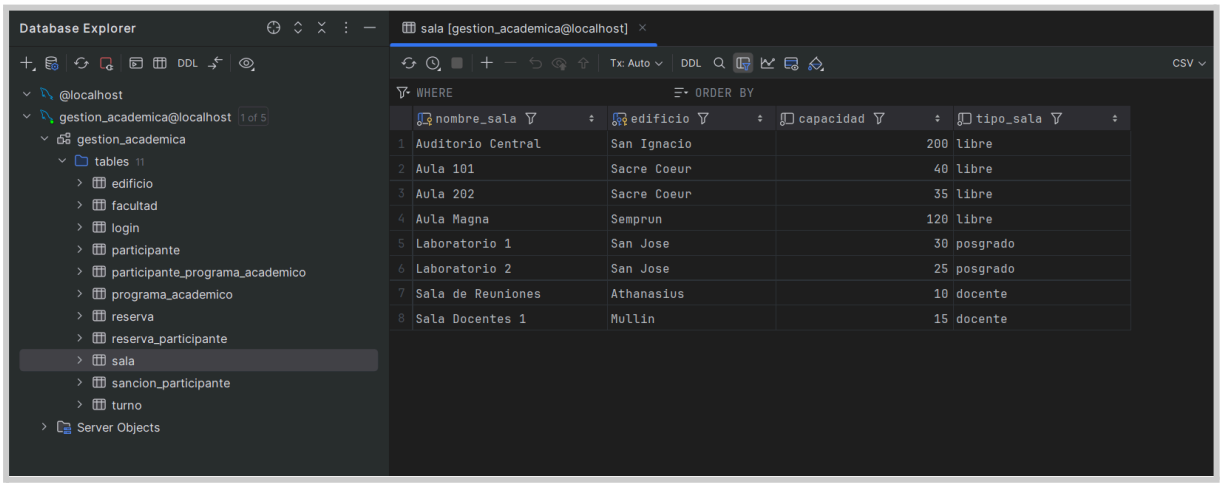

Database Explorer
participante [gestion_academica@localhost]
WHERE
ORDER BY
ci
nombre
apellido
email
11111111 Ana López ana@example.com
40298377 Sofia Mendez sofia.mendez@ucu.edu.uy
43782910 Ana Silva ana.silva@ucu.edu.uy
45111223 Carlos Lopez carlos.lopez@ucu.edu.uy
46293847 Lucia Suarez lucia.suarez@ucu.edu.uy
48923123 Juan Perez juan.perez@ucu.edu.uy
49011892 Rodrigo Fernandez rodrigo.fernandez@ucu.edu.uy
52201984 Maria Garcia maria.garcia@ucu.edu.uy
45134134 Matias Pérez matipere@example.com

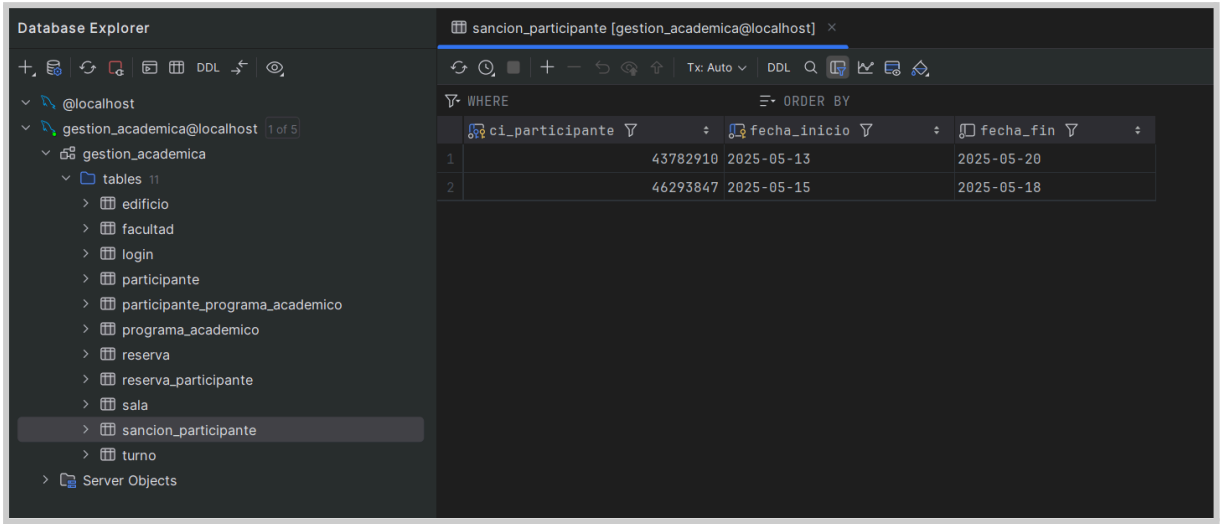

Database Explorer
participa..._academico [gestion_academica@localhost]
@localhost
gestion_academica@localhost 1 of 5
gestion_academica
tables 11
edificio
facultad
login
participante
participante_programa_academico
programa_academico
reserva
reserva_participante
sala
sancion_participante
turno
Server Objects
WHERE
ORDER BY
id_alumno_programa ci_participante nombre_programa rol
1 1 48923123 Ingenieria en Informatica alumno
2 2 52201984 Administracion de Empresas alumno
3 3 43782910 Maestria en Psicologia Organizacional alumno
4 4 49011892 Ingenieria en Informatica docente
5 5 40298377 Licenciatura en Educacion docente
6 6 45111223 Derecho alumno
7 7 46293847 Contador Publico alumno

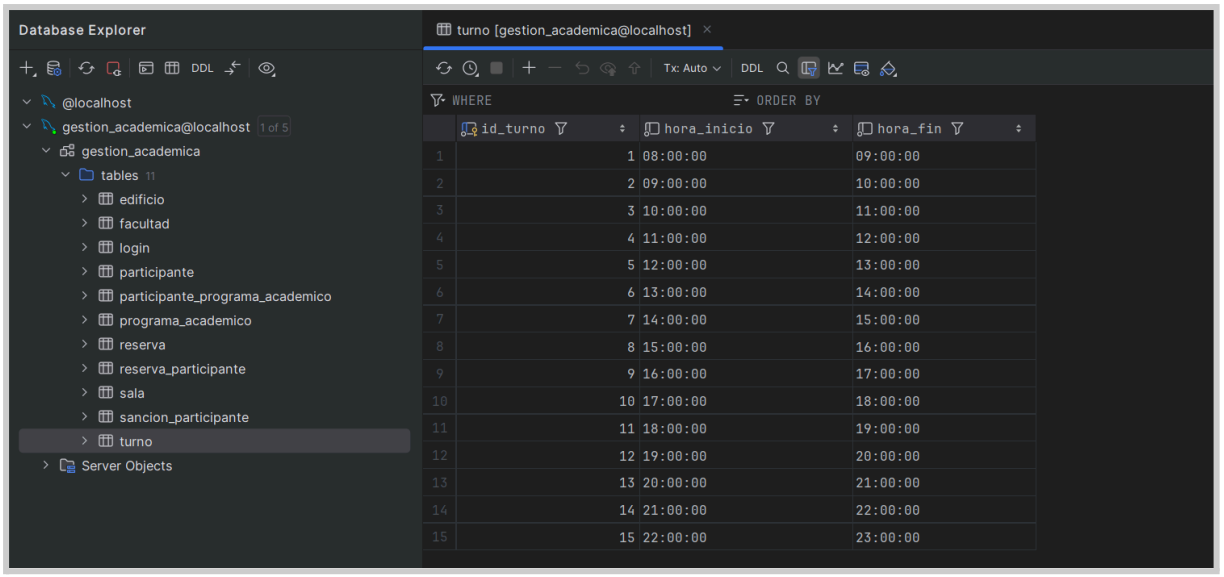

Database Explorer
programa_academico [gestion_academica@localhost]
@localhost
gestion_academica@localhost 1 of 5
gestion_academica
tables 11
edificio
facultad
login
participante
participante_programa_academico
programa_academico
reserva
reserva_participante
sala
sancion_participante
turno
Server Objects
WHERE
ORDER BY
nombre_programa id_facultad tipo
1 Administracion de Empresas 1 grado
2 Contador Publico 1 grado
3 Derecho 4 grado
4 Diploma en Gestion Educativa 5 posgrado
5 Ingenieria en Informatica 2 grado
6 Licenciatura en Educacion 5 grado
7 Maestria en Psicologia Organizacional 3 posgrado


Database Explorer
reserva [gestion_academica@localhost]
@localhost
gestion_academica@localhost 1 of 5
gestion_academica
tables 11
edificio
facultad
login
participante
participante_programa_academico
programa_academico
reserva
reserva_participante
sala
sancion_participante
turno
Server Objects
WHERE
ORDER BY
id_reserva nombre_sala edificio fecha id_turno estado
1 1 Laboratorio 1 San Jose 2025-05-10 1 finalizada
2 2 Aula Magna Semprun 2025-05-11 2 finalizada
3 3 Aula 101 Sacre Coeur 2025-05-12 3 activa
4 4 Sala Docentes 1 Mullin 2025-05-13 4 activa
5 5 Auditorio Central San Ignacio 2025-05-14 5 activa
6 6 Sala de Reuniones Athanasius 2025-05-14 2 cancelada
7 7 Aula 101 Sacre Coeur 2025-11-24 1 activa
8 8 Aula 101 Sacre Coeur 2025-11-25 1 activa
9 9 Auditorio Central San Ignacio 2025-12-01 8 activa


Database Explorer
@localhost
gestion_academica@localhost 1 of 5
gestion_academica
tables 11
edificio
facultad
login
participante
participante_programa_academico
programa_academico
reserva
reserva_participante
sala
sancion_participante
turno
Server Objects
reserva_participante [gestion_academica@localhost]
WHERE
ORDER BY
ci_participante | id_reserva | fecha_solicitud_reserva | asistencia
40298377 | 4 | 2025-05-12 10:15:00 | 1
43782910 | 3 | 2025-05-11 09:45:00 | 0
46293847 | 5 | 2025-05-12 11:00:00 | 1
48923123 | 1 | 2025-05-09 10:00:00 | 1
49011892 | 4 | 2025-05-12 08:00:00 | 1
52201984 | 2 | 2025-05-10 12:30:00 | 1


Database Explorer
@localhost
gestion_academica@localhost 1 of 5
gestion_academica
tables 11
edificio
facultad
login
participante
participante_programa_academico
programa_academico
reserva
reserva_participante
sala
sancion_participante
turno
Server Objects
sala [gestion_academica@localhost]
WHERE
ORDER BY
nombre_sala | edificio | capacidad | tipo_sala
Auditorio Central | San Ignacio | 200 | libre
Aula 101 | Sacre Coeur | 40 | libre
Aula 202 | Sacre Coeur | 35 | libre
Aula Magna | Semprun | 120 | libre
Laboratorio 1 | San Jose | 30 | posgrado
Laboratorio 2 | San Jose | 25 | posgrado
Sala de Reuniones | Athanasius | 10 | docente
Sala Docentes 1 | Mullin | 15 | docente


Database Explorer
@localhost
gestion_academica@localhost 1 of 5
gestion_academica
tables 11
edificio
facultad
login
participante
participante_programa_academico
programa_academico
reserva
reserva_participante
sala
sancion_participante
turno
Server Objects
sancion_participante [gestion_academica@localhost]
WHERE
ORDER BY
ci_participante | fecha_inicio | fecha_fin
43782910 | 2025-05-13 | 2025-05-20
46293847 | 2025-05-15 | 2025-05-18

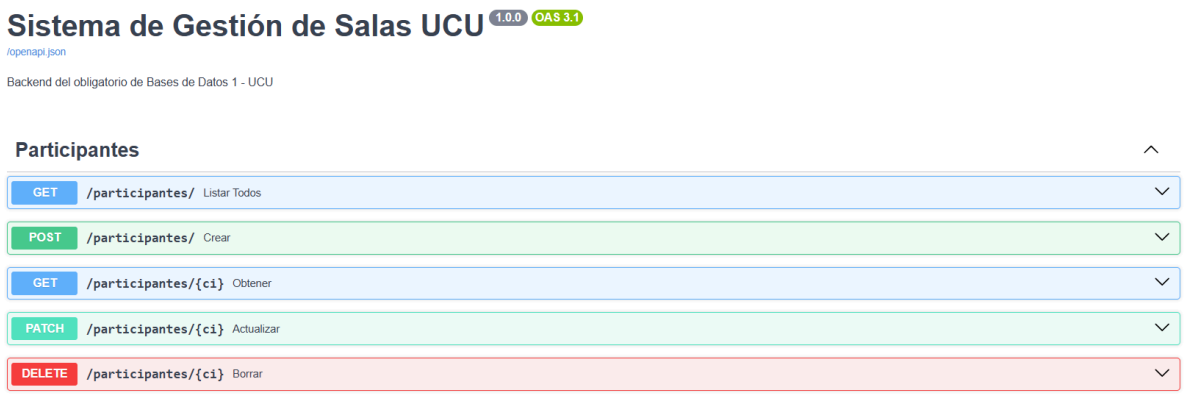


5. Backend (FastAPI)

La API se implementó modularmente con:

- routes/
- models/
- services/
- utils/helpers.py
- utils/validators.py

5.1. Endpoints incluidos



5.2. Reglas de negocio implementadas

Se implementaron restricciones como:

- *Reservas solo en bloques de 1 hora*
- *Máximo 2 horas diarias (para grado)*
- *Máximo 3 reservas activas por semana (para grado)*
- *Salas posgrado disponibles solo para posgrado/docentes*

- *Salas docentes exclusivas para docentes*
- *Capacidad de sala no superada*
- *Asistencia obligatoria*
- *Sanción automática de 2 meses* si nadie asiste

Estas reglas están implementadas tanto en forma de *triggers* en la DB MySQL, como en *validators* en los servicios del backend Python.

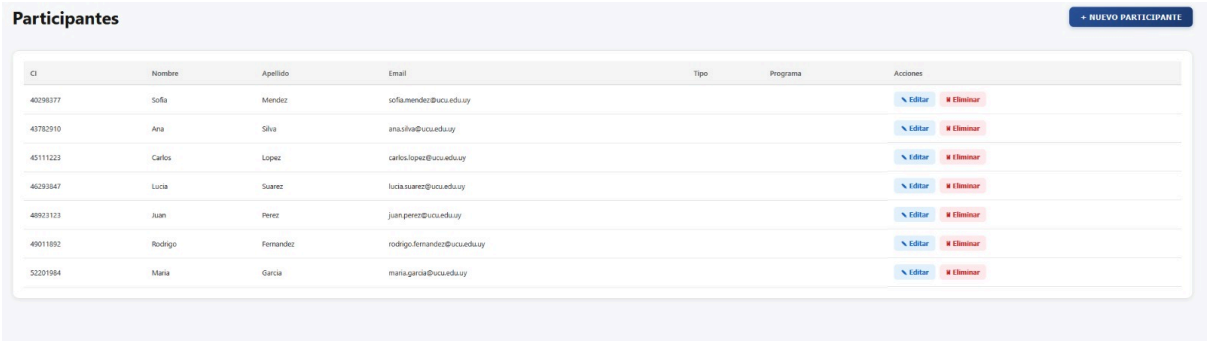
Capturas para insertar aquí

- Swagger mostrando todos los endpoints
- Ejemplos reales de POST /reservas
- Ejemplos reales de PATCH asistencia
- Mensajes de error de validación

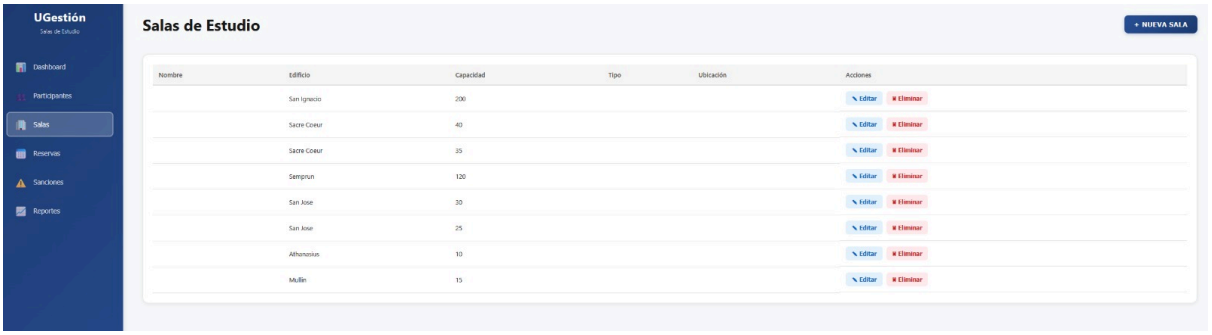
6. Frontend (React + Vite + Tailwind)

El frontend implementa:

- Pantalla de login
- Pantalla de reservas actuales



- Formulario de creación de reservas
- Gestión de salas



- Gestión de participantes
- Vista de sanciones

El objetivo fue entregar una interfaz clara que facilite a los funcionarios de UCU el uso del sistema.

Dependencias principales (package.json)

- Vista inicial
- ABM participantes
- ABM salas
- Reservas
- Sanciones
- Errores de validación del front

7. Validaciones y Seguridad

Seguridad

- Contraseñas hasheadas con bcrypt en el backend
- No se guardan contraseñas en texto plano
- No se usa ORM → consultas SQL controladas manualmente
- Sanitización de inputs
- Validaciones Pydantic en todos los modelos
- Validaciones adicionales en base de datos con CHECK + FK

Validaciones de reglas del negocio

Ejecutadas en utils/validators.py:

- Bloques estrictos de 1 hora
- No superposición de horarios
- Restricción de 2 horas por día
- Restricción de 3 reservas por semana
- Restricciones según tipo de sala
- Capacidad de sala
- Sanciones activas impiden reservar

8. Bitácora del Trabajo

Resumen realista del proceso:

1. Creación del script SQL en MySQL y pruebas en Docker.
2. Diseño inicial del modelo relacional según la letra.
3. Implementación del backend base (routers, services, db connection).
4. Implementación progresiva de reglas de negocio.
5. Creación de endpoints adicionales (asistencia, sanciones, reportes).
6. Construcción del frontend con Vite + React + Tailwind.
7. Instalación de entornos, Docker y estructura del proyecto
8. Conexión front–back y pruebas de integración.
9. Corrección de errores detectados en swagger.
10. Generación del informe, instructivo y capturas de pantalla.

9. Decisiones de Diseño

- Elegimos FastAPI por su velocidad, claridad, tipado y compatibilidad con Pydantic.
- Se evitó el uso de ORM para respetar la letra del curso.
- Se modularizó por servicios para mantener un código limpio y testeable.

- Todas las reglas de negocio se centralizaron en un módulo de validadores para evitar duplicación y mejorar el mantenimiento.
- Se optó por React + Vite por eficiencia y facilidad para construir SPAs modernas.
- Se dockerizó todo el entorno para tener una reproducibilidad total del sistema mucho más fácil.

10. Bibliografía

- Documentación oficial de FastAPI
- Documentación de MySQL 8
- Pydantic – documentación oficial
- Docker & Docker Compose docs
- TailwindCSS docs
- bcrypt documentation
- Consigna oficial del curso (UCU, Base de Datos 1)

11. Conclusión

Se intentó realizar un sistema robusto y consistente, que use las tecnologías más apropiadas y que funcione acorde a lo pedido. Tanto el uso de Swagger, MySQL y el frontend fueron fundamentales, y las múltiples verificaciones en las distintas capas fueron capaces de generar seguridad y un sistema más sólido .

Las capturas a ser agregadas demostrarán:

- Creación correcta de la base
- Uso real de todos los endpoints
- Validaciones funcionando
- Sanciones generadas automáticamente
- Frontend operativo