GESTIÓN DE RESERVAS SALAS DE ESTUDIO

Profesores:
Cristian Dinardi
Leandro Barral

Realizado por:
Facundo Diaz
Marcos Devincenzi

Avance: 31/10/2025
Entrega: 23/11/2025
Bases de datos I

facundo.diaz@correo.ucu.edu.uy
marcos.devincenzi@correo.ucu.edu.uy

INTRODUCCIÓN

El presente trabajo tiene como objetivo el diseño e implementación de un sistema de información destinado a la gestión de salas de estudio de la Universidad Católica del Uruguay. En la situación planteada por la consigna, la administración de las reservas se realiza de manera manual mediante planillas físicas, lo que dificulta el control, el seguimiento y la eficiencia en el uso de los espacios disponibles.

OBJETIVO GENERAL

El propósito de este proyecto es desarrollar una base de datos relacional en MySQL que permita registrar, organizar y consultar la información vinculada a reservas, participantes, turnos y sanciones.

El desarrollo de este sistema contribuye no solo a la mejora de los procesos administrativos internos, sino también al uso equitativo de los recursos físicos de la institución, fortaleciendo la trazabilidad y la transparencia en la asignación de salas.

Este trabajo integra los principales contenidos abordados en el curso, tales como el diseño lógico de bases de datos, la implementación de restricciones de seguridad, el uso de consultas de agregación y el manejo de claves primarias y foráneas.

OBJETIVOS ESPECÍFICOS

1. Desarrollar un repositorio en GitHub que registre el progreso del proyecto, facilite el control de versiones y documente los scripts y el código generado.

2. Modelar el sistema mediante un diseño lógico y físico de base de datos, definiendo entidades, relaciones y claves adecuadas para garantizar la integridad referencial.

3. Implementar consultas SQL que generen reportes útiles para la gestión institucional, aplicando funciones de agrupamiento, filtrado y combinación de datos.

4. Cargar datos maestros coherentes con la realidad universitaria, asegurando consistencia entre facultades, programas, edificios, salas y participantes.

5. Preparar la base de datos para su futura integración con un backend desarrollado en Python, estableciendo la estructura necesaria para la conexión con la aplicación.

6. Prototipar una interfaz (frontend) que permita operar las funciones centrales del sistema (ABM y reportes), priorizando la claridad de navegación.

7. Dockerizar la solución, coordinando al menos los servicios de aplicación (backend) y base de datos (MySQL), y opcionalmente el frontend.

DESARROLLO DEL PROYECTO

FASE 1: REPOSITORIO Y CONTROL DE VERSIONES

Previo al desarrollo del sistema, se llevó a cabo una fase inicial dedicada a la preparación del entorno de trabajo. El propósito de esta etapa fue organizar la estructura del proyecto y establecer un control de versiones confiable mediante Git y GitHub.

Se creó un repositorio público con el fin de centralizar los archivos, facilitar la colaboración entre los integrantes y documentar de manera adecuada el avance del proyecto.

FASE 2: DISEÑO E IMPLEMENTACIÓN DE LA BASE DE DATOS

La base de datos fue diseñada siguiendo los principios del modelo relacional de Codd, aplicando los conceptos de normalización, integridad referencial y coherencia semántica adquiridos en el curso.
El objetivo principal consistió en garantizar que la estructura reflejara correctamente las entidades y reglas definidas en la consigna.

El modelo se implementó en MySQL, utilizando tipos de datos adecuados, claves primarias y foráneas, así como restricciones de integridad para asegurar consistencia entre tablas.

Las principales entidades creadas fueron:

- Facultad
- Programa_academico
- Participante
- Participante_programa_academico
- Edificio
- Sala
- Turno
- Reserva
- Reserva_participante
- Sancion_participante
- Login

Cada tabla fue definida en el archivo db/schema.sql, utilizando nombres descriptivos y consistentes para mantener claridad semántica.

Se elaboró el archivo db/inserts.sql, el cual contiene datos simulados coherentes con la estructura académica de la UCU. Estos datos permitieron demostrar el funcionamiento del sistema y probar las consultas solicitadas por la consigna.

Asimismo, se desarrollaron las ocho consultas indicadas en el enunciado más las 3 adicionales pedidas, todas incluidas en db/consultas.sql, debidamente probadas y documentadas para garantizar que devuelvan información relevante para la toma de decisiones.

Consultas adicionales:

- Horarios Fantasma: Identifica combinaciones de salas y turnos que nunca han sido reservadas. Esta consulta utiliza un CROSS JOIN entre salas y turnos, combinado con NOT EXISTS para filtrar aquellas combinaciones sin reservas resigtradas. Es util para detectar turnos con baja demanda en sallas especificas, y posibles ajustes en horarios de disponibilidad.
- Sala camaleon: Detecta salas con alta diversidad interdisciplinaria, mostrando cuales son utilizadas por multiples facultades. Utiliza GROUP_CONCAT para agregar nombres de facultades en una lista separada por comas, y HAVING para filtrar solo salas usadas por mas de una facultad. Faculita identificacoin de espacios que fomenten colaboracion interfacultades, y analisis de patrones de uso compartido.
- Patrones de uso por franja horaria: Analiza preferencias horarias segmentadas en 4 franjas (madrugadores, mañana, tarde, noche) según tipo de programa (grado/posgrado) y rol (alumno/docente). Emplea CASE para clasificar turnos y multiples JOIN para relacionar reservas con informacion academica. Permite comprender habitos de reserva por perful y patrones culturales academicos

FASE 3: DESARROLLO DEL BACKEND Y VALIDACIONES

Durante esta etapa se implementó el backend en Python, encargado de gestionar la interacción entre la base de datos MySQL y la lógica de negocio del sistema. El código se organizó en la carpeta **/backend**, distribuido en módulos independientes:

- **db_connection.py:** gestiona la conexión con MySQL mediante variables de entorno y el conector oficial.

- **validaciones.py:** implementa reglas del sistema, tales como límite de horas diarias, capacidad de la sala, solapamientos, validación de tipo de sala y verificación de sanciones activas.
- **lógica.py:** centraliza la creación de reservas, aplicando todas las validaciones necesarias antes de registrar operaciones.
- **logs.py:** registra cada acción ejecutada en la tabla *log_acciones*, permitiendo mantener trazabilidad.

La necesidad de registrar acciones exitosas y fallidas motivó la implementación de un manejo de errores mediante bloques *try/except/finally*. Este mecanismo asegura que, ante cualquier fallo de conexión o violación de integridad, se ejecute un **ROLLBACK**, conservando la consistencia de la base y dejando constancia del incidente en el log.

En esta fase también se desarrolló **reportes.py**, módulo que implementa las ocho consultas requeridas, convirtiéndolas en funciones Python que ejecutan dinámicamente las sentencias SQL y presentan los resultados de manera legible.

FASE 4: CARGA MASIVA DE DATOS

Para realizar pruebas integrales, se cargaron datos maestros que permitieran simular escenarios reales: horarios, salas, edificios, participantes y usuarios de prueba. La selección de esta información se basó tanto en el conocimiento institucional del equipo como en investigación complementaria.

FASE 5: APLICACIÓN Y MENU INTERACTIVO

Se construyó el archivo **app.py**, encargado de integrar las funciones del backend en una interfaz de consola interactiva. El sistema permite:

- Crear nuevas reservas aplicando todas las validaciones y registrando cada acción.
- Ejecutar los reportes desde un submenú específico.
- Cerrar sesión y finalizar la aplicación de manera segura.

El menú fue diseñado para facilitar la navegación y demostrar el funcionamiento completo del sistema durante la instancia de defensa.

Aunque esta versión prioriza la interfaz por consola, el backend quedó estructurado de forma modular, permitiendo su integración futura con un frontend web.

FASE 6: IMPLEMENTACIÓN DE FRONTEND

Durante dos semanas, el equipo desarrolló un frontend con el objetivo de mejorar la presentación visual del proyecto y obtener una vista gráfica del funcionamiento del sistema. La herramienta seleccionada fue **Flask**, logrando conectar la interfaz con la base de datos y visualizar datos reales.

Sin embargo, surgieron dificultades relacionadas con la compatibilidad entre dispositivos, lo que generó inconsistencias en su funcionamiento. Tras consultar al cuerpo docente, se optó por

continuar exclusivamente con la interfaz por consola, centrando los esfuerzos en su robustez, estabilidad y completitud.

Durante esta fase se experimentó, por primera vez, con Docker para intentar uniformizar los entornos de ejecución, aunque sin los resultados esperados.

FASE 7: DOCKERIZACION

Se diseñó una arquitectura completa utilizando **Docker Compose**, conformada por dos servicios principales:

- **python_app** (aplicación)
- **mysql_local** (base de datos)

Se configuraron volúmenes, variables de entorno y archivos SQL de inicialización. Asimismo, se integró la lógica del sistema dentro del contenedor y se validó la conexión mediante *docker exec*.

Las principales dificultades incluyeron:

- Uso de nombres inválidos para la base de datos (por ejemplo, guiones).
- Orden incorrecto de ejecución entre scripts de esquema y consultas.
- Problemas con contraseñas que contenían caracteres especiales.

- Ejecución automática del programa dentro del contenedor, impidiendo la interacción, lo que obligó a separar el proceso en varias etapas.

BITACORA DE TRABAJO:

Semana 1: Inicio del proyecto

Avances

- Lectura detallada y análisis de la consigna.
- Identificación de las entidades principales: participante, sala, reserva, turno, sanciones, edificio, entre otras.
- Elaboración del primer borrador del modelo SQL.
- Discusión sobre claves primarias, claves foráneas y restricciones iniciales (tipo de sala, límites de reservas, límites de horarios, sanciones, etc.).

Dificultades

- La consigna presentaba ambigüedades acerca de ciertas relaciones, como la identificación de salas por nombre o por nombre+edificio.
- Surgieron dudas sobre el modelado de asistencia, sanciones, turnos y participación múltiple.

Semana 2: Creación de la base de datos e inserciones iniciales

Avances

- Construcción de la primera versión del esquema SQL en MySQL mediante DataGrip.
- Carga de datos iniciales de prueba (edificios, salas, participantes, entre otros).

Dificultades

- Varias claves foráneas fallaron debido a inconsistencias en nombres entre tablas.
- Algunos *inserts* no respetaban campos obligatorios, como salas sin edificio o sin capacidad asignada.
- Se presentaron problemas con el orden de creación de tablas y claves foráneas.

Semana 3: Ajustes del modelo, inserciones completas y consultas

- **Avances**
- Ajustes del modelo según errores detectados con datos reales.
- Inserción de nuevas salas, participantes y reservas simuladas para pruebas.
- Desarrollo de las primeras consultas SQL solicitadas por la consigna.

- **Dificultades**
- Los datos iniciales eran insuficientes para obtener resultados significativos, lo que requirió agregar manualmente más información.
- Se generaron problemas con consultas que requerían agregaciones o uso de *GROUP BY*.
- El equipo presentó enfoques distintos debido a problemas de commits en GitHub, lo que dificultó la unificación de las consultas.

- Existieron dudas sobre la forma de calcular el uso de salas, así como sobre el tratamiento de reservas canceladas (eliminarlas o marcarlas como "canceladas").
- Se encontraron dificultades en la construcción de consultas complejas que combinaban múltiples *joins* o subconsultas.

Semana 4: Desarrollo del Backend (Python + Lógica del sistema)

Avances

- Inicio de la implementación del backend en Python.
- Establecimiento de la conexión con MySQL sin la utilización de un ORM.

- Creación de los módulos principales:
  - *lógica.py*
  - *db_connection.py*
  - *validaciones.py*
- Implementación de funcionalidades iniciales como login, ABM de participantes, ABM de salas y lectura de reservas.
- Se decidió organizar el proyecto distribuyendo las funciones según su responsabilidad: las reglas y validaciones en *validaciones.py*, la conexión y credenciales en *db_connection.py*, y los procedimientos principales en *lógica.py*.

Dificultades

- Error recurrente “ModuleNotFoundError: dotenv” al ejecutar el sistema desde distintos entornos.
- Problemas con imports relativos y absolutos.
- Validaciones duplicadas entre MySQL y Python.
- Dificultad para organizar la lógica de negocio de manera clara.
- Manejo inadecuado de errores cuando MySQL rechazaba operaciones, lo que motivó posteriormente la implementación de *try/except* con *rollback*.

Semana 5: Lógica de reservas y validaciones complejas

Avances

- Implementación de reglas clave:
    - Validación del tipo de sala.
    - Verificación de capacidad.
    - Control de solapamientos de reservas.
    - Se avanzó con la implementación del registro de asistencia dentro del esquema SQL.

Dificultades

- Las validaciones requerían la combinación de múltiples tablas, volviéndose más complejas de lo previsto.
- Parte de la lógica se volvió extensa y repetitiva.
- MySQL mostraba errores en consultas basadas en fechas debido a diferencias en formatos.
- La implementación de sanciones fue postergada, ya que aún no se tenía claridad completa sobre su funcionamiento o viabilidad en el sistema.

Semana 6: Desarrollo del Frontend / Consola

Avances

- Desarrollo de la primera interfaz por consola para validar el correcto funcionamiento del backend.
- Tras confirmar que las funciones operaban correctamente, los integrantes investigaron y desarrollaron dos propuestas de frontend con buena estética y funcionalidad, conectadas parcialmente a la base de datos mediante Flask.

Dificultades

- El proyecto quedó estancado debido a dificultades para integrar el frontend con el backend y problemas de compatibilidad entre dispositivos.

- Ante la falta de conocimiento técnico en desarrollo web y las limitaciones de tiempo, se decidió volver al uso exclusivo del sistema basado en consola.
- Durante esta fase, el equipo probó herramientas como Docker por primera vez, principalmente para intentar compatibilizar el frontend entre sistemas operativos, aunque sin éxito.
- Tras una consulta con el equipo docente, se confirmó que un sistema de consola robusto, estable y completo era una solución válida e incluso potencialmente superior a un frontend incompleto.

Semana 7: Dockerización del proyecto

Avances

- Se preparó una arquitectura funcional basada en Docker Compose con dos servicios:
  - *python_app* (aplicación principal)
  - *mysql_local* (base de datos)
- Configuración de volúmenes, archivos de inicialización SQL y variables de entorno.
- Integración total del backend con MySQL dentro del contenedor.
- Pruebas de conexión y ejecución utilizando *docker exec*.

Dificultades

- Fallos debido a nombres inválidos en la base de datos (por ejemplo, el uso de guiones).
- El orden incorrecto de ejecución de scripts hacía que las consultas se ejecutaran antes del esquema, lo que obligó a renumerar los archivos.

- Errores por contraseñas que contenían caracteres especiales, lo que requirió simplificarlas.
- El programa iniciaba automáticamente dentro del contenedor, pero la consola de Docker no permitía interacción, lo que obligó a separar la ejecución en pasos manuales.

CONCLUSIÓN

El proyecto nos permitió diseñar e implementar un sistema integral para la gestión de salas de estudio de la Universidad Católica del Uruguay, abordando todas las etapas necesarias para su desarrollo: modelado de datos, construcción de la base de datos en MySQL, implementación del backend en Python, elaboración de consultas, generación de reportes y creación de una interfaz de usuario funcional. A lo largo del proceso, se aplicaron los conceptos fundamentales del curso, tales como normalización, integridad referencial, manejo de claves, validaciones y buenas prácticas de programación.

El sistema resultante consolida la información relevante de reservas, salas, participantes, turnos y sanciones en un entorno confiable y estructurado, asegurando trazabilidad y transparencia en su operación. Asimismo, la incorporación de mecanismos de registro de acciones, validaciones automáticas y control de errores contribuye a garantizar la consistencia de los datos y la eficiencia de las operaciones.

Si bien se realizaron exploraciones iniciales hacia un frontend web y la dockerización del entorno, la solución final priorizó la robustez del backend y la interfaz por consola, logrando un sistema completamente operativo y alineado con los objetivos de la consigna. En conjunto, el trabajo desarrollado evidencia la integración de conocimientos teóricos y prácticos, y constituye una base sólida para futuras mejoras o ampliaciones del sistema.

BIBLIOGRAFÍA

Dalto, S. (2021, mayo 15). *Curso de SQL desde CERO (Completo)* [Video]. YouTube. https://www.youtube.com/watch?v=DFg1V-rO6Pg

Pallets Project. (s. f.). *Flask documentation — PDF version* [PDF]. https://flask-cn.readthedocs.io/_/downloads/en/stable/pdf/

Oracle Corporation. (2025, June 25). *MySQL Connector/Python Developer Guide* [PDF]. https://downloads.mysql.com/docs/connector-python-en.a4.pdf