Victor Sosa, Domingo Leiva, Joaquín García.

Informe Trabajo Obligatorio

Sistema de Gestión de Reserva de Salas de Estudio

1. Fundamentación de decisiones de implementación

El sistema fue desarrollado utilizando Python (Flask) para el backend y HTML + JavaScript para el frontend. Se empleó una arquitectura modular basada en Blueprints y un enfoque relacional con MySQL para garantizar la integridad de datos y las reglas de negocio. Las validaciones implementadas contemplan límites de reservas, salas exclusivas según rol, registro de asistencia y aplicación automática de sanciones, respetando exactamente lo solicitado en la letra del obligatorio.

Las validaciones del sistema se implementaron directamente en el backend para garantizar la integridad de las reglas de negocio indicadas en la letra.

Se controla el límite de dos horas diarias y tres reservas semanales por participante mediante consultas SQL agrupadas por fecha.

Las salas exclusivas se restringen comparando el tipo de sala y el rol del participante (grado, posgrado o docente).

La inasistencia se registra por participante, y al finalizar una reserva se verifica si nadie asistió para aplicar automáticamente una sanción válida por dos meses.

También se bloquea la creación de reservas si un participante se encuentra sancionado dentro del rango de fechas correspondiente.

2. Mejoras implementadas y consideradas en el modelo de datos

Mejoras implementadas

- **Eliminación de redundancias:** Se evitó almacenar información duplicada, manteniendo datos normalizados en tablas específicas (por ejemplo, participante, sala, programa_académico).
- **Simplificación del control de asistencia:** Se incorporó el atributo asistencia directamente en la tabla reserva_participante, permitiendo registrar la presencia o ausencia de cada participante sin requerir tablas intermedias adicionales.
- **Sistema de sanciones basado en rangos de fechas:** En lugar de un modelo estático, las sanciones se registran con fecha_inicio y fecha_fin, lo que facilita verificar automáticamente si un usuario está habilitado o no para reservar.
- **Modelo relacional completo y consistente:** Se reforzó la integridad referencial mediante claves foráneas bien definidas, garantizando que una reserva siempre esté asociada a una sala, turno y participantes válidos.
- **Optimización en la gestión de roles y programas:** La estructura participante_programa_academico permite manejar roles (alumno, docente) y tipos de programa (grado, posgrado) sin modificar la tabla principal de participantes.

Mejoras consideradas pero no implementadas

- **Índices adicionales en columnas críticas** (como fecha, id_reserva o ci_participante) para acelerar aún más las consultas en escenarios de alto volumen.
- **Sistema de auditoría** para registrar automáticamente cambios en reservas, asistencias o sanciones.
- **Manejo avanzado de concurrencia** para evitar conflictos cuando múltiples usuarios intentan reservar la misma sala y turno simultáneamente.
- **Estructura de logs centralizados** para depuración avanzada y trazabilidad.
- **Dockerización.**

3. Bitácora del trabajo realizado

- Semana 1 (31/10) – Diseño y análisis inicial: interpretación detallada de la letra, identificación de entidades, reglas de negocio y diseño del modelo conceptual.
- Semana 2 – Base de datos: creación física de tablas, definición de claves, validaciones, comprobación de integridad referencial e inserción de datos iniciales.
- Semana 3 – Backend: implementación del servidor Flask, rutas, validaciones principales (límite diario, semanal, sanciones, roles, salas exclusivas) y manejo de errores.
- Semana 4 – Frontend: construcción de interfaces HTML, integración con backend mediante Fetch API, manejo de CORS y pruebas de interacción.
- Semana 5 – Integración final: pruebas completas de extremo a extremo, ajustes finos, depuración, validación de reglas especiales (docentes/posgrado), documentación e instructivo.

4. Bibliografía

- Flask Documentation – https://flask.palletsprojects.com/
- MDN Web Docs: Fetch API – https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- ChatGPT (OpenAI) – Asistencia en diseño, depuración y documentación – https://chat.openai.com
- Visual Studio Code Documentation – https://code.visualstudio.com/docs
- Python Standard Library (datetime) – https://docs.python.org/3/library/datetime.html
- HTML Living Standard – WHATWG – https://html.spec.whatwg.org/