**Sprint 1 - Día 1**

* Objetivo: Montar base de datos, clases modelo y estructura del proyecto
* Tareas estimadas:  
  + Crear pom.xml y estructura Maven
  + Script SQL y ERD
  + Clases: Sala, Empleado, Reserva
* Reflexión:  
  + Todo completado. Buen inicio. Para mañana se plantea implementar DAO y lógica CRUD.

**Sprint 1 - Día 2**

* Objetivo: Implementar la conexión a la base de datos MySQL y desarrollar el CRUD completo para la entidad Sala, utilizando buenas prácticas y estructura con clases DAO.
* Tareas estimadas:  
  + Crear la clase ConexionBD con parámetros reutilizables para la conexión.
  + Desarrollar la clase SalaDAO con los métodos:
  + insertarSala(Sala sala)
  + obtenerSalas()
  + actualizarSala(Sala sala)
  + eliminarSala(int id)
  + Verificar manualmente el funcionamiento de los métodos CRUD.
* Reflexión:  
  + Todas las tareas fueron completadas correctamente.
  + Se estableció la conexión con la base de datos y se accedió sin errores.
  + El CRUD de salas funciona como se esperaba.
  + Se decidió mantener ConexionBD fuera del paquete modelo para una mejor organización del código.
  + Para mañana se planea implementar la lógica CRUD de empleados con la misma estructura de DAO.

**Sprint 1 - Día 3**

* Objetivo: Implementar la funcionalidad CRUD completa para la entidad **Empleado**, siguiendo la misma estructura y patrones aplicados anteriormente con la entidad **Sala**.
* Tareas estimadas:  
  + Crear clase Empleado en el paquete modelo.
  + Crear clase EmpleadoDAO con los siguientes métodos:
  + insertarEmpleado(Empleado empleado)
  + obtenerEmpleados()
  + actualizarEmpleado(Empleado empleado)
  + eliminarEmpleado(int id)
  + Validar manualmente que las operaciones CRUD funcionen correctamente en la base de datos MySQL.
* Reflexión:  
  + Todas las tareas planificadas fueron completadas con éxito.
  + Se mantuvo la estructura organizada del código (modelo y DAO separados).
  + Los métodos funcionan correctamente y son reutilizables.
  + Para el próximo día se planea abordar la entidad Reserva, que requiere gestionar relaciones entre Empleado y Sala, y controlar conflictos de horario.

**Sprint 2 - Día 4**

* Objetivo: Desarrollar la entidad **Reserva** y su correspondiente clase ReservaDAO, implementando las funcionalidades CRUD y controlando los conflictos de horario entre reservas para una misma sala.
* Tareas estimadas:
  + Crear la clase Reserva en el paquete modelo.
  + Crear la clase ReservaDAO con los métodos:
  + insertarReserva(Reserva reserva)
  + obtenerReservas()
  + eliminarReserva(int id)
  + hayConflicto(Reserva reserva) ← verificación de solapamiento de horarios
  + Probar manualmente que el sistema rechace reservas dobles y permita reservas válidas.
* Reflexión:  
  + Todas las tareas se completaron con éxito.
  + Se implementó correctamente la detección de conflictos de horario para evitar reservas simultáneas en una misma sala.
  + La estructura de código sigue siendo clara y modular.
  + Para el último día se plantea:
  + Implementar pruebas unitarias con JUnit
  + Crear el archivo README.md
  + Finalizar la documentación y empaquetado para la entrega

**Sprint 2 - Día 5**

### Objetivo: Realizar pruebas unitarias con JUnit de las entidades Sala, Empleado y Reserva, y dejar el proyecto listo para entrega.

### Tareas estimadas:

* + Crear tests unitarios con JUnit para SalaDAO, EmpleadoDAO y ReservaDAO.
  + Comprobar funcionamiento de los tests utilizando: mvn test
  + Preparar archivo README.md con instrucciones de uso y entrega.
* Reflexión:
  + Se implementaron correctamente las pruebas unitarias.
  + El proyecto ahora permite comprobar de forma automática que las funcionalidades CRUD funcionan.
  + Se finalizó el README.md y se subió el repositorio con la estructura ordenada.
  + El proyecto queda listo para entrega.