**🧩 1. Documentación — Backend (Java Spring Boot)**

**🏗️ Arquitectura General**

El backend está desarrollado con **Spring Boot** siguiendo el patrón **MVC (Modelo-Vista-Controlador)**, estructurado en paquetes:

com.example.asistencia

┣ controller/

┣ service/

┣ model/

┣ repo/

┗ dto/

**🔹 Funcionalidad Principal**

El módulo implementa la gestión de **asistencias estudiantiles**, relacionando estudiantes, grupos y sus registros de asistencia.

**⚙️ Tecnologías utilizadas**

* **Java 17**
* **Spring Boot 3**
* **Spring Data JPA**
* **Spring Web**
* **Spring Validation**
* **MySQL / MariaDB**
* **Lombok (opcional)**
* **Postman (para pruebas)**
* **Maven**

**🧱 Estructura del Módulo de Asistencia**

**📄 Modelos**

* **Estudiante:** contiene id, nombre, identificacion, correo.
* **Grupo:** contiene id, nombre, materia.
* **Asistencia:** relaciona estudiante + grupo + fecha + estado (PRESENTE, AUSENTE, JUSTIFICADO).

**📄 DTOs**

* **AsistenciaDTO:** usado para crear registros (entrada desde el frontend).
* **AsistenciaResponseDTO:** usado para responder al frontend con información expandida (nombre del estudiante, grupo, etc.).

**📄 Repositorios**

Extienden JpaRepository para operaciones CRUD.  
Incluyen métodos personalizados como:

boolean existsByFechaAndEstudianteIdAndGrupoId(LocalDate fecha, Long estudianteId, Long grupoId);

List<Asistencia> findByGrupoIdAndFecha(Long grupoId, LocalDate fecha);

**💡 Service Layer — AsistenciaService**

Contiene la lógica de negocio principal:

* Validación de fechas (no futuras).
* Validación de existencia de estudiante y grupo.
* Prevención de registros duplicados.
* Conversión entre entidad y DTO.

El uso de **@Transactional** garantiza la integridad de las operaciones en base de datos.  
Ejemplo:

@Transactional

public Asistencia crear(AsistenciaDTO dto) {

// Validaciones y guardado atómico

}

**💬 Controller — AsistenciaController**

Gestiona los endpoints REST disponibles:

| **Método** | **Endpoint** | **Descripción** |
| --- | --- | --- |
| POST | /api/asistencias | Crear nueva asistencia |
| GET | /api/asistencias/estudiante/{id} | Listar asistencias de un estudiante |
| GET | /api/asistencias/grupo/{id}/fecha/{fecha} | Listar asistencias por grupo y fecha |
| PUT | /api/asistencias/{id}/estado | Actualizar estado |
| DELETE | /api/asistencias/{id} | Eliminar registro |
| GET | /api/asistencias | Obtener todas (para el frontend) |

**🌐 CORS y Seguridad**

Se habilitó acceso al frontend (puerto 5173) mediante una clase WebConfig:

@Configuration

public class WebConfig {

@Bean

public WebMvcConfigurer corsConfigurer() {

return new WebMvcConfigurer() {

@Override

public void addCorsMappings(CorsRegistry registry) {

registry.addMapping("/\*\*").allowedOrigins("http://localhost:5173").allowedMethods("\*");

}

};

}

}

**🧪 Pruebas**

* **Postman:** se validaron los endpoints POST, PUT, DELETE, GET.
* **Respuestas 404 / 409 / 400:** correctamente gestionadas con ResponseStatusException.