GUÍA DE ESTUDIO SPRING BOOT - LEAGUEPLAY

1. Spring Boot Basics

- Spring Boot simplifica el desarrollo de aplicaciones Spring mediante configuración automática y dependencias predefinidas.
- Se ejecuta como una aplicación Java estándar con una clase main.

2. Estructura del Proyecto

- LeaguePlay está organizado en paquetes:
  - controllers: Maneja las peticiones HTTP.
  - services: Lógica de negocio.
  - models: Entidades JPA (tablas de base de datos).
  - dto: Objetos de transferencia de datos (intercambio entre frontend/backend).
  - repositories: Interfaces para el acceso a datos mediante Spring Data JPA.
  - config: Configuración de seguridad.

3. Controladores (Controllers)

- Definidos con @RestController y @RequestMapping.
- Por ejemplo, AuthController expone endpoints como /api/auth/register.
- Usan objetos @RequestBody (DTOs) para recibir datos.

4. DTOs

- RegisterDTO encapsula datos del formulario de registro: username, email, password, repeatPassword.
- Sirve para separar la lógica del modelo de base de datos del intercambio de datos.

5. Servicios (Services)

- Usan @Service y encapsulan lógica como el registro de usuarios.

- Ejemplo: AuthService se encarga de codificar la contraseña, crear el objeto User, guardarlo en la BBDD.

6. Entidades (Models)

- Clases anotadas con @Entity que representan tablas de base de datos.

- Por ejemplo: User tiene campos como id, username, email, password, createAt.

- Se persisten automáticamente mediante JPA y repositories.

7. Repositorios (Repositories)

- Interfaces que extienden JpaRepository<T, ID>.

- Por ejemplo: UserRepository permite buscar por username o email.

8. Spring Security

- Se configura con SecurityConfig, anotada con @Configuration y @EnableWebSecurity.

- Usamos un SecurityFilterChain para permitir acceso libre a /api/auth/** y proteger el resto.

- Se desactiva CSRF por simplicidad en desarrollo.

9. Seguridad básica (por ahora)

- Seguridad básica HTTP con .httpBasic() mientras se implementa el login real con token o sesiones.

- El sistema usa BCrypt para cifrar las contraseñas.

10. Manejo de errores comunes

- Error 400: problema en el cuerpo JSON o petición malformada.

- Error 401/403: problemas de autenticación/autorización.

- Error 500: errores internos como duplicidad de datos (ej. username/email repetido).

Consejos:

- Usa Postman en modo "raw JSON" y asegúrate de que los headers sean correctos.

- Añade validaciones previas para evitar errores 500 innecesarios (ej: comprobar si el username ya existe).