mobydigital.

Evaluación N°2 Backend

JUnit, Mockito, JPA, Buenas Prácticas en el código, Postman y Swagger

Fecha límite de entrega: *30/10/25*

Sistema para Gestión de Votaciones

En el marco del contexto electoral actual en Argentina, el objetivo de esta evaluación será desarrollar una **API REST con Java y Spring Boot** que permita gestionar información básica relacionada con un **sistema de votaciones**, manteniendo una arquitectura limpia, probada y documentada.

Esta evaluación busca integrar todos los contenidos vistos en la primera parte del Roadmap (manejo de Java, Git, estructuras de proyectos, y desarrollo de APIs) y profundizar en los nuevos conocimientos adquiridos durante el segundo tramo del Roadmap.

Objetivos de la Evaluación

El proyecto deberá demostrar dominio de los siguientes conceptos y tecnologías:

- **Pruebas unitarias** con **JUnit y Mockito** (mockeo de dependencias en servicios y controladores).
- **Persistencia de datos** mediante **Spring Data JPA**.
- **Modelado de entidades y relaciones** (mappings con JPA).
- **Buenas prácticas de código** y análisis estático mediante **SonarLint / SonarQube**.
- **Documentación y testeo de la API** con **Postman** y **Swagger (OpenAPI)**.
- **Gestión de código limpio y mantenible** en un repositorio personal de GitHub.

Enunciado del Proyecto

Tema: Sistema de Gestión de Votaciones

Desarrollar una API REST que permita gestionar un **sistema de votaciones** a nivel local. El sistema deberá permitir registrar **candidatos**, **partidos políticos**, y **votos** emitidos por electores.

Requerimientos Técnicos

1. Clases (Modelado con JPA)

Deberán modelarse al menos las siguientes clases con sus relaciones:

- **PartidoPolitico**
    - id (Long, autogenerado)
    - nombre (String)
    - sigla (String)
- **Candidato**
    - id (Long, autogenerado)
    - nombreCompleto (String)
    - partido (ManyToOne → PartidoPolitico)
- **Voto**
    - id (Long, autogenerado)
    - candidato (ManyToOne → Candidato)
    - fechaEmision (LocalDateTime)

Nota: la base de datos deberá implementarse en memoria usando **H2** y deberá persistir los datos temporalmente mientras se ejecuta la aplicación.

2. Endpoints REST

La API deberá exponer endpoints que permitan:

- Crear, listar, obtener y eliminar **partidos políticos**
- Crear, listar, obtener y eliminar **candidatos**
- Registrar un **voto**
- Consultar la **cantidad total de votos** por candidato o por partido

Los endpoints deben estar correctamente estructurados según buenas prácticas REST (por ejemplo: `/api/candidatos`, `/api/partidos`, `/api/votos`).

3. Capa de Servicios y Repositorios

- Utilizar **Spring Data JPA** para las operaciones de persistencia.
- Implementar una **capa de servicio intermedia** entre los controladores y los repositorios, donde se maneje la lógica de negocio.
- Aplicar buenas prácticas de diseño y uso adecuado de **interfaces y dependencias**.

4. Testing

- Implementar **pruebas unitarias** con **JUnit 5 y Mockito** para los servicios y controladores.
- Mockear correctamente las dependencias (`@Mock`, `@InjectMocks`).
- Se valorará la cobertura de código y la claridad de los tests.

5. Documentación y Pruebas de la API

- Generar la documentación de la API utilizando **Swagger** mediante anotaciones (`@Operation`, `@ApiResponse`, etc.).

- Entregar una **colección de Postman** con:
    - Ejemplos de request y response para cada endpoint.
    - Datos válidos para pruebas.

6. Buenas Prácticas de Código

- Analizar el proyecto con **SonarLint** en IntelliJ.
- No debe existir ***ningún issue*** ni ***code smells*** al momento de la entrega.
- Seguir convenciones de nomenclatura, separación de responsabilidades y legibilidad del código.

7. Entrega

Cada uno deberá:

1. Subir su proyecto a un **repositorio personal en GitHub** (asegurarse de agregar como colaboradores a los mentores para poder visualizarlo).
2. Entregar:
    - Enlace al repositorio.
    - Colección de Postman exportada (`.json`). Si quieren, pueden crear un paquete que se llame `/docs` en la raíz del proyecto, para colocar el archivo de la colección ahí.
3. Incluir un **README.md** con:
    - Descripción breve del proyecto.
    - Instrucciones de ejecución.
    - Datos de prueba.
    - Captura del análisis SonarLint.

Plazo de Entrega

La evaluación podrá entregarse hasta el **30 de octubre** inclusive. En caso de tener alguna complicación, por favor, comunicarse con alguno de los mentores para notificar la situación.

Nota: recuerden que también se evaluarán los conceptos de la primera parte del Roadmap, así que aprovechen esta instancia para demostrar todo lo que aprendieron y son capaces de hacer 💪

¡Muchos éxitos chicos! 🚀🐋