CHALLENGE BACKEND - Java Spring Boot (API) 🚀

Objetivo

Desarrollar una API para explorar los principales íconos geográficos del mundo, la cual permitirá conocer y modificar los íconos geográficos y entender en qué ciudades o países estos se encuentran. Como por ejemplo, la torre Eiffel de París, la Estatua de la Libertad en Nueva York, el Coliseo en Italia. Por otro lado, deberá exponer la información para que cualquier frontend pueda consumirla.

👉 Utilizar Spring Boot.
👉 No es necesario armar el Frontend.
👉 Las rutas deberán seguir el patrón REST.
👉 Utilizar la librería Spring Security.

⚠️ ¡No es indispensable hacer todo!
Mientras más completes, mayor puntaje obtendrás, pero puedes enviar la app hasta el estadío que la tengas en base a tu cono

Requerimientos técnicos

1. Modelado de Base de Datos

- **Íconos geográficos**: deberá tener,
  - Imagen.
  - Denominación.
  - Fecha de creación.
  - Altura.
  - Historia.
  - Ciudad en la que se encuentra y otras con réplicas.
- **Ciudad/País**: deberá tener,
  - Imagen.
  - Denominación.
  - Cantidad de habitantes.
  - Superficie total (m2)
  - Íconos geográficos asociados.
- **Continente:** deberá tener,

- Imagen.
- Denominación.
- Ciudades con íconos geográficos asociadas.

1. Autenticación de Usuarios

Para realizar peticiones a los endpoints subsiguientes el usuario deberá contar con un token que obtendrá al autenticarse. Para ello, deberán desarrollarse los endpoints de registro y login, que permitan obtener el token.

2. Listado de Íconos geográficos

El listado deberá mostrar:

- Imagen (no es necesario que esté persistida en la base de datos).
- Denominación.

3. Creación, Edición y Eliminación de Íconos geográficos (CRUD)

Deberán existir las operaciones básicas de creación, edición y eliminación de íconos geográficos.

4. Detalle del Ícono geográfico

En el detalle deberán listarse todos los atributos del ícono geográfico, como así también los países en los que se encuentran.

5. Búsqueda de Íconos geográficos

Deberá permitir buscar por denominación, y filtrar por fecha de creación, altura o ciudades/países en las que participó. Para especificar el término de búsqueda o filtros se deberán enviar como parámetros de query:

- GET /icons?name=denominación
- GET /icons?date=fecha de creación
- GET /icons?cities=idCity

6. Listado de Ciudades/Países

Deberá mostrar solamente los campos imagen, denominación y cantidad de habitantes.

El endpoint deberá ser:

- GET /cities

7. Detalle de Ciudad /País con sus íconos geográficos

Devolverá todos los campos de la ciudad o país junto a los íconos geográficos asociados a la misma.

8. Creación, Edición y Eliminación de Ciudades/Países (CRUD)

Deberán existir las operaciones básicas de creación, edición y eliminación de ciudades o países.

9. Búsqueda de Ciudades/Países

Deberá permitir buscar por denominación, y filtrar por continente. Además, permitir ordenar los resultados por fecha de creación de forma ascendiente o descendiente.

El término de búsqueda, filtro u ordenación se deberán especificar como parámetros de query:

- /cities?name=denominación
- /cities?continent=idContinent
- /cities?order=ASC | DESC

10. Envío de emails

Al registrarse en el sitio, el usuario deberá recibir un email de bienvenida. Es recomendable, la utilización de algún servicio de terceros como SendGrid.

Documentación

Es deseable documentar los endpoints utilizando alguna herramienta como Postman o Swagger.

Tests

De forma opcional, se podrán agregar tests de los diferentes endpoints de la APP, verificando posibles escenarios de error:

- Campos faltantes o con un formato inválido en BODY de las peticiones
- Acceso a recursos inexistentes en endpoints de detalle

Los tests pueden realizarse utilizando JUnit y Mockito.