API de Franquicias

Se requiere construir una API que permita gestionar una red de franquicias. Cada franquicia tiene un nombre y una lista de sucursales. Cada sucursal también tiene un nombre y una lista de productos disponibles. Cada producto tiene un nombre y un valor numérico que representa el stock disponible.

Recomendaciones clave:

- Usar **Spring WebFlux** con programación **reactiva**.
- Implementar los endpoints utilizando **RouterFunctions** y **Handlers**, en lugar de @RestController.
- Manejar errores y validaciones desde el **handler** y el **dominio**, evitando mensajes técnicos hacia el cliente.
- Documentar todos los endpoints utilizando **OpenAPI/Swagger**.
- Seguir la estructura de carpetas definida por el **Arqueotipo o el Scaffold Clean Architecture de Bancolombia**:
  - https://bancolombia.github.io/scaffold-clean-architecture/
  - https://github.com/nanitagtj/resilient-api/tree/master
- Escribir todo el código en **inglés**, incluyendo nombres de clases, funciones y variables.

Criterios de Aceptación

1. El proyecto debe estar desarrollado en **Spring WebFlux** (preferiblemente usando **Gradle**).
2. Exponer un endpoint para crear una nueva franquicia.
3. Exponer un endpoint para agregar una nueva sucursal a una franquicia existente.
4. Exponer un endpoint para agregar un producto a una sucursal específica.
5. Exponer un endpoint para eliminar un producto de una sucursal.
6. Exponer un endpoint para modificar el stock de un producto.
7. Exponer un endpoint que devuelva, para una franquicia específica, el producto con mayor stock por cada sucursal. La respuesta debe incluir la sucursal a la que pertenece cada producto.
8. Utilizar un sistema de persistencia como **Redis**, **MySQL**, **MongoDB** o **DynamoDB**, desplegado en algún proveedor de nube. La elección queda a tu criterio.

Puntos Extra

- Plus si la aplicación está empacada usando **Docker**.
- Plus si se incluye un endpoint para actualizar el nombre de una franquicia.
- Plus si se incluye un endpoint para actualizar el nombre de una sucursal.
- Plus si se incluye un endpoint para actualizar el nombre de un producto.
- Plus si se implementa infraestructura como código usando **Terraform**
- Plus si toda la solución está completamente desplegada en la nube.

Notas Importantes

- Se valorará el uso adecuado de **Git** durante el desarrollo. La prueba debe ser compartida en un repositorio de código público (GitHub, Bitbucket, etc.).
- El proyecto debe incluir un archivo README.md que explique cómo ejecutar la aplicación de forma local y cómo está estructurada la solución.

Estudiar para entrevista

- Arquitectura hexagonal
- Scaffold de bancolombia
- Programación reactiva, también diferenciar entre Mono y Flux
- Programación funcional
- Terraform, modulos, variables, providers, etc…
- AWS, ECR, ECS, ALB, API Gateway
- Principios SOLID
- Test Unitarios, JUnit, Mockito, diferencias entre prueba unitaria y de integración