Preparcial (30%) – CRUD con API NestJS para Planificación de Viajes

Objetivo general

El objetivo de este taller es integrar los conocimientos fundamentales de NestJS para diseñar una API REST modular que permita:

Consumir información de países desde una API externa (RestCountries).

Almacenar esos países en una base de datos local a manera de caché.

Crear y gestionar planes de viaje asociados a países específicos.

Poner en práctica conceptos de módulos, controladores, servicios, providers, DTOs y validación.

Este preparcial se enfoca exclusivamente en el backend. La API debe poder ser probada mediante herramientas como Postman, Thunder Client o similares. No se requiere implementar un frontend.

Preparación del Preparcial

Para el desarrollo del taller, cada estudiante deberá

Crear un nuevo proyecto en NestJS.

Instalar y configurar las dependencias necesarias para:

Consumir servicios HTTP hacia la API externa.

Conectarse a una base de datos.

Utilizar validación de datos en los DTOs.

Elegir y configurar una base de datos de su preferencia (por ejemplo, SQLite, Postgres, MySQL, MongoDB, etc.) junto con la librería/ORM que desee (TypeORM, Prisma, Mongoose, u otro).

Verificar que el proyecto se ejecute correctamente y que la API sea accesible en un puerto local.

No se proporcionará un backend base ni una configuración fija de base de datos: la elección y configuración de esta hace parte del ejercicio.

Enunciado del reto

Diseñar y desarrollar una API REST en NestJS para una aplicaci√≥n de planificaci√≥n de viajes, que incluya al menos los siguientes elementos:

- Un m√≥dulo encargado de gestionar pa√≠ses (CountriesModule), utilizando datos obtenidos desde la API externa RestCountries y almacen√°ndolos en una base de datos local.
- Un m√≥dulo encargado de gestionar planes de viaje (TravelPlansModule), permitiendo crear y consultar planes de viaje asociados a un pa√≠s.
- Un provider que encapsule el consumo de la API externa RestCountries, de forma que el m√≥dulo de pa√≠ses no dependa directamente de detalles de implementaci√≥n externos (URLs, formato de respuesta, etc.).

La aplicaci√≥n debe cumplir con la idea de "cach√©": cuando se consulte un pa√≠s, debe buscar primero en la base de datos local y, solo si no existe, acudir a la API externa para obtenerlo y almacenarlo.

Requisitos mínimos

Módulo de países (CountriesModule)

El sistema debe contar con un módulo dedicado a la gestión de países, que:

- Defina una entidad o modelo de país con al menos los siguientes atributos:
- Código de país en formato alpha-3 (por ejemplo, “COL”, “FRA”).
- Nombre del país.
- Región y subregión.
- Capital.
- Población.
- Alguna representación de la bandera (por ejemplo, una URL).
- Información de creación y última actualización en la base de datos.
- Exponga al menos los siguientes endpoints:
- Un endpoint para listar todos los países almacenados en la base de datos.
- Un endpoint para consultar un país por su código alpha-3, con el siguiente comportamiento:
- Buscar primero el país en la base de datos.
- Si el país ya existe en la base, devolverlo indicando que el origen de la información es la caché local.
- Si el país no existe en la base de datos:
  - Consultar la API externa RestCountries.
  - Extraer solo los campos necesarios.
  - Guardar el país en la base de datos.
  - Devolver la información indicando que el origen de la información es la API externa.
- Limite la información que se almacena y se expone del país a los campos definidos para el modelo, sin incluir todos los datos que devuelve RestCountries.

Provider para la API externa

La API externa no debe consumirse directamente desde el servicio de países, sino a través de un provider especializado, que:

- Defina una interfaz o contrato que represente el servicio de información de países (por ejemplo, una operación que permita obtener un país por su código).
- Implemente ese contrato utilizando la API de RestCountries, limitando los campos que se solicitan.
- Sea inyectado en el servicio de países mediante el sistema de inyección de dependencias de NestJS.

El objetivo es separar la lógica propia del dominio (gestionar países y su caché) de los detalles de infraestructura (llamadas HTTP, URL de la API, etc.).

Módulo de planes de viaje (TravelPlansModule)

El sistema debe contar con un módulo para gestionar planes de viaje, que:

- Defina una entidad o modelo de plan de viaje con al menos los siguientes atributos:
- Un identificador propio del plan de viaje.
- El código alpha-3 del país destino, asociado a un país existente.
- Un título o nombre del viaje.
- Fecha de inicio y fecha de fin del viaje.

- Notas o comentarios opcionales.
- Información de creación del registro.
    - Exponga al menos los siguientes endpoints:
- Un endpoint para crear un nuevo plan de viaje, que reciba los datos necesarios y:
- Valide la información de entrada (por ejemplo, formato de fechas, presencia de campos obligatorios, etc.).
- Verifique que el país asociado exista en la caché de países; si no existe, deberá solicitarlo a través del módulo de países para que sea creado.
- Guarde el plan de viaje en la base de datos.
    - Un endpoint para listar todos los planes de viaje registrados.
    - Un endpoint para consultar un plan de viaje específico por su identificador.
- Haga uso de DTOs y mecanismos de validación propios de NestJS para estructurar y validar los datos de entrada y salida.

Puntos a desarrollar

Paso 1 – Diseño del modelo de datos

1. Identificar y documentar los atributos que formarán parte del modelo de Country y de TravelPlan.
2. Establecer la relación lógica entre un plan de viaje y su país correspondiente (por código alpha-3 o relación directa según el ORM elegido).

Paso 2 – Implementación del módulo de países

1. Crear el módulo, servicio y controlador encargados de la gestión de países.
2. Implementar la lógica de caché:
3. Consulta en base de datos.
4. Consulta a la API externa solo cuando sea necesario.
5. Almacenamiento local de los países.
6. Asegurar que la API de países exponga solo la información definida en el modelo.

Paso 3 – Implementación del provider externo

1. Definir una abstracción para el servicio que proporciona información de países (contrato).
2. Implementar esa abstracción consumiendo la API de RestCountries, limitando los campos solicitados.
3. Registrar este provider en el módulo de países e inyectarlo en el servicio correspondiente.

Paso 4 – Implementación del módulo de planes de viaje

1. Crear el módulo, servicio y controlador de planes de viaje.
2. Definir DTOs para la creación de planes de viaje, incluyendo las validaciones necesarias.
3. Conectar el módulo de planes con el módulo de países, reutilizando la lógica de caché antes de guardar un plan.

Entrega

La entrega de este preparcial se realizará por medio de bloqueneon, se debe enviar un link a un repositorio público que contenga la solución.

El README de la entrega debe incluir:

Cómo ejecutar el proyecto: Instalación, configuración de la base de datos elegida y comando para correr la API.

Descripción mínima de la API: módulos tiene (Countries y TravelPlans) y propósito general.

Documentación de endpoints: Rutas implementadas y ejemplos breves de uso.

Explicación del provider externo: Cómo se consultan los países desde RestCountries.

Modelo de datos: Campos principales de Country y TravelPlan.

Pruebas básicas sugeridas: Consultar un país no cacheado / cacheado y crear un plan de viaje.