El objetivo es construir una API RESTful en NestJS para una aplicación de gestión de tareas (To-Do List). La API debe permitir a los usuarios autenticarse, gestionar sus tareas de manera privada (solo visibles para ellos) y proteger los endpoints mediante un sistema de sesión. Además, la API debe incluir seguridad con API key y persistencia en una base de datos.

Requerimientos

Autenticación y Manejo de Sesión:
Crear un módulo auth con un endpoint de POST /auth/login.
Implementar el login de usuario mediante un email y contraseña. Al iniciar sesión correctamente, el endpoint debe generar una sesión única para el usuario autenticado, la session debe tener una vigencia 10 minutos.

Endpoints CRUD de Tareas (solo accesibles para el usuario autenticado):
Crear un módulo tasks para gestionar las tareas.
El modelo de una tarea debe contener los siguientes campos:
id (UUID, autogenerado)
title (string, requerido)
description (string, opcional)
status (enum: "pending", "in-progress", "completed";

valor por defecto: "pending")
createdAt (fecha de creación)
updatedAt (fecha de actualización)
userId (relación con el usuario que creó la tarea)
Implementar los siguientes endpoints en el controlador de tareas:
POST /tasks: Crear una nueva tarea.
GET /tasks: Obtener todas las tareas del usuario autenticado.
GET /tasks/:id: Obtener una tarea por su ID, solo si pertenece al usuario autenticado.

PUT /tasks/:id: Actualizar una tarea existente, solo si pertenece al usuario autenticado.
DELETE /tasks/:id: Eliminar una tarea, solo si pertenece al usuario autenticado.

Persistencia de Datos:
Usar una base de datos relacional (por ejemplo, PostgreSQL o MySQL) para almacenar usuarios y tareas.
Configurar TypeORM como ORM para manejar la persistencia de datos.

Seguridad con API Key:

Se debe validar que el api key sea valido

Validación de Sesión en Endpoints:
Todos los endpoints de tareas deben validar que la sesión es válida antes de responder.

Documentación:
Generar la documentación de la API usando Swagger.

Buenas Prácticas:
Aplicar principios de SOLID y arquitectura hexagonal.
Manejar excepciones de forma adecuada utilizando los filtros de excepción de NestJS.
Incluir un archivo README.md que contenga:
Descripción del proyecto.
Instrucciones para configurar y ejecutar la API.
Ejemplos de uso de cada endpoint.
(Opcional) Contenerización con Docker

Criterios de Evaluación

Funcionalidad: La API debe cumplir con todos los requisitos funcionales, incluyendo la autenticación, seguridad con API key, manejo de sesiones y operaciones CRUD.
Código Limpio y Buenas Prácticas
Documentación: La documentación debe ser clara, completa y fácil de seguir.
Opcional (Docker): La contenerización es un plus, pero no es un requisito obligatorio.

Entrega

El candidato debe entregar un repositorio de Git (o enlace a un repositorio público como GitHub, GitLab, etc.) con el proyecto completo, incluyendo la documentación en el archivo README.md y las configuraciones necesarias.