E-commerce Riwi SportsLine

Tabla de contenido

E-commerce Riwi SportsLine *1*

Épica: Migración del e-commerce Riwi SportsLine a NestJS 2

Semana 1 - Historia de Usuario: Fundamentos de NestJS y migración del setup base 3

Semana 2 - Historia de Usuario: Integración de ORM y persistencia con TypeORM 4

Semana 3 - Historia de Usuario: Arquitectura modular y DTOs 5

Semana 4 - Historia de Usuario: Middleware, filtros e interceptores 6

Semana 5 - Historia de Usuario: Autenticación con JWT, roles y permisos desde base de datos 7

Semana 6 - Historia de Usuario: Autenticaciones avanzadas (x-api-key y OAuth) 8

Semana 7 - Historia de Usuario: Pruebas y análisis estático 9

Épica: Migración del e-commerce Riwi SportsLine a NestJS

Descripción:
La empresa **Riwi SportsLine** busca evolucionar su arquitectura backend migrando el proyecto desarrollado en **Express** hacia el **framework NestJS**, adoptando principios de modularidad, inyección de dependencias, pruebas automatizadas avanzadas y mejores prácticas de seguridad.
El objetivo es que el equipo viva la experiencia de **migrar un proyecto existente** a un entorno empresarial más escalable y mantenible, aprovechando el **CLI y arquitectura de NestJS**.

Semana 1 - Historia de Usuario: Fundamentos de NestJS y migración del setup base

Descripción:

Como desarrollador backend quiero comprender la estructura y fundamentos de **NestJS**, instalando su CLI y migrando la base del proyecto para iniciar la conversión del backend Express hacia la arquitectura modular de Nest.

Criterios de aceptación:

- Proyecto creado con **Nest CLI** e inicializado desde el fork de GitHub.
- Integración de **TypeScript**, **ESLint**, **Prettier** y configuración de entorno (`.env`) bajo buenas prácticas.
- Migración del setup de conexión a **PostgreSQL** con TypeORM.
- Validación del arranque del servidor con variables configuradas profesionalmente (config module).

Tareas base de migración:

1. Crear el nuevo proyecto con `nest new riwi-sportsline` y vincularlo con el fork del repositorio original.
2. Configurar las variables de entorno (`ConfigModule`) y migrar las del proyecto Express.
3. Sustituir Sequelize por TypeORM con una entidad base (ej. Usuario).
4. Documentar el proceso inicial en el README del nuevo fork.

Semana 2 - Historia de Usuario: Integración de ORM y persistencia con TypeORM

Descripción:
Como desarrollador quiero usar **TypeORM** para manejar las entidades y relaciones del ecommerce, reemplazando Sequelize y aprovechando los decoradores de entidades de NestJS.

Criterios de aceptación:

- Entidades de Usuario, Producto, Cliente y Pedido creadas con **TypeORM**.
- Relaciones correctamente definidas (OneToMany, ManyToOne, etc.).
- Migraciones y seeders implementados con CLI de TypeORM.
- Repositorios personalizados para operaciones CRUD.

Tareas base de migración:

1. Migrar modelos de Sequelize a entidades de TypeORM.
2. Configurar las relaciones entre entidades.
3. Implementar migraciones y seeds iniciales.
4. Validar consultas básicas desde los servicios.

Semana 3 - Historia de Usuario: Arquitectura modular y DTOs

Descripción:
Como desarrollador necesito estructurar el proyecto bajo una **arquitectura modular** en Nest, implementando **controladores, servicios y DTOs**, garantizando una separación de responsabilidades y un manejo profesional de variables de entorno.

Criterios de aceptación:

- Creación de módulos de `usuarios`, `productos`, y `clientes` usando el **CLI de Nest**. • Uso de DTOs con `class-validator` y `class-transformer`.

- Configuración robusta de `.env` y validación con `ConfigModule`.
- Código estructurado con principios **SOLID** y dependencias inyectadas correctamente.

Tareas base de migración:

1. Generar los módulos, controladores y servicios de usuario, producto y cliente.
2. Migrar los DTO existentes en Express al formato compatible con Nest.
3. Centralizar la configuración de entorno (puerto, DB, claves, etc.).
4. Actualizar los controladores para usar inyección de dependencias.
5. Integrar pruebas unitarias

Semana 4 - Historia de Usuario: Middleware, filtros e interceptores

Descripción:
Como desarrollador quiero aprovechar el sistema de **middleware, exception filters, guards, interceptors y pipes** de NestJS para mejorar la robustez y control del flujo de peticiones.

Criterios de aceptación:

- Implementación de middleware de logging y validación.
- Uso de ExceptionFilter global.
- Guards personalizados para roles.

- Interceptors para formateo de respuestas y manejo de tiempo de ejecución.

Tareas base de migración:

1. Implementar un middleware global de auditoría.
2. Crear ExceptionFilter para errores HTTP.
3. Implementar Guards basados en roles y permisos.
4. Añadir interceptors personalizados (ej. logging de tiempo).
5. Integrar pruebas unitarias

Semana 5 - Historia de Usuario: Autenticación con JWT, roles y permisos desde base de datos

Descripción:
Como administrador deseo un sistema de autenticación con **JWT + Refresh Token**, donde los **roles y permisos** provengan de la base de datos y no estén quemados en código.

Criterios de aceptación:

- Autenticación y autorización implementadas con Passport y JWT.
- Refresh Token funcional.
- Roles y permisos gestionados desde base de datos.
- Protección de rutas mediante Guards y decoradores personalizados.

Tareas base de migración:

1. Configurar módulo `Auth` y estrategias JWT + Refresh Token.
2. Migrar el sistema de roles y permisos desde Express a BD.
3. Implementar guards y decoradores para proteger endpoints.
4. Actualizar documentación Swagger con autenticación.

Semana 6 - Historia de Usuario: Autenticaciones avanzadas (x-api-key y OAuth)

Descripción:
Como arquitecto de software quiero implementar autenticaciones adicionales mediante **xapi-key** y **OAuth**, permitiendo que sistemas externos se integren de forma segura al ecommerce.

Criterios de aceptación:

- Implementación de autenticación basada en x-api-key.
- Integración OAuth2 para login con terceros.
- Control de permisos por API key.
- Documentación de flujos de autenticación avanzada.

Tareas base de migración:

1. Crear modulo para autenticación x-api-key.
2. Implementar autenticación OAuth2 (por ejemplo, con Google).
3. Agregar validación de scopes y permisos por key.
4. Documentar los nuevos flujos en Swagger.

Semana 7 - Historia de Usuario: Pruebas y análisis estático

Descripción:
Como líder técnico necesito asegurar la calidad y mantenibilidad del código con **Swagger**, **SonarQube** y **linters**.

Criterios de aceptación:

- Swagger actualizado y documentando DTOs, respuestas y errores (Obligatorio).
- Pruebas de **caja blanca y negra** (WorkShop - Aplicación)
- Análisis de calidad con **SonarQube** (WorkShop - Aplicación).
- Configuración de linters y pre-commit hooks (WorkShop - Aplicación).

Tareas base de migración:

1. Configurar análisis de código estático con SonarQube.
2. Realizar
3. Implementar husky/pre-commit para control de calidad.