MÓDULO 7. DESARROLLO AVANZADO DE BACKEND Y APIS

Tarea 4: Seguridad, pruebas y despliegue

Diego Matilla De La Cierva

Descripción del Desarrollo

El proyecto desarrollado consiste en una aplicación backend completa que implementa una API RESTful centrada en la gestión de productos. Este sistema se ha construido empleando Node.js como entorno de ejecución y Express como framework principal para el enrutado y la creación de los controladores de la aplicación. La persistencia de datos se ha resuelto mediante el uso de Sequelize, un ORM que facilita la interacción con bases de datos SQL, y en este caso particular se ha utilizado tanto MySQL en entornos de desarrollo locales como PostgreSQL en producción (Heroku), adaptando la configuración para que sea flexible según el entorno de ejecución.

Desde el inicio, se definió una estructura de carpetas clara que separa la responsabilidad de cada parte del código. Por un lado, el directorio models contiene las definiciones de los modelos Sequelize, concretamente el modelo Product, que representa los registros de productos en la base de datos, y el modelo User, que gestiona la autenticación y almacenamiento seguro de las credenciales de los usuarios. El hashing de contraseñas se lleva a cabo con la biblioteca bcrypt, que genera un salt único por usuario, de manera que las contraseñas no quedan almacenadas en texto plano. Por otro lado, los controladores se encuentran en la carpeta controllers, donde se define la lógica de negocio de cada ruta, como la creación de productos, su modificación o eliminación, así como las funciones de registro y login de los usuarios. En la carpeta routes se declaran las rutas de la API, organizadas por dominio funcional (productos y autenticación), mientras que en middlewares se implementa el middleware de protección mediante JWT, que verifica que las peticiones a rutas sensibles incluyan un token válido antes de permitir el acceso.

El ciclo de vida de la aplicación comienza con el registro de un usuario. Cuando un usuario nuevo se registra, la contraseña se cifra automáticamente antes de ser almacenada. Posteriormente, el usuario puede iniciar sesión y obtener un token JWT que se utilizará en todas las peticiones a rutas protegidas, por ejemplo, para crear, editar o eliminar productos. Este sistema de autenticación garantiza que solo los usuarios autenticados puedan realizar operaciones de modificación, mientras que la lectura de datos (consultas públicas) permanece accesible sin necesidad de autenticación. El servidor Express expone los endpoints de la API y gestiona las solicitudes HTTP, además de incluir middlewares que procesan las cabeceras, los cuerpos JSON y los tokens de autorización.

Para asegurar la calidad y fiabilidad del sistema, se han desarrollado pruebas automatizadas utilizando Jest y Supertest. Las pruebas cubren tanto la autenticación como las operaciones principales del CRUD de productos. Este enfoque permite verificar que la API funciona correctamente y que las rutas protegidas no permiten accesos sin autorización. Asimismo, se ha incorporado un flujo de integración continua con GitHub Actions, de modo que cada vez que se realiza un cambio en el repositorio, se ejecutan las pruebas de manera automática, garantizando que el código desplegado se mantiene estable y funcional.

En términos de despliegue, la aplicación se ha preparado para funcionar de forma contenerizada mediante Docker, facilitando su ejecución en entornos locales o en servidores.

Conclusiones y Observaciones

A modo de conclusión, este proyecto ha permitido experimentar de forma práctica con muchos de los componentes esenciales que intervienen en el desarrollo de una aplicación backend robusta. A lo largo del proceso se ha podido comprobar que la construcción de un backend serio y profesional no consiste únicamente en exponer unas cuantas rutas que gestionen datos, sino en articular un conjunto de capas y prácticas que garanticen la seguridad, la integridad de la información, la escalabilidad y la facilidad de mantenimiento de todo el sistema.

En este sentido, la autenticación mediante JSON Web Tokens ha resultado clave para controlar el acceso a los recursos y permitir que solo los usuarios que se identifican correctamente puedan realizar operaciones de modificación o eliminación. Este enfoque refleja un estándar de seguridad ampliamente utilizado en aplicaciones reales y se complementa con la protección del almacenamiento de contraseñas mediante hashing con salt, que previene que datos sensibles queden expuestos aunque la base de datos sufriera un acceso no autorizado. Además, la inclusión de middlewares específicos para validar tokens y sanitizar datos de entrada contribuye a minimizar riesgos de ataques comunes como inyecciones o el uso indebido de las rutas.

Otro aspecto fundamental del proyecto ha sido la incorporación de pruebas automatizadas. Estas pruebas no solo ayudan a detectar errores de forma temprana, sino que también transmiten confianza a la hora de evolucionar el sistema, dado que cualquier cambio en el código se valida automáticamente antes de ser desplegado. La integración continua con GitHub Actions refuerza esta práctica, permitiendo que el ciclo de desarrollo sea más ágil y predecible. Este proceso evidencia que el testing es un pilar imprescindible si se aspira a desarrollar software fiable y mantenible.

El despliegue en la nube, concretamente en Heroku, ha permitido comprobar la importancia de preparar adecuadamente la configuración del entorno, gestionando variables sensibles como las credenciales de la base de datos y adaptando el código para funcionar correctamente en producción con conexiones seguras y escalables. La posibilidad de contenerizar la aplicación con Docker proporciona, además, una vía alternativa de distribución y despliegue que resulta muy valiosa en proyectos más grandes o cuando se trabaja con microservicios.