09/04/2025

Moises Pinzon Xiques

Solución

Prueba Técnica

1. Pregunta Teórica sobre Principios de Backend y RESTful ●

¿Qué es REST y cuáles son los principales principios que debes seguir al diseñar un servicio RESTful? Explica cómo manejarías el versionado de una API REST y por qué es importante.

respuesta

(Transferencia de estado representacional)principios son :(Representational State Transfer) es un estilo de arquitectura para el desarrollo de servicios web que utiliza HTTP y se basa en recursos identificados por URLs. Los principales principios son:

1. **Apátrida** : Cada solicitud debe contener toda la información necesaria para procesarla.
2. **Cliente-servidor** : Separación entre cliente y servidor, permitiendo la escalabilidad.
3. **Cacheable** : Las respuestas deben indicar si pueden ser almacenadas en caché.
4. **Interfaz uniforme**: Uso de métodos HTTP estándar (GET, POST, PUT, DELETE).
5. **Sistema en capas** : Puede haber múltiples capas entre el cliente y el servidor.

Versionado de API: Se puede manejar mediante la inclusión de la versión en la URL (`/api/v1/resource`). Es importante para asegurar la compatibilidad con los clientes existentes mientras se introducen nuevas funcionalidades sin afectar a los usuarios anteriores.

2. Ejercicio de Implementación de un Servicio API RESTful en Java ●

Crea un servicio RESTful en Java utilizando Spring Boot (o cualquier framework que prefiera el candidato)
. El servicio debe permitir realizar las siguientes operaciones sobre una lista de "productos":

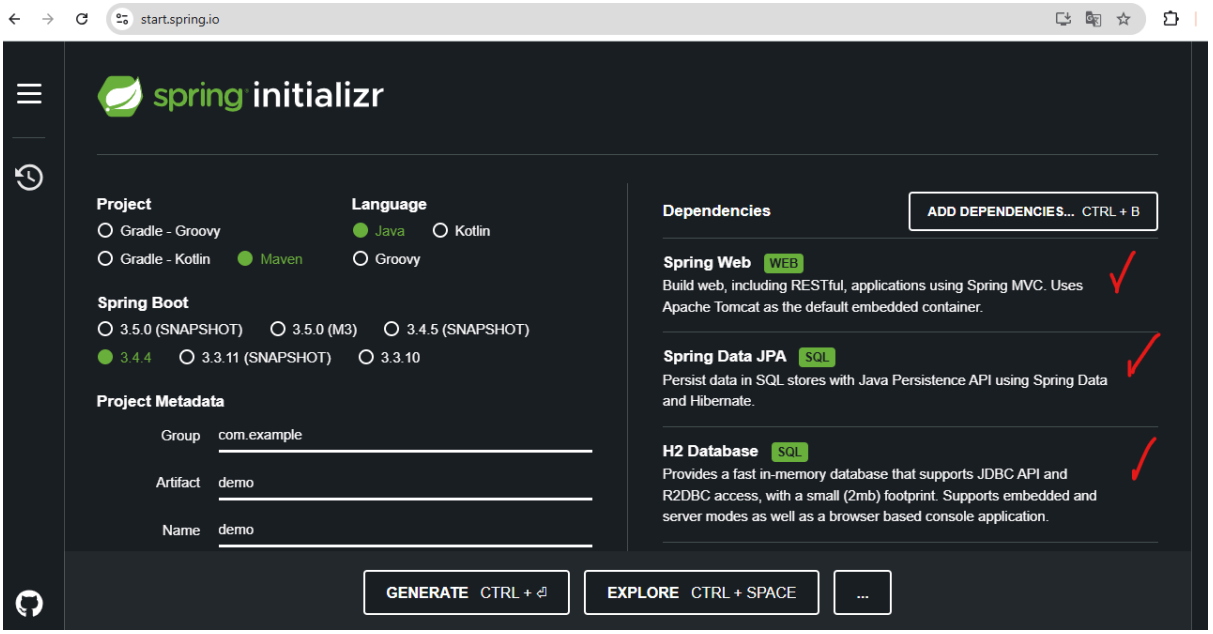
respuesta

Creación de Proyecto con spring Boot
Se crea proyecto con Dependencias necesarias

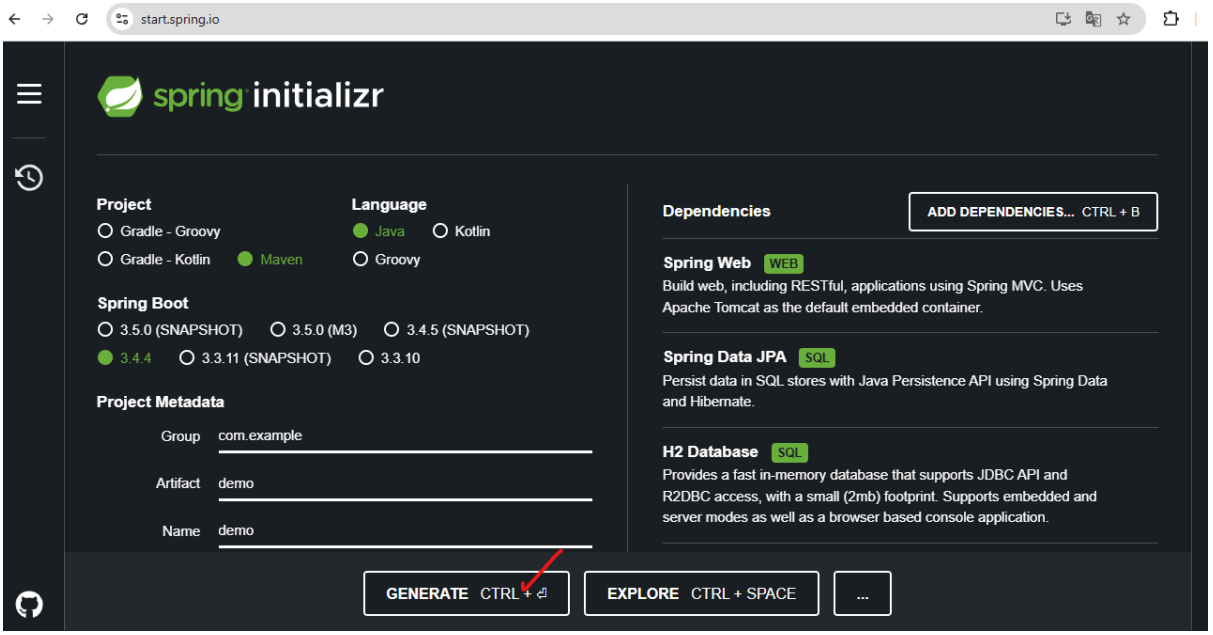
*Validar versión de java , de lo contrario Instalar Java

```
C:\Users\Acer A5\Documents>java --version
java 21.0.4 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 21.0.4+8-LTS-274)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.4+8-LTS-274, mixed mode, sharing)
```

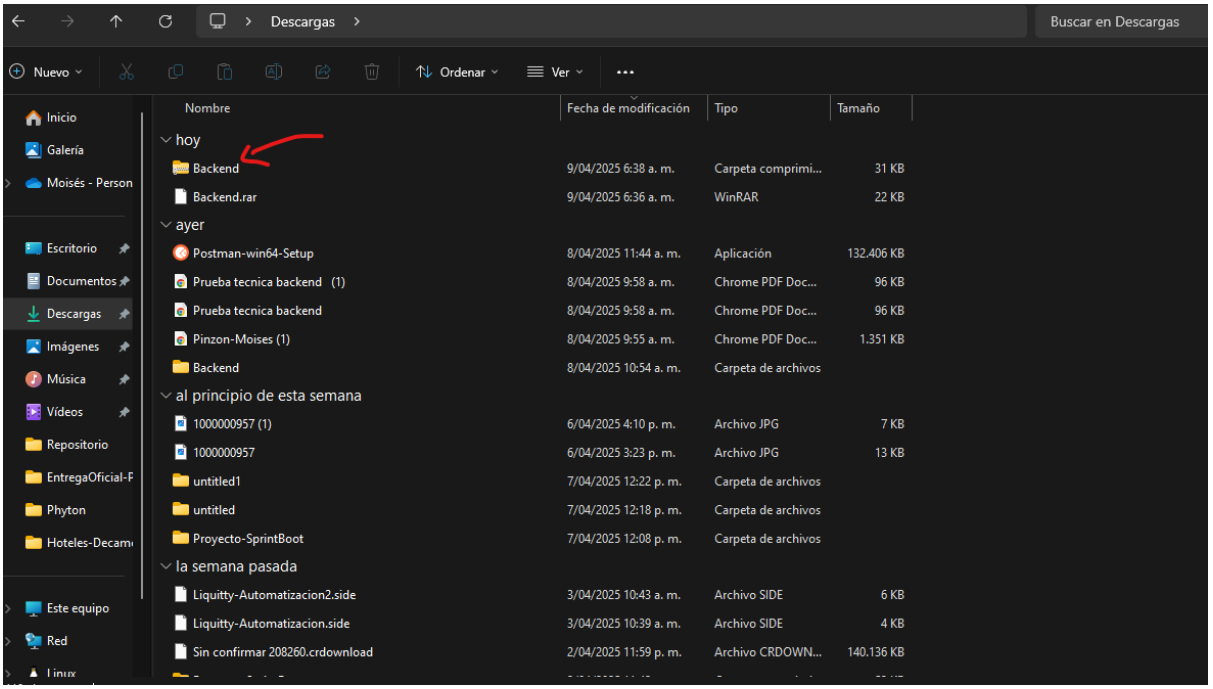
página principal de sprint Boot y Creación de Proyecto, con dependencias necesarias



-Luego descarga Proyecto



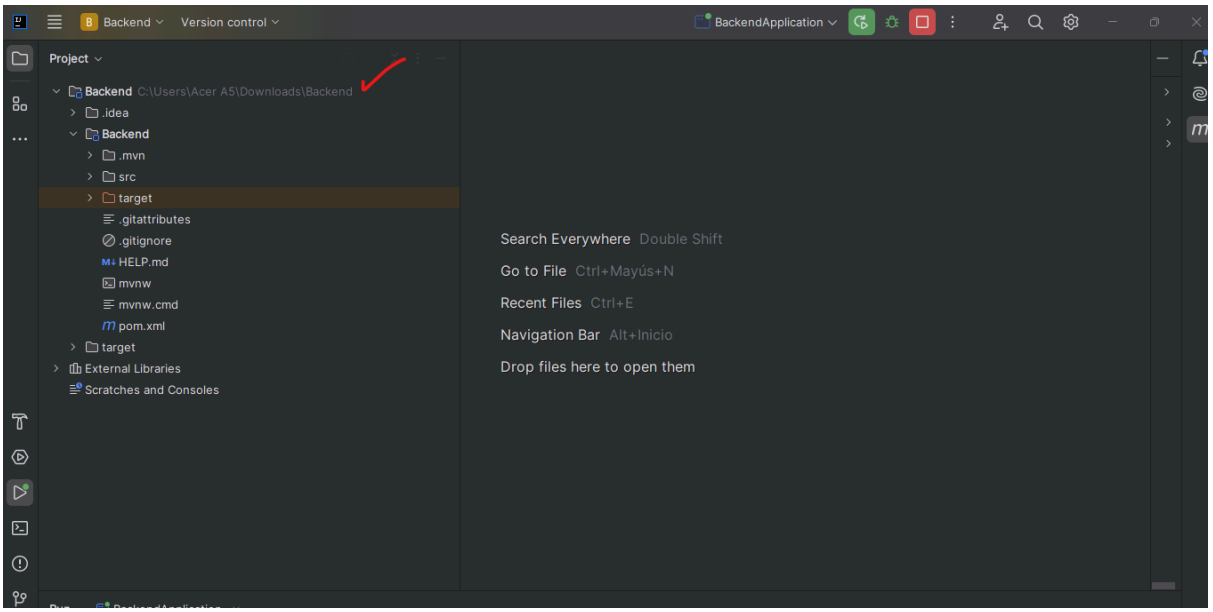
-Una vez descargado el proyecto
Buscamos ubicación del archivo y se descomprime



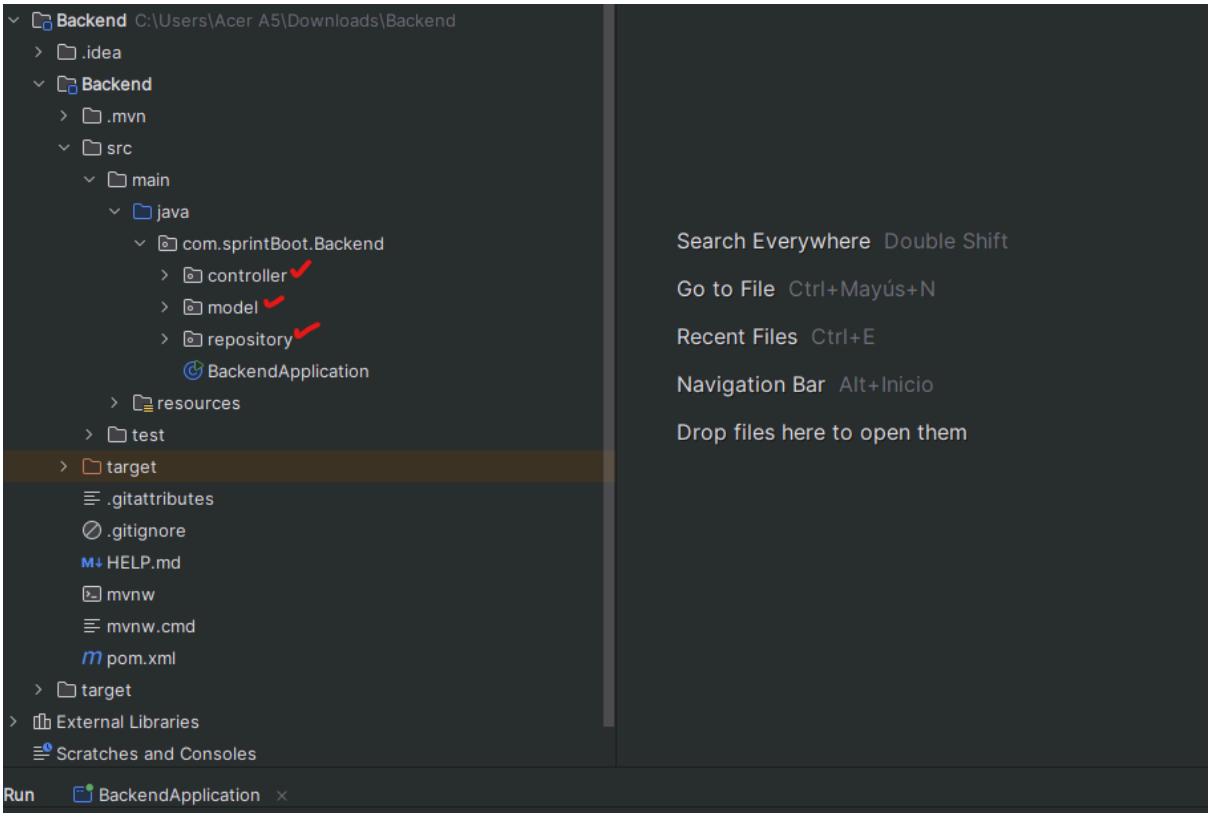
Importante descargar ide, Para visualizar Proyecto, Recomendado Intellij Idea

https://www.jetbrains.com/es-es/idea/download/?section=windows

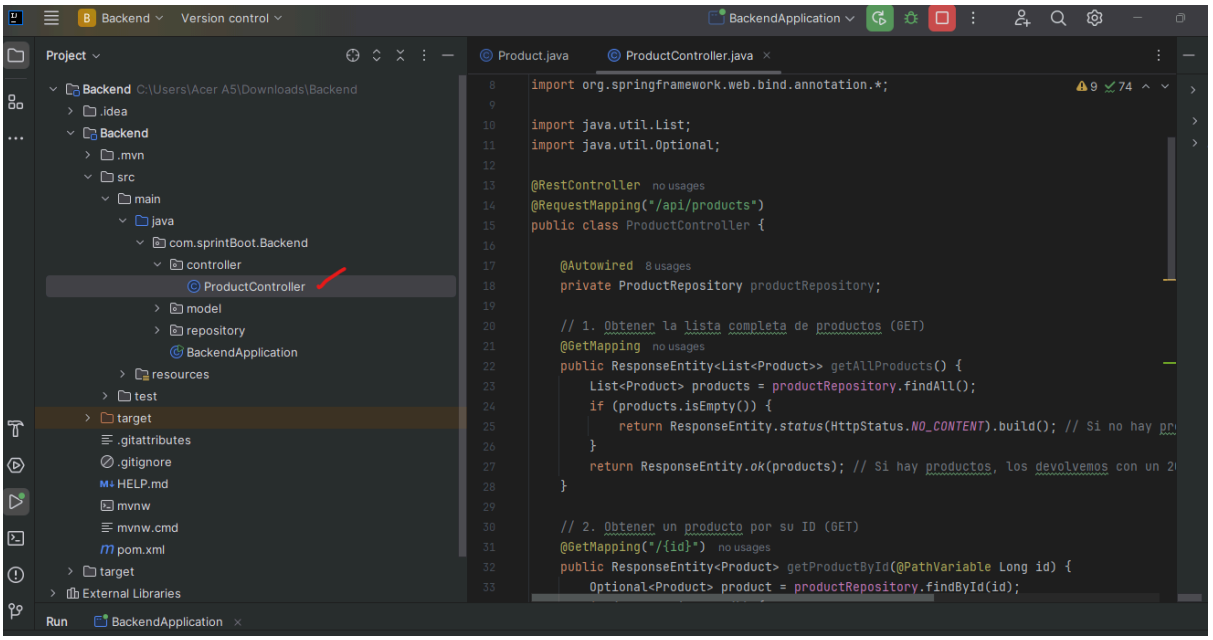
Luego abrimos proyecto con Intelij Idea



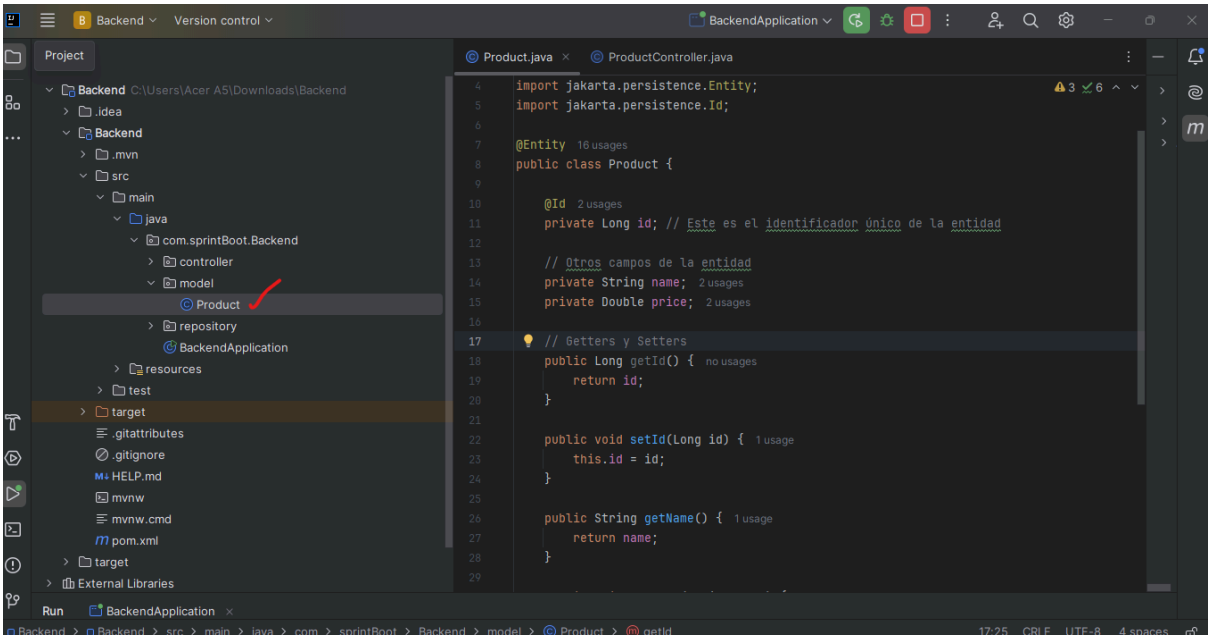
Ahora sigue el desarrollo de las clases , y métodos necesarios para que los servicios funcionen, a continuación creo carpetas



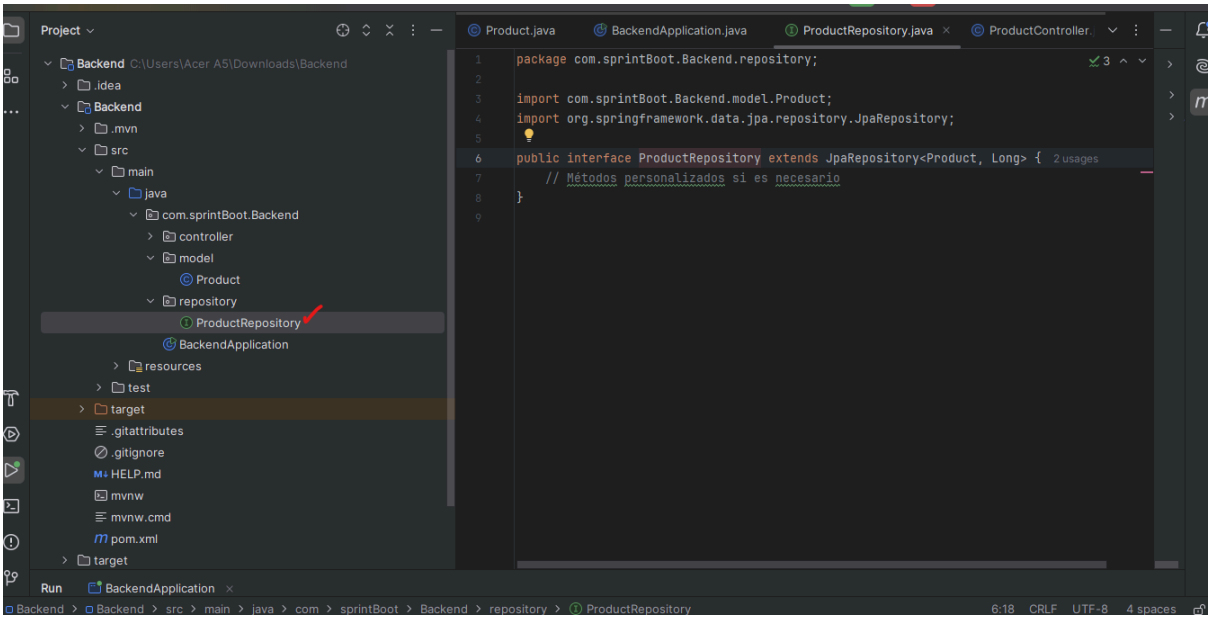
Controlador



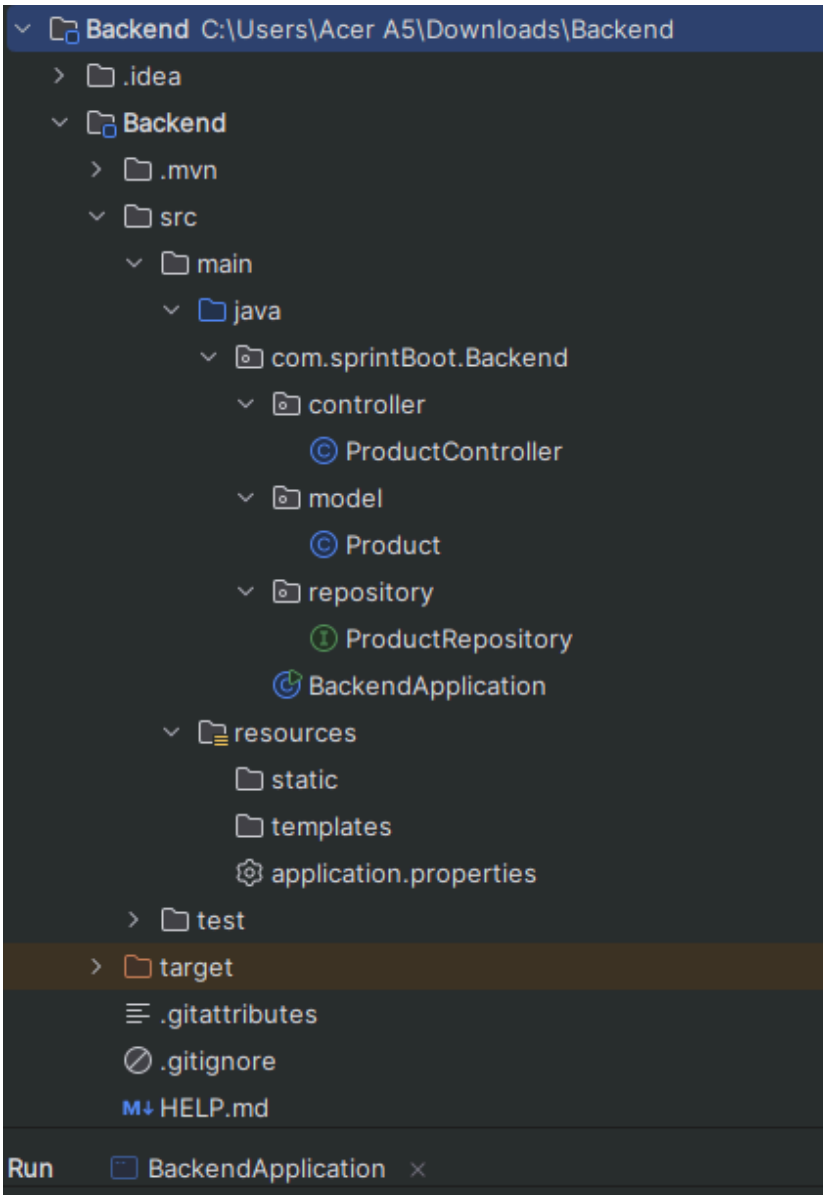
modelo



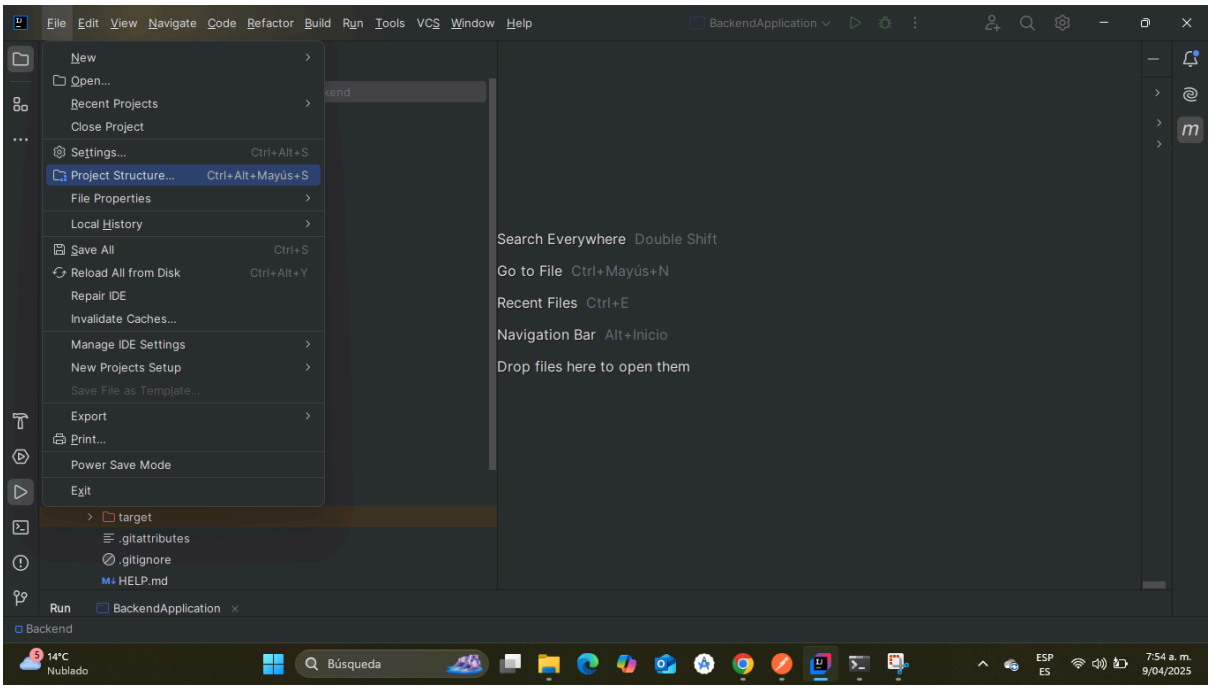
repository



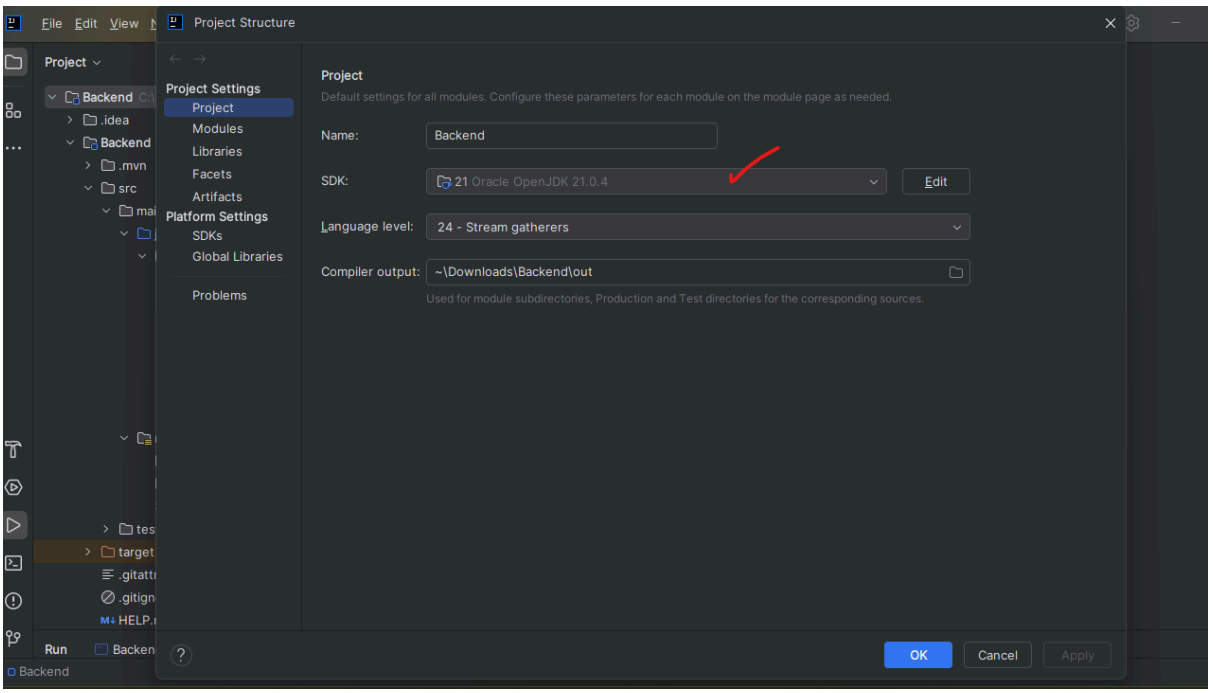
Ejecutar Proyecto



Para ejecutar proyecto validamos primero que en la estructura ya tenga java , nos dirigimos a opcion **File , y Project structure**

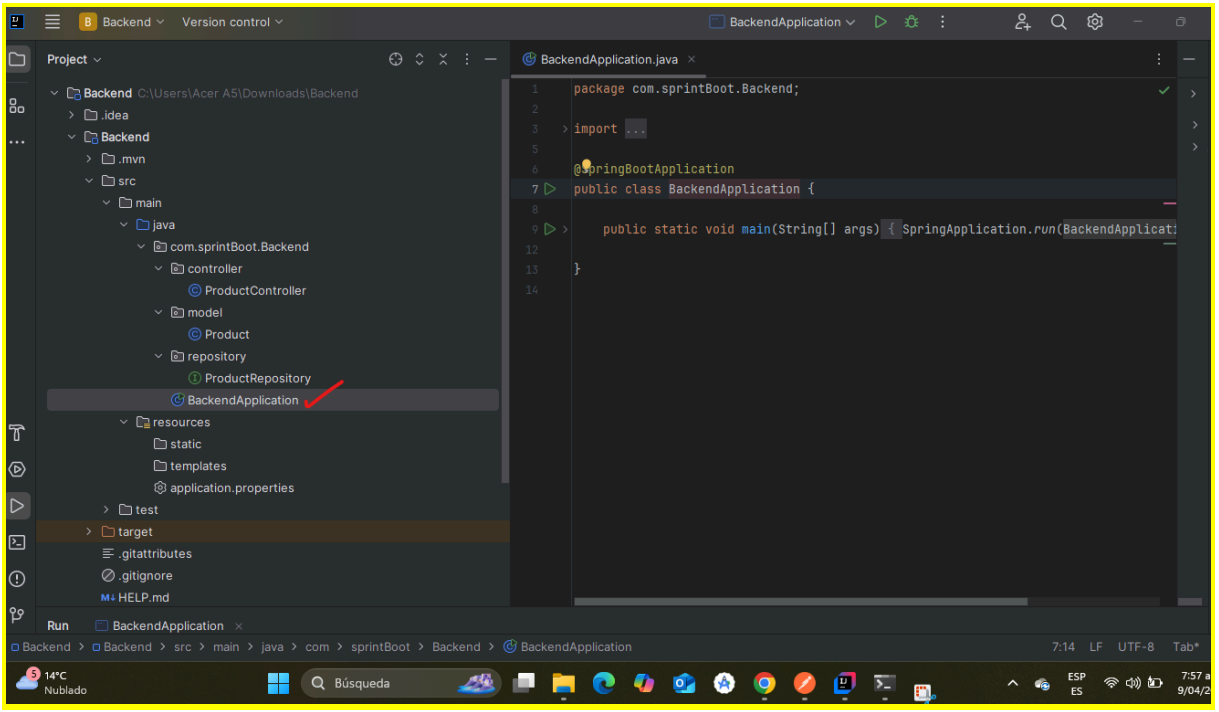


Se valida que tenga la versión de java correcta ,

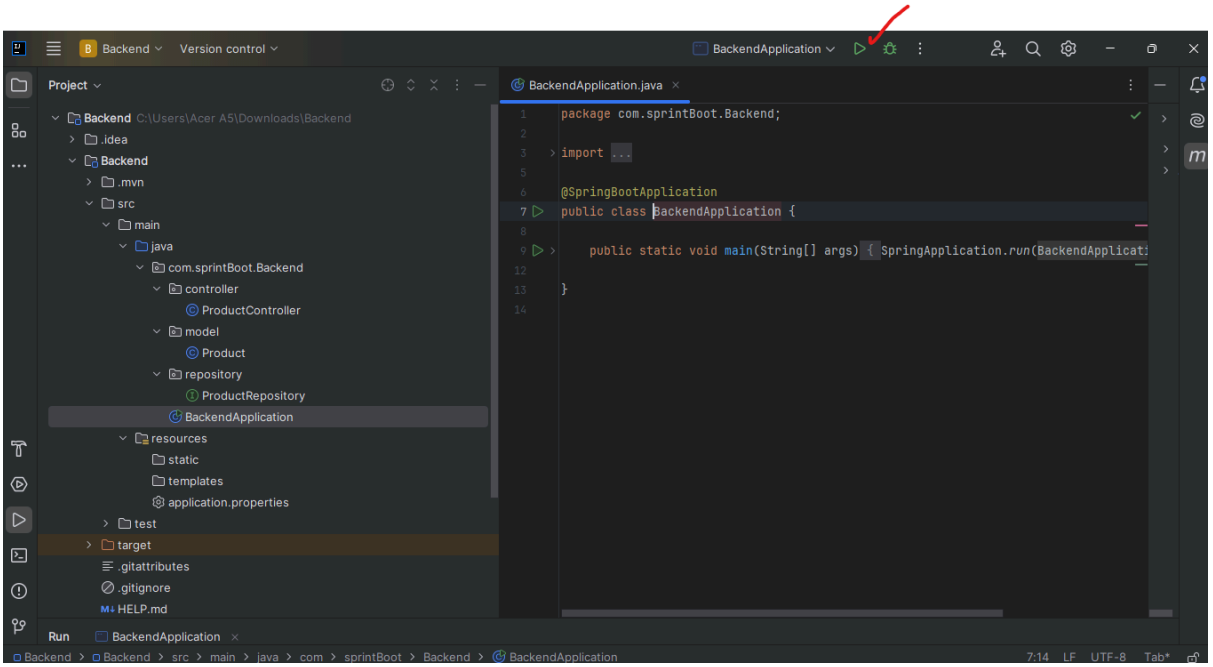


Una vez validada la estructura haora ejecutar proyecto,

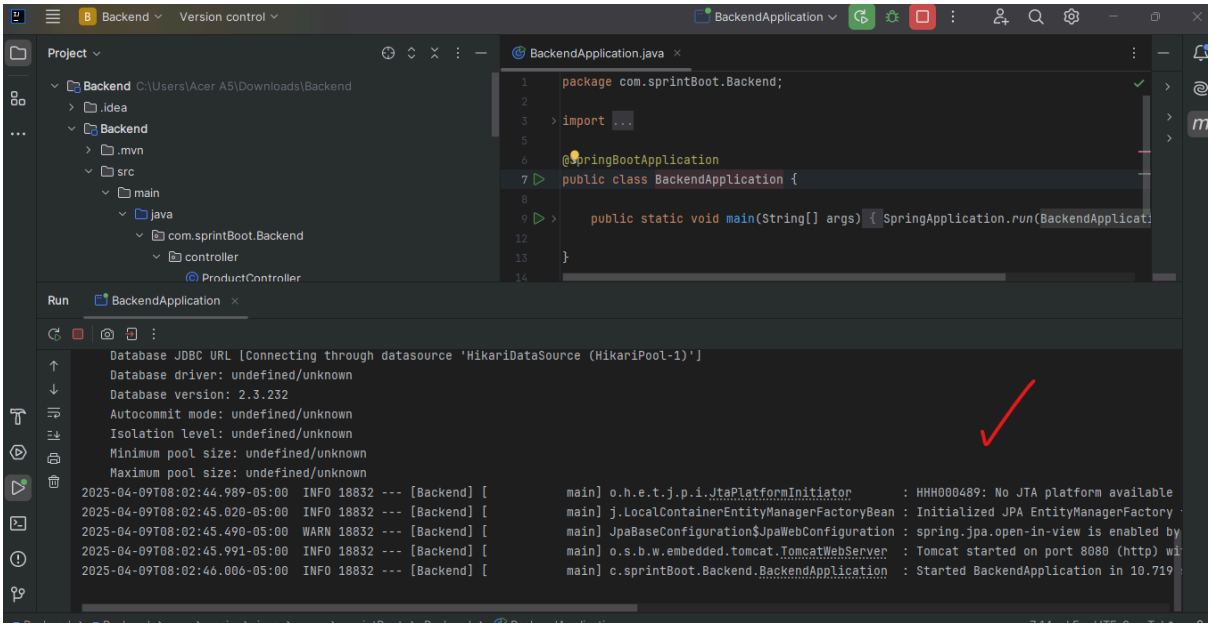
Selecionar
BackendApplication



Luego Seleccionar la flecha verde para ejecutar proyecto



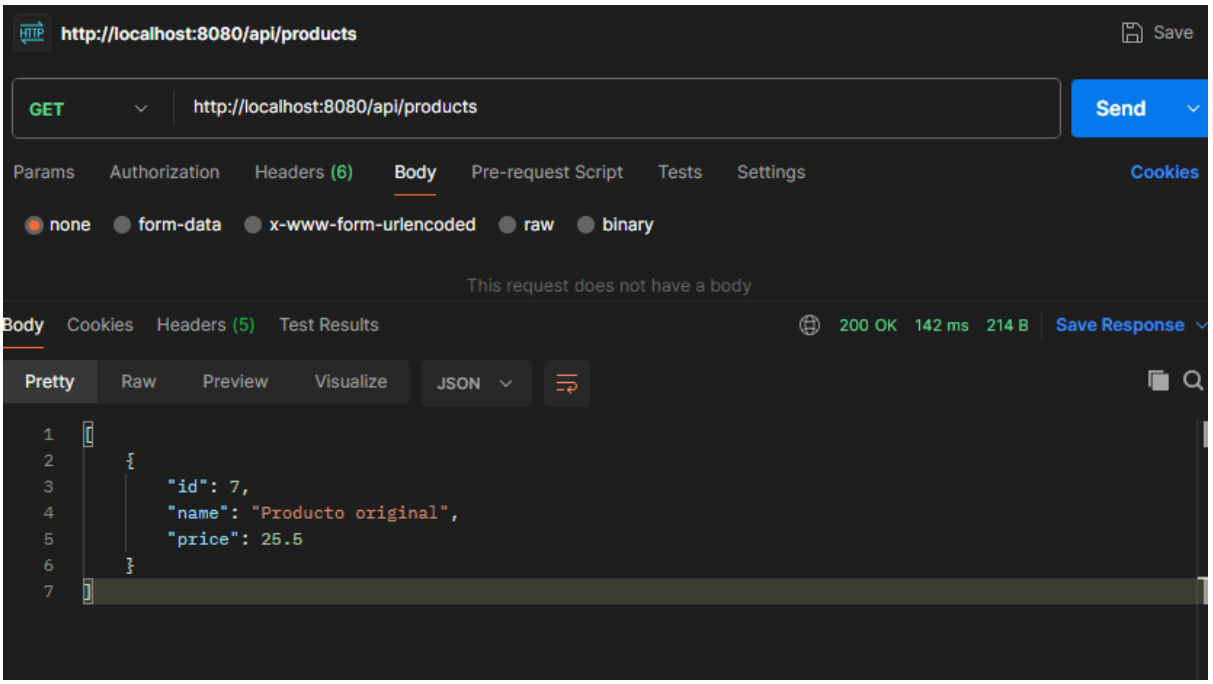
proyecto funcionando



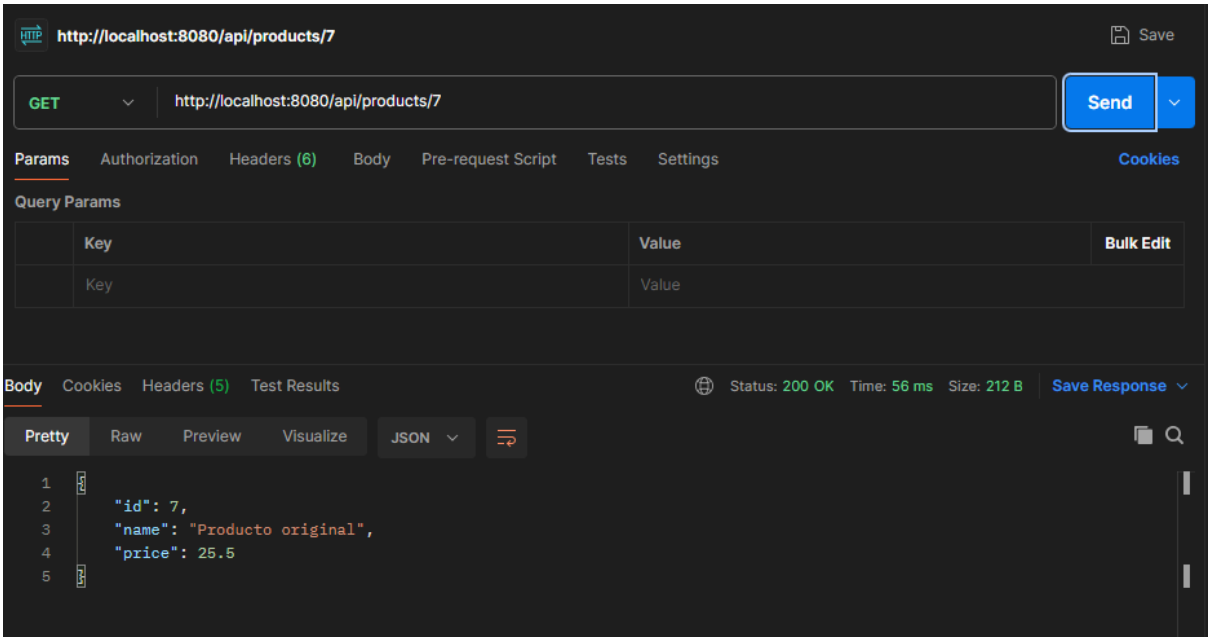
Haora ya se pueden validar todos los servicios creados en postman

1. Obtener la lista completa de productos (GET).

http://localhost:8080/api/products

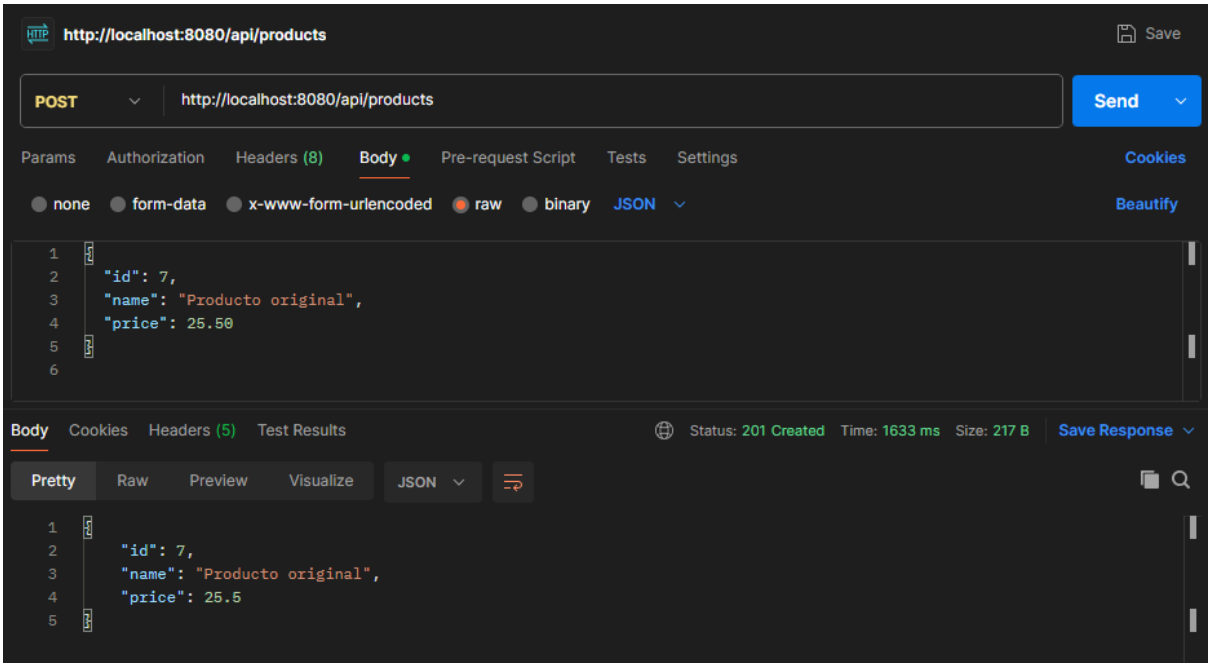


2. Obtener un producto por su ID (GET).
http://localhost:8080/api/products/7

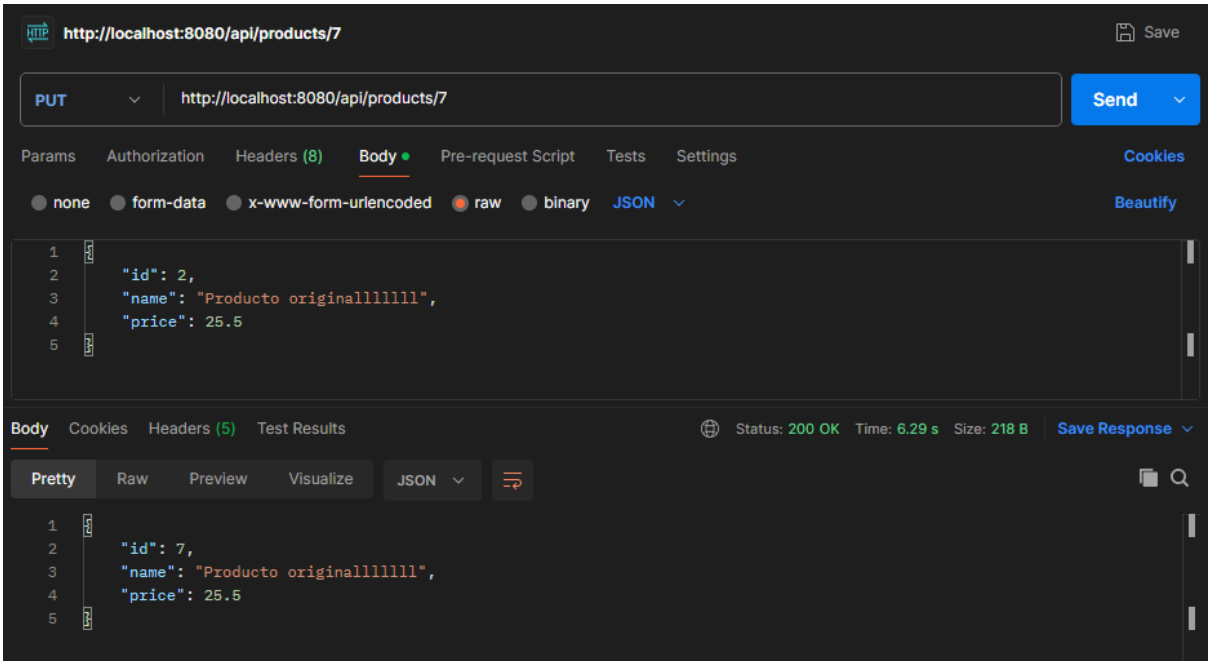


3. Crear un nuevo producto (POST).
http://localhost:8080/api/products

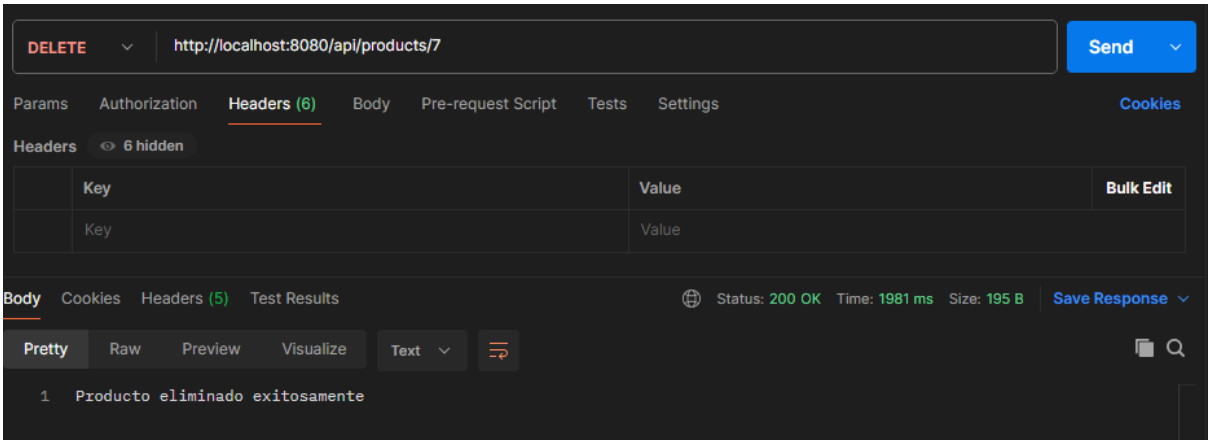
{
"id": 7,
"name": "Producto original",
"price": 25.50
}



4. Actualizar la información de un producto (PUT).
http://localhost:8080/api/products/7



5. Eliminar un producto (DELETE).
http://localhost:8080/api/products/7



3. Pregunta sobre Optimización de Rendimiento

● Imagina que tienes una API que consulta una base de datos para obtener una lista de productos. Al realizar consultas muy frecuentes, has notado que la aplicación comienza a tener tiempos de respuesta muy lentos. ¿Qué estrategias implementarías para mejorar el rendimiento de esta API? Explica tu enfoque paso a paso.

respuesta

Para mejorar el rendimiento de la API, implementaría las siguientes estrategias:

Cacheo de respuestas : Guardaría en caché las respuestas frecuentes (por ejemplo, usando Redis) para evitar consultas repetidas a la base de datos.

Optimización de consultas : Revisaría las consultas SQL, agregando índices a las columnas más consultadas para acelerar la búsqueda.

Paginación : Implementaría paginación en las respuestas para limitar la cantidad de datos devueltos por consulta.

Limitación de consultaslimitadores de tasa (rate) para evitar picos de carga excesiva en la base de datos .: Usaría limitadores de tasa (rate limiting) para evitar picos de carga excesiva en la base de datos.

Base de datos escalables : Si el tráfico es muy alto, consideraría dividir la base de datos o usar bases de datos más escalables.

4. Ejercicio Práctico de Lectura y Depuración de Logs

● Archivo de log simulado de una aplicación Java que presenta un error común de rendimiento o un error de conexión a la base de datos.

respuesta

Solución:

Revisar la configuración de la base de datos:

Verificar si hay problemas en la configuración de conexión a la base de datos, como credenciales incorrectas, dirección del servidor incorrecta o puertos bloqueados.

Asegurarse de que el pool de conexiones esté correctamente configurado para manejar múltiples conexiones concurrentes sin agotar los recursos.

Ajustar el tiempo de espera (timeout):

Revisar y ajustar los valores de timeout de conexión tanto en la configuración de la base de datos como en el código de la aplicación, para que la conexión no falle tan rápidamente.

Aumentar el tiempo de espera en la configuración de la base de datos para conexiones lentas.

Monitorear y optimizar la base de datos:

Asegurarse de que la base de datos esté optimizando las consultas, ya que los errores de conexión podrían deberse a sobrecarga o falta de recursos en el servidor de base de datos.

Utilizar herramientas de monitoreo para identificar cuellos de botella o problemas en el servidor de base de datos.

Reintentar la conexión y manejo de errores:

Implementar un mecanismo de reintentos más robusto cuando haya fallos de conexión a la base de datos. Esto puede incluir una lógica que intente reconectar varias veces antes de devolver un error al usuario.

Optimización de la API:

Mejorar los tiempos de respuesta de las APIs asegurando que las consultas a la base de datos estén optimizadas y utilizando técnicas como caché para reducir la carga en la base de datos.

Conclusión

El problema se centra en la conexión y el rendimiento de la base de datos, lo que afecta directamente a la API y provoca tiempos de respuesta elevados y errores internos del servidor. Para resolverlo, se deben revisar y mejorar tanto la configuración de la base de datos como la gestión de la conexión y el rendimiento de la aplicación.

5. Ejercicio de Integración con AWS (Simulado)

● Imagina que estás implementando una aplicación backend en AWS. Esta aplicación debe ser escalable y contar con las siguientes características:
1. Despliegue en EC2 o Elastic Beanstalk.
2. Almacenamiento de archivos estáticos utilizando S3.
3. Monitorización y alertas de la aplicación usando CloudWatch.
4. Seguridad con IAM para otorgar permisos. ● Describe los pasos que seguirías para configurar estos servicios en AWS y cómo gestionarías la seguridad, el escalado y la monitorización.

respuesta

Despliegue (EC2 o Elastic Beanstalk) :

- Usaría Elastic Beanstalk para simplificar el despliegue**para** simplificar el despliegue y gestión automática del entorno. Si necesito mayor control, usaría EC2 con una configuración de escalado automático .con una configuración de escalado automático.

Almacenamiento en S3 :

- Crearía un bucket en S3 para almacenar archivos est**ápara** almacenar archivos estáticos y configurar políticas de acceso adecuadas para que solo los servicios necesarios puedan acceder a él.

Monitorización y alertas (CloudWatch) :

- Configuraría CloudWatch para monitorear el rendimiento de la**para** monitorear el rendimiento de la aplicación y establecer alertas para recibir notificaciones si algún parámetro supera los umbrales definidos.

Seguridad (IAM) :

- Usaría IAM para crear rolespara crear roles y políticas que otorguen permisos mínimos necesarios a los servicios (por ejemplo, acceso a S3 o CloudWatch) y aseguraría que las claves y credenciales estén protegidas.

Escalado : Usaría autoescalado tanto en EC2 como en Elastic Beanstalk para ajustarse automáticamente a la demanda del tráfico.

6. Pregunta sobre Microservicios

● En un entorno de microservicios, ¿cómo manejarías la comunicación entre los diferentes servicios? Explica las ventajas y desventajas de usar REST frente a gRPC o RabbitMQ para la comunicación entre microservicios.

respuesta

En un entorno de microservicios, manejaría la comunicación mediante REST , gRPC o RabbitMQ según el caso :o RabbitMQ según el caso:

- **REST : Ideal para comunicación síncrona y servicios expuestos a clientes externos. Es fácil de usar y bien soportado, pero puede ser más lento y menos eficiente para comunicaciones de alta frecuencia.**

  Ventaja : Sencillo y ampliamente adoptado.

  Desventaja : Mayor latencia, no ideal para comunicación de alta eficiencia.

- **gRPC : Perfecto para comunicación síncrona entre microservicios internos, especialmente cuando se necesita alta velocidad y eficiencia.**

  Ventaja : Rápido, eficiente (basado en HTTP/2) y soporta contratos estrictos.

  Desventaja : Requiere mayor configuración y es menos amigable para clientes externos.

- **RabbitMQ : Usado para comunicación asíncrona, ideal para tareas desacopladas y procesamiento en segundo plano.**

  Ventaja : Desacopla servicios y mejora la resiliencia.

Desventaja : Mayor complejidad en la gestión de colas y manejo de errores.

La elección depende del caso de uso: REST para interacción externa, gRPC para microservicios internos de alto rendimiento y RabbitMQ para tareas asíncronas.

7. Pregunta sobre Herramientas y Frameworks

● ¿Qué herramientas o frameworks adicionales has utilizado para mejorar la calidad del código y la automatización de pruebas en aplicaciones backend? Explica cómo los has utilizado en proyectos anteriores

.
respuesta

1. **JUnit y Mockito**: Usado para pruebas unitarias y mockeo de dependencias externas (bases de datos, servicios) en la lógica de negocio.

2. **Spring Boot Test**: Para pruebas de integración de APIs REST, asegurando que los endpoints funcionen correctamente.

3. **SonarQube**: Analiza el código estático para identificar problemas de calidad como errores, vulnerabilidades y código duplicado.

4. **Docker**: Crea entornos aislados para pruebas de integración, asegurando que la app funcione de la misma forma en cualquier máquina.

5. **GitLab CI / Jenkins**: Automatiza pruebas y despliegues en pipelines de CI/CD, asegurando calidad continua en el código.

6. **Liquibase**: Gestiona cambios en esquemas de bases de datos de manera controlada y versionada.

7. **Postman**: Para pruebas manuales de APIs y automatización de pruebas de endpoints en el pipeline.

Estas herramientas y frameworks mejoran la calidad del código, la automatización de pruebas y facilitan despliegues seguros.