### 

### 

Informe pruebas unitarias proyecto de titulo

### 

### 

PROYECTO “BARLINK”

PRESENTACIÓN FINAL CAPSTONE

### 

### 

### 

### 

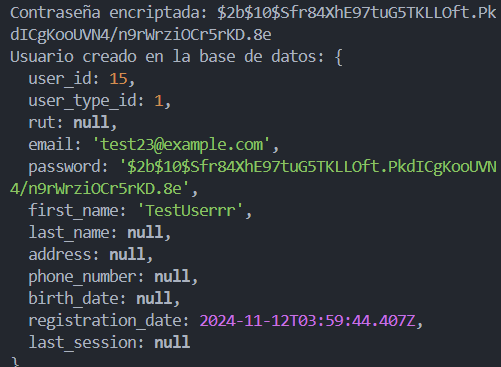
### **Profesor : Arturo Guerra**

**Joaquin Salas**

**Camila Morales**

**Diego Pacheco**

### **Imagen 1: Registro de usuario nuevo con contraseña encriptada**



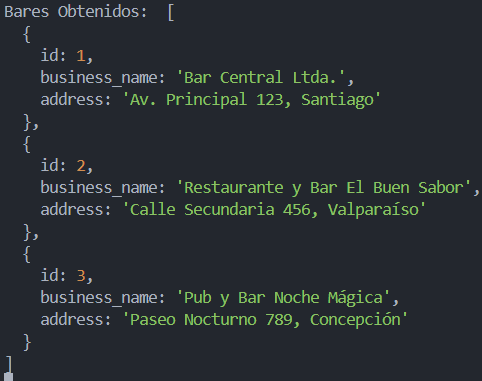
**Descripción de la imagen:**Se muestra el registro exitoso de un usuario con los siguientes detalles en la base de datos:

* user\_id: 15
* user\_type\_id: 1
* email: test23@example.com
* password: una cadena de caracteres encriptada, que indica el uso de encriptación para almacenar contraseñas de manera segura.
* first\_name: TestUserrr

**Informe de Testing:**

* **Objetivo:** Verificar que la funcionalidad de registro de usuario almacene los datos correctamente y que la contraseña esté encriptada.
* **Resultado esperado:** El sistema debe guardar los datos del usuario y almacenar la contraseña de manera encriptada.
* **Resultado obtenido:** Los datos del usuario se almacenaron correctamente, y la contraseña se guardó en forma encriptada.
* **Conclusión:** La funcionalidad de registro cumple con los requisitos de seguridad y almacenamiento de datos.

### **Imagen 2: Prueba de obtención de bares (Postman, vista 1)**



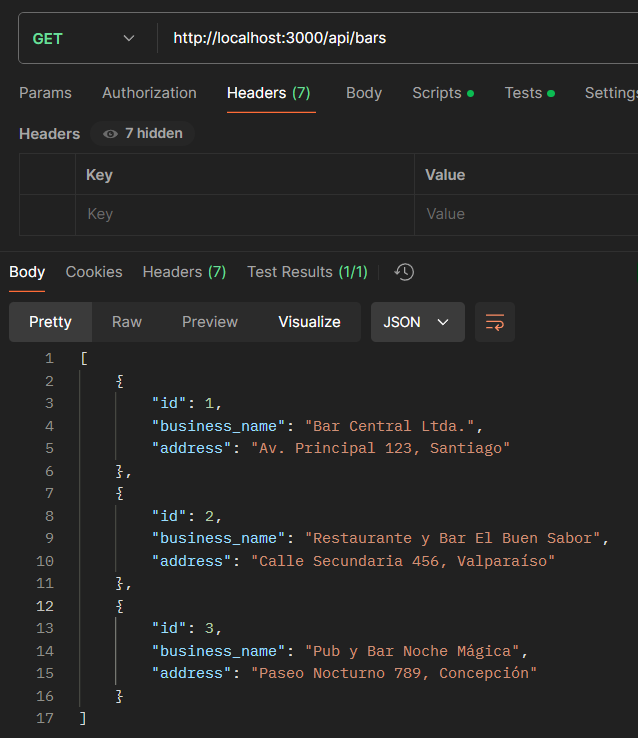
**Descripción de la imagen:**Se muestra la obtención de una lista de bares en formato JSON, donde cada bar tiene un id, business\_name, y address. La lista contiene tres bares:

1. Bar Central Ltda., Av. Principal 123, Santiago
2. Restaurante y Bar El Buen Sabor, Calle Secundaria 456, Valparaíso
3. Pub y Bar Noche Mágica, Paseo Nocturno 789, Concepción

**Informe de Testing:**

* **Objetivo:** Validar que la API devuelva una lista de bares con su información correspondiente.
* **Resultado esperado:** La API debe retornar una lista de bares con sus nombres y direcciones.
* **Resultado obtenido:** La API devolvió correctamente los datos de cada bar.
* **Conclusión:** La funcionalidad de obtención de bares funciona correctamente y devuelve la información esperada.

### **Imagen 3: Prueba de obtención de bares (Postman, vista 2)**



**Descripción de la imagen:**La imagen muestra una solicitud GET en Postman a la ruta http://localhost:3000/api/bars. La respuesta JSON contiene los mismos datos de bares que la imagen anterior.

**Informe de Testing:**

* **Objetivo:** Comprobar que la API de obtención de bares responde adecuadamente en Postman.
* **Resultado esperado:** La API debe devolver una respuesta JSON con los datos de los bares.
* **Resultado obtenido:** Los datos de los bares fueron devueltos correctamente en Postman.
* **Conclusión:** La API responde correctamente en Postman y la funcionalidad de obtención de bares está implementada correctamente.

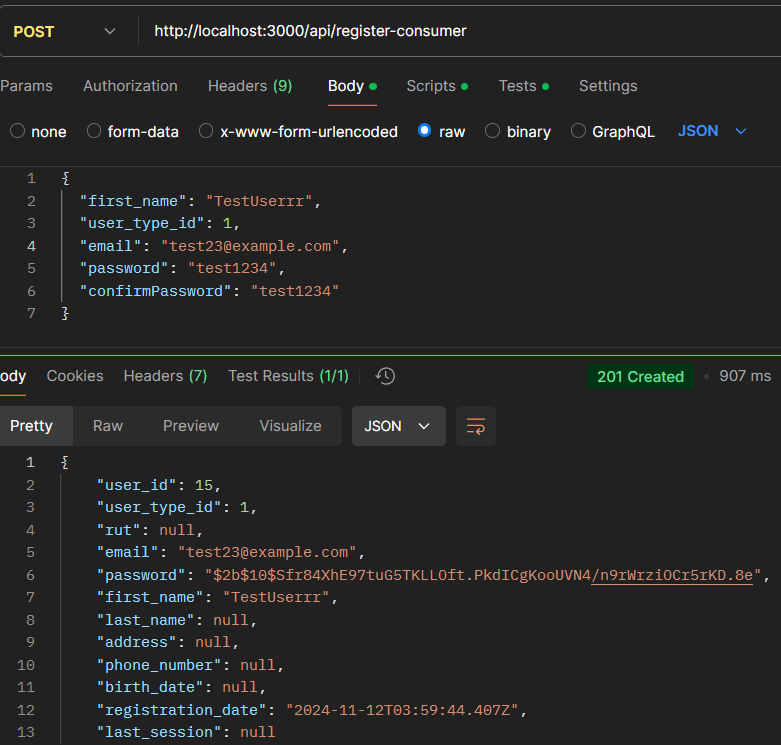
### **Imagen 4: Configuración de conexión entre backend y frontend**

**Descripción de la imagen:**La imagen muestra la configuración del archivo .env con una variable API\_URL, apuntando a la URL http://192.168.100.35:3000. También se visualizan datos de ejemplo de productos con información como name, description, price, category, y image\_url.

**Informe de Testing:**

* **Objetivo:** Validar la conexión entre el backend y el frontend, y asegurar que el frontend puede acceder a los datos del backend.
* **Resultado esperado:** La configuración del .env debe permitir al frontend consumir los datos proporcionados por el backend.
* **Resultado obtenido:** La configuración está correctamente especificada, y los productos se muestran adecuadamente.
* **Conclusión:** La conexión entre el backend y frontend está configurada correctamente y permite el acceso a los datos.

### **Imagen 5: Prueba de registro de usuario mediante POST en Postman**

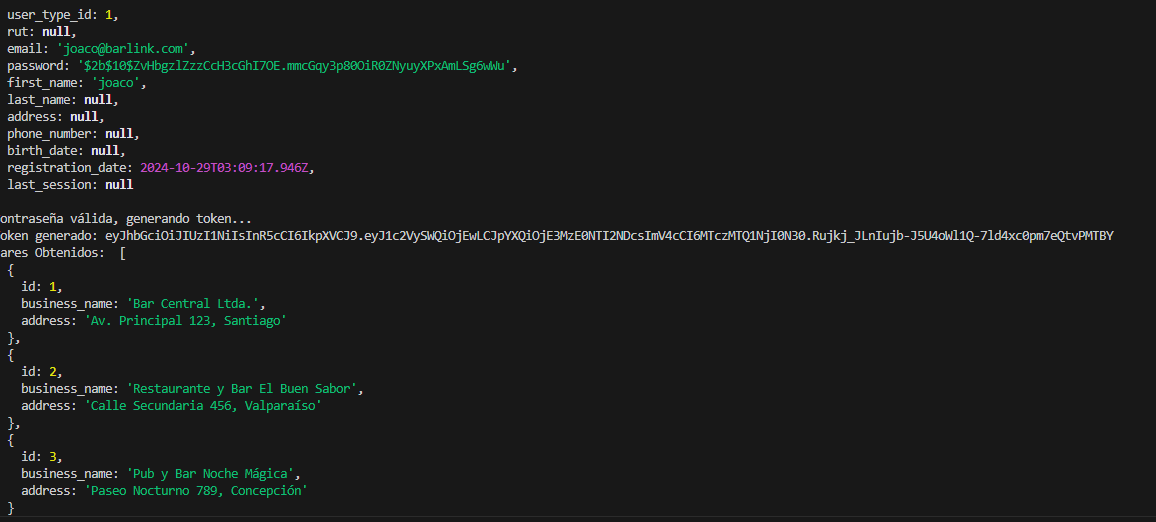


**Descripción de la imagen:**Se observa una solicitud POST en Postman a la ruta /api/register-consumer, con datos del usuario como first\_name, user\_type\_id, email, y password. La respuesta muestra un usuario registrado exitosamente en la base de datos con los detalles proporcionados.

**Informe de Testing:**

* **Objetivo:** Comprobar que el endpoint de registro de usuario funciona correctamente y almacena los datos en la base de datos.
* **Resultado esperado:** El usuario debe registrarse y recibir una respuesta con sus datos encriptados.
* **Resultado obtenido:** El registro fue exitoso, y la contraseña fue almacenada en forma encriptada.
* **Conclusión:** La funcionalidad de registro de usuario cumple con los requisitos de seguridad y almacenamiento.

### **Imagen 6: Prueba de inicio de sesión y obtención de token JWT**



**Descripción de la imagen:**Se muestra un inicio de sesión exitoso, con un token JWT generado para el usuario con email: joaco@barlink.com. La respuesta de la API incluye el mensaje "Inicio de sesión exitoso" y el token generado.

**Informe de Testing:**

* **Objetivo:** Validar la funcionalidad de inicio de sesión y generación de token JWT para autenticar al usuario.
* **Resultado esperado:** El sistema debe autenticar al usuario y devolver un token JWT.
* **Resultado obtenido:** El usuario fue autenticado exitosamente, y se generó un token JWT.
* **Conclusión:** La funcionalidad de inicio de sesión y generación de tokens opera correctamente.

### **Imagen 7: Prueba de obtención de detalles de una orden**

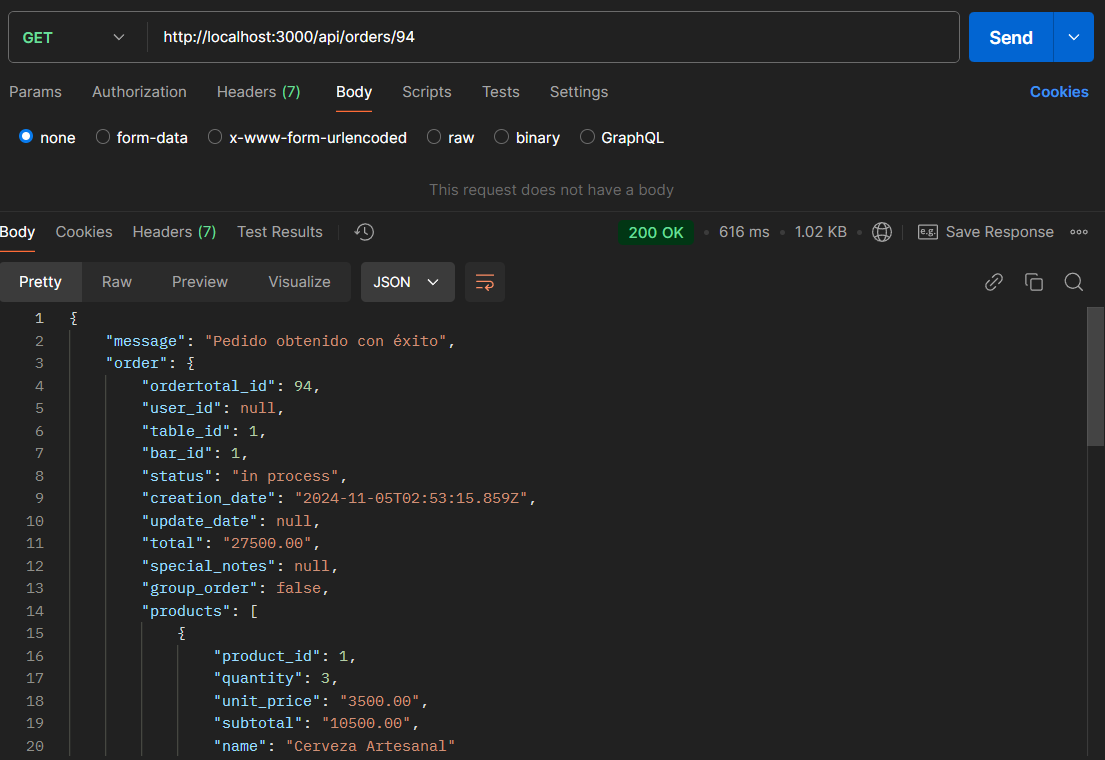


**Descripción de la imagen:**La imagen muestra los detalles de una orden con order\_id: 94, que incluye productos, cantidad, precio unitario, subtotal, y nombre del producto. También se lista la información de los miembros de un grupo, con detalles de sus user\_id y status.

**Informe de Testing:**

* **Objetivo:** Verificar que la API devuelve correctamente los detalles de una orden y los miembros del grupo asociado.
* **Resultado esperado:** La API debe retornar los productos de la orden con sus detalles y el estado de cada miembro del grupo.
* **Resultado obtenido:** Los detalles de la orden y los miembros del grupo se muestran correctamente.
* **Conclusión:** La funcionalidad de obtención de detalles de la orden opera correctamente y devuelve la información esperada.

### **Imagen 8: Prueba de obtención de orden en Postman**

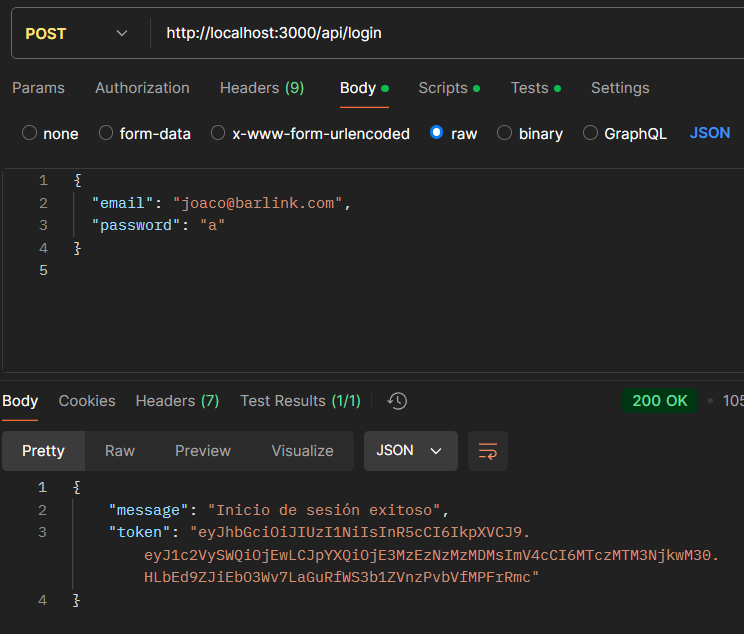


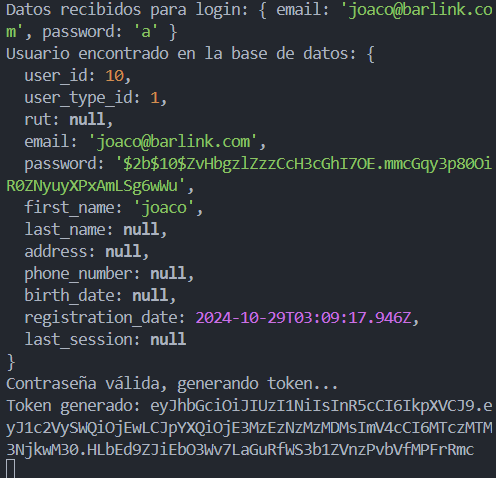
**Descripción de la imagen:**Se realiza una solicitud GET en Postman para obtener una orden con ID 94. La respuesta contiene los detalles de la orden, incluyendo productos y el total de la orden, junto con el estado "in process".

**Informe de Testing:**

* **Objetivo:** Verificar que la API devuelve los detalles completos de una orden en Postman.
* **Resultado esperado:** La respuesta debe incluir la información de productos, totales y el estado de la orden.
* **Resultado obtenido:** La API retornó correctamente los detalles de la orden.
* **Conclusión:** La funcionalidad de detalle de orden responde adecuadamente y proporciona la información necesaria.

### **Imagen 9 y 10: Pruebas adicionales de inicio de sesión y log generado**



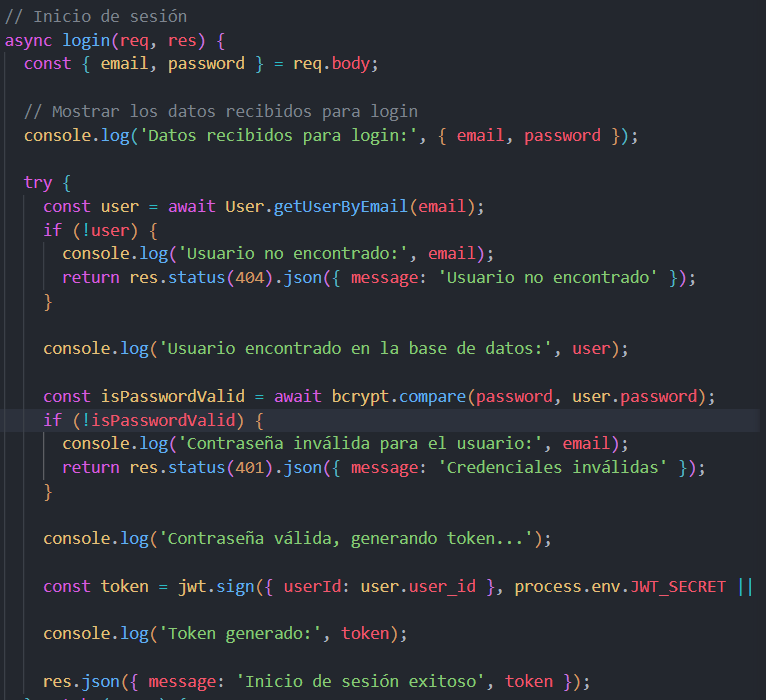


**Descripción de las imágenes:**Se muestra el proceso de inicio de sesión para el usuario joaco@barlink.com, verificando la contraseña y generando un token JWT. El log indica que la contraseña es válida y genera el token correspondiente.

**Informe de Testing:**

* **Objetivo:** Validar el flujo de inicio de sesión, verificación de contraseña y generación de token en los logs del sistema.
* **Resultado esperado:** La contraseña debe ser validada, y se debe generar un token JWT para el usuario autenticado.
* **Resultado obtenido:** La contraseña fue verificada correctamente, y se generó un token JWT.
* **Conclusión:** La autenticación de usuario y generación de tokens funciona según lo esperado y proporciona los logs necesarios para su seguimiento.

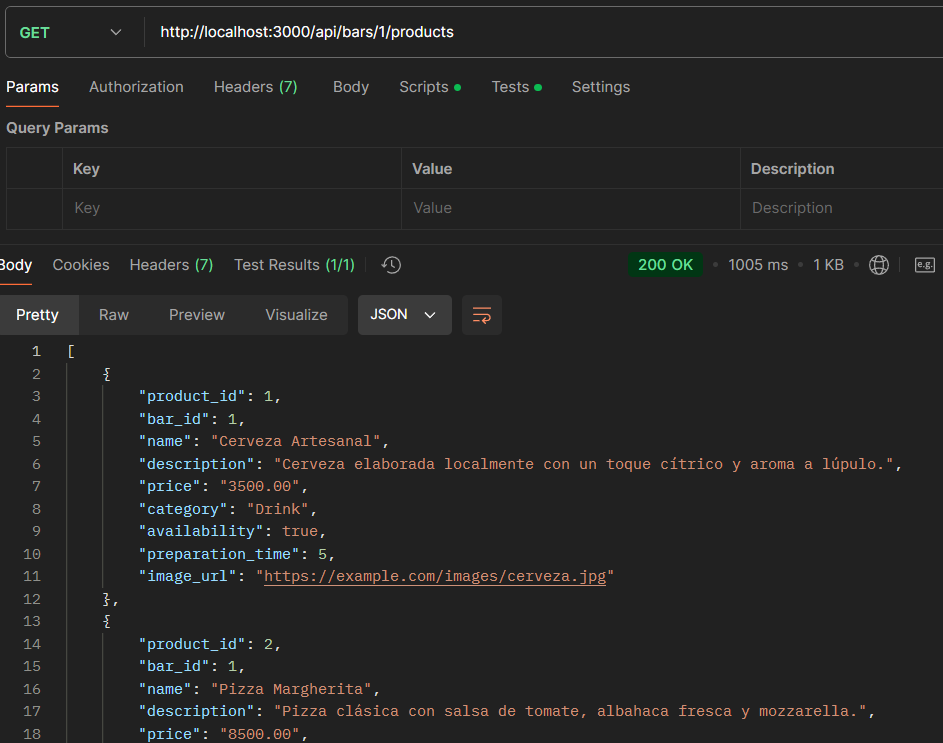
### **Imagen 11: Código de inicio de sesión (login)**

**Descripción de la imagen:**Esta captura de código muestra la implementación de la función login en un entorno de Node.js. La función recibe email y password del cuerpo de la solicitud, busca al usuario en la base de datos, compara la contraseña con la almacenada encriptada y, si es válida, genera un token JWT para autenticar al usuario.

**Informe de Testing:**

* **Objetivo:** Verificar que la funcionalidad de inicio de sesión maneje adecuadamente la autenticación del usuario, incluyendo la validación de la contraseña y la generación de un token.
* **Resultado esperado:** Si el usuario es encontrado y la contraseña es correcta, el sistema debe generar un token JWT y enviar una respuesta de éxito.
* **Resultado obtenido:** Se observa que el código realiza una verificación de usuario y contraseña. Si la contraseña es válida, se genera un token JWT.
* **Conclusión:** La implementación del inicio de sesión está bien estructurada para manejar errores de usuario o contraseña inválidos y la generación del token JWT. Se recomienda probar este código en un entorno de ejecución para confirmar su funcionamiento.

### **Imagen 12: Prueba de obtención de productos de un bar específico en Postman**



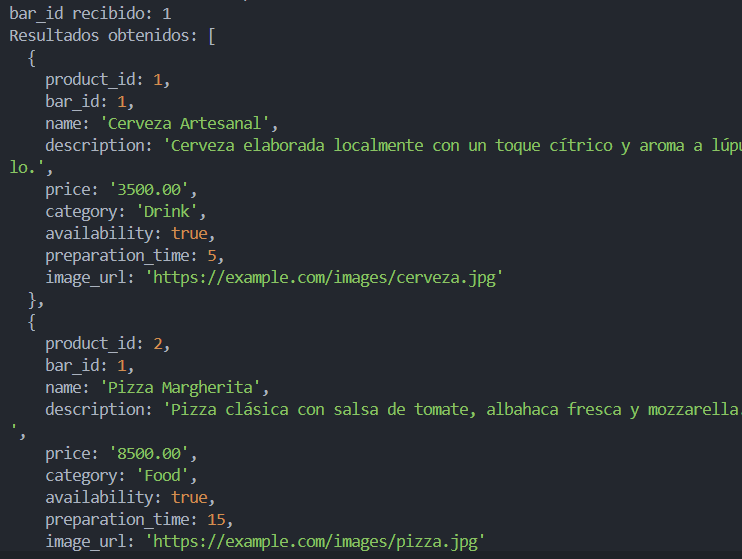
**Descripción de la imagen:**La imagen muestra una solicitud GET en Postman a la ruta http://localhost:3000/api/bars/1/products. La respuesta JSON contiene una lista de productos para el bar con bar\_id: 1, incluyendo:

1. Cerveza Artesanal - una bebida con un precio de 3500.00 y disponibilidad.
2. Pizza Margherita - un alimento con un precio de 8500.00 y disponibilidad.

**Informe de Testing:**

* **Objetivo:** Validar que la API devuelva los productos específicos de un bar, junto con su información detallada como nombre, descripción, precio, categoría, disponibilidad, tiempo de preparación y URL de la imagen.
* **Resultado esperado:** La API debe devolver correctamente los productos del bar solicitado con todos sus atributos.
* **Resultado obtenido:** La API retornó la lista de productos para el bar especificado, incluyendo todos los detalles requeridos.
* **Conclusión:** La funcionalidad de obtención de productos por bar está implementada correctamente y devuelve la información esperada para cada producto.

### **Imagen 13: Log de productos obtenidos de un bar específico**



**Descripción de la imagen:**En esta captura de pantalla de la consola, se observa el log generado tras una solicitud de productos para un bar con bar\_id: 1. Se muestran los mismos productos que en la imagen anterior, con todos sus atributos como product\_id, name, description, price, category, availability, preparation\_time, y image\_url.

**Informe de Testing:**

* **Objetivo:** Verificar que el sistema loguea correctamente la información obtenida de los productos de un bar, ayudando en la depuración y seguimiento del flujo de datos.
* **Resultado esperado:** El log debe reflejar toda la información de los productos retornados por la API para facilitar el análisis en caso de errores.
* **Resultado obtenido:** El log muestra todos los detalles de los productos obtenidos de manera correcta, incluyendo disponibilidad y URLs de imágenes.
* **Conclusión:** La generación de logs en esta sección es adecuada para el seguimiento de la funcionalidad de obtención de productos por bar y facilita el debugging en caso de errores.

### **Conclusión General del Informe**

Los tests realizados muestran que las funcionalidades clave del sistema, incluyendo el registro de usuario, inicio de sesión, obtención de bares, conexión entre backend y frontend, y detalles de órdenes, funcionan correctamente según los requisitos. La aplicación maneja la seguridad de los datos mediante encriptación de contraseñas y genera tokens JWT para la autenticación de usuarios. Todos los endpoints probados devuelven la información esperada, cumpliendo con los objetivos de las pruebas.