

Manual Técnico y de Usuario

API Tienda Online Backend

Documentación de Instalación, Configuración y Consumo de Servicios
REST

Departamento de Desarrollo

18 de noviembre de 2025

Índice general

1. Introducción	2
1.1. Descripción General	2
1.2. Tecnologías Utilizadas	2
2. Instalación y Configuración	3
2.1. Requisitos Previos	3
2.2. Pasos de Instalación	3
2.2.1. 1. Clonar el Repositorio	3
2.2.2. 2. Instalar Dependencias	3
2.2.3. 3. Configuración de Variables de Entorno	3
2.2.4. 4. Ejecución del Servidor	4
3. Autenticación y Seguridad	5
3.1. Flujo de Autenticación	5
3.2. Cabecera de Autorización	5
3.3. Roles	5
4. Endpoints de la API	6
4.1. Módulo de Usuarios (Auth)	6
4.2. Módulo de Productos	7
4.3. Módulo de Categorías	7
4.4. Módulo de Pedidos	7
4.5. Módulo de Imágenes	8
5. Estructura del Proyecto	9

Capítulo 1

Introducción

1.1. Descripción General

El presente documento detalla el funcionamiento técnico del sistema **Tienda Online Backend**. Este proyecto es una API RESTful desarrollada para gestionar la lógica de negocio de una plataforma de comercio electrónico.

El sistema permite la gestión de usuarios, productos, categorías, pedidos y la carga de imágenes, implementando seguridad mediante autenticación basada en tokens (JWT).

1.2. Tecnologías Utilizadas

El núcleo del sistema está construido utilizando las siguientes tecnologías:

- **Entorno de Ejecución:** Node.js
- **Framework Web:** Express.js
- **Base de Datos:** PostgreSQL
- **Seguridad:** Bcrypt.js (hashing) y JSON Web Tokens (JWT)
- **ORM/Driver:** pg (node-postgres)

Capítulo 2

Instalación y Configuración

2.1. Requisitos Previos

Para ejecutar este sistema localmente, asegúrese de tener instalado:

1. **Node.js** (versión 14 o superior).
2. **PostgreSQL** (versión 12 o superior).
3. Un cliente de API como Postman o Insomnia (opcional para pruebas).

2.2. Pasos de Instalación

2.2.1. 1. Clonar el Repositorio

Obtenga el código fuente en su máquina local:

```
1 git clone <url_del_repositorio>
2 cd tienda_online_bankend
```

2.2.2. 2. Instalar Dependencias

Ejecute el siguiente comando para instalar las librerías necesarias listadas en el `package.json`:

```
1 npm install
```

2.2.3. 3. Configuración de Variables de Entorno

Cree un archivo llamado `.env` en la raíz del proyecto. Este archivo debe contener las credenciales de su base de datos y la clave secreta para JWT.

```
1 PORT=3000
2 DB_USER=postgres
3 DB_HOST=localhost
4 DB_NAME=tienda_online
5 DB_PASSWORD=su_contrasena
6 DB_PORT=5432
7 JWT_SECRET=clave_super_secreta
```

Listing 2.1: Ejemplo de archivo `.env`

2.2.4. 4. Ejecución del Servidor

Para iniciar el servidor en modo de desarrollo (usando `nodemon` para recarga automática):

```
1 npm run dev
```

Si ve el mensaje `Servidor corriendo en puerto 3000`, la instalación ha sido exitosa.

Capítulo 3

Autenticación y Seguridad

El sistema utiliza **JSON Web Tokens (JWT)** para asegurar los endpoints sensibles.

3.1. Flujo de Autenticación

1. El usuario envía sus credenciales (email y contraseña) al endpoint de login.
2. El servidor valida las credenciales contra la base de datos (comparando el hash `bcrypt`).
3. Si son correctas, el servidor responde con un `token`.
4. El cliente debe almacenar este token y enviarlo en la cabecera `Authorization` en las peticiones subsiguientes.

3.2. Cabecera de Autorización

Para consumir rutas protegidas (como crear productos o ver pedidos), debe incluir la cabecera:

```
Authorization: Bearer <su_token_jwt_aqui>
```

3.3. Roles

El sistema cuenta con un middleware (`adminMiddleware.js`) que verifica si el usuario tiene rol de administrador antes de permitir operaciones críticas como la eliminación de productos.

Capítulo 4

Endpoints de la API

A continuación, se detallan las rutas disponibles organizadas por módulos.

4.1. Módulo de Usuarios (Auth)

Controlador: `UsuarioController.js`

Registro de Usuario

- **Método:** POST
- **Ruta:** /api/usuarios/registro
- **Body (JSON):**

```
1 {  
2   "nombre": "Juan Perez",  
3   "email": "juan@example.com",  
4   "password": "123456",  
5   "direccion": "Calle Falsa 123",  
6   "telefono": "555-5555"  
7 }  
8
```

Iniciar Sesión (Login)

- **Método:** POST
- **Ruta:** /api/usuarios/login
- **Body (JSON):**

```
1 {  
2   "email": "juan@example.com",  
3   "password": "123456"  
4 }  
5
```

- **Respuesta Exitosa:** Retorna el objeto usuario y el token.

4.2. Módulo de Productos

Controlador: `ProductoController.js`. Requiere autenticación para escritura.

Listar Productos

- **Método:** GET
- **Ruta:** `/api/productos`
- **Descripción:** Obtiene todos los productos disponibles.

Crear Producto (Admin)

- **Método:** POST
- **Ruta:** `/api/productos`
- **Header:** `Authorization: Bearer <token>`
- **Body (JSON):**

```
1 {
2   "nombre": "Laptop Gamer",
3   "descripcion": "Potente laptop para juegos",
4   "precio": 1500.00,
5   "stock": 10,
6   "categoria_id": 1
7 }
```

4.3. Módulo de Categorías

Controlador: `CategoriaController.js`

- **GET /api/categorias:** Listar todas las categorías.
- **POST /api/categorias:** Crear nueva categoría (Admin).
- **PUT /api/categorias/:id:** Actualizar categoría.
- **DELETE /api/categorias/:id:** Eliminar categoría.

4.4. Módulo de Pedidos

Controlador: `PedidoController.js`

Crear Pedido

Permite a un usuario autenticado generar una orden de compra.

- **Método:** POST
- **Ruta:** /api/pedidos
- **Body:** Debe contener el ID del usuario y los detalles de los productos (IDs y cantidades).

4.5. Módulo de Imágenes

Controlador: `ImagenProductoController.js`

Rutas para la gestión de archivos multimedia asociados a los productos.

- **POST /api/imagenes/upload:** Subida de archivos (multipart/form-data).

Capítulo 5

Estructura del Proyecto

El proyecto sigue una arquitectura por capas para facilitar el mantenimiento:

src/app.js Punto de entrada de la aplicación. Configura Express y Middlewares.

src/config/ Configuraciones globales (Base de datos).

src/controllers/ Lógica que maneja las peticiones HTTP y respuestas.

src/dao/ (Data Access Object) Capa de acceso directo a la base de datos PostgreSQL.

src/dto/ (Data Transfer Object) Modelado de datos para transferir entre capas.

src/routes/ Definición de las rutas de la API y asignación de controladores.

src/middlewares/ Funciones intermedias (Autenticación, Validación de Admin).