API de Productos - Node.js con Firebase

API REST para gestión de productos con autenticación JWT y base de datos Firebase Firestore.

🚀 Características

- **CRUD completo** para productos
- **Autenticación JWT** con Bearer tokens
- **Base de datos Firebase Firestore** en la nube
- **Manejo de errores** con códigos de estado HTTP apropiados
- **Middleware de autenticación** para rutas protegidas
- **Validación de datos** en servicios
- **Arquitectura en capas** (Controladores, Servicios, Modelos)

💼 Requisitos

- Node.js 18 o superior
- Cuenta de Firebase con proyecto configurado
- npm o yarn

🛠️ Instalación

1. **Clona el repositorio e instala dependencias:**

```
npm install
```

2. **Configura las variables de entorno:** Crea un archivo `.env` en la raíz del proyecto con las siguientes variables:

```
# Configuración del servidor
PORT=3000

# JWT Secret Key
JWT_SECRET=tu_clave_secreta_muy_segura_aqui

# Firebase Configuration
FIREBASE_API_KEY=tu_firebase_api_key
FIREBASE_AUTH_DOMAIN=tu_proyecto.firebaseapp.com
FIREBASE_PROJECT_ID=tu_proyecto_id
FIREBASE_STORAGE_BUCKET=tu_proyecto.appspot.com
FIREBASE_MESSAGING_SENDER_ID=123456789
FIREBASE_APP_ID=1:123456789:web:abcdef123456

# Credenciales de usuario para autenticación
ADMIN_EMAIL=admin@ejemplo.com
```

```
ADMIN_PASSWORD=admin123
```

3. **Configura Firebase:**

- La colección se llama `products`
- Obtener la configuración de tu proyecto y actualiza las variables de entorno

4. **Inicia el servidor:**

```
npm start
```

📁 Estructura del proyecto

```
project-root/
├── config/
│   └── firebase.js          # Configuración de Firebase
├── controllers/
│   ├── auth.controller.js    # Controlador de autenticación
│   └── product.controller.js # Controlador de productos
├── middleware/
│   └── auth.middleware.js    # Middleware de autenticación
├── models/
│   └── Product.model.js      # Modelo de datos de productos
├── routes/
│   ├── auth.routes.js        # Rutas de autenticación
│   └── products.routes.js    # Rutas de productos
├── services/
│   ├── auth.service.js       # Lógica de negocio de autenticación
│   └── product.service.js    # Lógica de negocio de productos
├── .env                      # Variables de entorno
├── index.js                  # Punto de entrada de la aplicación
└── package.json
```

🔐 Autenticación

Para acceder a las rutas protegidas, debes incluir el token JWT en el header:

```
Authorization: Bearer <tu_token_jwt>
```

Obtener token:

```
POST /auth/login
{
  "email": "admin@ejemplo.com",
  "password": "admin123"
}
```

📡 Endpoints

Autenticación

- `POST /auth/login` - Autenticar usuario
- `GET /auth/verify` - Verificar token

Productos

- `GET /api/products` - Obtener todos los productos (público)
- `GET /api/products/:id` - Obtener producto por ID (público)
- `POST /api/products/create` - Crear producto (protegido)
- `PUT /api/products/:id` - Actualizar producto (protegido)
- `DELETE /api/products/:id` - Eliminar producto (protegido)

📝 Ejemplos de uso

Crear un producto:

```
POST /api/products/create
Authorization: Bearer <token>
{
  "name": "Producto de ejemplo",
  "description": "Descripción del producto",
  "price": 29.99,
  "category": "Electrónicos",
  "stock": 100,
  "active": true,
  "imageUrl": "https://ejemplo.com/imagen.jpg"
}
```

Obtener todos los productos:

```
GET /api/products
```

Actualizar un producto:

```
PUT /api/products/:id
Authorization: Bearer <token>
{
  "name": "Producto actualizado",
  "price": 39.99
}
```

📊 Estructura de datos de productos

```
{
  "id": "documento_id_firebase",
  "name": "Nombre del producto",
  "description": "Descripción del producto",
```

```
  "price": 29.99,
  "category": "Categoría",
  "stock": 100,
  "active": true,
  "imageUrl": "URL de la imagen",
  "createdAt": "2023-12-01T10:00:00.000Z",
  "updatedAt": "2023-12-01T10:00:00.000Z"
}
```

🚨 Códigos de estado HTTP

- `200` - Éxito
- `201` - Creado exitosamente
- `400` - Solicitud incorrecta
- `401` - No autorizado (token inválido o faltante)
- `403` - Prohibido (token expirado)
- `404` - Recurso no encontrado
- `500` - Error interno del servidor

🎁 Dependencias principales

- **express**: Framework web
- **cors**: Middleware para CORS
- **body-parser**: Parser de cuerpos de petición
- **dotenv**: Variables de entorno
- **firebase**: SDK de Firebase
- **jsonwebtoken**: Tokens JWT

🛡 Seguridad

- Tokens JWT con expiración de 24 horas
- Validación de datos en servicios
- Middleware de autenticación para rutas protegidas
- Manejo seguro de errores sin exponer información sensible