**Productos APP**

Contents

[**Requerimientos – IMPORTANTE** 2](#_Toc211906824)

[**Pasos para empezar** 2](#_Toc211906825)

[Usuarios disponibles para usar la aplicación 3](#_Toc211906826)

[Permisos de los usuarios 3](#_Toc211906827)

[Rutas backend 3](#_Toc211906828)

[**Obtener todos los usuarios** 3](#_Toc211906829)

[**Login de usuario** 3](#_Toc211906830)

[**Registro de usuario** 3](#_Toc211906831)

[**Obtener todos los productos** 4](#_Toc211906832)

[**Obtener producto por ID** 4](#_Toc211906833)

[**Crear producto** 5](#_Toc211906834)

[**Actualizar producto** 5](#_Toc211906835)

[**Borrar producto** 6](#_Toc211906836)

# **Requerimientos – IMPORTANTE**

* Tener docker
* Tener los puertos 3000, 5000 y 3306 libres
* En caso de tener uno o más puertos ocupados
  1. Abrir consola o línea de comando como administrador
  2. Ejecutar ‘netstat -ano | findstr PUERTO’, ejemplo ‘netstat -ano | findstr 3306’
  3. Busca la línea correspondiente al puerto 3306 y anota el número de la última columna (PID).
  4. Termina el proceso que esta utilizando ese puerto ejecutando ‘taskkill /PID <PID> /F’ en la línea de comando, ejemplo ‘taskkill /PID 4405 /F’

# **Pasos para empezar**

1. Clonar el Repositorio
2. Abrir la linea de comando en el directorio del Repositorio
3. Ejecutar ‘docker compose up’
4. Esperar hasta que todo los servicios esten arriba

# Usuarios disponibles para usar la aplicación

Usuarios precreados con los seeders del backend, pueden probar la appweb directamente sin tener que crear usuarios.

Usuario – Admin (Permiso para todo)

Email: admin@example.com

Password: 123456

Usuario – usuario (Permiso solo para vista)

Email: usuario@example.com

Password: abcdef

# Permisos de los usuarios

* ADMIN: permiso de vista, creación, edición, eliminación de productos y generación de reportes.
* USUARIO: permiso de vista.

# Rutas backend

**Obtener todos los usuarios**

* **Ruta:** GET /api/usuario
* **Permisos:** Solo usuarios con rol Admin
* **Descripción:** Devuelve la lista completa de usuarios registrados en la base de datos.
* **Request:** No requiere cuerpo.
* **Response:** 200 OK

[

{

"Id": 1,

"Nombre": "Juan Perez",

"Email": "juan@mail.com",

"Rol": "Admin"

},

{

"Id": 2,

"Nombre": "Maria Lopez",

"Email": "maria@mail.com",

"Rol": "User"

}

]

* **Errores posibles:**
  + 401 Unauthorized → Si no se proporciona token válido.
  + 403 Forbidden → Si el usuario no tiene rol Admin.

**Login de usuario**

* **Ruta:** POST /api/usuario/login
* **Permisos:** Público (no requiere token)
* **Descripción:** Permite a un usuario autenticarse. Valida credenciales y devuelve un token JWT.
* **Request Body:**

{

"Email": "usuario@mail.com",

"Password": "123456"

}

* **Response:** 200 OK

{

"msg": "login exitoso",

"token": "eyJhbGciOiJIUzI1NiIsInR..."

}

* **Errores posibles:**
  + 400 Bad Request → Si el body no es válido.
  + 401 Unauthorized → Si las credenciales son incorrectas.

**Registro de usuario**

* **Ruta:** POST /api/usuario/registro
* **Permisos:** Público (no requiere token)
* **Descripción:** Crea un nuevo usuario. La contraseña se guarda encriptada y devuelve un token JWT.
* **Request Body:**

{

"Nombre": "Nuevo Usuario",

"Email": "nuevo@mail.com",

"Password": "123456",

"Rol": "User"

}

* **Response:** 200 OK

{

"msg": "registro exitoso",

"token": "eyJhbGciOiJIUzI1NiIsInR..."

}

* **Errores posibles:**
  + 400 Bad Request → Si los datos no cumplen las validaciones.
  + 409 Conflict → Si el email ya está registrado (opcional según implementación).

**Obtener todos los productos**

* **Ruta:** GET /api/producto
* **Permisos:** Público (no requiere token)
* **Descripción:** Devuelve la lista de todos los productos, incluyendo información del usuario que los creó o modificó.
* **Request:** No requiere cuerpo.
* **Response:** 200 OK

[

{

"Id": "guid",

"Nombre": "Producto 1",

"Descripcion": "Descripción del producto",

"Precio": 100.0,

"Estado": true,

"UsuarioCreacionId": "guid",

"UsuarioCreacion": {

"Id": "guid",

"Nombre": "Nombre Usuario",

"Email": "usuario@mail.com"

},

"FechaCreacion": "2025-10-21T16:30:00",

"UsuarioModificacionId": "guid",

"FechaModificacion": "2025-10-21T17:00:00"

}

]

**Obtener producto por ID**

* **Ruta:** GET /api/producto/{id}
* **Permisos:** Público (no requiere token)
* **Descripción:** Devuelve un solo producto por su ID, con información del usuario que lo creó o modificó.
* **Request:** Parametro id en la URL (GUID del producto)
* **Response:** 200 OK si existe, 404 Not Found si no existe.

**Crear producto**

* **Ruta:** POST /api/producto
* **Permisos:** Solo usuarios con rol Admin
* **Descripción:** Crea un nuevo producto. Se obtiene el ID del usuario desde el token y se guarda como UsuarioCreacionId.
* **Request Body:**

{

"Nombre": "Producto Nuevo",

"Descripcion": "Descripción del producto",

"Precio": 120.0,

"Estado": true

}

* **Response:** 200 OK

{

"msg": "producto agregado",

"data": {

"Id": "guid",

"Nombre": "Producto Nuevo",

"Descripcion": "Descripción del producto",

"Precio": 120.0,

"Estado": true,

"UsuarioCreacionId": "guid",

"FechaCreacion": "2025-10-21T16:30:00"

}

}

* **Errores posibles:**
  + 400 Bad Request → Si los datos no son válidos.
  + 401 Unauthorized → Si el usuario no envía token válido.
  + 403 Forbidden → Si el usuario no es Admin.

**Actualizar producto**

* **Ruta:** PUT /api/producto/{id}
* **Permisos:** Solo usuarios con rol Admin
* **Descripción:** Actualiza los campos de un producto existente. Guarda el usuario que modifica y la fecha de modificación.
* **Request Body:** Igual que para crear producto.
* **Response:** 200 OK con el producto actualizado.
* **Errores posibles:**
  + 400 Bad Request → Si los datos no son válidos.
  + 401 Unauthorized → Si no envía token válido.
  + 403 Forbidden → Si no es Admin.
  + 404 Not Found → Si el producto no existe.

**Borrar producto**

* **Ruta:** DELETE /api/producto/{id}
* **Permisos:** Solo usuarios con rol Admin
* **Descripción:** Elimina un producto existente por su ID.
* **Request:** Parametro id en la URL.
* **Response:** 200 OK con el producto eliminado.
* **Errores posibles:**
  + 401 Unauthorized → Si no envía token válido.
  + 403 Forbidden → Si no es Admin.
  + 404 Not Found → Si el producto no existe.