Documentación endpoints:

Product Handler

Base URL: /products

1. Obtener Todos los Productos (Paginado)

Recupera el catálogo completo de productos. Es vital usar la paginación para no sobrecargar el servidor, ya que el catálogo puede ser muy grande.

- **Método:** GET
- **URL:** /
- **Parámetros de Consulta (Query Params):**
  - limit (Integer, Opcional): Cantidad de resultados (Máx 100).
  - offset (Integer, Opcional): Número de registros a saltar (para la paginación).

Ejemplo de Solicitud:

GET /products?limit=10&offset=0

Respuesta Exitosa (200 OK):

JSON

```
None
{
    "status": 200,
    "message": "Products found: 2",
    "data": [
        {
            "id": 1,
            "name": "Ibuprofeno 600mg",
            "type": "Antiinflamatorio",
            "price": 7.50,
            "stock": 50,
            "expiration_date": "2025-12-31"
        },
        {
            "id": 2,
            "name": "Paracetamol 500mg",
            "type": "Analgésico",
            "price": 5.00,
```

```
            "stock": 100,
            "expiration_date": "2026-05-20"
        }
    ]
}
```

---

2. Buscar Productos por Nombre

Realiza una búsqueda parcial (coincidencias "like") por el nombre del producto. Útil para la barra de búsqueda del frontend.

- **Método:** GET
- **URL:** /search/
- **Parámetros de Consulta (Query Params):**
    - name (String, Requerido): El nombre o fragmento a buscar.

Ejemplo de Solicitud:

GET /products/search/?name=Para

Respuesta Exitosa (200 OK):

JSON

```
None
{
    "status": 200,
    "message": "Products found: 1",
    "data": [
        {
            "id": 2,
            "name": "Paracetamol 500mg",
            "type": "Analgésico",
            "use_case": "Fiebre y dolor leve",
            "price": 5.00
        }
    ]
}
```

---

3. Obtener Producto por ID

Obtiene el detalle completo de un producto específico.

- **Método:** GET
- **URL:** /:id
- **Parámetros de Ruta:**
    - id (Integer): ID único del producto.

Respuesta Exitosa (200 OK):

JSON

```
None
{
    "status": 200,
    "message": "Product found",
    "data": {
        "id": 1,
        "name": "Ibuprofeno 600mg",
        "type": "Antiinflamatorio",
        "use_case": "Dolor e inflamación",
        "warnings": "Tomar con comida",
        "contraindications": "Úlcera gástrica",
        "stock": 50,
        "price": 7.50
    }
}
```

---

4. Crear Producto

Registra un nuevo medicamento en el sistema. Estos datos alimentarán posteriormente la base de conocimiento (Vector Store) del Chatbot.

- **Método:** POST
- **URL:** /
- **Body (JSON):**

JSON

```
None
{
    "name": "Amoxicilina 500mg",
    "type": "Antibiótico",
    "use_case": "Infecciones bacterianas",
    "warnings": "Completar el tratamiento",
    "contraindications": "Alergia a penicilinas",
    "expiration_date": "2026-01-01",
```

```
    "price": 12.00,
    "stock": 30
}
```

Errores Comunes:

- 400 Bad Request: Si el producto ya existe o faltan campos obligatorios.

---

5. Actualizar Producto

Modifica los datos de un producto existente. Es posible enviar solo los campos que se desean cambiar (Actualización parcial).

- **Método:** PUT
- **URL:** /:id
- **Parámetros de Ruta:**
    - id (Integer): ID del producto.
- **Body (JSON):**

JSON

```
None
{
    "stock": 25,
    "price": 13.50
}
```

---

6. Eliminar Producto

Elimina permanentemente un producto del inventario.

- **Método:** DELETE
- **URL:** /:id
- **Parámetros de Ruta:**
    - id (Integer): ID del producto a eliminar.

Respuesta Exitosa (200 OK):

JSON

```
None
{
    "status": 200,
    "message": "Product deleted successfully"
```

```
}
```

User Handler

Base URL: /users

1. Registrar Usuario (Sign Up)

Crea una nueva cuenta de usuario en el sistema.

- **Método:** POST
- **URL:** /
- **Body (JSON) - CreateUserDTO:**

JSON

```
None
{
    "username": "juanperez",           // Requerido
    "email": "juan@example.com",       // Requerido, único
    "password": "Password123!",        // Requerido (Min 8 chars, mayus, num)
    "role": "user"                     // Requerido ('user' o 'admin')
}
```

Respuesta Exitosa (201 Created):

Devuelve un objeto UserDTO (sin la contraseña).

JSON

```
None
{
    "status": 201,
```

```
    "message": "User created successfully",
    "data": {
        "id": 1,
        "username": "juanperez",
        "email": "juan@example.com",
        "role": "user",
        "created_at": "2025-11-22T10:00:00.000Z"
    }
}
```

---

2. Iniciar Sesión (Login Local)

Autentica a un usuario mediante correo y contraseña.

- **Método:** POST
- **URL:** /login
- **Body (JSON):**

JSON

```
None
{
    "email": "juan@example.com",
    "password": "Password123!"
}
```

Respuesta Exitosa (200 OK):

JSON

```
None
{
    "status": 200,
    "message": "Login successful",
    "data": {
        "id": 1,
        "username": "juanperez",
        "email": "juan@example.com",
        "role": "user",
        "created_at": "..."
    }
}
```

Nota: Si el usuario no ha verificado su email, retornará error 400.

---

3. Obtener Todos los Usuarios

Devuelve la lista de usuarios registrados con paginación.

- **Método:** GET
- **URL:** /
- **Query Params:**
    - limit (int): Máximo de registros.
    - offset (int): Salto de registros.
      Respuesta (200 OK): Retorna un array de UserDTO.

---

4. Obtener Usuario por ID

- **Método:** GET
- URL: /:id
  Respuesta (200 OK): Retorna un único UserDTO.

---

5. Obtener Usuario por Email

Útil para validaciones o búsquedas administrativas.

- **Método:** GET
- **URL:** /email/
- **Query Params:**
    - email (string): El correo a buscar.
      Ejemplo: GET /users/email/?email=juan@example.com

---

6. Actualizar Perfil de Usuario

Modifica datos básicos del usuario. Solo se permiten los campos definidos en la lista blanca (username, email, role, is_verified).

- **Método:** PUT
- **URL:** /:id
- **Body (JSON) - UpdateUserDTO:**

JSON

```
None
{

    "username": "juan_actualizado",

    "role": "admin"

    // "is_verified": true // Solo admins deberían poder
enviar esto

}
```

Respuesta (200 OK): Retorna el UserDTO actualizado.

---

7. Cambiar Contraseña

Endpoint específico y seguro para la rotación de contraseñas. Requiere validar la contraseña anterior.

- **Método:** PUT
- **URL:** /:id/password
- **Body (JSON) - PasswordUpdateDTO:**

JSON

```
None
{

    "current_password": "OldPassword123!",

    "new_password": "NewPassword456!"

}
```

Respuesta (200 OK):

JSON

```
None
{

    "status": 200,

    "message": "Password updated successfully"

}
```

---

8. Eliminar Usuario

Borrado lógico o físico (según implementación del servicio) de la cuenta.

- **Método:** DELETE
- **URL:** /:id

---

9. Verificar Correo Electrónico

Endpoint al que accede el usuario al hacer clic en el enlace enviado a su correo.

- **Método:** GET
- **URL:** /verify-email/
- **Query Params:**
    - token (string): El token de verificación único.
      Respuesta (200 OK):

JSON

```
None
{

    "status": 200,

    "message": "User verified successfully"

}
```

---

10. Reenviar Correo de Verificación

Si el token expiró o el correo se perdió.

- **Método:** POST
- **URL:** /resend-verification-email/
- **Query Params:**
    - email (string): El correo del usuario.

---

11. Google OAuth (Social Login)

Estos endpoints manejan la autenticación con Google. No retornan JSON directo, sino que manejan redirecciones.

- **Iniciar Auth:** GET /auth/google
    - Redirige al usuario a la página de consentimiento de Google.
- **Callback:** GET /auth/google/callback
    - Google redirige aquí tras el login.
    - **Éxito:** El servidor redirige al Frontend: http://localhost:4200/auth/callback?token={JWT}
    - **Error:** Redirige al Frontend: http://localhost:4200/login/error?message={razon}

Chatbot Session Handler

Base URL: /chatbot-sessions

Este módulo administra las "salas" de conversación. Maneja la creación de nuevas sesiones, el historial por usuario y, lo más importante, **el punto de entrada principal para interactuar con la Inteligencia Artificial (RAG)**.

1. Iniciar Chat con el Bot (RAG)

Este es el endpoint principal de interacción. El usuario envía un mensaje, el sistema busca contexto relevante (productos), consulta a GPT-4 y devuelve una respuesta.

- **Funcionalidad Extra:** Si la sesión es nueva (no tiene título), este endpoint genera automáticamente un título resumen en segundo plano.
- **Método:** POST
- **URL:** /:sessionId/chat
- **Parámetros de Ruta:**
    - sessionId (Integer): ID de la sesión actual.
- **Body (JSON):**

JSON

```
None
{

    "message": "Tengo dolor de espalda, ¿qué puedo tomar?"

}
```

Respuesta Exitosa (200 OK):

Devuelve el par de mensajes (pregunta del usuario y respuesta del bot) para renderizar inmediatamente en el chat.

JSON

```
None
{

    "status": 200,

    "message": "Chat response generated successfully",

    "data": {

        "message": "Tengo dolor de espalda, ¿qué puedo
tomar?",

        "botResponse": "Para el dolor de espalda te
recomiendo..."

    }

}
```

---

2. Crear Nueva Sesión

Inicializa una sala de chat vacía para un usuario.

- **Método:** POST
- **URL:** /
- **Body (JSON):**

JSON

```
None
{

    "user_id": 11

}
```

Respuesta Exitosa (201 Created):

JSON

```
None
{

    "status": 201,

    "message": "ChatbotSession created successfully",

    "data": {

        "id": 5,

        "user_id": 11,

        "is_active": true,

        "title": null, // Será generado automáticamente tras el primer mensaje

        "created_at": "2025-11-22T10:00:00.000Z"

    }

}
```

---

3. Obtener Historial de Sesiones del Usuario

Obtiene todas las sesiones (activas y cerradas) de un usuario específico. Ideal para la barra lateral "Mis Conversaciones".

- **Método:** GET
- **URL:** /user/:user_id
- **Parámetros de Ruta:**
    - user_id (Integer): ID del usuario.

Respuesta Exitosa (200 OK):

JSON

```
None
{
    "status": 200,
    "message": "ChatbotSessions retrieved successfully",
    "data": [
        {
            "id": 5,
            "title": "Dolor de espalda y muscular",
            "is_active": true,
            "created_at": "..."
        },
        {
            "id": 3,
            "title": "Consulta sobre antibióticos",
            "is_active": false,
            "created_at": "..."
```

```
        }
    ]
}
```

---

4. Obtener Sesiones Activas

Filtra solo las sesiones que están abiertas (is_active: true).

- **Método:** GET
- **URL:** /active/:user_id

---

5. Obtener Sesión por ID

Recupera los metadatos de una sesión específica.

- **Método:** GET
- **URL:** /:id

---

6. Actualizar Sesión

Permite cerrar una sesión (finalizar chat) o cambiar su título manualmente.

- **Método:** PUT
- **URL:** /:id
- **Body (JSON) - UpdateChatbotSessionDTO:**
    - *Nota:* Solo se permiten los campos is_active y title.

JSON

```
None
{
    "is_active": false,
    "title": "Mi consulta personalizada"
}
```

---

7. Obtener Todas las Sesiones (Admin)

Endpoint administrativo para ver todas las sesiones del sistema con paginación.

- **Método:** GET
- **URL:** /
- **Query Params:**
  - limit (int): Límite de resultados.
  - offset (int): Desplazamiento.

---

8. Eliminar Sesión

Borra una sesión y (por cascada en la BD) todos sus mensajes asociados.

- **Método:** DELETE
- **URL:** /:id

Chatbot Message Handler

Base URL: /chatbot-messages

1. Obtener Mensajes por Sesión

Recupera el historial de conversación de una sesión específica.

- **Método:** GET
- **URL:** /session/:sessionId
- **Parámetros de Ruta:**
  - sessionId (Integer): ID de la sesión.
- **Query Params:**
  - limit (Integer, Opcional): Límite de mensajes a recuperar.

Respuesta Exitosa (200 OK) - ChatbotMessageDTO[]:

JSON

```
None
{

    "status": 200,

    "message": "Messages retrieved successfully",
```

```
    "data": [
        {
            "id": 10,
            "session_id": 5,
            "sender": "user",
            "message": "Hola, tengo gripe",
            "created_at": "2025-11-22T10:00:00.000Z"
        },
        {
            "id": 11,
            "session_id": 5,
            "sender": "bot",
            "message": "Entiendo, ¿tienes fiebre?",
            "created_at": "2025-11-22T10:00:05.000Z"
        }
    ]
}
```


---

2. Regenerar Mensaje

Endpoint avanzado. Permite editar un mensaje de usuario erróneo.

Comportamiento:

1. Actualiza el texto del mensaje original.
2. **Elimina** todos los mensajes posteriores a ese (para mantener coherencia temporal).
3. Invoca al RAG para generar una **nueva respuesta** basada en la corrección.
4. Guarda la nueva respuesta del bot.

- **Método:** PUT
- **URL:** /:id/regenerate
- **Parámetros de Ruta:**
    - id (Integer): ID del mensaje del usuario a corregir.
- **Body (JSON):**

JSON

```
None
{

    "message": "Corrección: No tengo gripe, tengo alergia"

}
```

Respuesta Exitosa (200 OK):

JSON

```
None
{

    "status": 200,

    "message": "Message updated and bot response generated
successfully",

    "data": {

        "message": "Corrección: No tengo gripe, tengo
alergia",

        "botResponse": "Ah, entiendo. Para la alergia te
recomendaría Loratadina..."

    }

}
```

---

3. Crear Mensaje (Manual)

Inserta un mensaje directamente en la base de datos **sin invocar a la IA**. Útil para logs, mensajes del sistema o migraciones.

- **Método:** POST
- **URL:** /
- **Body (JSON) - CreateChatbotMessageDTO:**

JSON

```
None
{

    "session_id": 5,

    "sender": "user",   // 'user' o 'bot'

    "message": "Mensaje insertado manualmente"

}
```

Respuesta Exitosa (201 Created):

Devuelve el objeto ChatbotMessageDTO creado.

---

4. Actualizar Mensaje (Simple)

Edita únicamente el texto de un mensaje en la base de datos. **NO** dispara el RAG ni borra el historial futuro. Útil para correcciones menores de ortografía que no cambian el contexto.

- **Método:** PUT
- **URL:** /:id
- **Body (JSON) - UpdateChatbotMessageDTO:**

JSON

```
None
{

    "message": "Texto corregido"

}
```

Respuesta Exitosa (200 OK):

Devuelve el objeto ChatbotMessageDTO actualizado.

---

5. Obtener Mensaje por ID

Recupera un único mensaje por su identificador primario.

- **Método:** GET
- **URL:** /:id

Respuesta Exitosa (200 OK) - ChatbotMessageDTO:

JSON

```
None
{
    "status": 200,
    "message": "ChatbotMessage retrieved successfully",
    "data": {
        "id": 10,
        "session_id": 5,
        "sender": "user",
        "message": "Hola",
        "created_at": "2025-11-22T10:00:00.000Z"
    }
}
```

---

6. Eliminar Mensaje

Elimina un mensaje específico.

- **Método:** DELETE
- **URL:** /:id

Respuesta Exitosa (200 OK):

JSON

```
None
{
    "status": 200,
    "message": "ChatbotMessage deleted successfully",
    "data": true
}
```

Anexo: Definición de DTOs

Estructuras de datos utilizadas en este módulo.

CreateChatbotMessageDTO

TypeScript

```
None
interface CreateChatbotMessageDTO {
  session_id: number;
  sender: string; // 'user' | 'bot'
  message: string;
}
```

UpdateChatbotMessageDTO

TypeScript

```
None
interface UpdateChatbotMessageDTO {
  message?: string; // Único campo permitido para actualizar
}
```

ChatbotMessageDTO (Respuesta)

TypeScript

```
None
interface ChatbotMessageDTO {

  id: number;

  session_id: number;

  sender: string;

  message: string;

  created_at: Date;

}
```