

Documentación del backend – Sistema de gestión hospitalaria

ANGEL DAVID REYES TELLEZ
LUIS IVAN MARQUEZ AZUARA
BRAYN KALID REYES SILVA
ALDO TOLENTINO DOMINGUEZ
IRVING MORALES

Índice

Introducción.....	2
Tecnologías utilizadas.....	4
Estructura del proyecto.....	5
Endpoints principales.....	6
Documentación interactiva	9
Instalación y configuración.....	18

Introducción

El backend del Sistema de Gestión Hospitalaria fue desarrollado con FastAPI, un framework moderno de Python diseñado para construir APIs rápidas y eficientes. Esta API proporciona los endpoints necesarios para gestionar espacios médicos, autenticación de usuarios y comunicación con la base de datos, siguiendo estándares RESTful.

Objetivos principales:

- Exponer servicios seguros y escalables para el frontend.
 - Garantizar autenticación mediante JWT (JSON Web Tokens).
 - Ofrecer documentación automática e interactiva mediante Swagger UI y ReDoc.
 - Integrarse con una base de datos relacional mediante SQLAlchemy.
-

Tecnologías Utilizadas

Tecnología	Versión	Descripción
FastAPI	0.111.0	Framework para construir APIs con Python, soporta async y validación automática de datos.
Uvicorn	0.30.1	Servidor ASGI para ejecutar FastAPI en producción con alto rendimiento.
SQLAlchemy	2.0.32	ORM para interactuar con la base de datos de forma segura y eficiente.
PyJWT	2.9.0	Generación y validación de tokens JWT para autenticación.
Pydantic	2.7.4	Validación de datos y manejo de esquemas para requests/responses.
Python-dotenv	1.0.1	Manejo de variables de entorno (.env) para configuración sensible.
MySQLclient	2.2.4	Adaptador para conectar con bases de datos MySQL/MariaDB.
Bcrypt	4.0.1	Encriptación segura de contraseñas.
Swagger UI	(Incluido)	Documentación interactiva automática integrada en FastAPI (/docs).

Características Clave

1. Documentación Autogenerada: Accesible en /docs (Swagger) y /redoc.
2. Autenticación JWT: Endpoints protegidos con tokens renovables.
3. Async/Await: Soporte para operaciones asíncronas (consultas a BD, llamadas externas).
4. Validación Estricta: Uso de Pydantic para esquemas de datos (evita errores comunes).

Estructura del proyecto

Hospital_	Config	
backend_		
API-main	Crud	Pediatria
	Models	Pediatria
	Routes	Pediatria
	Schemas	Pediatria

Endpoints principales

A. Consumibles Médicos

Endpoint	Método	Autenticación	Descripción
GET /consumibles/	GET	JWT	Lista paginada de consumibles.
POST /consumibles/	POST	Público	Crea un nuevo consumible.
PUT /consumible/{id}	PUT	JWT	Actualiza un consumible existente.

Ejemplo de JSON (POST /consumibles/)

```
{  
  "nombre": "Jeringas 10ml",  
  "cantidad": 100,  
  "tipo": "desechable"  
}
```

B. Usuarios

Módulo	Endpoint	Método	Autenticación	Descripción
Usuarios	POST /users/	POST	Público	Crea un nuevo usuario.
	GET /users/	GET	JWT	Lista paginada de usuarios.
	GET /user/{id}	GET	JWT	Consulta un usuario por ID.
	PUT /user/{id}	PUT	JWT	Actualiza un usuario existente.
	DELETE /user/{id}	DELETE	JWT	Elimina un usuario por ID.

Ejemplo en json

POST /medicamentos/ (Crear medicamento):

```
{
  "Nombre_comercial": "Paracetamol",
  "Presentación": "Tabletas 500mg",
  "Cantidad": 1000,
  "Principio_activo": "Acetaminofén"
}
```

Respuesta exitosa (201 Created):

```
{
  "ID": 1,
  "Nombre_comercial": "Paracetamol",
  "Presentación": "Tabletas 500mg",
  "Cantidad": 1000,
  "Principio_activo": "Acetaminofén"
}
```

C. Areas_Medicas

Módulo	Endpoint	Método	Autenticación	Descripción
Areas_Medicinas	GET /areas_medicas/	GET	JWT	Lista paginada de áreas médicas.
	GET /area_medica/{id}	GET	JWT	Obtiene un área

				medica por ID.
	POST /areas_medicas/	POST	Público	Crea una nueva área medica.
	PUT /área_medica/{ID}	PUT	JWT	Actualiza un área médica existente.
	DELETE /Area_medica/{ID}	DELETE	JWT	Elimina un área Medica por ID.

Ejemplo en json

POST /areas_medicas/ (Crear área médica):

```
{
  "Nombre": "Cardiología",
  "Descripcion": "Área especializada en enfermedades del corazón"
}
```

Respuesta exitosa (201 Created):

```
{
  "id": "card-001",
  "Nombre": "Cardiología",
  "Descripción": "Área especializada en enfermedades del corazón"
}
```


Documentación Interactiva con Swagger UI

FastAPI genera automáticamente documentación interactiva para **todos los endpoints** (incluyendo los 20+ existentes) a través de:

Swagger UI: `http://localhost:8000/docs` (pruebas interactivas).

Todos los endpoints están organizados por módulos (Medicamentos, Áreas Médicas, etc.) y muestran:

- Parámetros requeridos.
- Ejemplos de requests/responses.

Áreas Médicas

Áreas Médicas			^
GET	/areas_medicas/	Listar áreas médicas	🔒 ✓
POST	/areas_medicas/	Registrar área médica	🔒 ✓
GET	/area_medica/{id}	Consultar área médica por ID	🔒 ✓
PUT	/area_medica/{id}	Actualizar área médica	🔒 ✓
DELETE	/area_medica/{id}	Eliminar área médica	🔒 ✓

1. GET /areas_medicas/

Descripción:

Obtiene una lista paginada de todas las áreas médicas registradas en el sistema.

Autenticación:

Requiere token JWT válido.

Parámetros Query:

Parámetro	Tipo	Requerido	Valor por defecto	Descripción
skip	integer	No	0	Registros a omitir
limit	integer	No	10	Límite de resultados

Respuesta Exitosa (200 OK):

```
[  
  {
```

```
"id": "card-001",  
"Nombre": "Cardiología",  
"Descripcion": "Área especializada en corazón"  
}  
]
```

2. POST /areas_medicas/

Descripción:

Registra una nueva área médica en el sistema.

Autenticación:

Público (no requiere autenticación).

Request Body:

```
{  
  "Nombre": "Pediatria",  
  "Descripción": "Área médica infantil"  
}
```

Validaciones:

- Nombre debe ser único (no duplicados)
- Nombre máximo 100 caracteres

Respuesta Exitosa (201 Created):

```
{  
  "id": "ped-001",  
  "Nombre": "Pediatria",  
  "Descripción": "Área médica infantil"  
}
```

3. PUT /areas_medicas/{id}

Descripción:

Actualiza los datos de un área médica existente.

Autenticación:

Requiere token JWT válido.

Parámetros Path:

Parámetro	Tipo	Descripción
id	string	ID del área médica

Request Body:

```
{
  "Nombre": "Pediatría Actualizado",
  "Descripción": "Nueva descripción"
}
```

Respuesta Exitosa (200 OK):

```
{
  "id": "ped-001",
  "Nombre": "Pediatría Actualizado",
  "Descripción": "Nueva descripción"
}
```

Resumen

Endpoint	Método	Auth	Descripción
/areas_medicas/	GET	JWT	Lista áreas médicas
/areas_medicas/	POST	Público	Crea nueva área
/areas_medicas/{id}	PUT	JWT	Actualiza área

.

Consumibles Médicos

Consumibles			^
GET	/consumibles/	Listar consumibles	🔒 ▼
POST	/consumibles/	Registrar consumible	▼
GET	/consumible/{id}	Consultar consumible por ID	🔒 ▼
PUT	/consumible/{id}	Actualizar consumible	🔒 ▼
DELETE	/consumible/{id}	Eliminar consumible	🔒 ▼

1. GET /consumibles/

Descripción:

Obtiene una lista paginada de todos los consumibles médicos registrados.

Autenticación:

Requiere token JWT válido.

Parámetros Query:

Parámetro	Tipo	Requerido	Valor por defecto	Descripción
skip	integer	No	0	Registros a omitir
limit	integer	No	10	Límite de resultados

Respuesta Exitosa (200 OK):

```
[
  {
    "nombre": "Guantes de látex",
    "descripción": "Guantes estériles de un solo uso, talla mediana",
    "tipo": "Material Quirúrgico",
    "departamento": "Cirugía",
    "cantidad_existencia": 150,
    "detalle": "Caja con 100 unidades",
    "estatus": true,
    "observaciones": "Usar antes del 2025-12-01",
    "espacio_medico": "Almacén Central",
    "fecha_registro": "2025-03-21T22:19:44.610Z",
    "fecha_actualizacion": "2025-04-01T10:00:00.000Z",
    "id": "3e79d145-397a-4e50-bb74-8c6a88c93fa2"
  }
]
```

]

2. POST /consumibles/

Descripción:

Registra un nuevo consumible médico en el sistema.

Autenticación:

Público (no requiere autenticación).

Request Body:

```
{
  "nombre": "Guantes de látex",
  "descripción": "Guantes estériles de un solo uso, talla mediana",
  "tipo": "Material Quirúrgico",
  "departamento": "Cirugía",
  "cantidad_existencia": 150,
  "detalle": "Caja con 100 unidades",
  "estatus": true,
  "observaciones": "Usar antes del 2025-12-01",
  "espacio_medico": "Almacén Central",
  "fecha_registro": "2025-03-21T22:19:44.610Z",
  "fecha_actualizacion": "2025-04-01T10:00:00.000Z"
}
```

Validaciones:

- nombre debe ser único
- cantidad debe ser número positivo

3. GET /consumible/{id}

Descripción:

Obtiene los detalles de un consumible específico por su ID.

Autenticación:

Requiere token JWT válido.

Parámetros Path:

Parámetro	Tipo	Descripción
id	string	ID del consumible

Respuesta Exitosa (200 OK):






```
{
  "nombre": "Guantes de látex",
  "descripción": "Guantes estériles de un solo uso, talla mediana",
  "tipo": "Material Quirúrgico",
  "departamento": "Cirugía",
  "cantidad_existencia": 150,
  "detalle": "Caja con 100 unidades",
  "estatus": true,
  "observaciones": "Usar antes del 2025-12-01",
  "espacio_medico": "Almacén Central",
  "fecha_registro": "2025-03-21T22:19:44.610Z",
  "fecha_actualizacion": "2025-04-01T10:00:00.000Z",
  "id": "3e79d145-397a-4e50-bb74-8c6a88c93fa2"
}
```

Tabla Resumen

Endpoint	Método	Auth	Descripción
/consumibles/	GET	JWT	Lista consumibles
/consumibles/	POST	Público	Crea nuevo consumible
/consumible/{id}	GET	JWT	Consulta consumible

Usuarios

Usuarios

POST	/users/	Crear usuario		▼
GET	/users/	Listar usuarios		▼
POST	/user/{id}	Consultar usuario por ID		▼
PUT	/user/{id}	Actualizar usuario		▼
DELETE	/user/{id}	Eliminar usuario		▼

1. POST /users/ - Crear Usuario

Descripción:

Registra un nuevo usuario en el sistema.

Request Body (Ejemplo):

```
{
  "Persona_ID": "2d2f0e84-19d0-4fa4-81e3-8ddcbd2b94e0",
  "Nombre_Usuario": "juanperez",
  "Correo_Electronico": "juan.perez@example.com",
  "Contraseña": "MiContrasenaSegura123",
  "Numero_Telefonico_Movil": "5551234567",
  "Estatus": "Activo"
}
```

Validaciones:

- Correo_Electronico: Formato válido y único.
- Contraseña: Mínimo 8 caracteres (mayúsculas, números y símbolos).
- Estatus: Solo acepta "Activo" o "Inactivo".

Respuesta Exitosa (201 Created):

```
{
  "ID": "1a2b3c4d-5e6f-7890-abcd-1234567890ef",
  "Nombre_Usuario": "juanperez",
}
```

```
"Correo_Electronico": "juan.perez@example.com",  
"Estatus": "Activo",  
"Fecha_Registro": "2025-03-21T22:19:44.610Z"  
}
```

2. GET /users/ - Listar Usuarios

Descripción:

Obtiene una lista paginada de usuarios.

Parámetros Query:

Parámetro	Tipo	Requerido	Valor por Defecto
skip	integer	No	0
limit	integer	No	10

Respuesta Exitosa (200 OK):

```
[  
  {  
    "ID": "1a2b3c4d-5e6f-7890-abcd-1234567890ef",  
    "Nombre_Usuario": "juanperez",  
    "Correo_Electronico": "juan.perez@example.com",  
    "Estatus": "Activo"  
  }  
]
```

3. PUT /user/{id} - Actualizar Usuario

Descripción:

Modifica los datos de un usuario existente.

Parámetros Path:

Parámetro	Tipo	Descripción
id	string	ID del usuario.

Request Body (Ejemplo):

```
{  
  "Numero_Telefonico_Movil": "5557654321",  
  "Estatus": "Inactivo"  
}
```

Respuesta Exitosa (200 OK):

```
{  
  "mensaje": "Usuario actualizado",  
  "ID": "1a2b3c4d-5e6f-7890-abcd-1234567890ef",  
  "Cambios": {  
    "Numero_Telefonico_Movil": "5557654321",  
    "Estatus": "Inactivo"  
  }  
}
```

Instalación y Configuración del Backend

Requisitos Previos

- **Python 3.10+**
- **Git**
- **MySQL:** Base de datos configurada y en ejecución.

Pasos para Despliegue

1. Crear y activar entorno virtual (Windows)

```
python -m venv venv
```

```
venv\Scripts\activate
```

2. Instalar dependencias

```
pip install -r requirements.txt
```

3. Configurar variables de entorno (.env)

```
"DATABASE_URL=mysql+pymysql://usuario:contraseña@localhost/hospital"
```

4. Levantar servidor

```
uvicorn app.main:app --reload
```

Configuración Clave

Variable	Ejemplo	Descripción
DATABASE_URL	mysql+pymysql://root:1234@localhost/hospital	URL de conexión a MySQL.

Verificación

1. Accede a la documentación interactiva:

- Swagger UI: <https://integradora-backend-linux.onrender.com/docs/>