INTRODUCCIÓN

Funcionalidad

La API **OdontoCare** es una aplicación diseñada para modernizar las operaciones de una red de clínicas dentales, permitiendo la gestión de sus usuarios, pacientes y médicos, la disponibilidad de los odontólogos y la organización y creación de citas médicas en las diferentes clínicas de la red.

Descripción

OdontoCare es una API REST que utiliza **Flask** y está organizada mediante **Blueprints** para asegurar su modularidad y escalabilidad. Su estructura está basada en una arquitectura distribuida mediante contenedores **Docker**. La persistencia de los datos generados por la aplicación se garantiza mediante el uso de una base de datos **SQLite** gestionada a través de **SQLAlchemy** como ORM para gestionar las entidades, las relaciones entre ellas y las operaciones **CRUD**. Todas las respuestas generadas por la API se entregan en formato **JSON** y se han seguido estándares de buenas prácticas como: validación de datos (marshmallow), manejo de excepciones, paginación y la generación de esta documentación básica.

La URL de acceso a la API es: http://127.0.0.1:5001

La API implementa un sistema de autenticación segura mediante **tokens**, y autorización mediante el uso de **roles** asignados a los diferentes tipos de usuarios de la API, esto garantiza el acceso seguro y controlado a los diferentes recursos del sistema.

Se incluye un módulo independiente, el **Cliente Python**, que consume los servicios de la API mediante la biblioteca **requests** asegurándose de:

- hacer **login** en la API con el usuario **admin**,
- procesar los ficheros de **datos.csv** y enviar a la base de datos los registros de estos ficheros,
- crear la primera **cita médica** en el sistema,
- **imprimir por consola** un JSON con la información de la cita creada.

El **Cliente Python** realiza la carga inicial de registros en la base de datos procesando los diferentes archivos 'csv' y enviando cada registro por separado a los diferentes endpoints de la API mediante la biblioteca **requests** para demostrar la correcta interacción entre cliente y servidor.

Se han incluido los archivos **requirements.txt** para especificar todas las librerías y dependencias necesarios para el funcionamiento correcto del proyecto.

AUTENTICACIÓN

Los usuarios de la API (gestores, médicos y pacientes) pueden acceder a sus servicios identificándose mediante la utilización de un 'username' y una 'password'. El sistema controla su acceso mediante un mecanismo de autenticación basado en tokens (**JWT**), verifica que el usuario existe en la base de datos y se genera un token válido para ese usuario.

Este token se envía en el header: **Authorization: Bearer<token>.**

Todos los endpoints de la API que necesiten estar protegidos, validan este token para permitir el acceso del usuario a su funcionalidad y, además, controlan el tipo de usuario mediante el campo **rol** para evitar el acceso de usuarios no permitidos a algunas funcionalidades de la API.

ARQUITECTURA DE LA API

La API está implementada utilizando una arquitectura modular basada en **Blueprints** de Flask y está organizada en dos contenedores **Docker**:

- el **servicio de gestión**: que, a su vez, agrupa los módulos de:
    - autenticación (**auth_dp**) que está encargado de las operaciones relacionadas con el acceso seguro, y
    - administración y gestión de clínicas, pacientes y doctores (**admin_bp**) que está encargado de las tareas administrativas: gestión de usuarios, carga de datos, operaciones de consulta, etc.
- el **servicio de citas** (**citas_bp**): que es el encargado de la planificación, administración y control de las citas médicas, permitiendo la creación, actualización, consulta y eliminación de citas, la validación de disponibilidad de los doctores y las reglas de gestión que eviten duplicidad de citas en la agenda.

Las operaciones básicas del sistema de Gestión de Citas son:

- la **creación** de una nueva cita, realizando las operaciones de:
    - **comprobación** de que los usuarios implicados (doctor, paciente y centro médico) existen en la Base de Datos,
    - **disponibilidad** del doctor para evitar la doble reserva de citas.
- la **cancelación** de una cita agendada, verificando que la cita existe en la base de datos y que no está ya cancelada,
- la **obtención de listados** de citas con las restricciones siguientes:
    - los doctores sólo pueden listar sus propias citas,
    - la secretaria sólo puede listar las citas filtrándolas por fecha,
    - el administrador del sistema puede listar las citas filtrando por cualquier parámetro.
- la **modificación** de datos de una cita: permite modificar datos de la cita como la fecha, el doctor, el paciente o el centro médico.

FUNCIONALIDADES DE LA API (ENDPOINTS)

Endpoints y métodos disponibles

ruta de acceso básico '/': consulta al punto de acceso básico de la API que devuelve información básica de la API: autor, endpoints, versión y un mensaje de bienvenida.

URL: http://127.0.0.1:5001

Método: GET

Descripción: consulta que devuelve información básica de la API: autor, endpoints, versión y un mensaje de bienvenida.
Autenticación: no requerida.
Parámetros de entrada: no requeridos.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)

```
{
   "author": "Miguel Beigveder",
   "endpoints": {
       'main': '/servicio_gestion/modulos/main',
       'auth': '/servicio_gestion/modulos/auth',
       'admin': 'servicio_gestion/modulos/admin',
       'citas': 'servicio_citas/modulos/citas',
   },
   "message": "Bienvenido a la API RESTful de OdontoCare",
   "status": "success",
   "versión": "1.0"
}
```

Autenticación

login: consulta que permite autenticar a un usuario mediante credenciales (usuario/password/rol) y generar un token de acceso (JWT) para ser utilizado en las siguientes peticiones protegidas. Devuelve información básica de la API: autor, endpoints, versión y un mensaje de bienvenida.

URL: http://127.0.0.1:5001/auth/login

Método: POST

Descripción: consulta que permite autenticar a un usuario mediante credenciales (usuario/password/rol) generando un token de acceso (JWT) para ser utilizado en las siguientes peticiones protegidas que realice el usuario.
Devuelve información básica de la autenticación: token, rol y 'expiration date'.
Autenticación: requerida.
Parámetros de entrada:

```
{
  "username": "user_admin",
  "password": "pass_user_admin",
  "rol": "admin"
}
```

Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)

```
{
   "expires_at": "2026-01-19T23+00:00Z",
   "rol": "admin",
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyX2FkbWluIiwicm9sIjoiYWRtaW4iLC
JpYXQiOjE3Njg4MjM0NTksImV4cCI6MTc2ODg2NjY1OX0.eJD3WCuHRSVWeLwtUuB_zbbG_vG
OLt7T1PgyInyOKYc"
}
```

Errores:

- 404 NOT FOUND

```
{
   "error": "Usuario: user_admin_27 no encontrado."
}
```

- 401 UNAUTHORIZED

```
{
   "error": "Password incorrecta"
}
```

Servicio de Gestión

Listados:

listados doctores: consulta que obtiene una lista paginada de todos los doctores registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.

URL: http://127.0.0.1:5001/admin/doctores

Método: GET

Descripción: consulta que permite obtener un listado con los datos de todos los doctores incluidos en la base de datos. El listado se obtiene paginado. Se pueden indicar los datos de la paginación mediante query params.
Devuelve el listado paginado con la información básica de los doctores registrados: nombre, especialidad, id_usuario e id_doctor y los datos de la paginación.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada: no requeridos.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)


```
{
  "doctores": [
    {
      "especialidad": "Odontopediatría",
      "id_doctor": 1,
      "id_usuario": 4,
      "nombre": "Jesusa Delia Barrios"
    },
    {
      "especialidad": "Odontopediatría",
      "id_doctor": 2,
      "id_usuario": 5,
      "nombre": "Adelina Losa Fiol"
    },
    {
      "especialidad": "Endodoncia",
      "id_doctor": 3,
      "id_usuario": 6,
      "nombre": "Calixta Bastida Palacio"
    },
    {
      "especialidad": "Endodoncia",
      "id_doctor": 4,
      "id_usuario": 7,
      "nombre": "Inés Sánchez Torrents"
```

listados doctores: consulta que obtiene una lista paginada de todos los doctores registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.


```
      },
      {
        "especialidad": "Ortodoncia",
        "id_doctor": 5,
        "id_usuario": 8,
        "nombre": "Roberto Abascal Ripoll"
      }
  ],
  "pagination": {
    "current_page": 1,
    "per_page": 5,
    "total_doctores": 10,
    "total_pages": 2
  }
}
```


Respuesta exitosa a una paginación errónea: (200 OK)


```
{
  "error": "La página solicitada no contiene doctores."
}
```


Errores:

- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token inválido"
}
```


listados pacientes: consulta que obtiene una lista paginada de todos los pacientes registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.

URL: http://127.0.0.1:5001/admin/pacientes

Método: GET

Descripción: consulta que permite obtener un listado con los datos de todos los pacientes incluidos en la base de datos. El listado se obtiene paginado.
Devuelve el listado paginado con la información básica de los pacientes registrados: nombre, teléfono, estado, id_usuario e id_paciente y los datos de la paginación.

listados pacientes: consulta que obtiene una lista paginada de todos los pacientes registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.

Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada: no requeridos.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)


```
{
  "pacientes": [
    {
      "estado": "activo",
      "id_paciente": 1,
      "id_usuario": 13,
      "nombre": "Sebastián Lago Cadenas",
      "teléfono": "+34 887 37 06 11"
    },
    {
      "estado": "activo",
      "id_paciente": 2,
      "id_usuario": 14,
      "nombre": "Pío Coca Casado",
      "teléfono": "+34 883 53 71 25"
    },
    {
      "estado": "inactivo",
      "id_paciente": 3,
      "id_usuario": 15,
      "nombre": "Melania Rincón Vizcaíno",
      "teléfono": "+34 884 30 18 23"
    },
    {
      "estado": "inactivo",
      "id_paciente": 4,
      "id_usuario": 16,
      "nombre": "Ámbar Company Real",
      "teléfono": "+34 806 29 07 56"
    },
    {
      "estado": "activo",
      "id_paciente": 5,
      "id_usuario": 17,
      "nombre": "Roxana Múñiz Garmendia",
      "teléfono": "+34 975 01 53 66"
    }
  ],
```

listados pacientes: consulta que obtiene una lista paginada de todos los pacientes registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.


```
  "pagination": {
    "current_page": 1,
    "per_page": 5,
    "total pacientes": 20,
    "total_pages": 4
  }
}
```


Respuesta exitosa a una paginación errónea: (200 OK)


```
{
  "error": "La página solicitada no contiene pacientes."
}
```


Errores:

- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token inválido"
}
```


listados usuarios: consulta que obtiene una lista paginada de todos los usuarios registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.

URL: http://127.0.0.1:5001/admin/usuarios

Método: GET

Descripción: consulta que permite obtener un listado con los datos de todos los usuarios incluidos en la base de datos. El listado se obtiene paginado.
Devuelve el listado paginado con la información básica de los usuarios registrados: username, rol, id_usuario y los datos de la paginación.
Autenticación: requerida por Bearer Token.
Roles admitidos: ‘admin’ y ‘secretaria’
Parámetros de entrada: no requeridos.
Formato de respuesta: ‘application/json’.
Respuesta exitosa: (200 OK)


```
{
  "pagination": {
    "current_page": 1,
```

listados usuarios: consulta que obtiene una lista paginada de todos los usuarios registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.


```
    "per_page": 5,
    "total_pages": 7,
    "total_usuarios": 33
  },
  "usuarios": [
    {
      "id_usuario": 1,
      "rol": "admin",
      "username": "user_admin"
    },
    {
      "id_usuario": 2,
      "rol": "secretaria",
      "username": "user_secretaria"
    },
    {
      "id_usuario": 3,
      "rol": "medico",
      "username": "user_medico_1"
    },
    {
      "id_usuario": 4,
      "rol": "medico",
      "username": "user_medico_2"
    },
    {
      "id_usuario": 5,
      "rol": "medico",
      "username": "user_medico_3"
    }
  ]
}
```


Respuesta exitosa a una paginación errónea: (200 OK)


```
{
  "error": "La página solicitada no contiene usuarios."
}
```


Errores:

- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
```

listados usuarios: consulta que obtiene una lista paginada de todos los usuarios registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.


```
  "mensaje": "Token inválido"
}
```


listados centros médicos: consulta que obtiene una lista paginada de los centros médicos registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.

URL: http://127.0.0.1:5001/admin/centros_medicos

Método: GET

Descripción: consulta que permite obtener un listado con los datos de los centros médicos incluidos en la base de datos. El listado se obtiene paginado.
Devuelve el listado paginado con la información básica de los centros médicos registrados: nombre, dirección, id_centro y los datos de la paginación.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'.
Parámetros de entrada: no requeridos.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)


```
{
  "centros médicos": [
    {
      "dirección": "Cuesta de Clemente Suarez, 3 Piso 7 Madrid, 28023",
      "id_centro": 1,
      "nombre": "Salud Dental"
    },
    {
      "dirección": "Glorieta Sebastián Rosales 64 Puerta 8 Madrid, 28009",
      "id_centro": 2,
      "nombre": "Bright Smiles"
    }
  ],
  "pagination": {
    "current_page": 1,
    "per_page": 5,
    "total_centros": 2,
    "total_pages": 1
  }
}
```


Respuesta exitosa a una paginación errónea: (200 OK)

listados centros médicos: consulta que obtiene una lista paginada de los centros médicos registrados en el sistema. Permite opcionalmente filtrar por página y modificar el número de registros por página.


```
{
  "error": "La página solicitada no contiene Centros Médicos."
}
```


Errores:

- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token inválido"
}
```


Registros individuales:

registro doctor: obtiene los detalles completos de un médico específico (doctor) registrado en el sistema, utilizando su identificador único (id_doctor).

URL: http://127.0.0.1:5001/admin/doctor/<int:id_doctor>

Método: GET

Descripción: consulta que permite obtener los datos individuales de un doctor identificándolo por su 'id_doctor'.
Devuelve la información básica del doctor registrado: nombre, especialidad, id_usuario e id_doctor.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada: id_doctor.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)


```
{
  "Doctor": {
    "especialidad": "Endodoncia",
    "id_doctor": 3,
    "id_usuario": 6,
    "nombre": "Calixta Bastida Palacio"
  }
}
```


Errores:

- 401 UNAUTHORIZED


```
{
```

registro doctor: obtiene los detalles completos de un médico específico (doctor) registrado en el sistema, utilizando su identificador único (id_doctor).


```
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token inválido"
}
```


- 404 NOT FOUND


```
{
  "error": "Doctor no encontrado "
}
```


registro paciente: obtiene los detalles completos de un paciente específico registrado en el sistema, utilizando su identificador único (id_paciente).

URL: http://127.0.0.1:5001/admin/paciente/<int:id_paciente>

Método: GET

Descripción: consulta que permite obtener los datos individuales de un paciente identificándolo por su 'id_paciente'.
Devuelve la información básica del paciente registrado: nombre, teléfono, estado, id_usuario e id_paciente.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada: id_paciente.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)


```
{
  "Paciente": {
    "estado": "activo",
    "id_paciente": 1,
    "id_usuario": 13,
    "nombre": "Sebastián Lago Cadenas",
    "teléfono": "+34 887 37 06 11"
  }
}
```


Errores:

- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED

registro paciente: obtiene los detalles completos de un paciente específico registrado en el sistema, utilizando su identificador único (id_paciente).

```
{
  "mensaje": "Token inválido"
}
```

- 404 NOT FOUND

```
{
  "error": "Paciente no encontrado "
}
```

registro usuario: obtiene los detalles completos de un usuario específico registrado en el sistema, utilizando su identificador único (id_usuario).

URL: http://127.0.0.1:5001/admin/usuario/<int:id_usuario>

Método: GET

Descripción: consulta que permite obtener los datos individuales de un usuario identificándolo por su 'id_usuario'.
Devuelve la información básica del usuario registrado: username, rol e id_usuario.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada: id_usuario.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)

```
{
  "Usuario": {
    "id_usuario": 5,
    "rol": "medico",
    "username": "user_medico_3"
  }
}
```

Errores:

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token no proporcionado"
}
```

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token inválido"
}
```

- 404 NOT FOUND

```
{
```

registro usuario: obtiene los detalles completos de un usuario específico registrado en el sistema, utilizando su identificador único (id_usuario).


```
  "error": "Usuario no encontrado "
}
```


registro centro_médico: obtiene los detalles completos de un centro médico específico registrado en el sistema, utilizando su identificador único (id_centro).

URL: http://127.0.0.1:5001/admin/centro_medico/<int:id_centro>

Método: GET

Descripción: consulta que permite obtener los datos de un centro médico identificándolo por su 'id_centro'.
Devuelve la información básica del centro médico registrado: nombre, dirección e id_centro.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada: id_centro.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)


```
{
  "Centro Médico": {
    "dirección": "Glorieta Sebastián Rosales 64 Puerta 8 Madrid, 28009",
    "id_centro": 2,
    "nombre": "Bright Smiles"
  }
}
```


Errores:

- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token inválido"
}
```


- 404 NOT FOUND


```
{
  "error": "Centro Médico no encontrado "
}
```

Creación de registro individual:

registro nuevo doctor: consulta que registra un nuevo doctor en el sistema. Crea un registro de usuario y otro de doctor en sus respectivas tablas con la información proporcionada (nombre, especialidad, rol, username y password) y devuelve los datos de ID del doctor creado.

URL: http://127.0.0.1:5001/admin/doctor

Método: POST

Descripción: consulta que permite registrar, en la tabla de doctores de la base de datos, un nuevo doctor. Antes de añadir el registro de doctor, lo añade como usuario en la tabla de usuarios de la base de datos.
Devuelve la información básica del usuario/doctor registrado: id_doctor, id_usuario y el mensaje 'Usuario y Doctor registrados exitosamente'.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada en el cuerpo de la solicitud:

```
{
 "username": "user_medico_111",
 "password": "pass_user_medico_111",
 "rol": "medico",
 "nombre": "Pablo Loco Flores",
 "especialidad": "Odontología"
}
```

Formato de respuesta: 'application/json'.
Respuesta exitosa: (201 CREATED)

```
{
  "id_doctor": 11,
  "id_usuario": 35,
  "message": "Usuario y Doctor registrados exitosamente"
}
```

Errores:

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token no proporcionado"
}
```

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token inválido"
}
```

- 400 BAD REQUEST

```
{
  "error": "El campo \"password\" no debe estar vacío."
}
```

- 400 BAD REQUEST

```
{
```

registro nuevo doctor: consulta que registra un nuevo doctor en el sistema. Crea un registro de usuario y otro de doctor en sus respectivas tablas con la información proporcionada (nombre, especialidad, rol, username y password) y devuelve los datos de ID del doctor creado.


```
  "rol": [
    "Must be one of: admin, medico, secretaria, paciente."
  ]
}
```


- 409 CONFLICT


```
{
  "error": "El usuario ya existe"
}
```


registro nuevo paciente: consulta que registra un nuevo paciente en el sistema. Crea un registro de usuario y otro de paciente en sus respectivas tablas con la información proporcionada (nombre, teléfono, rol, estado, username y password) y devuelve los datos de ID del paciente creado.

URL: http://127.0.0.1:5001/admin/paciente

Método: POST

Descripción: consulta que permite registrar, en la tabla de pacientes de la base de datos, un nuevo paciente. Antes de añadir el registro de paciente, lo añade como usuario en la tabla de usuarios de la base de datos.
Devuelve la información básica del usuario/paciente registrado: id_paciente, id_usuario y el mensaje 'Usuario y Paciente registrados exitosamente'.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada:


```
{
 "username": "user_paciente_33",
 "password": "pass_user_paciente_33",
 "rol": "paciente",
 "nombre": "Francisco Pérez Prou",
 "teléfono": "+34 689 43 22 19",
 "estado": "activo"
}
```


Formato de respuesta: 'application/json'.
Respuesta exitosa: (201 CREATED)


```
{
  "id_paciente": 21,
  "id_usuario": 36,
  "message": "Usuario y Paciente registrados exitosamente"
```

registro nuevo paciente: consulta que registra un nuevo paciente en el sistema. Crea un registro de usuario y otro de paciente en sus respectivas tablas con la información proporcionada (nombre, teléfono, rol, estado, username y password) y devuelve los datos de ID del paciente creado.

```
}
```

Errores:

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token no proporcionado"
}
```

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token inválido"
}
```

- 400 BAD REQUEST

```
{
  "error": "El campo \"password\" no debe estar vacío."
}
```

- 400 BAD REQUEST

```
{
  "teléfono": [
    "Length must be between 3 and 25."
  ]
}
```

- 409 CONFLICT

```
{
  "error": "El usuario ya existe"
}
```

registro nuevo usuario: consulta que registra un nuevo usuario en el sistema. Crea un registro de usuario en la tabla de usuarios con la información proporcionada (rol, username y password) y devuelve un mensaje informando de la creación del registro.

URL: http://127.0.0.1:5001/admin/usuario

Método: POST

Descripción: consulta que permite registrar, en la tabla de usuarios de la base de datos, un nuevo usuario.
Devuelve el mensaje: 'Usuario registrado exitosamente'.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada:

registro nuevo usuario: consulta que registra un nuevo usuario en el sistema. Crea un registro de usuario en la tabla de usuarios con la información proporcionada (rol, username y password) y devuelve un mensaje informando de la creación del registro.


```
{
 "username": "user_paciente_115",
 "password": "pass_user_paciente_115",
 "rol": "paciente"
}
```


Formato de respuesta: ‘application/json’.

Respuesta exitosa: (201 CREATED)


```
{
"message": "Usuario registrado exitosamente"
}
```


Errores:

- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token inválido"
}
```


- 400 BAD REQUEST


```
{
  "error": "El campo \"password\" no debe estar vacío."
}
```


- 400 BAD REQUEST


```
{
  "rol": [
    "Must be one of: admin, medico, secretaria, paciente."
  ]
}
```


- 409 CONFLICT


```
{
  "error": "El usuario ya existe"
}
```

registro nuevo centro_médico: consulta que registra un nuevo centro médico en el sistema. Crea un registro de centro médico en la tabla de centros médicos con la información proporcionada (nombre y dirección) y devuelve un mensaje informando de la creación del registro.

URL: http://127.0.0.1:5001/admin/centro_medico>

Método: POST

Descripción: consulta que permite registrar, en la tabla de centros médicos de la base de datos, un nuevo centro médico.
Devuelve el mensaje de 'Centro Médico registrado exitosamente'.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada:

```
{
  "nombre": "Sonrisa Luminosa",
  "dirección": "Calle Rosales 34 bajo, Madrid 28033"
}
```

Formato de respuesta: 'application/json'.
Respuesta exitosa: (201 CREATED)

```
{
"message": "Centro Médico registrado exitosamente"
}
```

Errores:

- 401 UNAUTHORIZED

```
{
   "mensaje": "Token no proporcionado"
}
```

- 401 UNAUTHORIZED

```
{
   "mensaje": "Token inválido"
}
```

- 400 BAD REQUEST

```
{
   "nombre": [
      " Length must be between 3 and 40."
   ]
}
```

- 409 CONFLICT

```
{
   "error": "El centro médico ya existe"
}
```

Identificación doctor por username:

identificación doctor por username: consulta que obtiene la información detallada de un doctor específico utilizando su nombre de usuario (username) único. Este endpoint se utiliza para consultar el perfil y las credenciales asociadas a un médico en el sistema.

URL: http://127.0.0.1:5001/admin/ doctor/username

Método: GET

Descripción: consulta que permite obtener los datos de un registro individual de doctor a partir de su username mediante 'query params'.
Devuelve la información básica del doctor registrado: nombre, especialidad, id_usuario e id_doctor.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'medico'
Parámetros de entrada: 'username' como query params.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)

```
{
  "especialidad": "Odontopediatría",
  "id_doctor": 2,
  "id_usuario": 5,
  "nombre": "Adelina Losa Fiol"
}
```

Errores:

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token no proporcionado"
}
```

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token inválido"
}
```

- 404 NOT FOUND

```
{
  "error": "Usuario no encontrado."
}
```

- 400 BAD REQUEST

```
{
  "error": "Se requiere el parámetro username"
}
```

Servicio de Citas

Listado de citas:

listado de citas: consulta que obtiene una lista de citas médicas (agenda) asociadas a un doctor específico, un paciente específico, una fecha, o el estado de las citas (activa, cancelada) dependiendo del rol del peticionario del listado y del query param que se indique en la consulta, permite visualizar la disponibilidad, citas programadas o historial de un médico.

URL: http://127.0.0.1:5001/citas/listar_citas

Método: GET

Descripción: consulta que permite listar las citas existentes mediante query params.
Devuelve la cita o el listado de citas según el rol del peticionario y el query param presente en la consulta:

- id_doctor:
  - rol = 'admin': lista las citas por el id_doctor proporcionado,
  - rol = 'medico': lista las citas por el id_doctor, sólo si coincide con el id_doctor del peticionario,
- fecha:
  - roles = 'admin' o 'secretaria': lista las citas por la fecha proporcionada,
- id_centro:
  - rol = 'admin': lista las citas por el id_centro proporcionado,
- estado:
  - rol = 'admin': lista las citas por el estado ('activa' o 'cancelada') proporcionado,
- id_paciente:
  - rol = 'admin': lista las citas del id_paciente proporcionado,

Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin', 'secretaria' y 'médico'.
Parámetros de entrada: admite uno o varios de los siguientes atributos en el cuerpo de la petición, se listan las citas según el rol del peticionario y el parámetro que se proporcione en la url de la consulta.

- fecha,
- id_paciente,
- id_doctor,
- id_centro.

Formato de respuesta: 'application/json'.
Respuesta exitosa: (200 OK)


```
{
  "Citas": [
    {
      "estado": "activa",
      "fecha": "Thu, 19 Feb 2026 10:25:00 GMT",
      "id_centro": 4,
      "id_cita": 5,
      "id_doctor": 7,
```

listado de citas: consulta que obtiene una lista de citas médicas (agenda) asociadas a un doctor específico, un paciente específico, una fecha, o el estado de las citas (activa, cancelada) dependiendo del rol del peticionario del listado y del query param que se indique en la consulta, permite visualizar la disponibilidad, citas programadas o historial de un médico.


```
        "id_paciente": 2,
        "id_usuario": 14,
        "motivo": "Revisión periódica"
      }
  ]
}
```


Errores:

- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token inválido"
}
```


- 403 FORBIDDEN


```
{
  "mensaje": "Permiso denegado"
}
```


- 200 OK


```
{
  "error": "La consulta no contiene ningún resultado."
}
```


- 201 CREATED


```
{
  "error": "El doctor no tiene citas agendadas"
}
```


- 400 BAD REQUEST


```
{
  "error": "No está autorizado a listar las citas de un paciente."
}
```


- 400 BAD REQUEST


```
{
  "error": " No está autorizado a ver las citas de otro doctor."
}
```

Creación:

agendar cita: consulta que crea una nueva cita en el sistema con los datos proporcionados en el cuerpo de la solicitud: fecha, motivo, estado, y los identificadores únicos de usuario, doctor, paciente y centro médico. Este endpoint valida los datos introducidos, la existencia de los registros y la disponibilidad del médico en esa fecha, antes de confirmar la cita.

URL: http://127.0.0.1:5001/citas/agendar

Método: POST

Descripción: consulta que permite agendar una cita a partir de los datos proporcionados en el cuerpo de la solicitud.
Devuelve la información básica de la cita: fecha, estado, motivo y los identificadores de cita, paciente, usuario, doctor y centro médico, además del mensaje: 'Cita registrada'.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'.
Parámetros de entrada:

```
{
 "fecha": "22-02-2026 10:15",
 "motivo": "Revisión periódica",
 "estado": "activa",
 "id_usuario": "14",
 "id_paciente": "2",
 "id_doctor": "7",
 "id_centro": "1"
}
```

Formato de respuesta: 'application/json'.
Respuesta exitosa: (201 CREATED)

```
{
   "cita": {
      "estado": "activa",
      "fecha": "Sun, 22 Feb 2026 10:15:00 GMT",
      "id_centro": 1,
      "id_cita": 7,
      "id_doctor": 7,
      "id_paciente": 2,
      "id_usuario": 14,
      "motivo": "Revisión periódica"
   },
   "message": "Cita registrada"
}
```

Errores:

- 400 BAD REQUEST

```
{
'message': 'Missing JSON in request'
}
```

agendar cita: consulta que crea una nueva cita en el sistema con los datos proporcionados en el cuerpo de la solicitud: fecha, motivo, estado, y los identificadores únicos de usuario, doctor, paciente y centro médico. Este endpoint valida los datos introducidos, la existencia de los registros y la disponibilidad del médico en esa fecha, antes de confirmar la cita.

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token no proporcionado"
}
```

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token inválido"
}
```

- 404 NOT FOUND

```
{
  "error": "Usuario no encontrado."
}
```

- 400 BAD REQUEST

```
{
  "id_centro": ["Not a valid integer."]
}
```

- 200 OK

```
{
  "error": "Error al obtener datos del paciente: 404",
  "message": "El paciente no existe en la base de datos"
}
```

- 200 OK

```
{
  "error": "Error al obtener datos del doctor: 404",
  "message": "El doctor no existe en la base de datos"
}
```

- 200 OK

```
{
  "error": "Error al obtener datos del centro médico: 404",
  "message": "El centro médico no existe en la base de datos"
}
```

- 200 OK

```
{
  "error": "El Doctor ya tiene una cita a esa hora en esa fecha"
}
```

Cancelación de cita:

cancelación de cita: consulta que cambia en el sistema el estado de una cita específica a "cancelada". Esta acción no es definitiva, y se ha elegido a diferencia de una eliminación (DELETE), para mantener el registro histórico de la cita con su nuevo estado.

URL: http://127.0.0.1:5001/citas/cancelar/<int:id_cita>

Método: PUT

Descripción: consulta que permite cancelar una cita existente a partir de su 'id_cita'.
Devuelve la información básica de la cita cancelada: estado, fecha, motivo y los identificadores únicos de centro, usuario, paciente, cita y doctor, además del mensaje: 'Cita cancelada'.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada: id_cita.
Formato de respuesta: 'application/json'.
Respuesta exitosa: (201 CREATED)

```
{
  "cita": {
    "estado": "cancelada",
    "fecha": "Sun, 22 Feb 2026 10:15:00 GMT",
    "id_centro": 1,
    "id_cita": 7,
    "id_doctor": 7,
    "id_paciente": 2,
    "id_usuario": 14,
    "motivo": "Revisión periódica"
  },
  "message": "Cita cancelada"
}
```

Errores:

- 200 OK

```
{
  "error": "La cita ya estaba cancelada."
}
```

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token no proporcionado"
}
```

- 401 UNAUTHORIZED

```
{
  "mensaje": "Token inválido"
}
```

- 404 NOT FOUND

```
{
  "error": "La cita que quieres cancelar, no existe."
```

cancelación de cita: consulta que cambia en el sistema el estado de una cita específica a "cancelada". Esta acción no es definitiva, y se ha elegido a diferencia de una eliminación (DELETE), para mantener el registro histórico de la cita con su nuevo estado.

```
}
```

Modificación de cita:

modificación de cita: consulta que actualiza los datos de una cita existente especificada por su ID único. Este endpoint permite la modificación completa de los campos de la cita (fecha, paciente, doctor, centro médico) dependiendo de los campos presentes en el cuerpo de la solicitud.

URL: http://127.0.0.1:5001/citas/modificar/<int:id_cita>

Método: PUT

Descripción: consulta que permite modificar una cita existente a partir de su id_cita.
Devuelve la información básica de la cita modificada: estado, fecha, motivo y los identificadores únicos de centro, usuario, paciente, cita y doctor, además del mensaje: 'Cita actualizada'.
Autenticación: requerida por Bearer Token.
Roles admitidos: 'admin' y 'secretaria'
Parámetros de entrada: admite uno o varios de los siguientes atributos en el cuerpo de la petición, se modifica la cita según el parámetro que se proporcione en el cuerpo de la consulta.

- fecha: verifica antes que el doctor no tenga una cita en esa fecha,
- id_paciente: verifica antes que el paciente no esté inactivo,
- id_doctor: verifica antes que el doctor no tenga una cita en esa fecha,
- id_centro: verifica que exista el centro médico.

Formato de respuesta: 'application/json'.
Respuesta exitosa: (201 CREATED)

```
{
  "cita": {
    "estado": "activa",
    "fecha": "Fri, 02 Jan 2026 08:00:00 GMT",
    "id_centro": 1,
    "id_cita": 4,
    "id_doctor": 7,
    "id_paciente": 6,
    "id_usuario": 18,
    "motivo": "revisión"
  },
  "message": "Cita actualizada"
}
```

Errores:

- 200 OK

modificación de cita: consulta que actualiza los datos de una cita existente especificada por su ID único. Este endpoint permite la modificación completa de los campos de la cita (fecha, paciente, doctor, centro médico) dependiendo de los campos presentes en el cuerpo de la solicitud.


```
{
  "error": "El Doctor ya tiene una cita a esa hora en esa fecha"
}
```


- 200 OK


```
{
  "error": "Error al obtener datos: 404"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token no proporcionado"
}
```


- 401 UNAUTHORIZED


```
{
  "mensaje": "Token inválido"
}
```