



# API TatasApp



## Creadores

- Alexander Aguilera
- Andrea Pino
- David Guentelican



## Arquitectura General






El proyecto está basado en una arquitectura de capas para APIs, que incluye:

- **Models:** Definición de tablas y relaciones (SQLAlchemy)
- **Schemas:** Validación y serialización de datos (Pydantic)
- **Routers:** Definición de rutas/endpoints y lógica de negocio
- **Utils:** Funciones auxiliares y validaciones personalizadas
- **Settings:** Configuración centralizada (DB, variables de entorno)
- **Auth:** Lógica de autenticación (hashing, JWT)

## Estructura de Carpetas

```
app/  
├─ main.py  
├─ models.py  
├─ routers/  
│   ├─ usuario.py  
│   ├─ familiar.py  
│   ├─ evento.py  
│   └─ alerta.py  
├─ schemas/  
│   ├─ usuario.py  
│   ├─ familiar.py  
│   ├─ evento.py  
│   └─ alerta.py  
├─ settings/  
│   ├─ config.py  
│   ├─ database.py  
│   └─ dependencies.py  
├─ auth/  
│   ├─ auth.py  
│   ├─ hashing.py  
│   └─ jwt.py  
└─ utils/  
    ├─ helpers.py  
    └─ validations.py
```

## Flujo de Trabajo

1.  Solicitud HTTP del cliente
2.  Router correspondiente recibe la petición y valida con un Schema
3.  Lógica de negocio y acceso a la base de datos
4.  Respuesta serializada al cliente
5.  Manejo de errores personalizado



# Endpoints y Ejemplos



## Usuarios

- **POST** /usuarios/registro\_usuario

Registrar usuario

```
{
  "nombres": "Juan",
  "apellidos": "Pérez",
  "fecha_nacimiento": "1950-01-01",
  "correo": "juan.perez@mail.com",
  "telefono": "912345678",
  "tipo_usuario": 1,
  "contrasena": "Password123",
  "direccion": {
    "direccion_texto": "Calle Falsa 123",
    "adicional": "Depto 4B"
  }
}
```

- **POST** /usuarios/login

Login usuario

```
{
  "correo": "juan.perez@mail.com",
  "contrasena": "Password123"
}
```

- **PATCH** /usuarios/editar-foto-perfil

```
{
  "id": 1,
  "foto_perfil": "https://url.com/foto.jpg"
}
```

- **PATCH** /usuarios/editar-datos

```
{
  "id": 1,
  "nombres": "Juan",
  "apellidos": "Pérez",
  "fecha_nacimiento": "1950-01-01",
  "telefono": "912345678",
  "direccion": {
    "direccion_texto": "Nueva dirección 456",
    "adicional": "Casa"
  }
}
```

- **PATCH** /usuarios/editar-correo

```
{
  "id": 1,
  "correo": "nuevo.correo@mail.com"
}
```

- **PATCH** /usuarios/editar-contrasena

```
{
  "id": 1,
  "contrasena": "NuevaPassword123"
}
```

## Rutas GET de Usuarios

- **GET** /usuarios/contactos-registrados

Retorna la lista de usuarios registrados como familiares (tipo\_usuario=2).

- **Ejemplo de respuesta:**

```
[
  {
    "id_usuario": 2,
    "telefono": "912345679",
    "tipo_usuario": 2
  }
]
```

- **GET** /usuarios/{usuario\_id}

Retorna los datos completos de un usuario por su ID.

- **Parámetros:**

- `usuario_id` (int, requerido): ID del usuario.

- **Ejemplo de respuesta:**

```
{
  "nombres": "Juan",
  "apellidos": "Pérez",
  "fecha_nacimiento": "1950-01-01",
  "correo": "juan.perez@mail.com",
  "telefono": "912345678",
  "tipo_usuario": 1,
  "direccion_rel": {
    "direccion_texto": "Calle Falsa 123",
    "adicional": "Depto 4B"
  }
}
```

- **GET** `/usuarios/foto-perfil/{usuario_id}`

Retorna la foto de perfil del usuario por su ID.

- **Parámetros:**

- `usuario_id` (int, requerido): ID del usuario.

- **Ejemplo de respuesta:**

```
{
  "id_usuario": 1,
  "foto_perfil": "https://url.com/foto.jpg"
}
```



## Familiares

- **POST** `/familiares/regarstrar-familiar`

```
{
  "adulto_mayor_id": 1,
  "familiar_id": 2
}
```

- **DELETE** `/familiares/eliminar-familiar/{adulto_mayor_id}/{familiar_id}`

Elimina la relación entre un adulto mayor y un familiar.

- **Parámetros:**

- `adulto_mayor_id` (int, requerido): ID del adulto mayor.

- `familiar_id` (int, requerido): ID del familiar.
- **Ejemplo de respuesta:**

```
{
  "status": "success",
  "message": "Familiar eliminado correctamente del grupo familiar"
}
```

## Rutas GET de Familiares

- **GET** `/familiares/familiares-adulto-mayor/{adulto_mayor_id}`

Retorna la lista de familiares asociados a un adulto mayor.

- **Parámetros:**
  - `adulto_mayor_id` (int, requerido): ID del adulto mayor.
- **Ejemplo de respuesta:**

```
[
  {
    "id_adulto_mayor": 1,
    "familiar_rel": {
      "id_usuario": 2,
      "nombres": "Andrea",
      "apellidos": "Pino",
      "correo": "andrea@mail.com",
      "telefono": "912345679",
      "foto_perfil": "https://url.com/foto.jpg"
    }
  }
]
```



## Eventos

- **POST** `/eventos/crear-evento`

```
{
  "usuario_id": 1,
  "nombre": "Cita médica",
  "descripcion": "Control anual",
  "fecha_hora": "2025-06-01T10:00:00",
  "tipo_evento": 1
}
```

- **PUT** /eventos/modificar/{evento\_id}

Modifica los datos de un evento existente.

- **Parámetros:**

- evento\_id (int, requerido): ID del evento a modificar.

- **Body:**

```
{
  "nombre": "Cita médica modificada",
  "descripcion": "Control anual actualizado",
  "fecha_hora": "2025-06-02T11:00:00",
  "tipo_evento": 1
}
```

- **Ejemplo de respuesta:**

```
{
  "status": "success",
  "message": "Evento modificado correctamente",
  "evento_id": 1
}
```

- **DELETE** /eventos/eliminar/{evento\_id}

Elimina un evento por su ID.

- **Parámetros:**

- evento\_id (int, requerido): ID del evento.

- **Ejemplo de respuesta:**

```
{
  "status": "success",
  "message": "Evento eliminado correctamente"
}
```

## Rutas GET de Eventos

- **GET** /eventos/listar?usuario\_id={id}

Retorna todos los eventos asociados al usuario indicado.

- **Parámetros:**

- usuario\_id (int, requerido): ID del usuario logueado.

- **Ejemplo de respuesta:**

```
[
  {
    "id": 1,
    "usuario_id": 1,
    "nombre": "Cita médica",
    "descripcion": "Control anual",
    "fecha_hora": "2025-06-01T10:00:00Z",
    "tipo_evento": 1,
    "tipo_evento_nombre": "Cita Médica"
  }
]
```

- **GET** /eventos/listar-por-familiar?familiar\_id={id}

Retorna los eventos del adulto mayor asociado a un familiar.

- **Parámetros:**

- familiar\_id (int, requerido): ID del usuario familiar.

- **Ejemplo de respuesta:**

```
[
  {
    "id": 2,
    "usuario_id": 1,
    "nombre": "Control de presión",
    "descripcion": "Medirse la presión",
    "fecha_hora": "2025-06-01T09:00:00Z",
    "tipo_evento": 3,
    "tipo_evento_nombre": "Evento Personal"
  }
]
```

- **GET** /eventos/proximos?usuario\_id={id}&minutos={minutos}

Retorna los eventos del usuario que ocurrirán en los próximos X minutos (por defecto 15).

- **Parámetros:**

- usuario\_id (int, requerido): ID del usuario logueado.



- minutos (int, opcional): Minutos hacia el futuro para buscar eventos (por defecto 15).
- **Ejemplo de respuesta:**

```
[
  {
    "id": 5,
    "usuario_id": 1,
    "nombre": "Toma de medicamentos",
    "descripcion": "Pastilla para la presión",
    "fecha_hora": "2025-06-01T10:05:00Z",
    "tipo_evento": 3,
    "tipo_evento_nombre": "Evento Personal"
  }
]
```

## Alertas

- **POST** /alertas/crear-alerta

```
{
  "usuario_id": 1,
  "ubicacion": "-33.4569,-70.6483",
  "mensaje": "¡Ayuda! Caí en la casa",
  "tipo_alerta": 3
}
```

- **PATCH** /alertas/actualizar-estado

```
{
  "id": 10,
  "estado_alerta": 1
}
```

## Rutas GET de Alertas

- **GET** /alertas/obtener-alertas-pendientes/{id\_familiar}

Retorna las alertas pendientes (estado=0) asociadas al adulto mayor de un familiar.

- **Parámetros:**
  - id\_familiar (int, requerido): ID del usuario familiar.
- **Ejemplo de respuesta:**

```
[
  {
    "id": 1,
    "usuario_id": 1,
    "ubicacion": "-33.4569,-70.6483",
    "mensaje": "¡Ayuda! Caí en la casa",
    "tipo_alerta": 3,
    "fecha_hora": "2025-06-01T10:00:00",
    "estado_alerta": 0
  }
]
```

- **GET** /alertas/obtener-alertas-historial/{id\_familiar}

Retorna el historial de alertas (estado=1) asociadas al adulto mayor de un familiar.





- **Parámetros:**

- `id_familiar` (int, requerido): ID del usuario familiar.


- **Ejemplo de respuesta:**

```
[
  {
    "id": 2,
    "usuario_id": 1,
    "ubicacion": "-33.4569,-70.6483",
    "mensaje": "Alerta entregada",
    "tipo_alerta": 2,
    "fecha_hora": "2025-06-01T09:00:00",
    "estado_alerta": 1
  }
]
```

## Características Adicionales

-  **PostgreSQL** como base de datos
-  **JWT** para autenticación
-  **CORS** configurado
-  Manejo personalizado de errores y excepciones

## Ventajas de la Arquitectura

-  Modularidad
-  Escalabilidad
-  Reutilización
-  Seguridad

## Dependencias principales

El proyecto utiliza las siguientes dependencias principales:

- **FastAPI** y **Starlette**: Framework para construir APIs web rápidas y asíncronas.
- **SQLAlchemy**: ORM para la gestión de la base de datos PostgreSQL.
- **psycopg2**: Driver para conectar con bases de datos PostgreSQL.
- **Pydantic**: Validación y serialización de datos.
- **Passlib** y **bcrypt**: Hashing seguro de contraseñas.
- **python-jose**: Manejo de autenticación y generación de tokens JWT.
- **python-dotenv**: Carga de variables de entorno desde archivos `.env`.
- **Uvicorn**: Servidor ASGI para ejecutar la aplicación.
- **CORS Middleware**: Permite el acceso controlado desde distintos orígenes.
- **email-validator** y **python-multipart**: Validación de emails y manejo de formularios/multipart.
- **orjson**, **ujson**: Serialización rápida de JSON.
- **rich**: Salida enriquecida en consola para desarrollo.
- Otras utilidades para manejo de tipos, seguridad y soporte de desarrollo.

Estas dependencias permiten el desarrollo, despliegue y operación segura de la API TatasApp.