

Dossier Técnico: API RESTful de GastanGO

Versión: 1.0

Fecha: 16 de octubre de 2025

Equipo: Grupo 14 - Joel Tapia

1. Introducción

Este documento detalla la especificación técnica de la API RESTful para el sistema GastanGO. La API centraliza y gestiona la lógica de negocio de la aplicación, incluyendo autenticación de usuarios, registro de transacciones financieras (ingresos y gastos), gestión de presupuestos y generación de reportes analíticos. Sirve como backend para las aplicaciones móvil y web, proporcionando una interfaz estandarizada para la manipulación y consulta de datos a través del protocolo HTTPS.

2. Listado de Endpoints

A continuación, se detallan los endpoints implementados, agrupados por módulo.

2.1. Módulo de Autenticación (/auth)

Endpoint: POST /auth/login

Descripción: Permite a un usuario autenticarse con su correo y contraseña. Devuelve un token JWT.

Códigos de respuesta:

- 200 OK: Autenticación exitosa.
- 401 Unauthorized: Credenciales incorrectas.

Ejemplo de solicitud:

```
{  
  "email": "usuario@mail.com",  
  "password": "123456"  
}
```

Ejemplo de respuesta:

```
{  
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."  
}
```

2.2. Módulo de Transacciones (/transactions)

GET /transactions: Devuelve la lista de transacciones. Parámetro opcional: month (filtra por mes).

POST /transactions: Crea una nueva transacción (ingreso o gasto).

Ejemplo de solicitud:

```
{  
  "type": "gasto",  
  "amount": 15.50,  
  "category": "Comida",  
  "description": "Almuerzo en restaurante",  
  "date": "2025-10-27T10:00:00Z"  
}
```

2.3. Módulo de Presupuestos (/budgets)

GET /budgets: Obtiene todos los presupuestos.

POST /budgets: Crea un nuevo presupuesto asociado a una categoría específica.

Ejemplo de solicitud:

```
{  
  "id": "bud-xyz-789",  
  "category": "Entretenimiento",  
  "limit": 200,  
  "spent": 50,  
  "startDate": "2025-10-01",  
  "endDate": "2025-10-31"  
}
```

2.4. Módulo de Reportes (/reports)

GET /reports/summary: Devuelve un resumen financiero mensual con ingresos, gastos y balance.

3. Evidencia de Pruebas

Se realizaron pruebas de los endpoints utilizando Swagger UI.

Prueba 1: Creación exitosa de una transacción (POST /transactions).

Transactions

POST /transactions Registrar una nueva transacción

Parameters

Cancel

Reset

No parameters

Request body required

application/json

Edit Value | Schema

```
{  "type": "gasto",  "amount": 25.75,  "category": "transporte",  "description": "Taxi al trabajo",  "date": "2025-10-16T08:30:00Z"}

```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/transactions' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "type": "gasto",
    "amount": 25.75,
    "category": "transporte",
    "description": "Taxi al trabajo",
    "date": "2025-10-16T08:30:00Z"
  }'

```

Request URL

http://localhost:3000/api/transactions

Server response

Code

Details

Undocumented

Failed to fetch.

Possible Reasons:

- CORS
- Network Failure
- URL scheme must be "http" or "https" for CORS request.

Responses

Prueba 2: Login con credenciales (POST /auth/login).

Authentication

POST

/auth/login

iniciar sesión de usuario

Parameters

No parameters

Request body required

application/json

Edit Value

Schema

```
{
  "email": "joeltapia10@gmail.com",
  "password": "contral23"
}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/auth/login' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "email": "joeltapia10@gmail.com",
    "password": "contral23"
  }'
```

Request URL

http://localhost:3000/api/auth/login

Server response

4. Repositorio

Repositorio del Código: <http://github.com/Jxel117/GastanGO>

5. Especificación OpenAPI 3.0

A continuación se presenta la definición YAML de la API:

Especificación OpenAPI 3.0 para la API de GastanGO

openapi: 3.0.0

info:

title: "GastanGO API"

description: "API RESTful para el Sistema de Gestión de Gastos GastanGO. Gestiona la lógica de negocio, usuarios, transacciones y notificaciones."

version: "1.0.0"

servers:

- url: "http://localhost:3000/api"

description: "Servidor de desarrollo local"

Definición de componentes reutilizables (esquemas de datos)

components:

schemas:

User:

type: object

properties:

id:

type: string

example: "user-123"

name:

type: string

example: "Joel Tapia"

email:

type: string

format: email

example: "joeltapia10@gmail.com"

Transaction:

type: object

properties:

id:

type: string

example: "txn-abc-456"

type:

type: string

enum: [ingreso, gasto]

example: "gasto"

amount:

type: number

format: float

example: 15.50

category:

type: string

example: "Comida"

description:

type: string

example: "Almuerzo en restaurante"

date:

type: string

format: date-time

example: "2025-10-27T10:00:00Z"

Budget:

type: object

properties:

id:

type: string

example: "bud-xyz-789"

category:

```
  type: string
  example: "Entretenimiento"
limit:
  type: number
  format: float
  example: 200.00
spent:
  type: number
  format: float
  example: 50.00
startDate:
  type: string
  format: date
  example: "2025-10-01"
endDate:
  type: string
  format: date
  example: "2025-10-31"
```

Definición de los Endpoints

paths:

--- Authentication Controller ---

/auth/login:

post:

summary: "Iniciar sesión de usuario"

tags: ["Authentication"]

requestBody:

required: true

content:

application/json:

schema:

type: object

properties:

email:

type: string

example: "joeltapia10@gmail.com"

password:

type: string

example: "contra123"

responses:

"200":

description: "Autenticación exitosa"

content:

application/json:

```
    schema:
      type: object
      properties:
        token:
          type: string
          example: "eyeyehakhdeyh376hneky3"
    "401":
      description: "Credenciales inválidas"
```

--- Transaction Controller ---

/transactions:

post:

summary: "Registrar una nueva transacción"

tags: ["Transactions"]

requestBody:

required: true

content:

application/json:

schema:

\$ref: "#/components/schemas/Transaction" # Reutilizamos el esquema

responses:

"201":

description: "Transacción creada exitosamente"

content:

application/json:

schema:

\$ref: "#/components/schemas/Transaction"

"400":

description: "Datos de entrada inválidos"

get:

summary: "Obtener lista de transacciones"

tags: ["Transactions"]

parameters:

- in: query

name: month

schema:

type: integer

description: "Filtrar transacciones por mes (e.g., 10 para Octubre)"

responses:

"200":

description: "Lista de transacciones"

content:

application/json:

schema:

```
    type: array
    items:
      $ref: "#/components/schemas/Transaction"
```

--- Budget Controller ---

/budgets:

post:

summary: "Crear un nuevo presupuesto"

tags: ["Budgets"]

requestBody:

required: true

content:

application/json:

schema:

\$ref: "#/components/schemas/Budget"

responses:

"201":

description: "Presupuesto creado"

content:

application/json:

schema:

\$ref: "#/components/schemas/Budget"

get:

summary: "Consultar todos los presupuestos"

tags: ["Budgets"]

responses:

"200":

description: "Lista de presupuestos"

content:

application/json:

schema:

type: array

items:

\$ref: "#/components/schemas/Budget"

--- Reporting Service ---

/reports/summary:

get:

summary: "Obtener resumen financiero del mes"

tags: ["Reports"]

responses:

"200":

description: "Resumen mensual"

content:

application/json:
schema:
 type: object
 properties:
 totalIncome:
 type: number
 example: 250.00
 totalExpenses:
 type: number
 example: 150.50
 balance:
 type: number
 example: 99.50