# **API REST - Gestión de Usuarios**

## **Descripción**

API RESTful desarrollada en Go para la gestión de usuarios con operaciones CRUD completas.

## **Tecnologías Utilizadas**

* **Lenguaje**: Go 1.24+
* **Framework**: Gin Web Framework
* **Base de Datos**: PostgreSQL
* **Arquitectura**: MVC + Repository Pattern

## **Instalación y Configuración**

### **Prerrequisitos**

* Go 1.24 o superior
* PostgreSQL 12+
* Git

### **Pasos de Instalación**

1. **Clonar el repositorio**

git clone https://github.com/tu-usuario/api-go-users.git

cd api-go-users

1. **Instalar dependencias**

go mod download

1. **Configurar variables de entorno** Crear archivo .env en la raíz del proyecto:

DB\_HOST=localhost

DB\_PORT=5432

DB\_NAME=mini\_api\_usuarios

DB\_USER=postgres

DB\_PASSWORD=\*\*\*\*

PORT=3001

1. **Crear la base de datos** Ejecutar el script database.sql (ver sección de Base de Datos)
2. **Ejecutar la aplicación**

go run cmd/server/main.go

La API estará disponible en: http://localhost:3001

## **Endpoints de la API**

### **Base URL**

http://localhost:3001/api

### **Información de la API**

* **GET** /api/ - Información general de la API

### **Gestión de Usuarios**

#### **1. Obtener todos los usuarios**

* **Endpoint**: GET /api/users
* **Descripción**: Obtiene la lista completa de usuarios
* **Respuesta exitosa** (200):

{

"success": true,

"message": "Usuarios obtenidos exitosamente",

"data": [

{

"id": 1,

"nombre": "Juan Pérez",

"email": "juan@example.com",

"telefono": "+57 300 1234567",

"fecha\_creacion": "2024-01-15T10:30:00Z",

"fecha\_actualizacion": "2024-01-15T10:30:00Z"

}

],

"count": 1

}

#### **2. Obtener usuario por ID**

* **Endpoint**: GET /api/users/ID
* **Descripción**: Obtiene un usuario específico por su ID
* **Parámetros**:
  + id (path parameter): ID del usuario
* **Respuesta exitosa** (200):

{

"success": true,

"message": "Usuario encontrado",

"data": {

"id": 1,

"nombre": "Juan Pérez",

"email": "juan@example.com",

"telefono": "+57 300 1234567",

"fecha\_creacion": "2024-01-15T10:30:00Z",

"fecha\_actualizacion": "2024-01-15T10:30:00Z"

}

}

* **Respuesta de error** (404):

{

"success": false,

"message": "Usuario no encontrado"

}

#### **3. Crear usuario**

* **Endpoint**: POST /api/users
* **Descripción**: Crea un nuevo usuario
* **Body** (JSON):

{

"nombre": "Ana García",

"email": "ana@example.com",

"telefono": "+57 300 7654321"

}

* **Campos requeridos**: nombre, email
* **Campos opcionales**: telefono
* **Respuesta exitosa** (201):

{

"success": true,

"message": "Usuario creado exitosamente",

"data": {

"id": 2,

"nombre": "Ana García",

"email": "ana@example.com",

"telefono": "+57 300 7654321",

"fecha\_creacion": "2024-01-15T11:00:00Z",

"fecha\_actualizacion": "2024-01-15T11:00:00Z"

}

}

#### **4. Actualizar usuario**

* **Endpoint**: PUT /api/users/:id
* **Descripción**: Actualiza un usuario existente
* **Parámetros**:
  + id (path parameter): ID del usuario
* **Body** (JSON):

{

"nombre": "Juan Carlos Pérez",

"email": "juan.carlos@example.com",

"telefono": "+57 300 1111111"

}

* **Respuesta exitosa** (200):

{

"success": true,

"message": "Usuario actualizado exitosamente",

"data": {

"id": 1,

"nombre": "Juan Carlos Pérez",

"email": "juan.carlos@example.com",

"telefono": "+57 300 1111111",

"fecha\_creacion": "2024-01-15T10:30:00Z",

"fecha\_actualizacion": "2024-01-15T11:30:00Z"

}

}

#### **5. Eliminar usuario**

* **Endpoint**: DELETE /api/users/:id
* **Descripción**: Elimina un usuario
* **Parámetros**:
  + id (path parameter): ID del usuario
* **Respuesta exitosa** (200):

{

"success": true,

"message": "Usuario eliminado exitosamente"

}

### **Health Check**

* **Endpoint**: GET /health
* **Descripción**: Verifica el estado de la API
* **Respuesta** (200):

{

"status": "OK",

"timestamp": {}

}

## **Códigos de Estado HTTP**

| **Código** | **Descripción** |
| --- | --- |
| 200 | OK - Operación exitosa |
| 201 | Created - Usuario creado exitosamente |
| 400 | Bad Request - Datos inválidos |
| 404 | Not Found - Usuario no encontrado |
| 409 | Conflict - Email ya registrado |
| 500 | Internal Server Error - Error del servidor |

## **Validaciones**

### **Campos Requeridos**

* nombre: No puede estar vacío
* email: Debe ser un email válido y único

### **Formato de Datos**

* email: Se convierte automáticamente a minúsculas
* telefono: Campo opcional, se limpia automáticamente

## **Ejemplos de Uso con curl**

### **Crear usuario**

curl -X POST http://localhost:3001/api/users \

-H "Content-Type: application/json" \

-d '{

"nombre": "María López",

"email": "maria@example.com",

"telefono": "+57 300 9876543"

}'

### **Obtener todos los usuarios**

curl -X GET http://localhost:3001/api/users/

### **Obtener usuario por ID**

curl -X GET http://localhost:3001/api/users/1

### **Actualizar usuario**

curl -X PUT http://localhost:3001/api/users/1 \

-H "Content-Type: application/json" \

-d '{

"nombre": "María Elena López",

"email": "maria.elena@example.com",

"telefono": "+57 300 5555555"

}'

### **Eliminar usuario**

curl -X DELETE http://localhost:3001/api/users/1

## **Dashboard Web**

La aplicación incluye un dashboard web accesible en http://localhost:3001 que permite:

* Visualizar todos los usuarios
* Crear nuevos usuarios
* Editar usuarios existentes
* Eliminar usuarios
* Interfaz responsive y moderna

## **Estructura del Proyecto**

api-go/

├── cmd/

│ └── server/

│ └── main.go

├── internal/

│ ├── config/

│ │ └── database.go

│ ├── controllers/

│ │ └── user\_controller.go

│ ├── models/

│ │ └── user\_model.go

│ └── routes/

│ └── user\_routes.go

├── pkg/

│ └── utils/

│ └── response.go

├── static/

│ └── index.html

├── .database.sql

├── .env

├── go.mod

├── go.sum

└── README.md

## **Características Adicionales**

* **CORS**: Configurado para permitir solicitudes desde cualquier origen
* **Logging**: Registro detallado de solicitudes y errores
* **Recovery**: Manejo automático de pánico en la aplicación
* **Archivos estáticos**: Servido del dashboard web
* **Pool de conexiones**: Configuración optimizada para PostgreSQL