🌐 Uso de Google Maps Distance Matrix

> ## via RestClient
>
> **Paso a Paso**

Este tutorial guía la construcción paso a paso de un microservicio básico en Spring Boot que recibe dos coordenadas geográficas y retorna la distancia en kilómetros entre ellas utilizando la API pública **Google Maps Distance Matrix**.

📦 1. Inicialización del proyecto

1.1. Spring Initializr

Acceder a https://start.spring.io/ y completar:

- **Project:** Maven
- **Language:** Java
- **Spring Boot:** 3.5.0
- **Group:** `utnfc.isi.back.spring`
- **Artifact:** `geoapi`
- **Name:** `geoapi`
- **Package name:** `utnfc.isi.back.spring.geoapi`
- **Packaging:** Jar
- **Java:** 21

1.2. Dependencias

Seleccionar las siguientes:

- Spring Web
- Spring Boot DevTools
- Lombok
- Validation
- Actuator

Descargar y descomprimir el proyecto generado.

> 🛠️ Luego de generar el proyecto, realizar los siguientes ajustes manuales:
>
> 1. Eliminar el archivo application.properties que se crea por defecto en src/main/resources.
> 2. Asegurarse de crear el archivo application.yml como se indica más adelante.
> 3. Agregar manualmente la dependencia de Swagger UI si no fue incluida automáticamente:

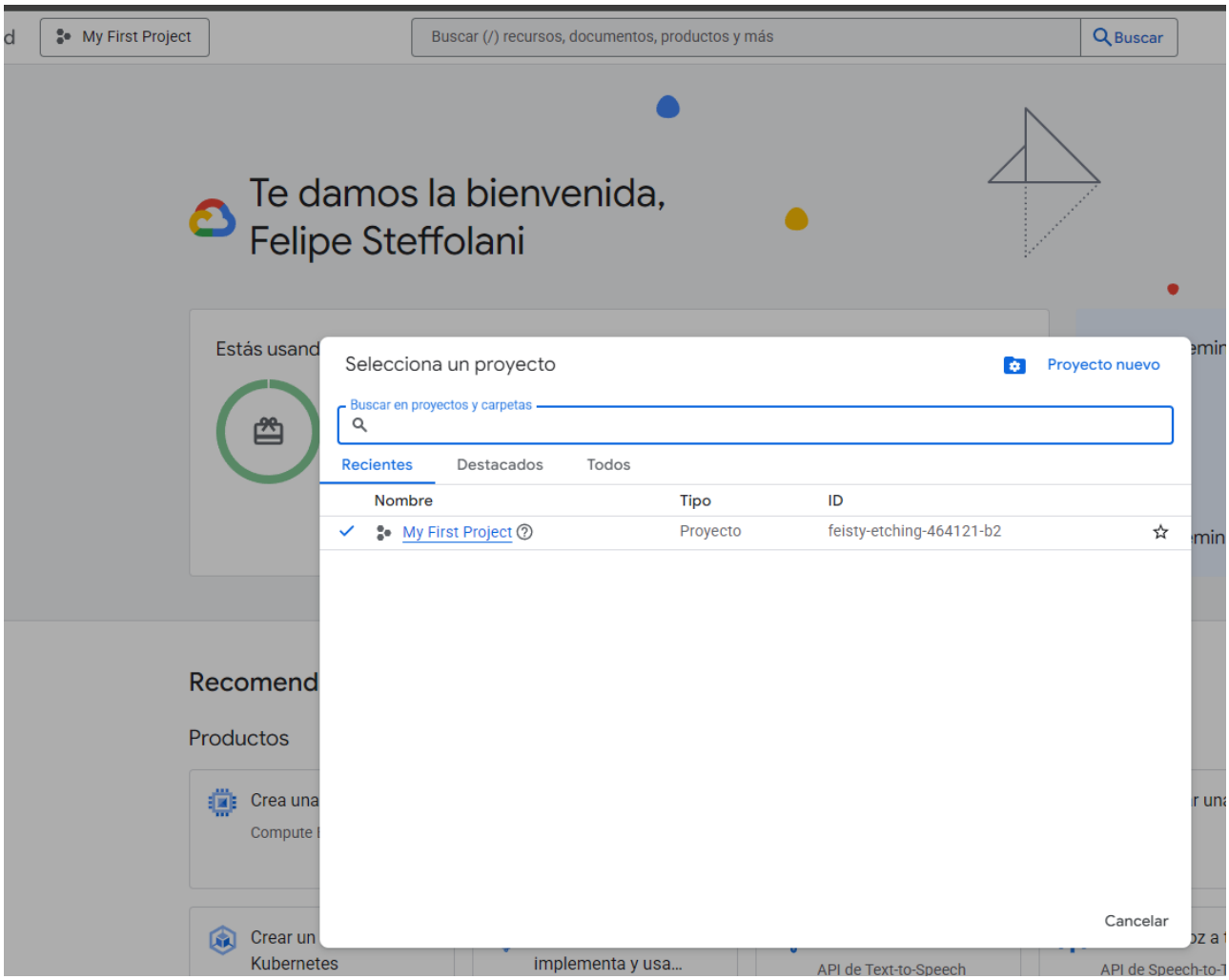
```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
```

```
    <version>2.5.0</version>
</dependency>
```

🛠️ 2. Requisitos de Google Cloud

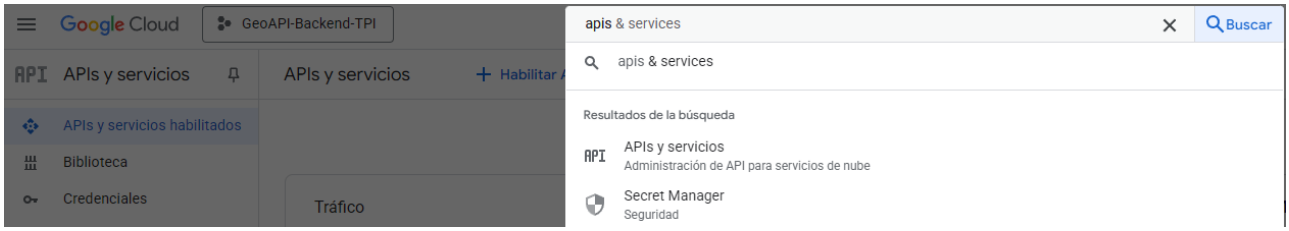
2.1 Crear cuenta y proyecto en Google Cloud Console

1. Ir a https://console.cloud.google.com/
2. Crear un nuevo proyecto (ej: `GeoAPI-Backend-TPI`)

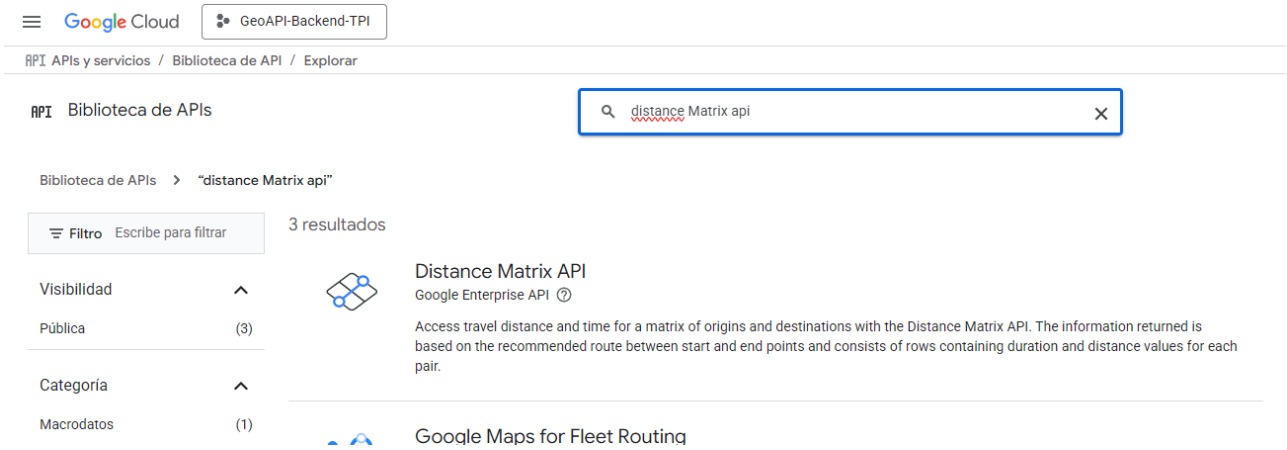


2.2 Habilitar Distance Matrix API

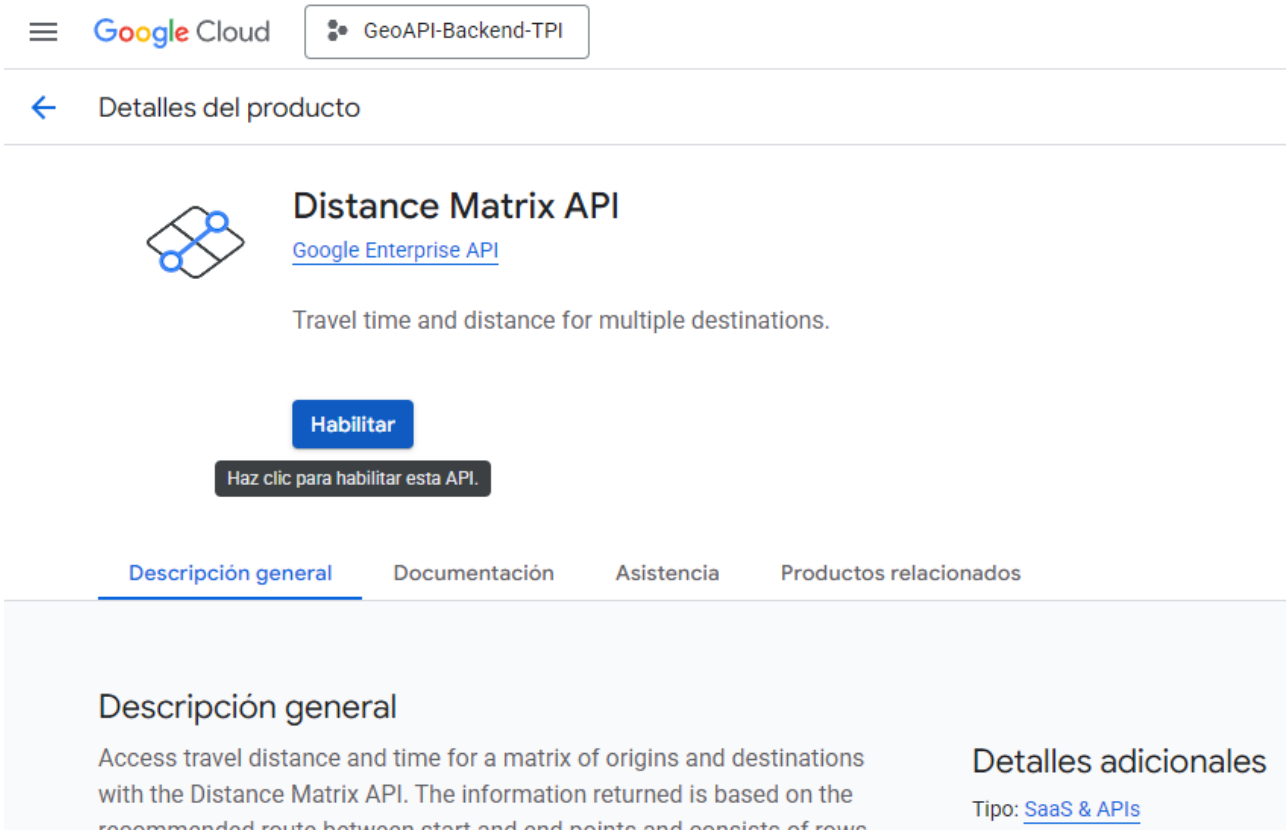
1. Ir a "APIs y Servicios > Biblioteca"



2. Buscar `Distance Matrix API`

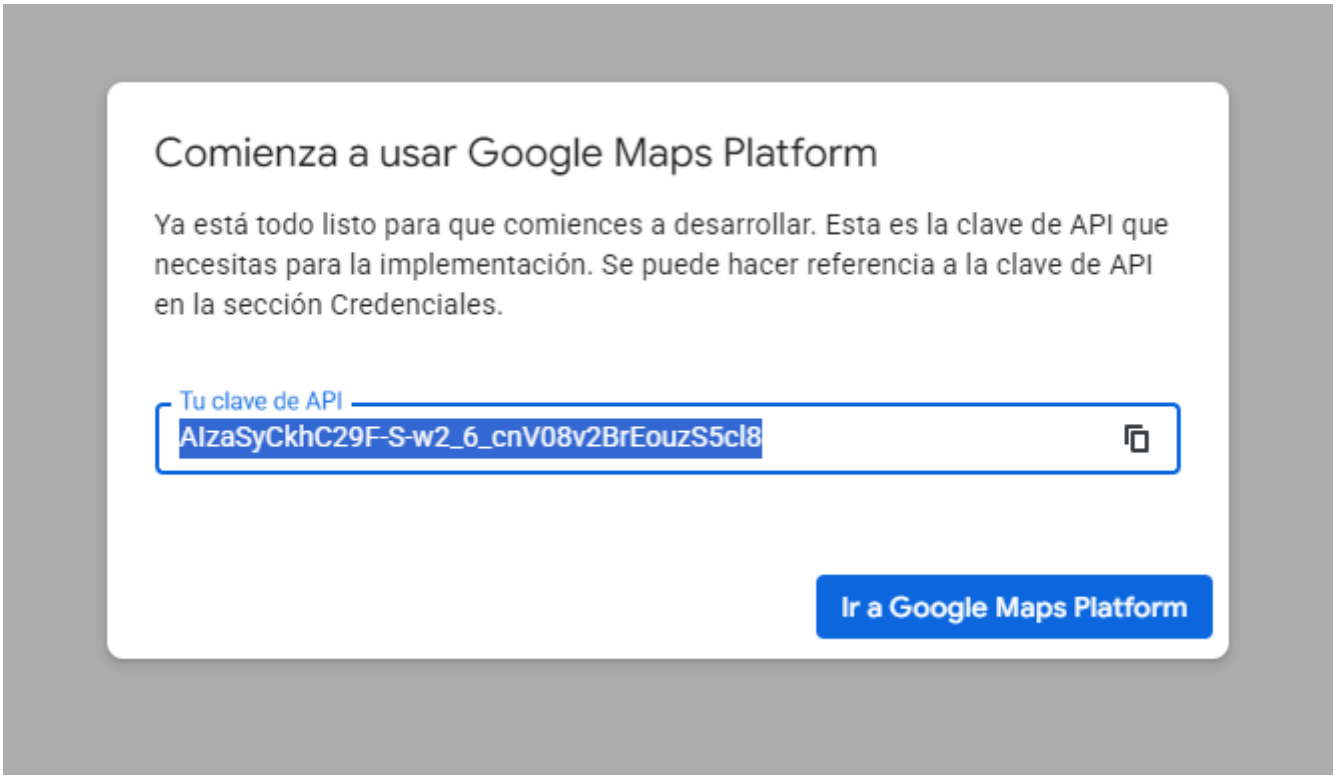


3. Hacer clic en **Habilitar**



2.3 Crear clave de API

Al habilitar la API te genera una Key automáticamente:



De todos modos puedes generar una cuando sea necesario:

1. Ir a "APIs y Servicios > Credenciales"
2. Crear una nueva **API Key**
3. Anotar y guardar (se usará en el `application.yml`)

⚙️ 3. Estructura del microservicio

3.1 Agregar configuración en `src/main/resources/application.yml`

```
server:
  port: 8080

springdoc:
  swagger-ui:
    path: /swagger-ui.html

google:
  maps:
    apikey: TU_API_KEY_AQUI
```

> 💡 Usar luego `@Value("${google.maps.apikey}")` para inyectar la clave.

3.2 Crear modelo DTO para respuesta simplificada

```
package utnfc.isi.back.spring.geoapi.model;
```

```
import lombok.Data;

@Data
public class DistanciaDTO {
    private String origen;
    private String destino;
    private double kilometros;
    private String duracionTexto;
}
```

3.3 Crear servicio `GeoService`

```
package utnfc.isi.back.spring.geoapi.service;

import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestClient;
import utnfc.isi.back.spring.geoapi.model.DistanciaDTO;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

@Service
@RequiredArgsConstructor
public class GeoService {

    @Value("${google.maps.apikey}")
    private String apiKey;

    private final RestClient.Builder builder;

    public DistanciaDTO calcularDistancia(String origen, String destino)
throws Exception {
        RestClient client =
builder.baseUrl("https://maps.googleapis.com/maps/api").build();

        String url = "/distancematrix/json?origins=" + origen +
                     "&destinations=" + destino +
                     "&units=metric&key=" + apiKey;

        ResponseEntity<String> response =
client.get().uri(url).retrieve().toEntity(String.class);

        ObjectMapper mapper = new ObjectMapper();
        JsonNode root = mapper.readTree(response.getBody());
        JsonNode leg = root.path("rows").get(0).path("elements").get(0);

        DistanciaDTO dto = new DistanciaDTO();
        dto.setOrigen(origen);
        dto.setDestino(destino);
```

```
            dto.setKilometros(leg.path("distance").path("value").asDouble() /
1000);
            dto.setDuracionTexto(leg.path("duration").path("text").asText());
            return dto;
        }
    }
```

3.4 Crear controlador REST

```
package utnfc.isi.back.spring.geoapi.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import utnfc.isi.back.spring.geoapi.model.DistanciaDTO;
import utnfc.isi.back.spring.geoapi.service.GeoService;

@RestController
@RequestMapping("/api/distancia")
@RequiredArgsConstructor
public class GeoController {

    private final GeoService geoService;

    @GetMapping
    public DistanciaDTO obtenerDistancia(
            @RequestParam String origen,
            @RequestParam String destino) throws Exception {
        return geoService.calcularDistancia(origen, destino);
    }
}
```

🔨 4. Compilación y ejecución del microservicio

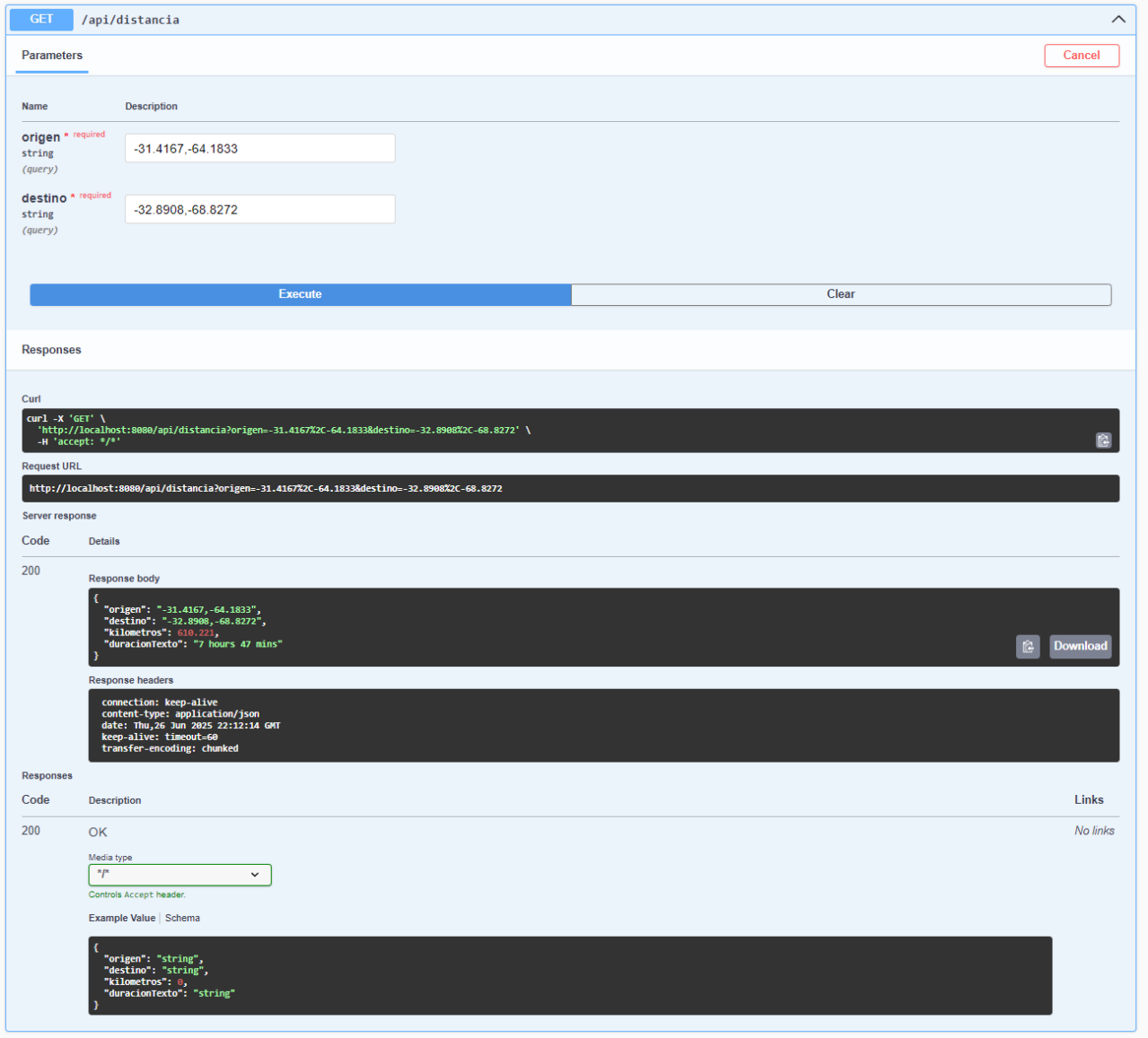
1. Abrir una terminal en el directorio raíz del proyecto.
2. Ejecutar el siguiente comando para compilar y correr el microservicio:

```
...$ mvn spring-boot:run
```

> Esto iniciará el servidor embebido en el puerto 8080 por defecto.

🔬 5. Pruebas desde Swagger UI

1. Acceder a: http://localhost:8080/swagger-ui.html
2. Probar endpoint `/api/distancia?origen=-31.4167,-64.1833&destino=-32.8908,-68.8272`



🧪 5. Ideas para ampliaciones

- Validar formato de coordenadas:
  Puede implementarse mediante anotaciones de Bean Validation, utilizando expresiones regulares con @Pattern para validar que las coordenadas ingresadas correspondan al formato correcto de latitud y longitud.
- Transformar una dirección en coordenada con Geocoding API:
  Utilizando la API de geocodificación de Google Maps se puede enviar una dirección completa (ej: "Av. Colón 750, Córdoba, Argentina") y obtener como respuesta un par lat/lng. El endpoint es /geocode/json?address=...&key=.... Esto permite luego usar esa coordenada como origen o destino dentro del microservicio.
- Registrar peticiones realizadas
- Agregar cálculo de costo por km como parte del TPI
- Y todo lo que se les ocurra sumar 😉

✍️ *Versión Septiembre 2025*