

🌐 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**)

The screenshot shows the Google Cloud Console interface. At the top, there's a navigation bar with a search bar and a 'Buscar' button. Below it, a welcome message says 'Te damos la bienvenida, Felipe Steffolani'. On the left, there's a sidebar with sections like 'Recomendaciones', 'Productos', and 'Biblioteca'. The main area has a 'My First Project' dashboard. A modal window titled 'Selección de proyecto' is overlaid on the page. It contains a search bar, a 'Recientes' tab, and a table with columns 'Nombre', 'Tipo', and 'ID'. One row is selected: 'My First Project' (Proyecto, feisty-etching-464121-b2). There are 'Cancelar' and 'Proyecto nuevo' buttons at the bottom right of the modal.

2.2 Habilitar Distance Matrix API

1. Ir a "APIs y Servicios > Biblioteca"

The screenshot shows the Google Cloud API library. The left sidebar has a navigation menu with 'Google Cloud' at the top, followed by 'APIs y servicios' (with a dropdown arrow), 'Biblioteca', and 'Credenciales'. The main right panel has a search bar with the placeholder 'Resultados de la búsqueda'. Below the search bar, there's a table with columns 'Nombre', 'Descripción', and 'Estado'. The first row in the table is 'APIs y servicios' with the subdescription 'Administración de API para servicios de nube'. There are also entries for 'Secret Manager' and 'Seguridad'.

2. Buscar Distance Matrix API

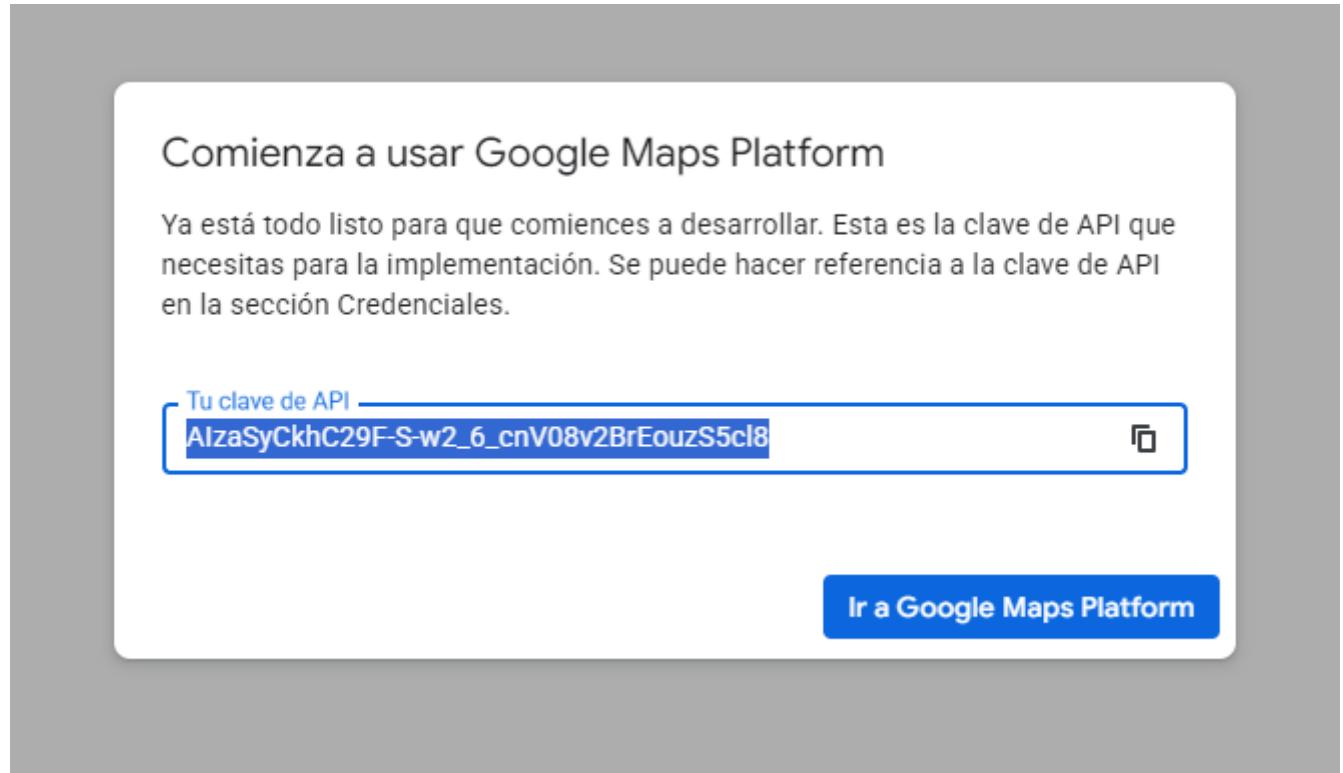
The screenshot shows the Google Cloud API Library interface. In the top navigation bar, there are links for 'API APIs y servicios / Biblioteca de API / Explorar'. A search bar contains the query 'distance Matrix api'. Below the search bar, the results are displayed under the heading 'Biblioteca de APIs > "distance Matrix api"'. It shows 3 resultados. On the left, there are filters for 'Visibilidad' (Publica, 3), 'Categoría' (Macrodatos, 1), and 'Escribe para filtrar'. The main result is 'Distance Matrix API' under 'Google Enterprise API'. Its description states: 'Access travel distance and time for a matrix of origins and destinations with the Distance Matrix API. The information returned is based on the recommended route between start and end points and consists of rows containing duration and distance values for each pair.' Another result, 'Google Maps for Fleet Routing', is partially visible below it.

3. Hacer clic en Habilitar

The screenshot shows the 'Distance Matrix API' product page. At the top, there is a back arrow labeled 'Detalles del producto' and a 'Habilitar' button. Below the button, a dark box contains the text 'Haz clic para habilitar esta API.'. At the bottom, there are tabs for 'Descripción general', 'Documentación', 'Asistencia', and 'Productos relacionados'. The 'Descripción general' tab is selected, showing the following text: 'Access travel distance and time for a matrix of origins and destinations with the Distance Matrix API. The information returned is based on the recommended route between start and end points and consists of rows containing duration and distance values for each pair.' To the right, under 'Detalles adicionales', it says 'Tipo: SaaS & APIs'.

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 injectar 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`

The screenshot shows a REST API documentation interface for a `GET /api/distancia` endpoint. The **Parameters** section includes two required query parameters: `origen` (value: -31.4167,-64.1833) and `destino` (value: -32.8908,-68.8272). Below the parameters are **Execute** and **Clear** buttons. The **Responses** section shows a successful response (status code 200) with a JSON body containing the distance matrix:

```
{
  "origen": "-31.4167,-64.1833",
  "destino": "-32.8908,-68.8272",
  "kilometros": 610.221,
  "duracionTexto": "7 hours 47 mins"
}
```

It also displays response headers including `connection: keep-alive`, `content-type: application/json`, `date: Thu, 26 Jun 2025 22:12:14 GMT`, `keep-alive: timeout=60`, and `transfer-encoding: chunked`.

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