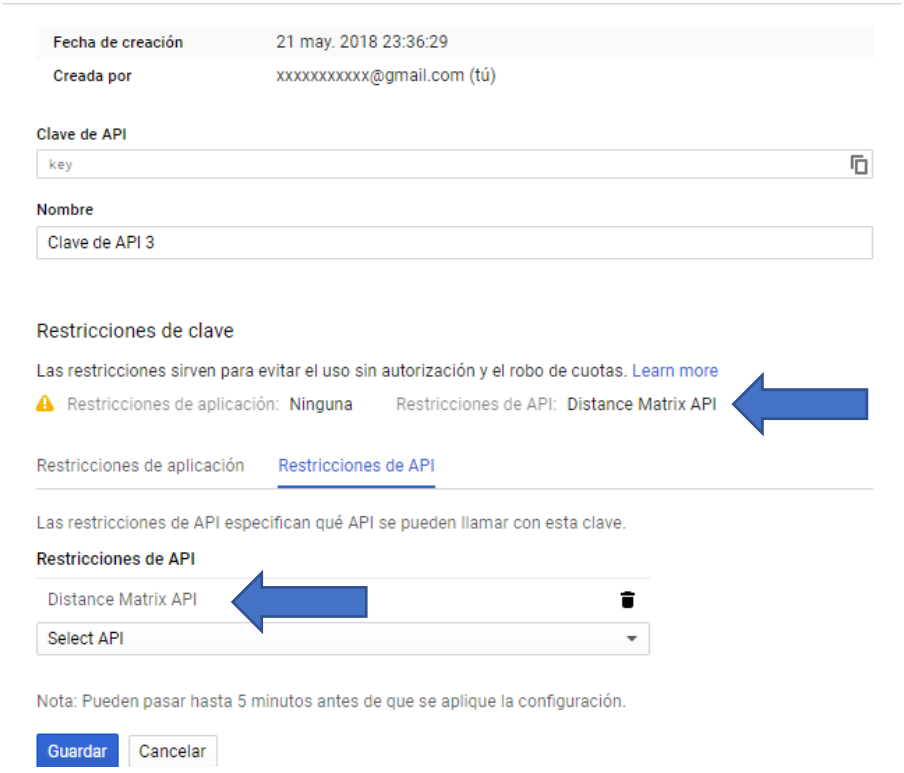
Consuming API in Spring

The purpose of this report is to explain how to consume an API in Spring Framework. For this project, we are going to consume two APIS: Distance Matrix API and Maps Embed API. The first one is used to calculate the distance between two given places, and the second one is used to show a map and the route between the same places.

Before to start to use them, we need to get the API keys. In our case, we are going to get two keys. Google let us administrate the keys and, if we want, we can use the same key for both APIs, but we will see later that one of the keys is public, so it's better to have two different keys to protect them.

Getting the keys is very simple. First, we need to create a project in https://appengine.google.com/. On the sidebar, we click on 'APIs y Servicios' and 'Biblioteca'. Then, we must search the APIs and click on 'Habilitar'. We must do this with both APIs and we can use them.
Later, to create two different keys, we click on 'APIs y Servicios' and 'Credenciales'. In this page, we can administrate the keys we own. We should have one by default. We can create another one clicking on 'Crear credencial'. Then, we can configure the restrictions of the key and use them in only one API.



As we can see in this image, this key is restricted to be used only in Distance Matrix API.

Now we can start consuming it in our project. Let's start with Distance Matrix API.

Distance Matrix API is an API that can give us information about the distance and estimated time between two places. When we call the API, it returns us the data in JSON format. Then, with a library called GSON, we convert that JSON into Java object which we can use to our project.

All this code is written in a class called GoogleMaps.java and it will contain two static method that calculate all the information we want. The first method is:

```
private static String getData(final String origin, final String destination) {
	final String key = "";
	final String uri =
"https://maps.googleapis.com/maps/api/distancematrix/json?units=metric&origins=" +
origin + "&destinations=" + destination + "&key=" + key;
	final RestTemplate restTemplate = new RestTemplate();
	final String result = restTemplate.getForObject(uri, String.class);
	return result;
}
```

First, what we do is to create the query we are going to use. The URL of the API is:

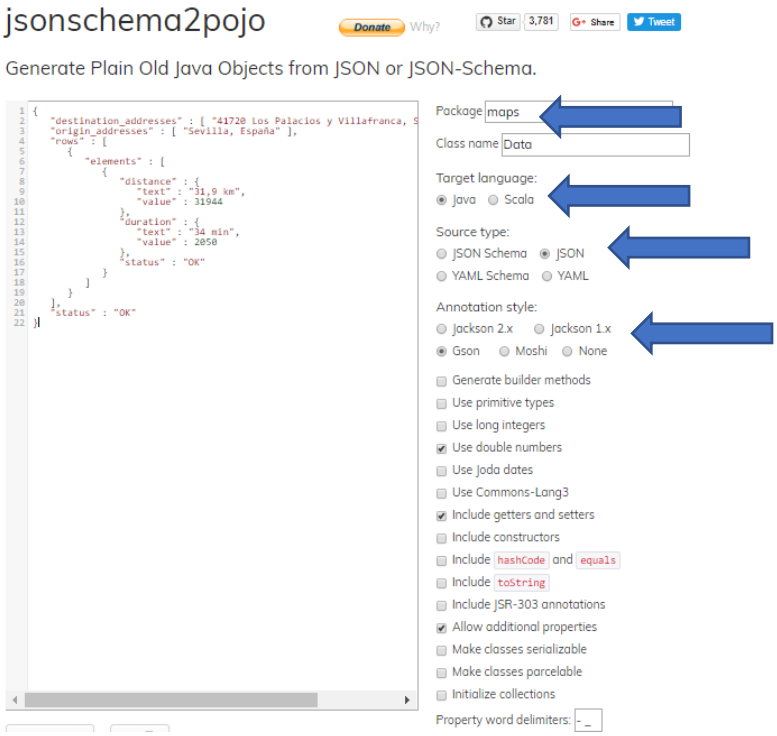
https://maps.googleapis.com/maps/api/distancematrix/.

Then, we must write the type of the format of the text we want. In our case, json:

And now we must give the parameters of our search. We are going to give units=metric, to obtain the result in meters, origin and destination, which are required and are given when the method is called, and the key, which we must have.

When we have the URL complete, now we create a RestTemplate object. This object is on Spring Framework and can be used to call an API Rest. To use it, we only must instance the object and call the method 'getForObject()'. The parameters are the URL and the type of the return value. In our case, the last parameter is the String class.

Now, we must convert the JSON into Java Object. To do this, we will use GSON. First, we need to create the Java classes that represents the JSON. To do this, we will use http://www.jsonschema2pojo.org/.



We must check Java language, JSON source and Gson annotation style. We press download and it will create the classes. Finally, we must copy them into our project.

Now, we must install GSON library. To do this, we must copy these lines into pom.xml:

```
<dependency>
	<groupId>com.google.code.gson</groupId>
	<artifactId>gson</artifactId>
	<version>2.8.4</version>
</dependency>
```

Automatically, Maven should start to download the library. If it doesn't, we click on Update Maven Project and it should download it.

Now we can get the data we want from the JSON String. To do this, we must create GSON object and the main class in the domain we download before:

```
final String json = GoogleMaps.getData(origin, destination);
final Data data = new Gson().fromJson(json, Data.class);
```

Now we can get the data we need to. In our case, the distance and the value:

```
final String json = GoogleMaps.getData(origin, destination);
		final Data data = new Gson().fromJson(json, Data.class);
		final List<Integer> res = new ArrayList<>();
		res.add(0,
data.getRows().get(0).getElements().get(0).getDistance().getValue());
		res.add(1,
data.getRows().get(0).getElements().get(0).getDuration().getValue());
		return res;
```

In our case, we will return them in a list with two integers. The first one is the distance and the second one is the duration.

We must consider the problem that the request has no result or any problem on it. If there is any problem, GSON throws an exception. So, we are going to use a try/catch sentence to solve this:

```
try {
	final String json = GoogleMaps.getData(origin, destination);
	final Data data = new Gson().fromJson(json, Data.class);
	final List<Integer> res = new ArrayList<>();
	res.add(0,
data.getRows().get(0).getElements().get(0).getDistance().getValue());
	res.add(1,
data.getRows().get(0).getElements().get(0).getDuration().getValue());
	return res;
} catch (final Throwable oops) {
	return null;
}
```

The method is complete now. But, in our case, the key is limited in use. This means that when we realize the performance tests, we will reach the limit very quickly. So, now we are going to add a Boolean that will tell us if use or not the API.

```
public static List<Integer> getDistanceAndDuration(final String origin, final String
destination, final boolean useApi) {
	if (useApi)
		try {
			final String json = GoogleMaps.getData(origin, destination);
			final Data data = new Gson().fromJson(json, Data.class);
			final List<Integer> res = new ArrayList<>();
			res.add(0,
data.getRows().get(0).getElements().get(0).getDistance().getValue());
			res.add(1,
data.getRows().get(0).getElements().get(0).getDuration().getValue());
			return res;
		} catch (final Throwable oops) {
			return null;
		}
	else {
		final List<Integer> res = new ArrayList<>();
		res.add(0, 1);
		res.add(1, 1);
		return res;
	}
}
```

If the Boolean is false, the distance and the estimated time will be always 1. Of course, these data are wrong, but they will be only used in performance tests.

The second API is easier to use.

```
<jstl:if test="${useApi}">
<iframe
  width="700"
  height="500"
  frameborder="0" style="border:0"
  src="https://www.google.com/maps/embed/v1/directions?key=&origin=<jstl:out
value="${request.origin}"/>&destination=<jstl:out value="${request.destination}"/>
&mode=driving" allowfullscreen>
</iframe>
</jstl:if>
```

We only must to add this iframe code with the source of the API. The source is https://www.google.com/maps/embed/v1/directions and the parameters are the same we use in the other API: origin, destination and key. In this one, we will add another one to tell that the route must be travelled by car, writing 'mode=driving'.

The iframe let us change other options, like the width and the height of the map, and the border.

This code is written inside a <jstl:if> statement to avoid be consumed on the performance tests.

Starting June 11, Google Maps will require to have a Google Cloud Billing Account. We have been warned against the problems that may arise if we give our credit card information, so we do not have a Google Cloud Billing Account. This means that this A+ will no longer work after June 11. In order to test the page after June 11, it is recommended to log in as admin and in the configuration view, disable the use of the API. By doing so, the system will provide false information when testing the calculation of price, distance and time, but all the request

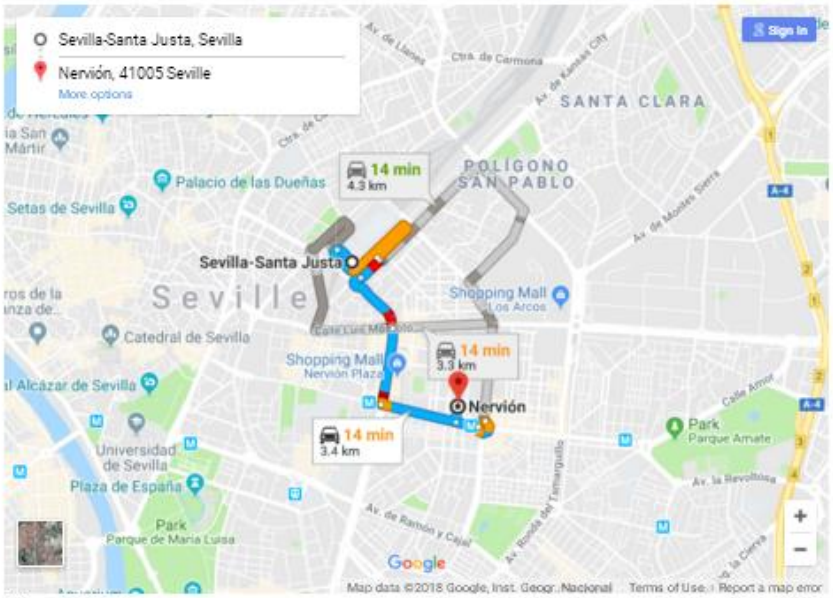
related functions will be usable. In case the page is tested after June 11, we will provide a screenshot showing the use of our API when making a request:

Request

Origin: Santa Justa, Sevilla, España

Destination: Nervión, Sevilla, España

Moment: 2018/11/30 00:00



Price: 12.19 €

Estimated time: 0h 13min (Estimated time can change according to the traffic and others issues)

Do you want to send the request?