

## **Manual técnico**

## Manual Técnico de AvantiParking

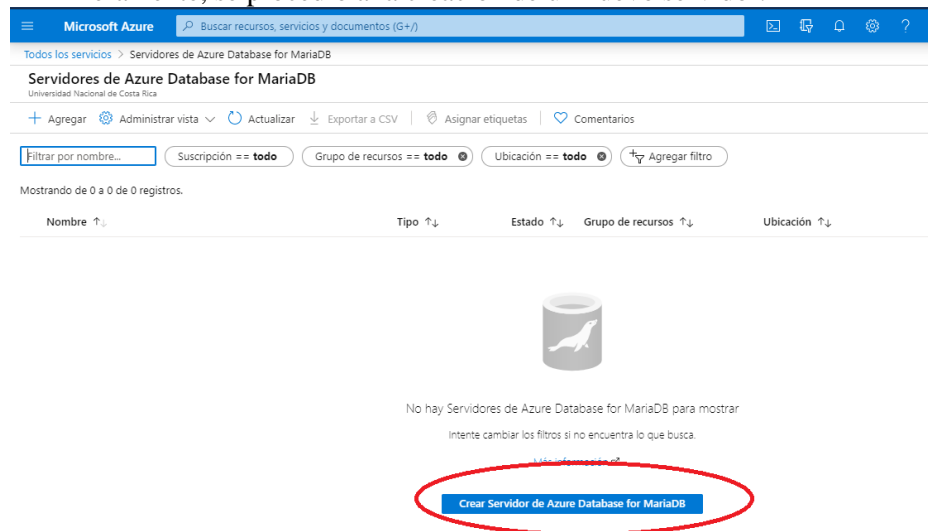
### Tabla de contenido

<i>Base de datos:</i> .....	3
<i>Spring Boot</i> .....	6
<i>Angular</i> .....	8
<i>Deploy de Spring Boot en Azure</i> .....	9
<i>Angular deploy</i> .....	11
<i>Referencias:</i> .....	13

## Base de datos:

Para la creación y montaje de la base de datos en un servidor se implementaron los servidores de Azure Database para MariaDB.

1-Primeramente, se procedió a la creación de un nuevo servidor.



## 2-Rellenamos las credenciales, de la siguiente manera:

The screenshot shows the 'Crear servidor MariaDB' form. It has two main sections: 'Detalles del proyecto' and 'Detalles del servidor'.  
**Detalles del proyecto:**  
- 'Suscripción': Azure subscription 1  
- 'Grupo de recursos': (Nuevo) avantiparking  
**Detalles del servidor:**  
- 'Nombre del servidor': avantiparking  
- 'Origen de datos': Ninguno (selected), Copia de seguridad (available)  
- 'Ubicación': (US) Este de EE. UU.  
- 'Versión': 10.3

Nombre del servidor \* ⓘ  ✓

Origen de datos \* ⓘ Ninguno Copia de seguridad

Ubicación \* ⓘ  ✓

Versión \* ⓘ  ✓

Proceso y almacenamiento ⓘ **Uso general**  
2 núcleos virtuales; 20 GB de almacenamiento  
[Configurar servidor](#)

Cuenta de administrador

Nombre de usuario de administrador \* ⓘ  ✓

Contraseña \* ⓘ  ✓

Confirmar contraseña \*  ✓

[Revisar y crear](#) [Siguiente: Etiquetas >](#)

20 gigas de almacenamiento, con 2 nucleos, el nombre del usuario administrador será AvantiDBAdmin y la contraseña Avantiparking2020

3-En la página siguiente del formulario dejamos las opciones por defecto:

Básico **Etiquetas** Revisar y crear

Las etiquetas son pares nombre-valor que permiten categorizar y consultar una facturación consolidada mediante la aplicación de la misma etiqueta en varios recursos y grupos de recursos. [Más información](#) ⓘ

Tenga en cuenta que si crea etiquetas y, después, cambia la configuración de los recursos en otras pestañas, las etiquetas se actualizan automáticamente.

Nombre ⓘ	Valor ⓘ	Recurso
<input type="text"/>	:	Server

[Revisar y crear](#) [< Anterior](#) [Siguiente: Revisar y crear >](#)

4-Por último, se mostrará un resumen de como nos quedaría la BD si todo esta correcto procedemos a realizar su creación.

---

**Crear servidor MariaDB**  
Microsoft

de Microsoft

[Términos de uso](#) | [Directiva de privacidad](#)

132.65 USD  
[Ver detalles de precio](#)

---

**Términos**

Al hacer clic en "Crear", (a) acepto los términos legales y las declaraciones de privacidad asociados a cada oferta de Marketplace enumerada previamente; (b) autorizo a Microsoft a facturar, de acuerdo con mi método de pago actual, las cuotas relativas a las ofertas con la misma frecuencia de facturación que mi suscripción de Azure; y (c) autorizo a Microsoft a compartir mi información de contacto y los datos de transacción y uso con los proveedores de dichas ofertas para fines de soporte técnico, facturación y otras actividades transaccionales. Microsoft no proporciona derechos sobre ofertas de terceros. Para obtener información adicional, consulte los [Términos de Azure Marketplace](#). [\[?\]](#)

**Básico**

Suscripción	Azure subscription 1
Grupo de recursos	avantiparking
Nombre del servidor	avantiparking
Origen de datos	None
Nombre de inicio de sesión del administrador del servidor	AvantiDBAdmin
Ubicación	Este de EE. UU.
Versión	10.3
Proceso y almacenamiento	GeneralPurpose, Gen5, 2 núcleos virtuales, 20 GB de almacenamiento
Período de retención de copia de seguridad	7 día(s)
Redundancia de copia de seguridad	Redundancia: Localmente
Crecimiento automático de almacenamiento	Enabled

**Etiquetas**

---

**Crear**

< Anterior

[Descargar una plantilla para la automatización](#)

## Spring Boot

Base de datos:

Dentro del proyecto de Spring Boot, necesitamos irnos a [src/main/resources/application.yml](#)

`datasource:`

```
url: jdbc:mysql://avantiparking-  
db.mariadb.database.azure.com:3306/avantiparking?useUnicode=true&useJDBCCompliantTimezoneShift=tr  
ue&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC  
username: master@avantiparking-db  
password: Aekm2020
```

Como vemos en el ejemplo de arriba, la base de datos se compone de una url, username y un password, así que lo unico que debemos hacer es poner nuestras credenciales en ese apartado.

Email:

Dentro del proyecto de Spring Boot, necesitamos irnos a [src/main/resources/application.yml](#)

`mail:`

```
host: in-v3.mailjet.com  
port: 587  
username: a9183edea445a2346abdb068d8eed198  
password: d9f033b49ac3cfe3711a8edd1881ef3e  
properties:  
  mail:  
    smtp:  
      auth: true  
      connectiontimeout: 5000  
      timeout: 5000  
      writetimeout: 5000  
      starttls:  
        enable: true
```

Como vemos en el ejemplo de arriba, el apartado email, solo debemos de colocar nuestras credenciales, host, port, username y password, además de poder modificar otros aspectos un poco más opcionales. Además, en el apartado [src/main/java/com.avantiparking.controller/Email\\_Controller.java](#) en la línea 44, debemos de colocar nuestro dominio, de apartado Front End, ejemplo:

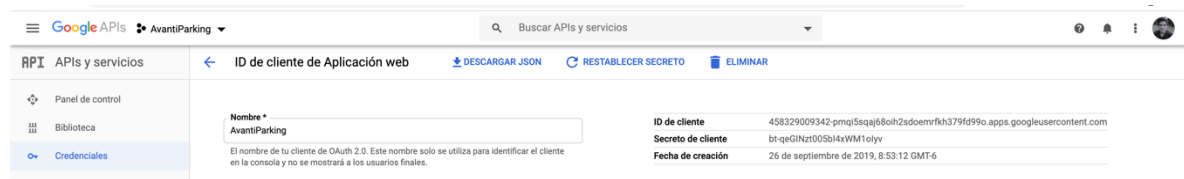
```
String emailHtml = i.TemplateEmailAvantiParking(email.getSubject(),email.getText(),email.getHtml(), "https://test-  
avantiparking.web.app/", true, "Go to AvantiParking");
```

Autenticación con Google:

Dentro del proyecto de Spring Boot, necesitamos irnos a [src/main/resources/application.yml](#)

```
security:
  oauth2:
    client:
      registration:
        google:
          clientId: 458329009342-pmqi5sqaj68oih2sdoemrfkh379fd99o.apps.googleusercontent.com
          clientSecret: bt-qeGINzt005bI4xWM1oIyv
          redirectUriTemplate: "{baseUrl}/oauth2/callback/{registrationId}"
          scope:
            - email
            - profile
```

Solo debemos colocar nuestras credenciales de la API de autenticación con google, en clientId y clientSecret



Las cuales obtenemos en la consola de google, después de crear un proyecto y habilitar la API de gmail

Además de esto, es importante que agreguemos el token y su expiración, como también agregar dominio/oauth2/redirect del Front End en la sección de redireccionamiento

```
app:
  auth:
    tokenSecret: 926D96C90030DD58429D2751AC1BDBBC
    tokenExpirationMsec: 864000000
  oauth2:
    authorizedRedirectUris:
      - http://localhost:4200/oauth2/redirect
      - https://test-avantiparking.web.app/c
```

## Angular

### API

Dentro del proyecto angular, necesitamos irnos a [src/app/services/api.modules.ts](#) debemos de agregar el dominio de nuestra aplicación Backend

```
export const domain: string = "https://avantiparking-1590019036260.azurewebsites.net";
```

Nota: No coloque "/" al final del dominio.

### Redireccionamiento

Dentro del proyecto angular, necesitamos irnos a [src/app/services/user.service.ts](#) debemos de agregar el dominio de nuestra aplicación FrontEnd para el redireccionamiento al autenticarnos con Google, de la siguiente manera:

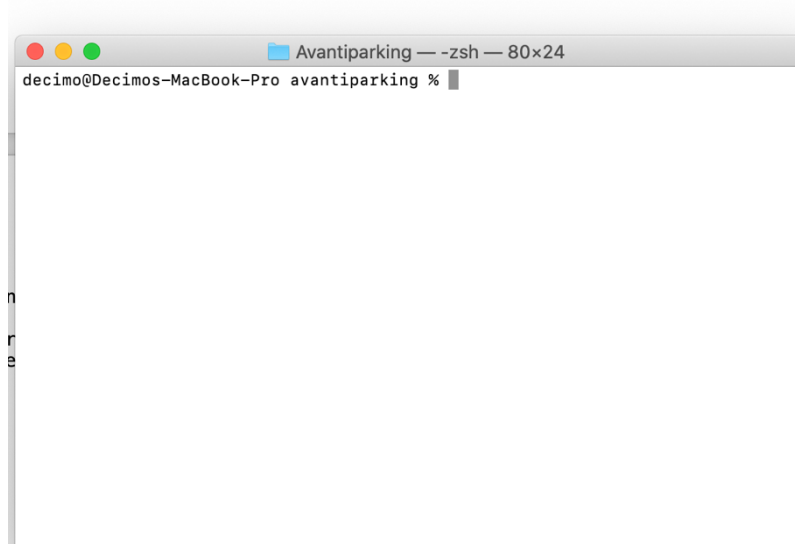
//Front End

```
public oauth2UrlRedirect = 'https://test-avantiparking.web.app/oauth2/redirect';
```



## Deploy de Spring Boot en Azure

1. Lo primero que debemos hacer, es irnos a la raíz del proyecto, pero con la terminal.



```
Avantiparking — zsh — 80x24
decimo@Decimos-MacBook-Pro avantiparking %
```

2. Previamente debemos de tener maven instalado en nuestra MAC, una vez, tengamos este requisito cumplido, solo debemos de colocar el siguiente comando y esperar a que termine:  
`mvn clean install`
3. Después de este comando, solo debemos de comenzar a configurar el proyecto, para su debido deploy, con el siguiente comando:

`mvn azure-webapp:config`

Seleccionamos 1. Application



```
Avantiparking — java -classpath /usr/local/Cellar/maven/3.6.3_1/libexec/boot/pl...
decimo@Decimos-MacBook-Pro avantiparking % mvn azure-webapp:config
[INFO] Scanning for projects...
[INFO] -----< com.avantiparking:AvantiParking >-----
[INFO] Building oauth2-demo 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- azure-webapp-maven-plugin:1.7.0:config (default-cli) @ AvantiParking ---
Please choose which part to config
1. Application
2. Runtime
3. DeploymentSlot
Enter index to use:
```

```
Avantiparking — java -classpath /usr/local/Cellar/maven/3.6.3_1/libexec/boot/pl...
[INFO]
[INFO] --- azure-webapp-maven-plugin:1.7.0:config (default-cli) @ AvantiParking
---
Please choose which part to config
1. Application
2. Runtime
3. DeploymentSlot
Enter index to use: 1
Define value for appName(Default: AvantiParking-1590019036260):
Define value for resourceGroup(Default: AvantiParking-1590019036260-rg):
Define value for region(Default: westeurope):
Define value for pricingTier(Default: P1v2):
1. b1
2. b2
3. b3
4. d1
5. f1
6. p1v2 [*]
7. p2v2
8. p3v2
9. s1
10. s2
11. s3
Enter index to use: 6
```

Que nos quede algo similar a la terminar de ejemplo.

Y se nos desplegara la siguiente información:

```
Please confirm webapp properties
AppName : AvantiParking-1590019036260
ResourceGroup : AvantiParking-1590019036260-rg
Region : westeurope
PricingTier : PremiumV2_P1v2
OS : Linux
RuntimeStack : JAVA 11-java11
Deploy to slot : false
Confirm (Y/N)? : Y
```

4. Al cual le daremos que Y y ya estaríamos listos para hacer el deploy, pero antes de hacerlo, debemos loguearnos con Azure, con el siguiente comando:

```
az login
```

5. Una vez logueados, solo debemos de colocar el siguiente comando para hacer el deploy:

```
mvn azure-webapp:deploy;
```

Y listo, ya tenemos en el aire nuestro proyecto Spring Boot

## Angular deploy

Lo primero que debemos hacer, es colocar el comando en nuestra terminal, en la raíz del proyecto Angular

`ng build`

Esto nos creará todo lo necesario para poder lanzar nuestro deploy, el cual se aloja en la carpeta `dist/avantiparking`.

Ahora creamos una carpeta, en la cual alojaremos nuestro deploy, y ingresaremos los siguientes comandos:

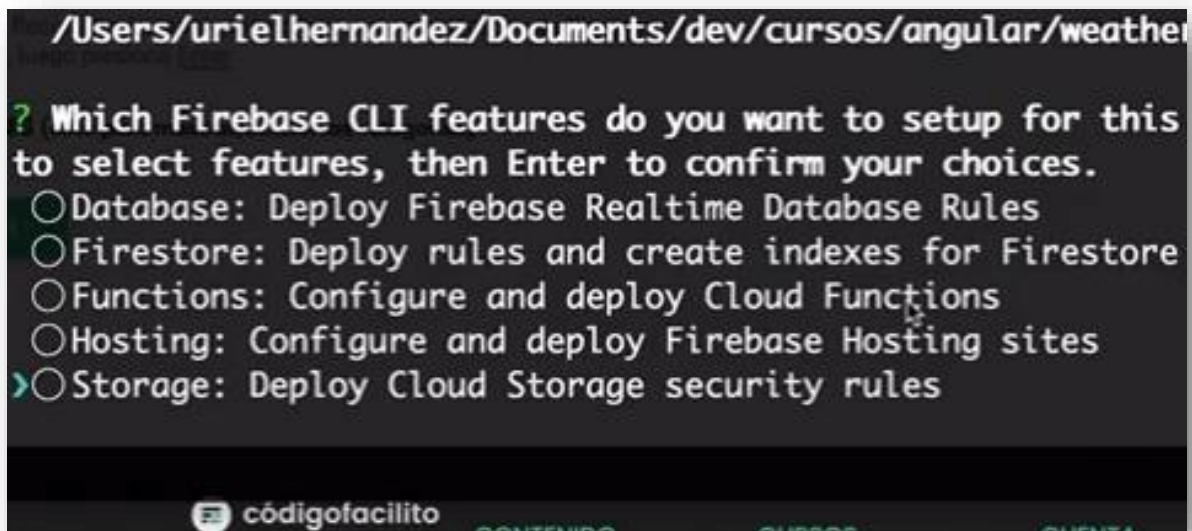
```
npm install -g firebase-tools
```

```
firebase login
```

```
firebase init
```

Este comando iniciará un asistente que te guiará por la configuración del proyecto, las opciones que debes seleccionar son:

1. Usando las flechas arriba/abajo selecciona qué característica de Firebase usarás, una vez que hayas seleccionado la del Hosting, presiona la barra espaciadora para activar el uso de esta característica, luego presiona Enter



Una vez seleccionado, debes seleccionar un nuevo hosting y lo creas en base al proyecto a necesitar

Un vez creado nuestro proyecto en Firebase, solo debemos mover los archivos de la carpeta `dist/avantiparking` a la carpeta `public` de nuestro deploy y escribir el siguiente comando:

```
firebase deploy
```

Referencias:

Deploy en Angular con FireBase:

<https://codigofacilito.com/articulos/deploy-angular-firebase>

Deploy en Spring Boot:

<https://www.youtube.com/watch?v=qFy9v454GyY&t=965s>