



## **VIRTUALIZACIÓN con Infraestructura en la Nube**

**Elaborado por:**

Edwin Paul Paute Chalco  
Juan Francisco Quizhpi Fajardo

**Docente:**

Cristian Fernando Timbi Sisalima

**Tema:**

Google Cloud

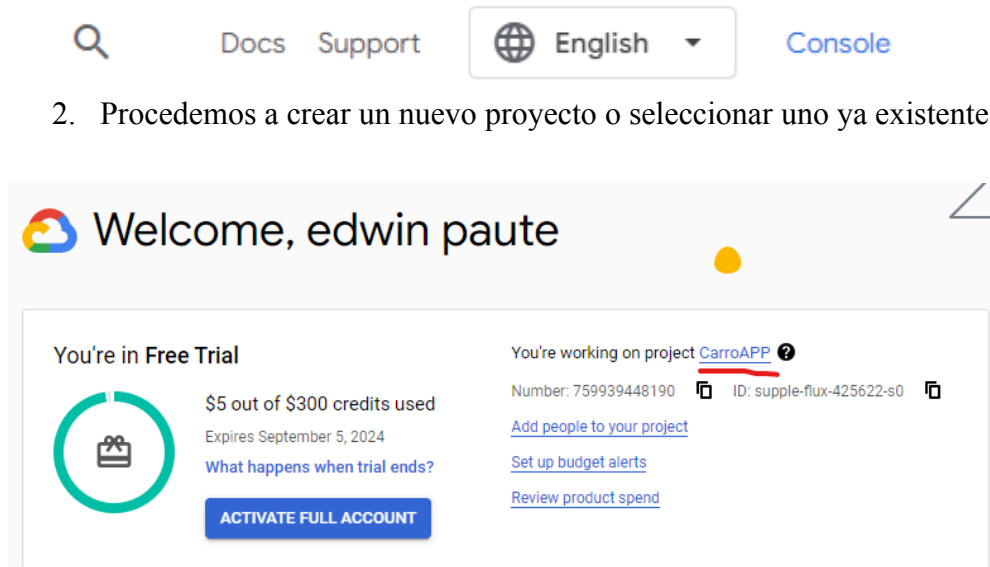
**Periodo Académico:**

## Recomendaciones iniciales

Para crear el proyecto al momento de configurar las máquinas debemos empezar por la VM de base de datos, VM del BackEnd y finalmente por la del FrontEnd

## Configuración de Google Platform

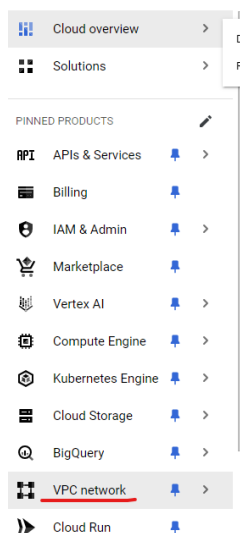
1. Accedemos a la consola de Google Cloud



2. Procedemos a crear un nuevo proyecto o seleccionar uno ya existente

## Configuración de VPC network

1. Navegamos a la sección de "VPC Network" y nos aseguramos que tenemos una red VPN configurada que pueda ser utilizada por las máquinas virtuales



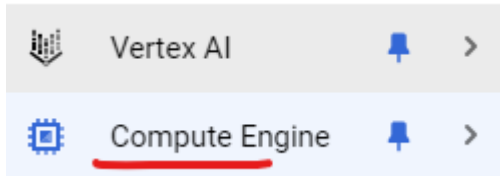
2. Creamos la VPC network

VPC networks [+ CREATE VPC NETWORK](#) [REFRESH](#)

NETWORKS IN CURRENT PROJECT SUBNETS IN CURRENT PROJECT

## Creación de las máquinas virtuales

1. Nos iremos al apartado de "Compute Engine"



2. Creamos las instancias para cada una de las partes de nuestro proyecto

### VM instances

[+ CREATE INSTANCE](#)

## Máquina virtual para FrontEnd o angular (Debian)

Name \*

web-servidor1

?

MANAGE TAGS AND LABELS

Region \*

us-east1 (South Carolina)

?

Region is permanent

Zone \*

us-east1-b

?

Zone is permanent

☒ E2

Low cost, day-to-day computing

0.25 - 32

1 - 128 GB

Based on availability

Boot disk ?

Name

web-servidor1

Type

New balanced persistent disk

Size

10 GB

License type ?

Free

Image

Debian GNU/Linux 11 (bullseye)

CHANGE

## Firewall ?

Add tags and firewall rules to allow specific network traffic from the Internet

- ☒ Allow HTTP traffic
- ☒ Allow HTTPS traffic

## Máquina virtual para BackEnd (Centos 7)

Name \*  
app-server1 ?

✓ MANAGE TAGS AND LABELS

Region \*  
us-east1 (South Carolina) ▼ ?  
Region is permanent

Zone \*  
us-east1-b ▼ ?  
Zone is permanent

☒ E2 Low cost, day-to-day computing 0.25 - 32 1 - 128 GB Based on availability

## Boot disk ?

Name	web-servidor1
Type	New balanced persistent disk
Size	10 GB
License type ?	Free
Image	Debian GNU/Linux 11 (bullseye)

CHANGE

## Boot disk ?

Name	app-server1
Type	New balanced persistent disk
Size	20 GB
License type ?	Free
Image	CentOS 7

## Firewall ?

Add tags and firewall rules to allow specific network traffic from the Internet

☒ Allow HTTP traffic

☒ Allow HTTPS traffic

## Máquina virtual para base de datos (Ubuntu)

Name \*  
ubuntu-focal-11 ?


✓ MANAGE TAGS AND LABELS

Region \*  
us-east1 (South Carolina) ▼ ?  
Region is permanent

Zone \*  
us-east1-b ▼ ?  
Zone is permanent

☒ E2 Low cost, day-to-day computing 0.25 - 32 1 - 128 GB Based on availability

## Boot disk ?

Name	ubuntu-focal-11
Type	New balanced persistent disk
Size	10 GB
License type ?	Free
Image	 Ubuntu 20.04 LTS

CHANGE

## Firewall ?

Add tags and firewall rules to allow specific network traffic from the Internet

☒ Allow HTTP traffic

☒ Allow HTTPS traffic

## Instancias creadas para el despliegue

VM instances

Filter Enter property name or value

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	<a href="#">app-server</a>	us-east1-b			10.142.0.6 ( <a href="#">nic0</a> )	34.74.78.120 ( <a href="#">nic0</a> )	SSH ▾ ⋮
<input type="checkbox"/>	✓	<a href="#">ubuntu-focal-1</a>	us-east1-b			10.142.0.5 ( <a href="#">nic0</a> )	34.73.48.35 ( <a href="#">nic0</a> )	SSH ▾ ⋮
<input type="checkbox"/>	✓	<a href="#">web-servidor</a>	us-east1-b			10.142.0.9 ( <a href="#">nic0</a> )	35.229.110.18 ( <a href="#">nic0</a> )	SSH ▾ ⋮

## Proceso para quitar las direcciones IP externas a las máquinas del backend y la base de datos.

Primero que le quitamos la dirección externa a la VM encargada de servir el backend

← Edit app-server instance

VM to Public IP: 2Gbps


### Network interfaces ?

Network interface is permanent

#### ^ Edit network interface

Network  
default

The instance is currently running X

 To use IPv6, you need an IPv6 subnet range. [LEARN MORE](#)

IP stack type  
☒ IPv4 (single-stack)  
☐ IPv4 and IPv6 (dual-stack)

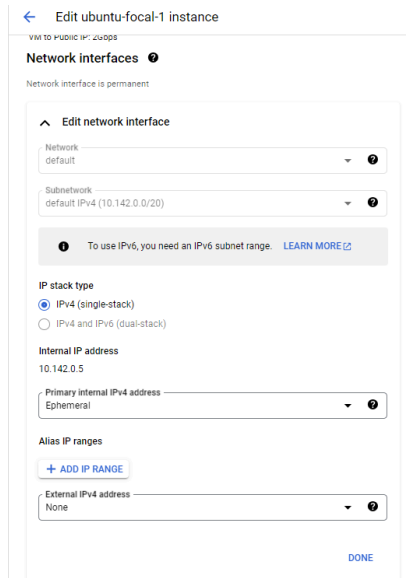
Internal IP address  
10.142.0.6  
Primary internal IPv4 address  
Ephemeral

Alias IP ranges  
[+ ADD IP RANGE](#)

External IPv4 address  
None

DONE

Luego repetimos el proceso para quitar la ip externa a la máquina encarga de la base de datos postgres



← Edit ubuntu-focal-1 instance

VM ID: PUBLIC IP: 200ps

### Network interfaces

Network interface is permanent

#### Edit network interface

Network: default

Subnetwork: default IPv4 (10.142.0.0/20)

To use IPv6, you need an IPv6 subnet range. [LEARN MORE](#)

**IP stack type**

☒ IPv4 (single-stack)

☐ IPv4 and IPv6 (dual-stack)

**Internal IP address**

10.142.0.5

Primary internal IPv4 address: Ephemeral

**Alias IP ranges**

[+ ADD IP RANGE](#)

External IPv4 address: None

DONE

Después de configurar las instancias tendremos que las direcciones IP externas para el backend y la base de datos se encontrarán deshabilitadas.

VM instances

Filter

Enter property name or value

<input type="checkbox"/>	Status	Name <span>↑</span>	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	<div>✓</div>	<a href="#">app-server</a>	us-east1-b			10.142.0.6 <a href="#">(nic0)</a>		SSH <div>▼</div> <div>⋮</div>
<input type="checkbox"/>	<div>✓</div>	<a href="#">ubuntu-focal-1</a>	us-east1-b	<div>💡 Save \$22 / mo</div>		10.142.0.5 <a href="#">(nic0)</a>		SSH <div>▼</div> <div>⋮</div>
<input type="checkbox"/>	<div>✓</div>	<a href="#">web-servidor</a>	us-east1-b			10.142.0.9 <a href="#">(nic0)</a>	35.229.110.18 <a href="#">(nic0)</a>	SSH <div>▼</div> <div>⋮</div>

## Configuración de la VM para el FrontEnd

En primer lugar vamos a ejecutar los siguientes comando para actualizar los paquetes de Nginx

```
sudo apt update
sudo apt install nginx -y
```

Luego ejecutaremos los comandos necesarios para la instalación de Git y Node.js

```
sudo apt install git -y
curl -sL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Ahora procedemos a ejecutar los comandos para la instalación de Angular para poder crear el proyecto en modo producción.

```
sudo npm install -g @angular/cli@17
```

Ahora procedemos a clonar nuestro proyecto de angular y entramos dentro de la carpeta donde se encuentra nuestro repositorio del frontend

```
git clone https://github.com/JuanQuizhpi/frontendcarroapp.git
cd frontendcarroapp
```

Dentro del repositorio procedemos a la construcción de la aplicación con los siguientes comandos

```
npm install
ng build --configuration=production
```

Ahora procederemos a la configuración de Nginx para servir la aplicación de Angular ejecutamos el siguiente comando

```
atlas_ep26@web-servidor:~$ sudo nano /etc/nginx/sites-available/default
```

Ahora procedemos a configurar nuestro servidor de la siguiente manera. Tomamos en cuenta también que el nombre del servidor será la ip de nuestro servidor web. Además la ruta que le pasamos es la de nuestra aplicación de Angular ejecutada en modo producción

```
server {
    listen 80;
#    listen [::]:80;

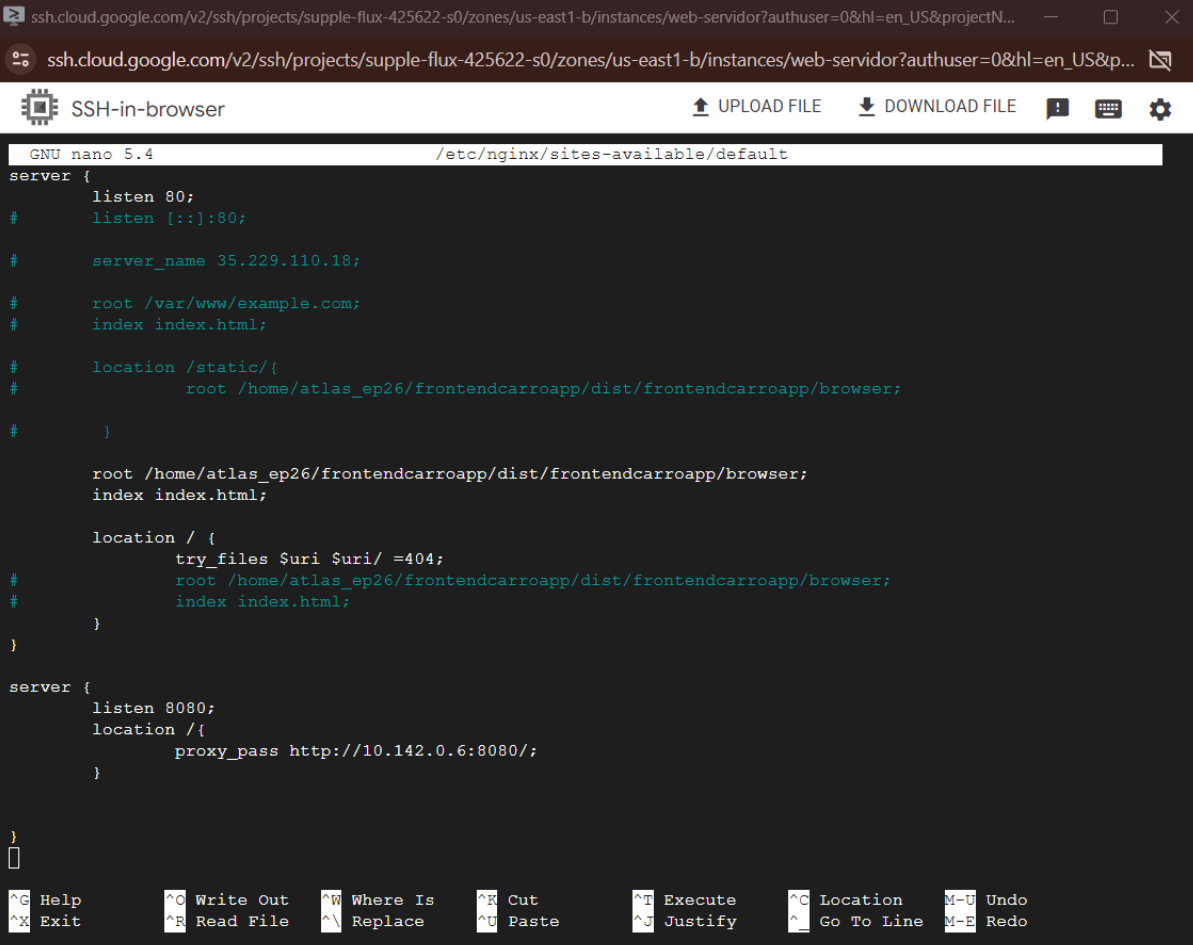
    server_name 35.229.110.18;

#    root /var/www/example.com;
#    index index.html;

    location /static/{
        root /home/atlas_ep26/frontendcarroapp/dist/frontendcarroapp/browser;
    }

    location / {
#        try_files $uri $uri/ =404;
        root /home/atlas_ep26/frontendcarroapp/dist/frontendcarroapp/browser;
        index index.html;
    }
}
```

En la configuración anterior trabajamos con las direcciones IP externas sin embargo para esta práctica se usará direcciones IP internas. De este modo tenemos que nuestra configuración queda de la siguiente manera.

A screenshot of a web browser displaying an SSH-in-browser interface. The browser's address bar shows a Google Cloud SSH session URL. Below the browser window, there's a header for 'SSH-in-browser' with 'UPLOAD FILE' and 'DOWNLOAD FILE' buttons. The main area shows a terminal window with the GNU nano 5.4 editor open to the file /etc/nginx/sites-available/default. The configuration contains two server blocks: the first listens on port 80 and serves static content from /home/atlas\_ep26/frontendcarroapp/dist/frontendcarroapp/browser; the second listens on port 8080 and acts as a proxy to http://10.142.0.6:8080/. A footer bar contains various keyboard shortcuts for nano editor functions like Help, Write Out, Where Is, Cut, Execute, Location, Undo, Exit, Read File, Replace, Paste, Justify, Go To Line, and Redo.

```
GNU nano 5.4 /etc/nginx/sites-available/default
server {
    listen 80;
    #
    #    listen [::]:80;
    #
    #    server_name 35.229.110.18;
    #
    #    root /var/www/example.com;
    #    index index.html;
    #
    #    location /static/{
    #        root /home/atlas_ep26/frontendcarroapp/dist/frontendcarroapp/browser;
    #
    #    }

    root /home/atlas_ep26/frontendcarroapp/dist/frontendcarroapp/browser;
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    #        root /home/atlas_ep26/frontendcarroapp/dist/frontendcarroapp/browser;
    #        index index.html;
    #    }
    }

server {
    listen 8080;
    location /{
        proxy_pass http://10.142.0.6:8080/;
    }
}
^G Help      ^C Write Out  ^W Where Is   ^R Cut        ^T Execute    ^G Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```

Lo que hacemos al configurar el proxy pass es reenviar las solicitudes HTTP recibidas a otro servidor, en este caso denominado servidor de destino o backend. En contexto Nginx actuará como un intermediario entre los clientes de frontend y el servidor de backend el cual se encargará del manejo de las solicitudes para consumir los servicios.

Ahora reiniciamos nuestro nginx y procedemos a ver el estatus de nuestro servidor de servidor con los siguientes comandos

```
sudo systemctl restart nginx
sudo systemctl status nginx
```



```

atlas_ep26@web-servidor:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2024-06-09 18:16:09 UTC; 7h ago
     Docs: man:nginx(8)
  Process: 14988 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 14989 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 14991 (nginx)
    Tasks: 3 (limit: 4691)
   Memory: 3.6M
      CPU: 99ms
   CGroup: /system.slice/nginx.service
           └─14991 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─14992 nginx: worker process
               └─14993 nginx: worker process

Jun 09 18:16:09 web-servidor systemd[1]: Starting A high performance web server and a reverse proxy server...
Jun 09 18:16:09 web-servidor systemd[1]: Started A high performance web server and a reverse proxy server.
lines 1-17/17 (END)

```

## Configuración de la VM del BackEnd

En primer lugar, tomando en cuenta que estamos trabajando en Centos los comandos van a variar para la configuración, seguiremos los siguientes pasos. Primero vemos que nuestro sistema está actualizado con el siguiente comando.

```
sudo yum update -y
```

Luego vamos a realizar la instalación de OpenJDK 11 para esto nos ayudaremos del siguiente comando

```
sudo yum install -y java-11-openjdk-devel
```

Ahora vamos a proceder a la instalación del WildFly para esto vamos a navegar hacia el directorio donde deseamos descargar el WildFly con el siguiente comando

```
cd /opt
```

Ahora vamos a proceder a descargar WildFly con ayuda del wget tomar en cuenta descargar este antes de usar su comando

```

sudo yum install wget
sudo wget https://github.com/wildfly/wildfly/releases/download/27.0.0.Final/wildfly-27.0.

```

Ahora procedemos a extraer el contenido del archivo descargado con el siguiente comando

```
sudo tar -xvzf wildfly-27.0.0.Final.tar.gz
```

Ahora cambiamos el nombre del directorio para facilitar el uso del mismo con el siguiente comando

```
sudo mv wildfly-27.0.0.Final /opt/wildfly
```

También cambiamos los permisos para permitir la ejecución del mismo con el siguiente comando

```
sudo chown -R atlas_ep26:atlas_ep26 /opt/wildfly
```

Ahora procedemos a descargar el driver de JDBC para PostgreSQL y lo ubicamos en el directorio de deployments con los siguientes comandos

```
cd /opt/wildfly/standalone/deployments  
sudo wget https://jdbc.postgresql.org/download/postgresql-42.3.1.jar
```

También para el correcto funcionamiento vamos a descargar las carpetas del siguiente repositorio y los vamos a ubicar en la carpeta modules del WildFly

```
sudo yum install -y git  
git clone https://github.com/JuanQuizhpi/libreriasJDBC.git  
cd libreriasJDBC  
sudo cp -r ./ /opt/wildfly/modules/
```

Ahora procedemos a configurar el standalone.xml con ayuda del siguiente comando

```
sudo nano /opt/wildfly/standalone/configuration/standalone.xml
```

Una vez dentro debemos tener configurado el standalone.xml de la siguiente manera. Recordar que cada vez que editemos ejecutaremos (Ctrl+X, Y, Enter) para salir. Además el tomar en cuenta reemplazar el localhost por el nombre de la máquina que se encarga de la base de datos. Además que carobase hace referencia a la base de datos creada en postgres. Así como el usuario y contraseña creada en postgres. También se puede considerar el uso de la ip interna en vez del el “ubuntu-focal-1” en este caso nos podríamos comunicar con ip interna en este caso con “10.142.0.5”

```
<datasource jndi-name="java:/jdbc/demojpaDS" pool-name="demojpaDS" enabled="true" use-java-context="true">
  <connection-url>jdbc:postgresql://ubuntu-focal-1:5432/carrobase</connection-url>
  <driver>postgresql</driver>
  <security>
    <user-name>carroadmin</user-name>
    <password>admin</password>
  </security>
</datasource>
```

```
<driver name="postgresql" module="org.postgresql">
  <xa-datasource-class>org.postgresql.xa.PGXADatasource</xa-datasource-class>
</driver>
```

```
<interfaces>
  <interface name="management">
    <inet-address value="{jboss.bind.address.management:0.0.0.0}"/>
  </interface>
  <interface name="public">
    <inet-address value="{jboss.bind.address:0.0.0.0}"/>
  </interface>
</interfaces>
```

Una vez completado esa parte abrimos los puertos necesarios para la administración de WildFly con los siguientes comandos

```
sudo firewall-cmd --zone=public --add-port=8080/tcp --permanent
sudo firewall-cmd --zone=public --add-port=9990/tcp --permanent
sudo firewall-cmd --reload
```

Ahora regresamos a la carpeta home en donde en un principio descargamos las librerías JDBC. Una vez ahí ejecutamos los siguientes comandos. Además en esta parte lo que vamos a hacer copiar el war en deployments del WildFly

```
git clone https://github.com/JuanQuizhpi/carrobe.git
cd carrobe
sudo cp target/carrobe.war /opt/wildfly/standalone/deployments/
```

Finalmente para poder iniciar el servidor de aplicaciones ejecutamos el siguiente comando

```
sudo sh /opt/wildfly/bin/standalone.sh -b 0.0.0.0
```

## Configuración del servidor de Base de datos

En primera instancia vamos a actualizar y instalar todo lo necesario para nuestra VM con los siguientes comandos

```
sudo apt update && sudo apt upgrade -y
```

Ahora procedemos a instalar postgres para este caso se instala la versión 12

```
sudo apt install -y postgresql postgresql-contrib
```

Ahora procedemos a iniciar el postgres con el siguiente comando

```
sudo systemctl start postgresql
```

Ahora verificamos el status de nuestro postgres con el siguiente comando

```
sudo systemctl status postgresql
```

Ahora ingresamos al archivo de configuración de postgres para permitir las conexiones desde otras máquinas. Usaremos el siguiente comando.

```
sudo nano /etc/postgresql/12/main/pg_hba.conf
```

Una vez dentro cambiamos la línea que controla el acceso a la base de datos para que permita conexiones de todas las IPs (0.0.0.0/0). Para la primera vez del **md5** en el usuario postgres debe ser **peer** o no tendremos acceso para crear la tabla.

```
local  all             postgres          md5
# TYPE  DATABASE  USER  ADDRESS  METHOD
# "local" is for Unix domain socket connections only
local  all             all               md5
# IPv4 local connections:
#host   all             all             127.0.0.1/32   md5
host   all             all             0.0.0.0/0     md5
# IPv6 local connections:
host   all             all             ::1/128        md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local  replication    all               md5
host   replication    all             127.0.0.1/32   md5
host   replication    all             ::1/128        md5
```

Ahora también vamos a entrar al siguiente archivo de configuraciones de postgres para permitir que todas las interfaces de red escuchen a PostgreSQL. Con el siguiente comando

```
sudo nano /etc/postgresql/12/main/postgresql.conf
```

Ahora buscaremos la siguiente línea y la configuramos de esta manera

```
listen_addresses = '*'           # what IP address(es) to listen on;
                                  # comma-separated list of addresses;
                                  # defaults to 'localhost'; use '*' for all
                                  # (change requires restart)
port = 5432                       # (change requires restart)
max_connections = 100             # (change requires restart)
#superuser_reserved_connections = 3 # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
# (change requires restart)
#unix_socket_group = ''           # (change requires restart)
```

Una vez terminado ese proceso ejecutamos el comando para reiniciar y aplicar las configuraciones

```
sudo systemctl restart postgresql
```

Ahora vamos a ingresar a la consola de postgres para crear el usuario con el siguiente comando.

```
sudo -u postgres psql
```

En esta parte ya estaremos en la consola y ejecutamos los siguientes comandos para crear la base de datos estos variarán en función del datasource ya configurado en el wildfly.

```
##Comandos en la consola de postgres

CREATE DATABASE carrobase;
CREATE USER carroadmin WITH PASSWORD 'admin';
GRANT ALL PRIVILEGES ON DATABASE carrobase TO carroadmin;
#Para salir de la consola de postgres
\q
```

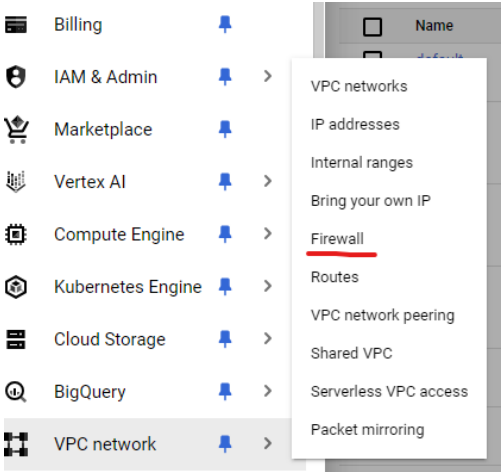
Ahora vamos a permitir las conexiones al puerto 5432 de PostgreSQL con el siguiente comando

```
sudo ufw allow 5432/tcp
```

Ahora nos vamos a conectar al postgres de nuevo para verificar que se creó la base con el siguiente comando

```
psql -U carroadmin carrobase -W
```

Creación de la regla de Firewall para el puerto 8080



Filter Enter property name or value										
<input type="checkbox"/>	Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network	↑	
<input type="checkbox"/>	<a href="#">default-allow-http</a>	Ingress	http-server	IP ranges:	tcp:8080	Allow	1000	<a href="#">default</a>		▼
<input type="checkbox"/>	<a href="#">default-allow-https</a>	Ingress	https-	IP ranges:	tcp:80	Allow	1000	<a href="#">default</a>		▼
<input type="checkbox"/>	<a href="#">default-allow-icmp</a>	Ingress	Apply to all	IP ranges:	icmp	Allow	65534	<a href="#">default</a>		▼
<input type="checkbox"/>	<a href="#">default-allow-internal</a>	Ingress	Apply to all	IP ranges:	tcp:0-65535 udp:0-65535 icmp	Allow	65534	<a href="#">default</a>		▼
<input type="checkbox"/>	<a href="#">default-allow-rdp</a>	Ingress	Apply to all	IP ranges:	tcp:3389	Allow	65534	<a href="#">default</a>		▼
<input type="checkbox"/>	<a href="#">default-allow-ssh</a>	Ingress	Apply to all	IP ranges:	tcp:22	Allow	65534	<a href="#">default</a>		▼

default-allow-http

Description

Logs

Turning on firewall logs can generate a large number of logs which can increase costs in Logging. [Learn more](#)

☐ On

☐ Off

Network

default

Priority \*1000

COMPARE?

Priority can be 0 - 65535

Direction

Ingress

Action on match

Allow

Targets  
Specified target tags ▼

Target tags \*  
http-server ✕

Source filter  
IPv4 ranges ▼ ?

Source IPv4 ranges \*  
0.0.0.0/0 ✕ ?

Second source filter  
None ▼ ?

Destination filter  
None ▼ ?

Protocols and ports ?

- ☐ Allow all
- ☒ Specified protocols and ports

☒ TCP

Ports  
8080

E.g. 20, 50-60

☐ UDP

Ports

E.g. all

☐ Other

Protocols

Separate multiple protocols by commas, e.g. ah, sctp

▼ [DISABLE RULE](#)

[SAVE](#) [CANCEL](#)

## Datos a tomar en cuenta

Para conectar el FrontEnd con el BackEnd en la ruta con la que consumimos los servicios cambiamos el **localhost** por la IP externa de nuestro **app-server**. En este caso se estará haciendo uso de la IP externa de nuestro servidor de aplicación.

```

@Injectable({
  providedIn: 'root'
})
export class CarroService {

  private apiUrl = 'http://34.74.78.120:8080/carrobe/rs/carros';

  constructor(private http: HttpClient) { }

  save(carro: Carro) {
    return this.http.post<any>("http://34.74.78.120:8080/carrobe/rs/carros", carro)
  }

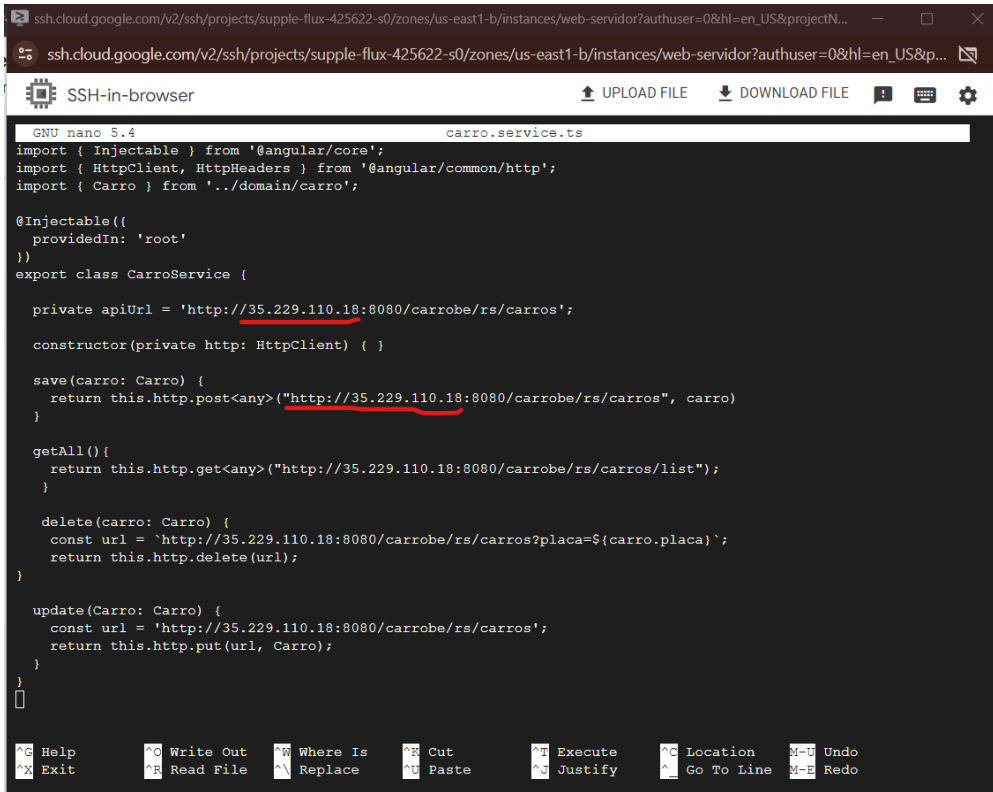
  getAll(){
    return this.http.get<any>("http://34.74.78.120:8080/carrobe/rs/carros/list");
  }

  delete(carro: Carro) {
    const url = `http://34.74.78.120:8080/carrobe/rs/carros?placa=${carro.placa}`;
    return this.http.delete(url);
  }

  update(Carro: Carro) {
    const url = 'http://34.74.78.120:8080/carrobe/rs/carros';
    return this.http.put(url, Carro);
  }
}

```

Ahora procedemos a configurar el frontend con su dirección IP externa propia. Ya que en la configuración de Nginx ya configuramos el proxy\_pass lo que hace es que cada vez que nuestro frontend reciba peticiones por el puerto 8080 este nos redirija a la dirección IP interna de nuestra VM del backend para consumir sus servicios. Tenemos entonces que nuestro archivo carro.services.ts queda de la siguiente manera.



```

GNU nano 5.4 carro.service.ts
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Carro } from '../domain/carro';

@Injectable({
  providedIn: 'root'
})
export class CarroService {

  private apiUrl = 'http://35.229.110.18:8080/carrobe/rs/carros';

  constructor(private http: HttpClient) { }

  save(carro: Carro) {
    return this.http.post<any>("http://35.229.110.18:8080/carrobe/rs/carros", carro)
  }

  getAll(){
    return this.http.get<any>("http://35.229.110.18:8080/carrobe/rs/carros/list");
  }

  delete(carro: Carro) {
    const url = `http://35.229.110.18:8080/carrobe/rs/carros?placa=${carro.placa}`;
    return this.http.delete(url);
  }

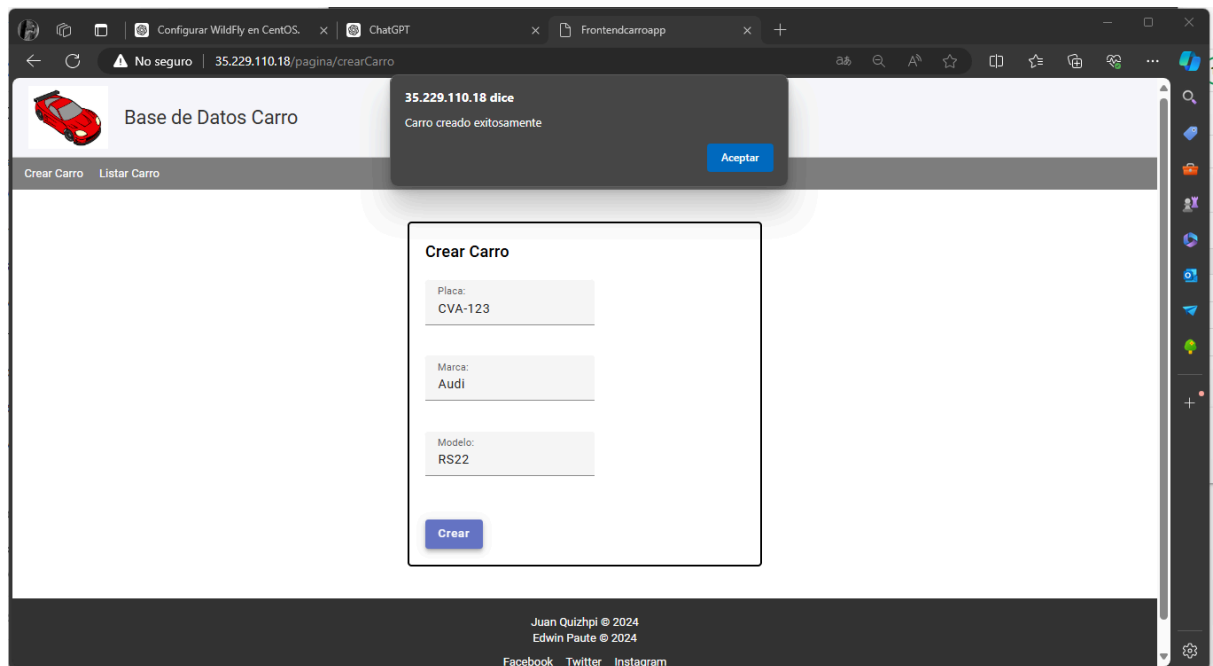
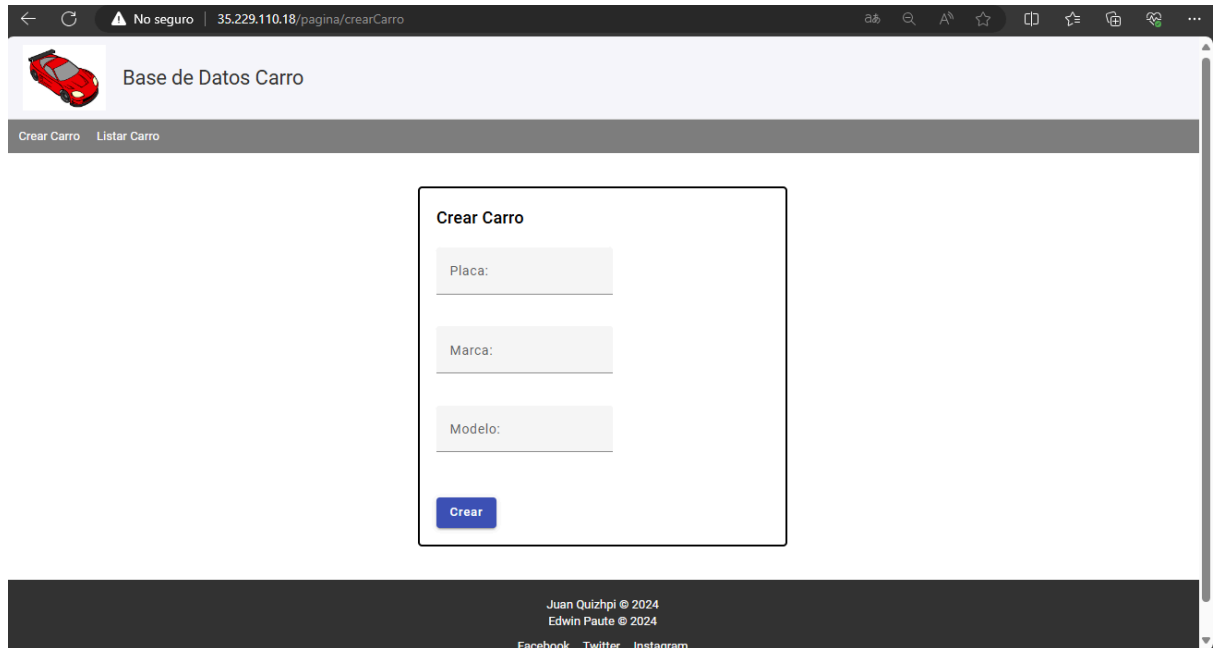
  update(Carro: Carro) {
    const url = 'http://35.229.110.18:8080/carrobe/rs/carros';
    return this.http.put(url, Carro);
  }
}

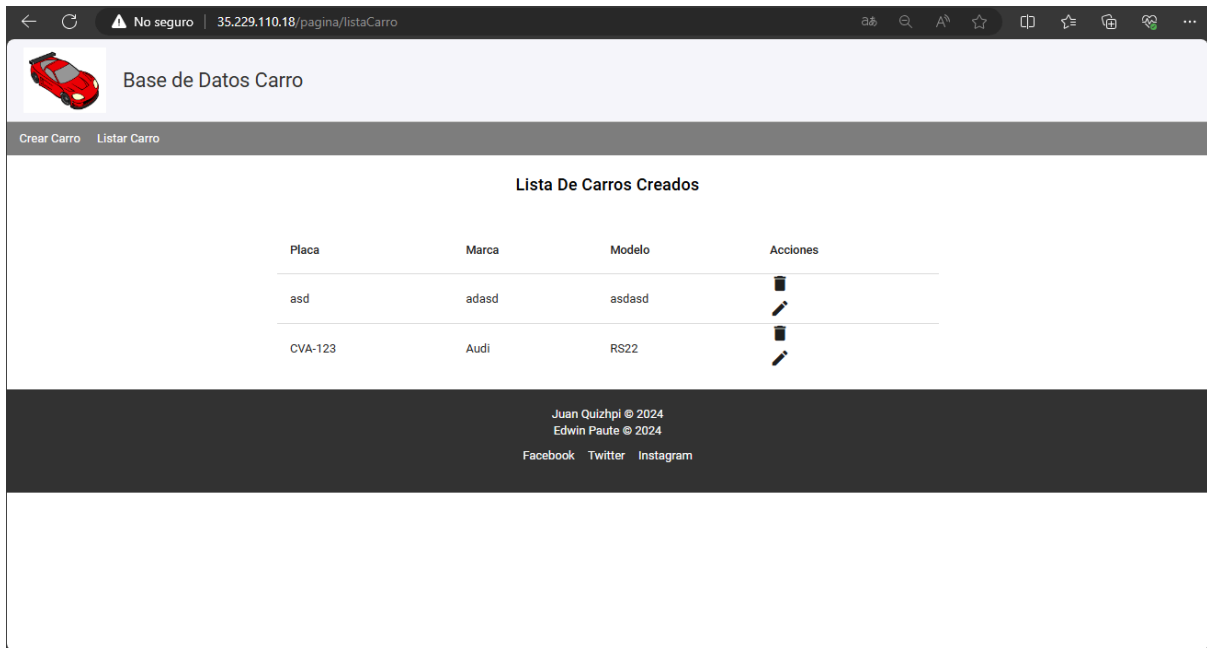
```



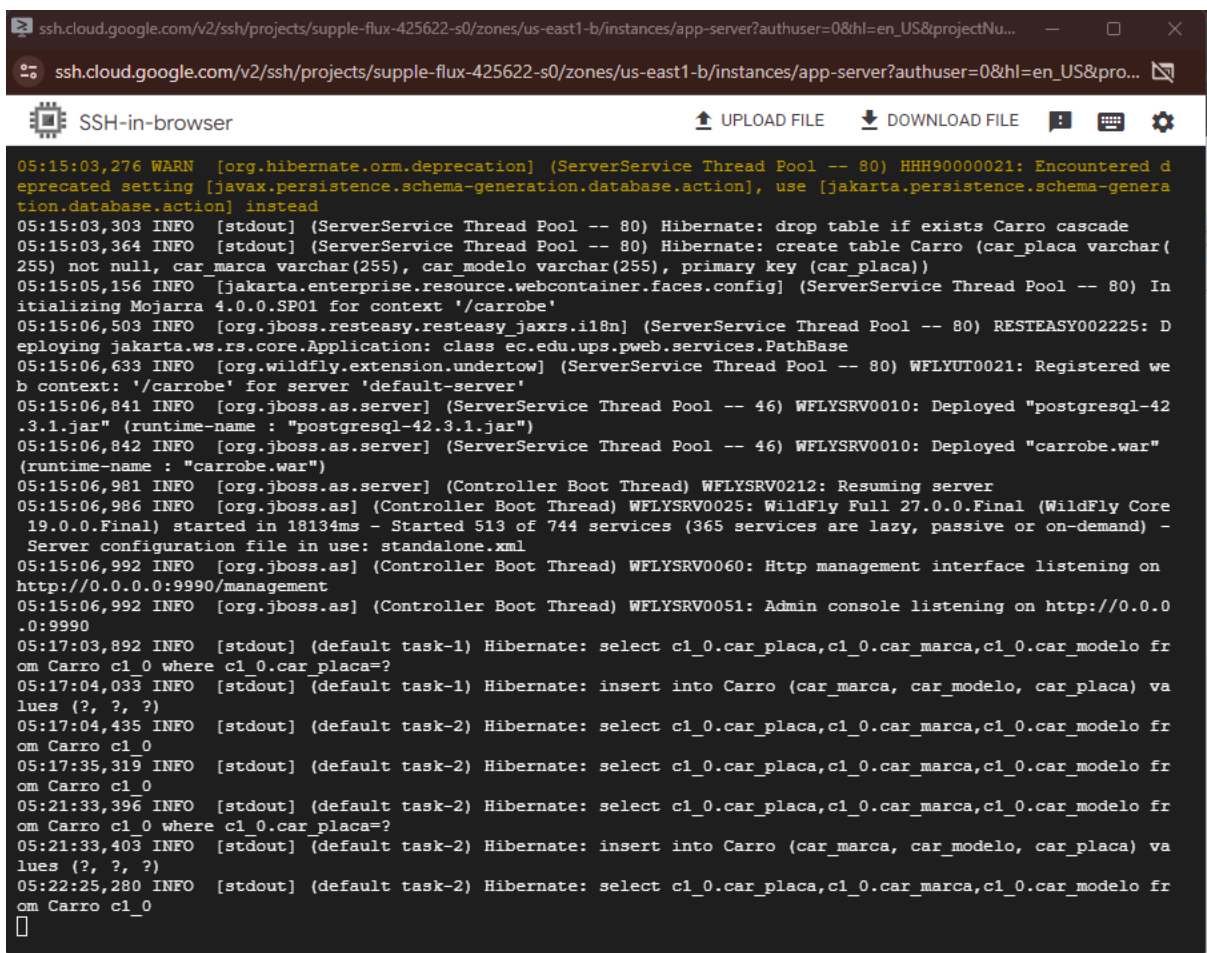
## Pruebas de la Aplicación

### Funcionamiento de nuestro FrontEnd





## Funcionamiento del BackEnd



## Funcionamiento de la Base de Datos

```
ssh.cloud.google.com/v2/ssh/projects/supple-flux-425622-s0/zones/us-east1-b/instances/ubuntu-focal-1?authuser=0&hl=en_US&projec...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

System information as of Mon Jun 10 05:24:03 UTC 2024

System load:  0.0          Processes:           112
Usage of /:   31.2% of 9.51GB Users logged in:       0
Memory usage: 7%          IPv4 address for ens4: 10.142.0.5
Swap usage:   0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Mon Jun 10 04:36:43 2024 from 35.235.243.209
atlas_ep26@ubuntu-focal-1:~$ psql -U carroadmin carrobase -W
Password:
psql (12.18 (Ubuntu 12.18-0ubuntu0.20.04.1))
Type "help" for help.

carrobase=> select * from carro;
 car_placa | car_marca | car_modelo
-----+-----+-----
    asd    |   adasd   |   asdasd
CVA-123    |   Audi    |   RS22
(2 rows)

carrobase=> 
```

Enlaces a los repositorios usados para el despliegue

<https://github.com/JuanQuizhpi/frontendcarroapp.git>

<https://github.com/JuanQuizhpi/libreriasJDBC.git>

<https://github.com/JuanQuizhpi/carrobe.git>

IP para probar el proyecto

[Frontendcarroapp](#)