ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

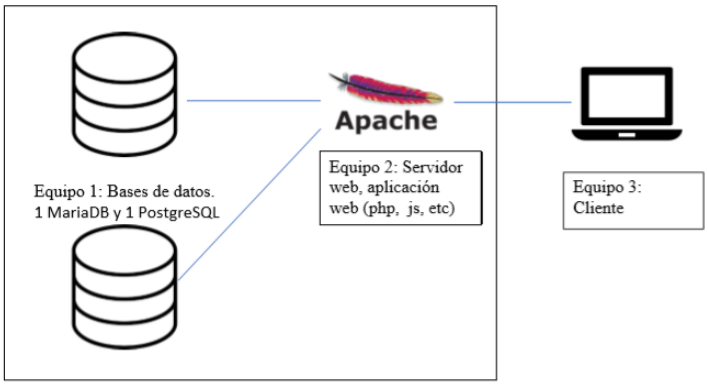
SERVICIO SOBRE REDES

ASIGNATURA:	Servicios sobre Redes
PROFESOR:	Ing. Juan Pablo Zaldumbide
PERÍODO ACADÉMICO:	2020-A

Prueba_1

TÍTULO:

Una Aplicación web



INTEGRANTES

Edison Jumbo
Jhonathan Pizarra

FECHA DE REALIZACIÓN: 19 / 08 / 2020

CALIFICACIÓN OBTENIDA:

FIRMA DEL PROFESOR:

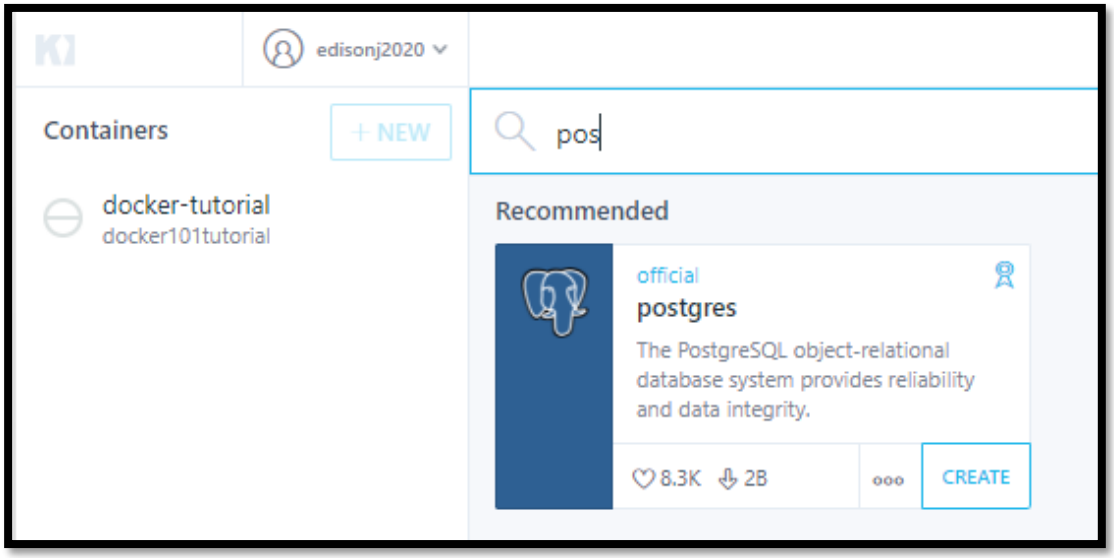
1 PROPÓSITO DE LA PRUEBA

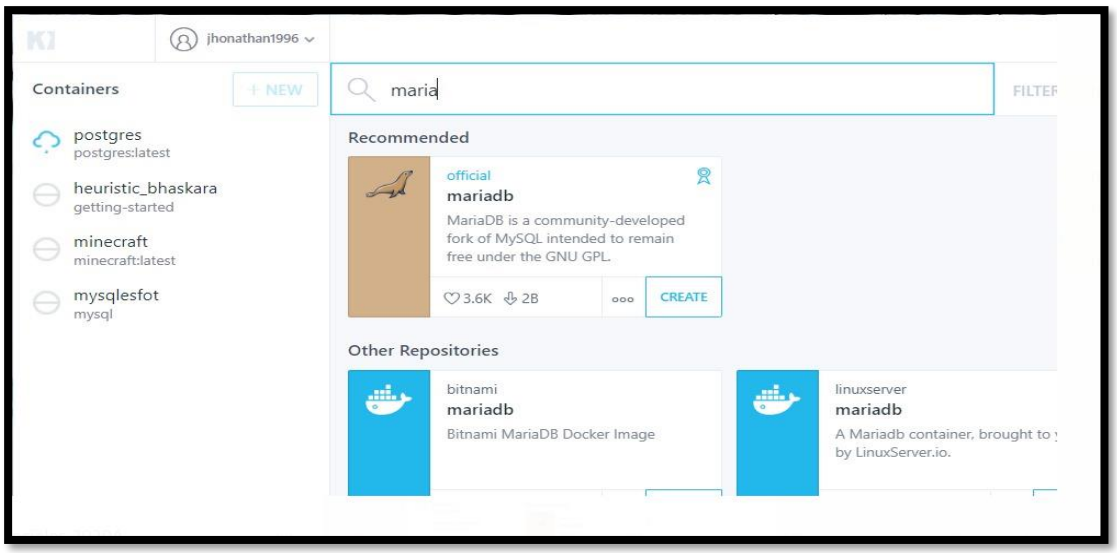
Utilizando docker y Hamachi implementar la siguiente arquitectura y una aplicación web sencilla que consuma elementos de las bases de datos, documentar los pasos que siguió en un repositorio de github:

2 DESARROLLO Y RESULTADOS DE LA PRÁCTICA

1. DOCKER

Instalación de las imágenes de las bases de datos Postgres y MaríaDB por medio de Docker.





2. HAMACHI

3. BASES DE DATOS

PosgreSQL

Este comando permite correr el programa en el puerto 5432:5432. Con el nombre yourContainerName.

- docker run --name some-postgres -e POSTGRES_PASSWORD=mysecretpassword -d postgres

```
C:\Users\Edison Jumbo>docker run -p 5432:5432 --name yourContainerName -e POSTGRES_PASSWORD=yourPa
ssword -d postgres
4bf70f2b66610f7a1abe73aed7870735a308034c6590c343f87a740500b3e2b0
```

Para verificar que el contenedor de postgreSQL esta corriedo se debe ingresar el comando.

- docker ps

```
C:\Users\Edison Jumbo>docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
      PORTS                     NAMES
4bf70f2b6661        postgres            "docker-entrypoint.s…"   5 minutes ago       Up 5 minutes
      0.0.0.0:5432->5432/tcp    yourContainerName
```

Para poder conectarse con el contenedor es necesario ingresar el **ID** del contenedor

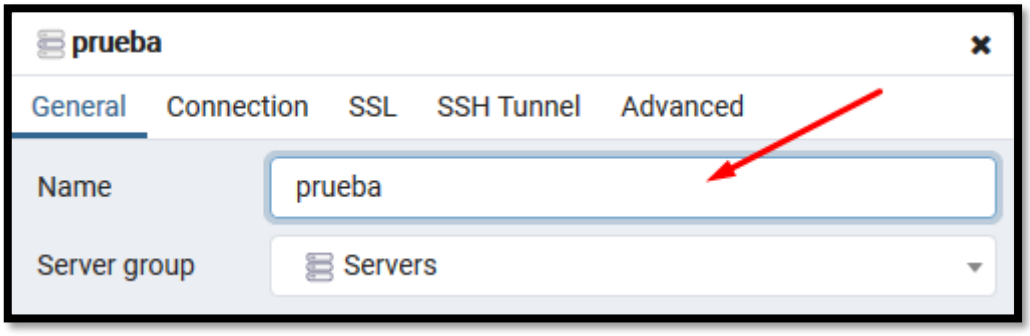
- docker exec -it [ID contenedor]bash

```
C:\Users\Edison Jumbo>docker exec -it 4bf70f2b6661 bash
```

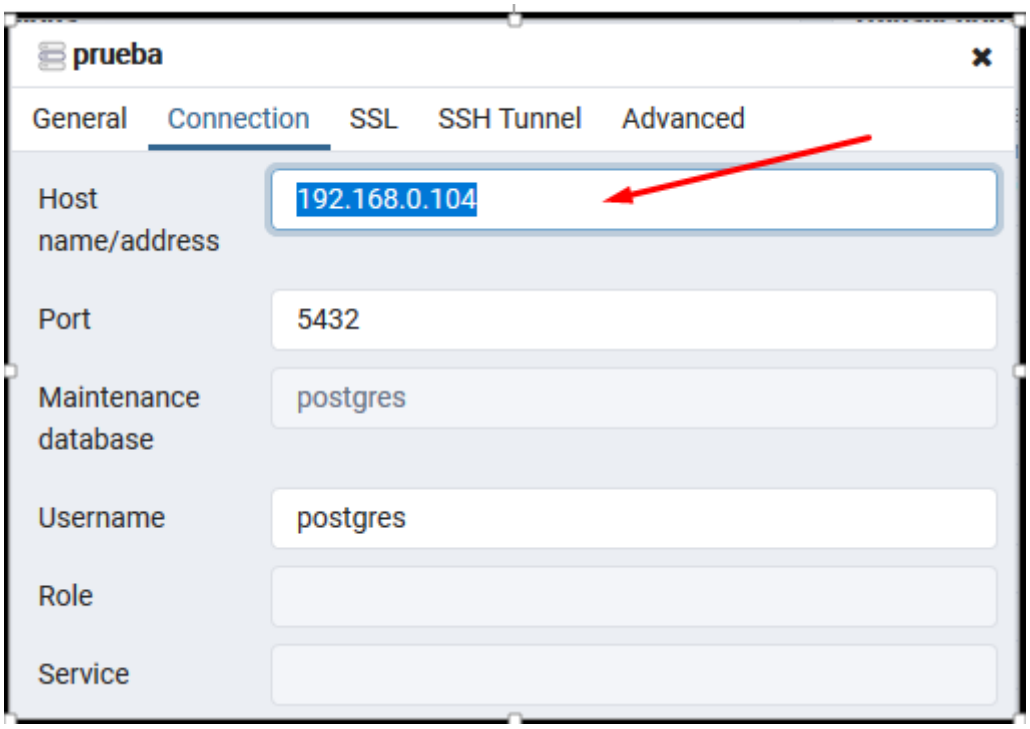
Configuración del cliente de PostgreSQL

Descargamos el cliente pgaAdmin
https://www.pgadmin.org/

Crear un nuevo servidor con cualquier nombre, en este caso lo llamaremos prueba.



En la opción CONNECTION, debemos ingresar la IP del equipo local y la contraseña se debe ser cualquiera.



Luego, se debe parar los servicios de postgresql por medio de este comando.

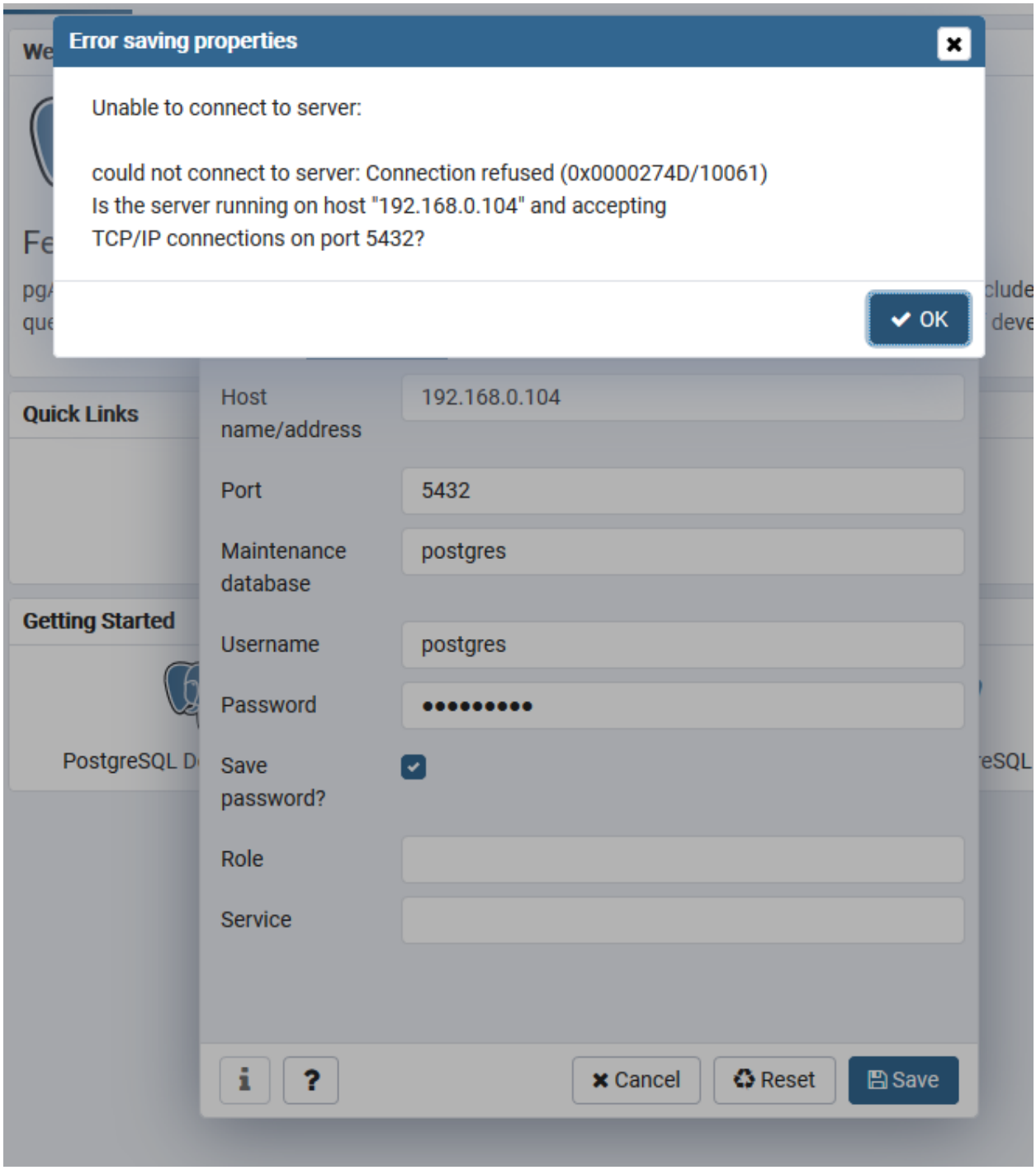
- docker stop [ID contenedor]

Luego, creamos un nuevo contenedor para la base de datos postgreSQL.

- docker rm [ID contenedor]

Finalmente agregamos el nuevo servidor con las configuraciones ingresadas.

- docker run --name **[ID contenedor]** -e POSTGRES_PASSWORD=**[password ingresada]** -d postgres



Coneccion postgresql y docker
https://elanderson.net/2018/02/setup-postgresql-on-windows-with-docker/

MariaDB con Docker

Primero de todo vamos a ver qué serie de imágenes tenemos disponibles. De la siguiente manera:

- docker search mariadb

Nos aparecerán unas cuantas. Nosotros utilizaremos la última versión oficial, llamada "mariadb:latest"

- `docker pull mariadb:latest`

Para crear el contenedor utilizando la imagen, primero de todo utilizaremos el parámetro "inspect" para que nos muestre los puertos que utiliza la imagen:

- docker inspect b1fe0881b739 | grep –i tcp

```
C:\Users\Edison Jumbo>docker inspect b28df07deb6c | grep -i tcp
'grep" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
```

Para crear el contenedor vamos a especificar varias cosas, la primera, especificaremos la clave de superusuario utilizando -e MYSQL_ROOT_PASSWORD=nuestra-pass; además le vamos a asignar un nombre utilizando --name mariadbtst , de la siguiente manera:

- docker run -dti -p 33306:3306 --name mariadbtst -e MYSQL_ROOT_PASSWORD=contrasea b1fe0881b739

```
C:\Users\Edison Jumbo>docker inspect b28df07deb6c | findstr -i tcp
                "3306/tcp": {}
                "3306/tcp": {}
```

Una vez creado ya lo podemos ver en funcionamiento:

- docker run -dti -p --name mariadbtst -e MYSQL_ROOT_PASSWORD=mariadb b28df07deb6c

```
C:\Users\Edison Jumbo>docker run -dti -p 33306:3306 --name mariadbtst -e MYSQL_ROOT_PASSWORD=mariadb b28df07deb6c
a43410e5e151d85fcfe6c18ad42253ea7680e1ffd6e799c449d95f653375ec26
```

4. SERVIDOR WEB

Por medio de XAMPP despliegaremos nuestro servidor web, entonces lo primero que necesitamos hacer es un script que recoja las credenciales de la base de datos de la máquina en dónde estén las bases de datos que a su vez están en Docker;

Lo primero que haremos será un Script para que pueda tomar los datos desde la base, es decir sean consumidos, en nuestro caso lo haremos con PHP, y le daremos algunos estilos:

Conexión a la base de datos:

```php
$conexion = pg_connect("host=192.168.0.104 dbname=prueba user=USUARIO password=password");
$nombres = pg_query($conexion, "SELECT nombre FROM estudiante WHERE edad < 30");
```

Script para el consumo:

File Edit View Selection Find Packages Help

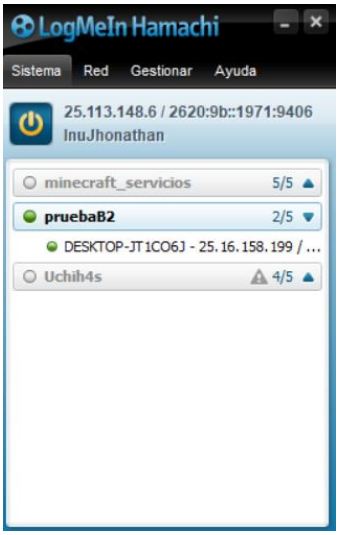
Project

PruebaVoIP
- recursos
  - conexion.php
  - empleados.php
  - index.php
- index.html

index.html | empleados.php | conexion.php

```php
$nombres = pg_query($conexion, "SELECT nombre FROM estudiante WHERE edad < 30");


//Leer datos desde Postgres
// base da datos: prueba
//tabla: estudiante: id, nombre, edad

function listarPersonas( $conexion )
    {
        $sql = "SELECT * FROM tbl_personas ORDER BY id";
        $ok = true;
        // Ejecutar la consulta:
         $rs = pg_query( $conexion, $sql );
        if( $rs )
        {
            // Obtener el número de filas:
             if( pg_num_rows($rs) > 0 )
            {
                echo "<p/>LISTADO DE EMPLEADOS<br/>";
                echo "==================<p />";
                // Recorrer el resource y mostrar los datos:
                 while( $obj = pg_fetch_object($rs) )
                    echo $obj->id." - ".$obj->nombre."<br />";
            }
            else
                echo "<p>No se encontraron estudiantes</p>";
        }
        else
            $ok = false;
        return $ok;
    }
```

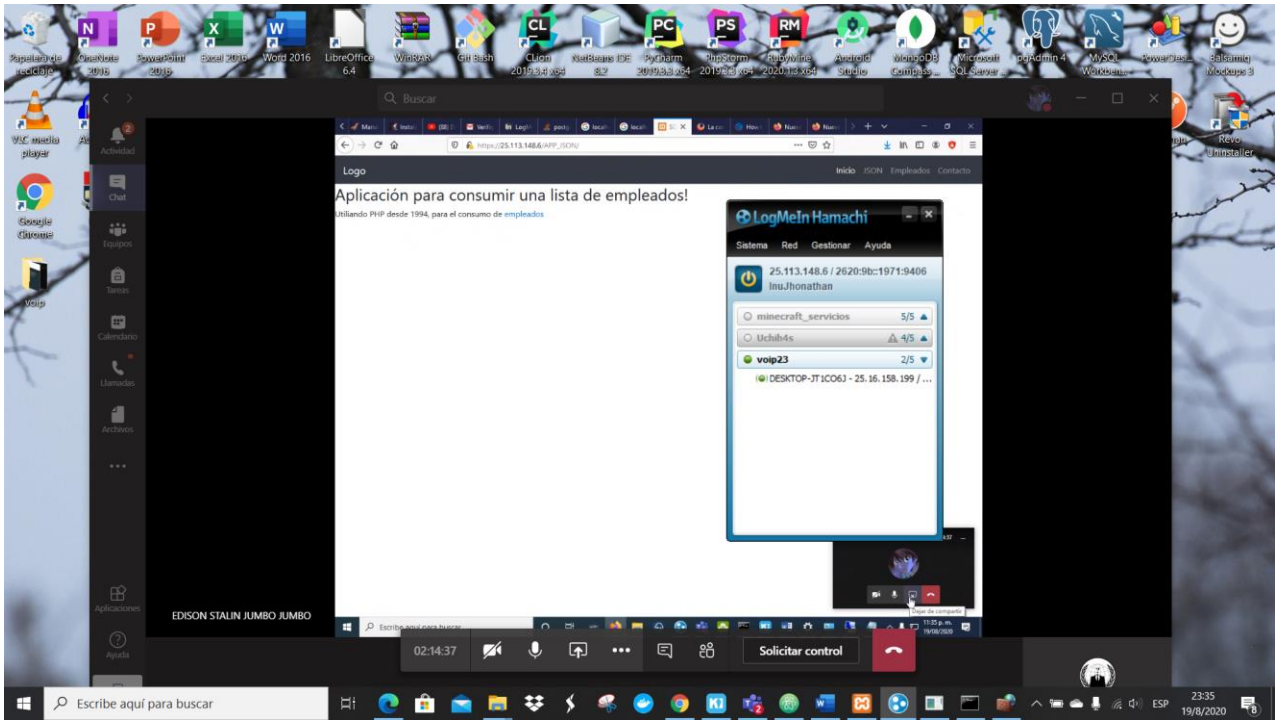
Necesitamos usar Hamachi para estar en red y poder acceder a sus servidor:



Estilos:

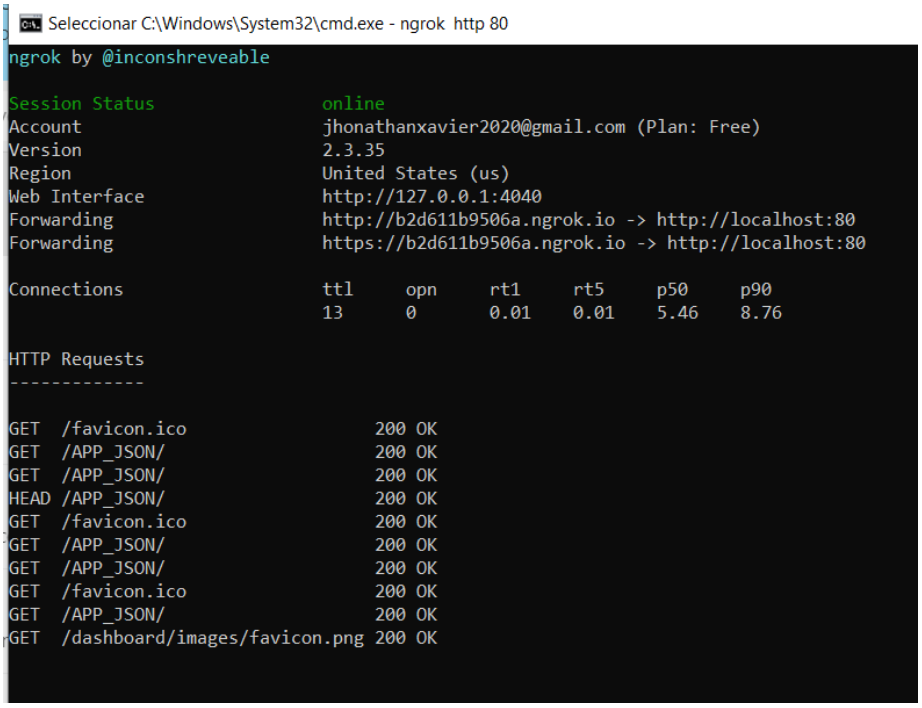
```
Project
PruebaVoIP
  recursos
    conexion.php
    empleados.php
    index.php
  index.html

index.html | empleados.php | index.php | conexion.p

<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="
<title>Prueba VoIP</title>
</head>
<body>
<!--NavBar-->
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="#">Logo</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="nav
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav ml-auto">
<li class="nav-item active">
<a class="nav-link" href="#">Inicio <span class="sr-only">(current)</span></a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">JSON</a>
</li>
<li class="nav-item">
<a class="nav-link" href="https://uchih4jhonathan.alwaysdata.net/recursos/empleados.php">Empleados</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Contacto</a>
</li>
</ul>
</div>
</nav>
<!--Fin Nav-->
<h2>Aplicación para consumir una lista de empleados!</h2>
<p>Utiliando PHP desde 1994, para el consumo de <a href="recursos/index.php">empleados</a></p>
```

Por medio de él ya podemos acceder remotamente al servidor



5. CLIENTE.

Utilizamos Nrock para generar una url y que de ese modo el cliente final pueda acceder a nuestro servicio:



Puedes acceder a nuestra aplicación desde: http://b2d611b9506a.ngrok.io/APP_JSON/

3 CONCLUSIONES

- Crea contenedores ligeros y portables para las aplicaciones software que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues.

- Hamachi es un popular programa que simula una red local entre varios equipos que permite la conexión e intercambiar archivos entre diferentes máquinas conectadas de una manera rápida y sencilla.

4 BIBLIOGRAFÍA

Repositorio de GitHub aquí: https://github.com/EdisonStalin/pruebaB2_Servicios