MEMORIA DE TRABAJO

1. OBJETIVO

Familiarización con conceptos de conteinerización, y programación asociada a los contenedores.

2. ALCANCE

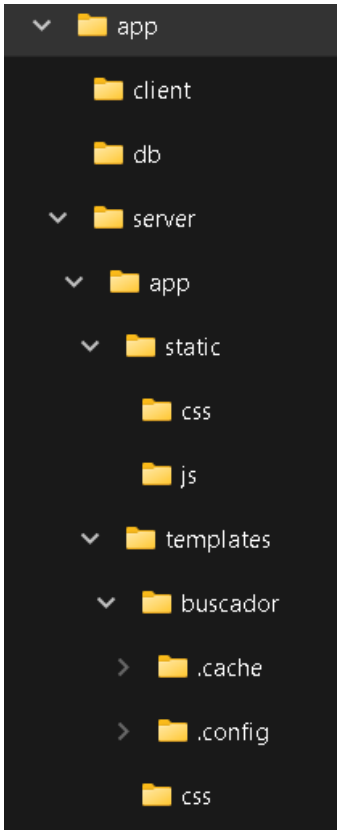
El alcance se encuentra detallado en el Archivo denominado *Ejercicio_evaluado.pdf*

3. ENTREGABLE

Las carpetas conteniendo todos los archivos necesarios para ejecutar lo detallado en el alcance.

4. DESARROLLO

A. Árbol de carpetas



B. Carpeta client

A continuación se indica el contendidos de los archivos alojados en la carpeta client

1. client.py

```
import requests
import random
import time
import logging as log

log.basicConfig(level=log.INFO)

# URL de tu servidor Flask
server_url = 'http://servidor:5000'

def generate_random_username():
    # Genera un nombre de usuario aleatorio
    return 'user' + str(random.randint(1, 1000))

def send_request():
    # Envia una solicitud al servidor Flask con un nombre de usuario
aleatorio
    username = generate_random_username()
    log.info("Registering User: %s", username)
    response = requests.post(f'{server_url}/formulario', data={'nombre':
username})

    # Imprime la respuesta del servidor
    log.info("SUCCESFULLY REGISTERED User: %s", username)

if __name__ == '__main__':
    while True:
        # Envia solicitudes indefinidamente
        send_request()

        # Duerme un tiempo aleatorio entre 1 y 5 segundos
        sleep_time = random.uniform(1, 5)
        time.sleep(sleep_time)
```

2. Dockerfile.dockerfile

```
# Establecemos la imagen base con Python
# De esta manera, evitamos tener que instalar manualmente Python.
FROM python:latest

# Establecemos el directorio de trabajo en el contenedor,
# es decir, a partir de ahora, trabajaremos sobre este directorio.
# Debemos considerar que esto hará un cambio de directorio.
WORKDIR /app

# Instalamos las dependencias de la aplicacion
# En este caso, solo necesitaremos Flask y el conector a MySQL.
RUN pip install requests
```

```
# Copiamos el codigo fuente de la aplicacion al contenedor
# Esto incluye tanto el archivo app.py como formulario.html
COPY client.py app/

# Finalmente, declaramos el comando a ejecutar por defecto al lanzar, el contenedor.
# Comando por defecto al ejecutar el contenedor
CMD ["python", "app/client.py"]
```

C.Carpeta db

Init.sql

```
-- Crea la base de datos
 CREATE DATABASE IF NOT EXISTS mydatabase;
 USE mydatabase;

 -- Crea una tabla
 CREATE TABLE usuarios (
 id_usuario INT AUTO_INCREMENT PRIMARY KEY,
 nombre VARCHAR(50),
 apellido VARCHAR(50)
 );

 -- Inserta valores por defecto en tabla usuarios
INSERT INTO usuarios (nombre, apellido) VALUES ('Juan','Alvarez');
INSERT INTO usuarios (nombre, apellido) VALUES ('Marta','Rodriguez');
INSERT INTO usuarios (nombre, apellido) VALUES ('Pedro','Seminario');
INSERT INTO usuarios (nombre, apellido) VALUES ('Luis','Garcia');
INSERT INTO usuarios (nombre, apellido) VALUES ('Karina','Almarales');
INSERT INTO usuarios (nombre, apellido) VALUES ('Andrea','Aybar');
INSERT INTO usuarios (nombre, apellido) VALUES ('Angel','De la Torre');
INSERT INTO usuarios (nombre, apellido) VALUES ('Julia','Aponte');
INSERT INTO usuarios (nombre, apellido) VALUES ('Domingo','Suarez');
INSERT INTO usuarios (nombre, apellido) VALUES ('Miguel','Guillen');
INSERT INTO usuarios (nombre, apellido) VALUES ('Maria','Rojas');

-- Crear la tabla de asignaturas
CREATE TABLE asignaturas (
  id_asignatura INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(60),
  descripción TEXT,
  id_usuario INT,
  FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario)
);

-- Inserta valores por defecto en tabla Asignaturas
INSERT INTO asignaturas (id_asignatura, nombre, descripcion) VALUES
('1','Programacion','asignatura para maestria Bigdata','1');
INSERT INTO asignaturas (id_asignatura, nombre, descripcion) VALUES ('2','Cloud','asignatura
para maestria Bigdata','1');
INSERT INTO asignaturas (id_asignatura, nombre, descripcion) VALUES
('3','Estadistica','asignatura para maestria Bigdata','1');
```

Dockerfile.dockerfile

```
FROM mysql:latest
```

```
# Establecemos variables de entorno para MySQL
ENV MYSQL_DATABASE=mydatabase
ENV MYSQL_ROOT_PASSWORD=ejemplo
ENV MYSQL_USER=test
ENV MYSQL_PASSWORD=ejemplo

# Copiamos el script de SQL a la imagen
COPY init.sql /docker-entrypoint-initdb.d/
```

C. Carpeta server

Dockerfile.dockerfile

```
# Establecemos la imagen base con Python
# De esta manera, evitamos tener que instalar manualmente Python.
FROM python:latest

# Establecemos el directorio de trabajo en el contenedor,
# es decir, a partir de ahora, trabajaremos sobre este directorio.
# Debemos considerar que esto hará un cambio de directorio.
WORKDIR /app

# Instalamos las dependencias de la aplicacion
# En este caso, solo necesitaremos Flask y el conector a MySQL.
RUN pip install Flask mysql-connector-python

# Copiamos el codigo fuente de la aplicacion al contenedor
# Esto incluye tanto el archivo server.py como formulario.html
COPY app/ app/



# Exponemos el puerto 5000 para Flask
# Esto no abre el puerto, simplemente es una forma de documentacion,
# Explica que el puerto está abierto.
EXPOSE 5000

# Finalmente, declaramos el comando a ejecutar por defecto al lanzar, el contenedor.
# Comando por defecto al ejecutar el contenedor
CMD ["python", "app/app.py"]
```

Carpeta server/app

app.py

```
from flask import Flask, request, render_template
import logging as log
import mysql.connector

log.basicConfig(level=log.INFO)

app = Flask(__name__)

# Configura la conexion a la base de datos MySQL
```

```
db_config = {
    'host': 'mysql', # Definido en el docker-compose en seccion posterior
    'user': 'root',
    'password': 'ejemplo',
    'database': 'mydatabase'
}

def store_data_db(data):
    try:
        # Conecta a la base de datos MySQL
        conn = mysql.connector.connect(**db_config)
        cursor = conn.cursor()

        # Inserta los datos en la tabla forms
        cursor.execute("INSERT INTO usuarios (nombre) VALUES (%s);", (data,))

        # Realiza commit y cierra la conexion
        conn.commit()
        conn.close()
        log.info("registrado exitosamente")
    except:
        log.warning("No se pudo insertar en la base de datos %s (Database may not be ready): "
%(data))

@app.route('/')
def index():
    # Esta funcion se asocia a la ruta raiz "/"
    return "Bienvenido Luis Angel a mi aplicacion web2!"

@app.route('/formulario', methods=[ 'GET','POST'])
def formulario():
   # Esta funcion se asocia a la ruta "/formulario"
    if request.method == 'POST':
        # Si se envia el formulario, procesamos los datos
        nombre = request.form['nombre']
        #apellido = request.form['apellido']

        # Llama a la funcion para almacenar datos en la base de datos
        store_data_db(nombre)

        mensaje = f"Hola, {nombre}. Bienvenido Luis a mi app web. Esta procesando la base de
datos"
        return mensaje
    #Si se accede por GET, mostramos el formulario
    return render_template('formulario.html')

@app.route('/buscador', methods=[ 'GET','POST'])
def buscador():
    try:
        # Esta funcion se asocia a la ruta "/buscador"
        if request.method == 'POST':
            # Si se envia el formulario, procesamos los datos
            nombre = request.form['nombre']
            mensaje = f"Hola, {nombre}. Bienvenido mi pagina de consultas"
            return mensaje
        #Si se accede por GET, mostramos el formulario
        return render_template('buscador.html')
    except:
            log.warning("No se pudo encontrar la pagina buscar.html")
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Carpeta server/app/templates

formulario.html

```
<!DOCTYPE html>
<html>
<head>
<title>ir al buscador tematico</title>
</head>
<body>
<h1>Visita mi buscador web</h1>
<a href="buscador"> ir a buscador</a>
<br>
<h1>Formulario de Bienvenida</h1>
<form method="POST" action="/formulario">
<label for="nombre">Nombre:</label>
<input type="text" id="nombre" name="nombre" required>
<label for="apellido">Apellido:</label>
<input type="text" id="apellido" name="apellido" required>
<br>
<input type="submit" value="Enviar">
</form>
</body>
</html>
```

buscador.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Search</title>
  <link rel="stylesheet" type="text/css" href="css/style.css">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css">
</head>
<body>

  <div class="container">

  <input type="text" placeholder="Buscar">

  <div class="btn">


  <i class="fa fa-search"></i>

  </div>


  </div>

<script language="Javascript" src="xsearch-5.2.js"></script>
<script language="Javascript" src="db.js"></script>
```

```
<script language="Javascript">
  initXsearch();

</script>
</body>
</html>
```

Carpeta server/app/templates/css

style.css

```
body {
  margin: 0;
  padding: 0;
  background: #FFFFFF;

color:#000000;
 link:#FFFFFF;
 vlink:#996699;
 alink:#996699;
}



.container {
  position: absolute;
  left: 54%;
  top: 16%;
  transform: translate(-50%,-50%);
  padding: 10px;

}

input {
  outline: none;
  box-sizing: border-box;
  height: 60px;
  width: 0px;
  padding: 10 10px ;
  color: #000;
  border-radius: 50px;
  font-size: 20px;
  border: 1px solid #D50000;
  transition: all .7s ease;
}

::placeholder {
  color: grey;
}

.btn {
  position: absolute;
  right: 0px;/*cambiar a 0*/
  top: 0px;
  width: 80px;
  height: 80px;
```

```
  background: #dd5f5f;
  line-height: 80px;
  border-radius: 50%;
  text-align: center;
  cursor: pointer;
  transition: .1;
}

.btn i {
  font-size: 25px;
  color: white;
  line-height: 80px;
  transition: all .5s ease;
}

.container:hover input {
  width: 350px;
}

.container:hover i{
  transform: rotate(-360deg);
}

.btn:hover{
  background: #cc0000;
}
```

5. CONCLUSIONES

A través del siguiente link se puede acceder al repositorio:

https://github.com/grandtranfers/cloud_docker