Actividad 4. Proyecto integrador. Etapa 1

Esta actividad consiste en aplicar los conocimientos adquiridos a lo largo del curso y retomar lo aprendido en asignaturas previas, de modo que integres diferentes disciplinas. De igual forma se toman como referente actividades elaboradas previamente, lo que garantiza la transversalidad de los contenidos revisados para fortalecer el desarrollo de competencias.

El proyecto consiste en instalar la infraestructura de un sistema distribuido bajo una arquitectura cliente servidor. Para aplicar el concepto de sistema distribuido de forma clara, deberán utilizarse por lo menos dos equipos de cómputo conectados en red.

El sistema distribuido a instalar constará de:

- Servidor de base de datos (Equipo servidor)
- Servidor Web (Equipo servidor)
- Cliente (Equipo cliente)

El objetivo de la primera parte del Proyecto integrador es realizar la instalación y configuración del equipo servidor:

- servidor de base de datos
- servidor web

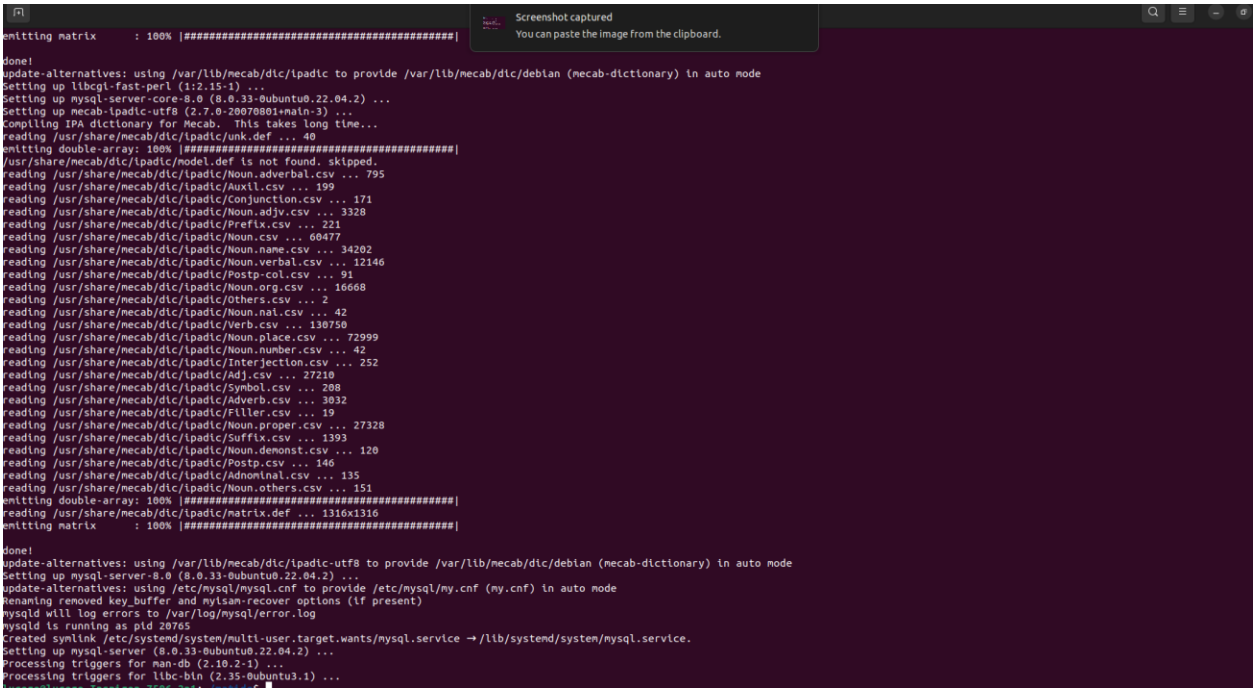
Explicación del proceso realizado

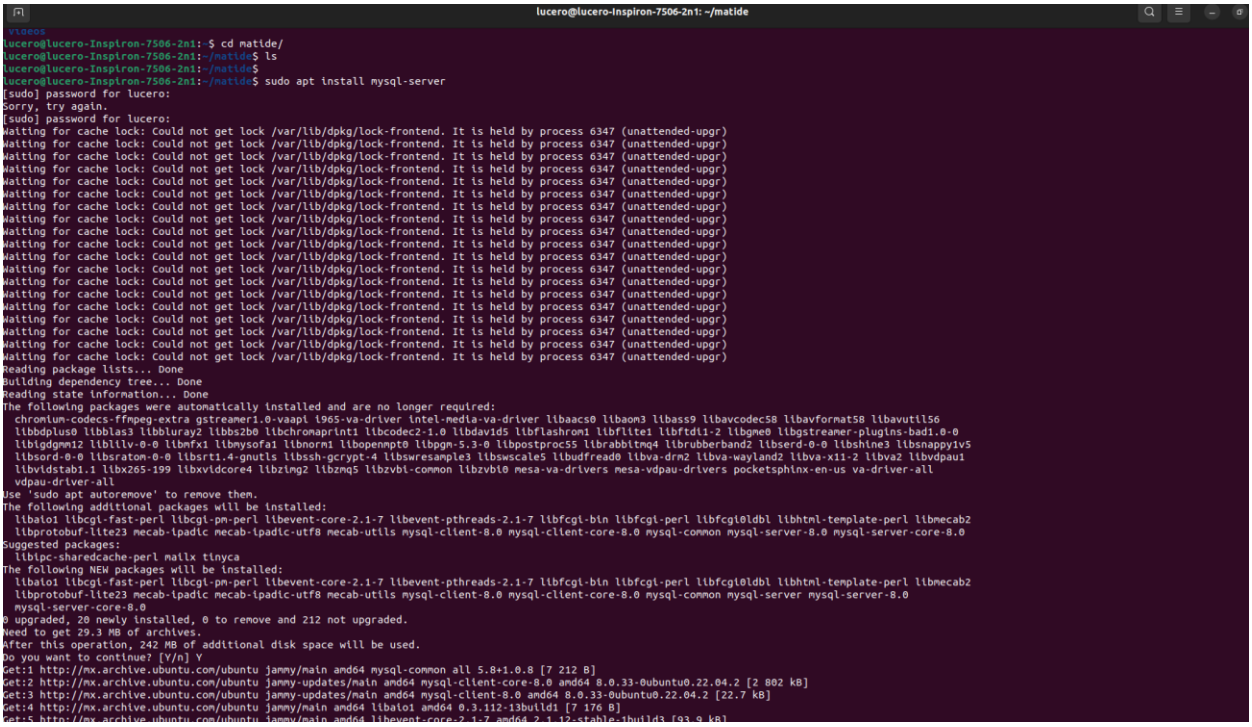
Instalación

Instala y configura el servidor de base de datos de tu preferencia en el equipo servidor. Se sugiere el servidor de MySQL Server, el cuál puedes descargar de la siguiente página: https://dev.mysql.com/downloads/

La base de datos se creo en Ubuntu.

Antes de instalar cualquier paquete, es recomendable actualizar el sistema operativo.





Durante la instalación, se te solicito contraseña

La instalación de MySQL en Ubuntu es relativamente fácil debido a que el paquete de MySQL Server está disponible en los repositorios oficiales de Ubuntu. Esto simplifica el

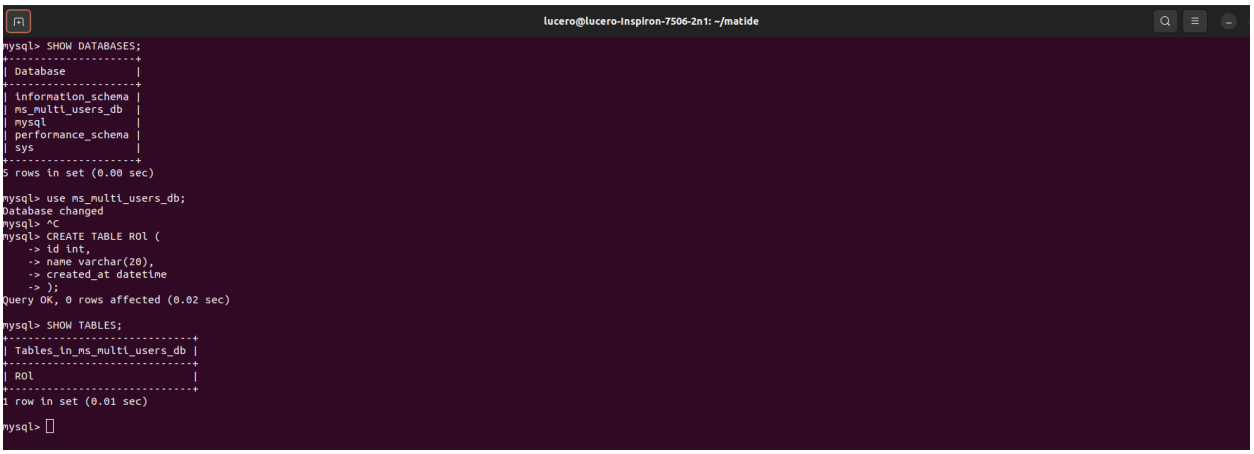
proceso de instalación y actualización del software a través del sistema de gestión de paquetes APT.

Creación de Base de datos

Crea una base de datos para control de acceso e inserta datos en ella, los usuarios deberán iniciar sesión mediante nombre de usuario y contraseña, cada usuario deberá tener un rol asignado (Roles: Administrador, operativo, general).

El siguiente enlace te puede servir de apoyo: https://evilnapsis.com/2016/09/05/3-modelos-de-base-de-datos-para-tabla-de-usuarios/

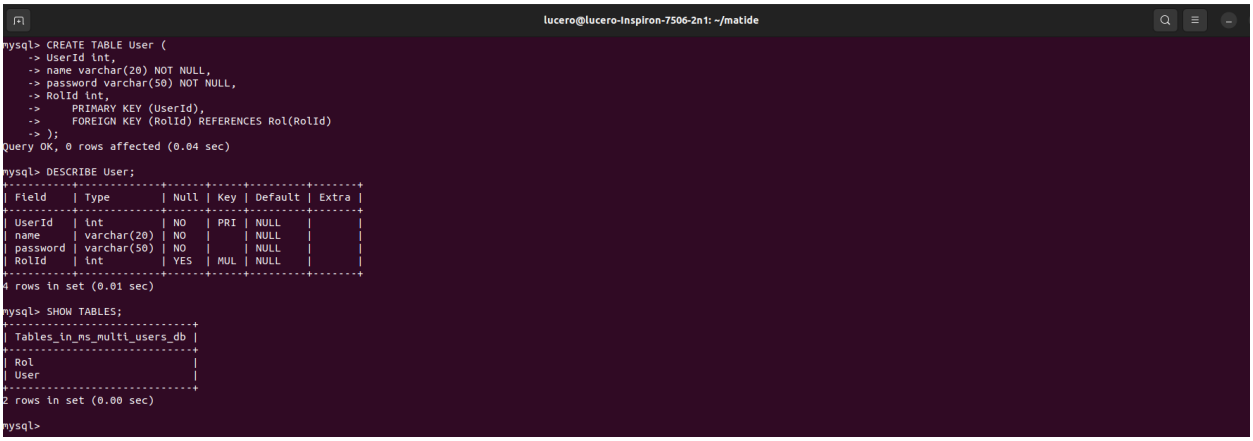
Como primer paso, revisamos las bases de datos que tenemos actualmente:



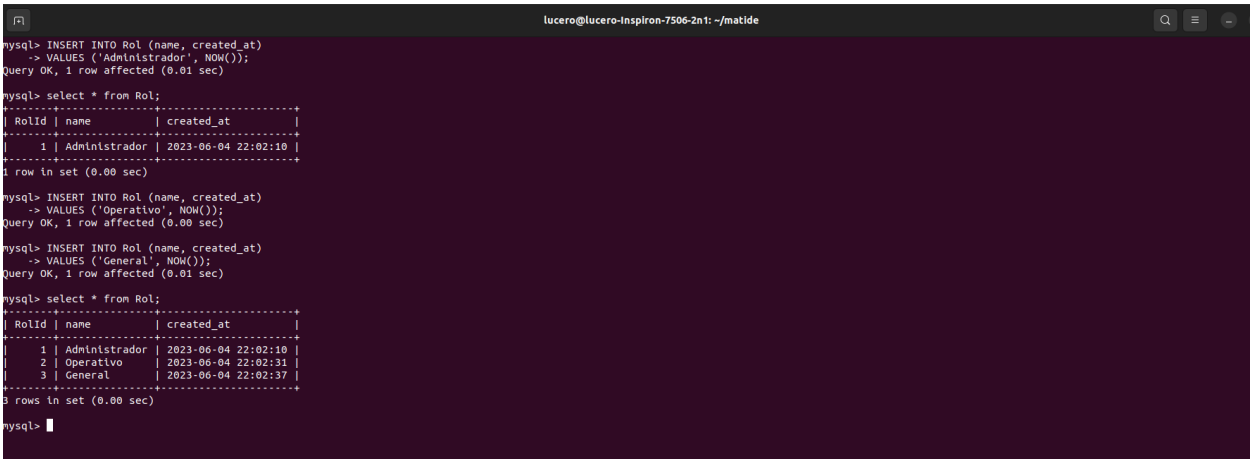
Creamos la tabla Rol

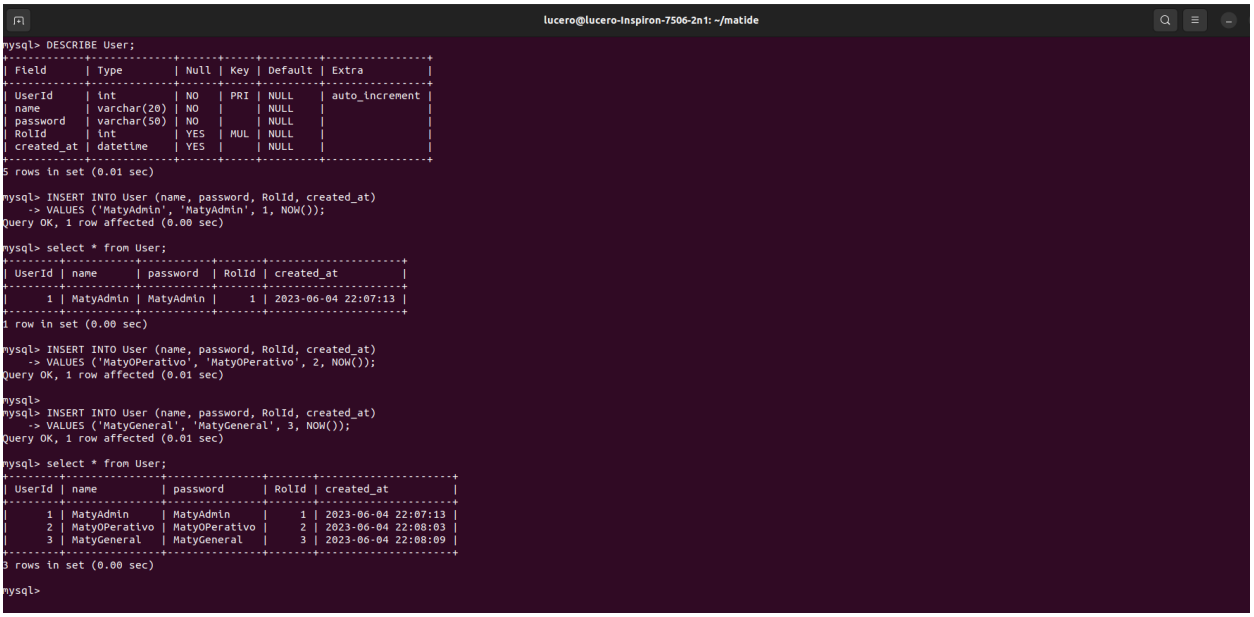
Mostramos la tabla:

Insertamos datos:



Mostramos los datos en la tabla Rol:





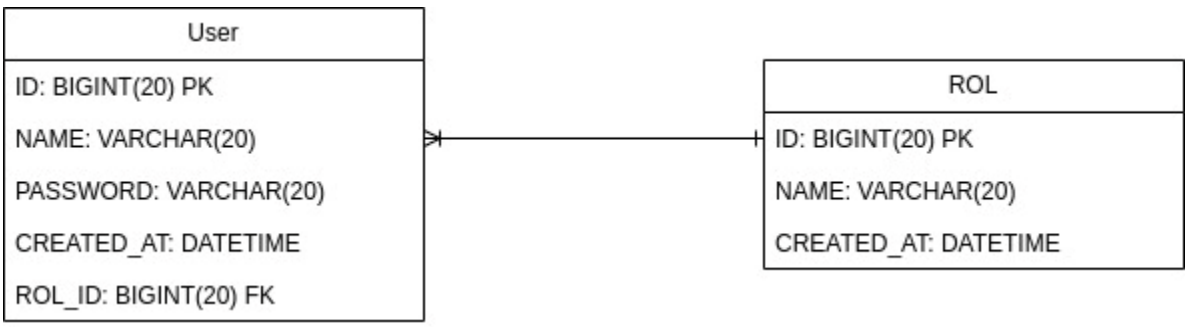
Script de base de datos generada, con las instrucciones de inserción de datos

SCRIPTS

```
CREATE DATABASE ms_multi_users_db;


SHOW DATABASES;



use ms_multi_users_db;



CREATE TABLE Rol (
	RolId int NOT NULL AUTO_INCREMENT,
	name varchar(20) NOT NULL,
```

```
      created_at datetime,
      PRIMARY KEY (RolId)
);


SHOW TABLES;

CREATE TABLE User (
      UserId int NOT NULL AUTO_INCREMENT,
      name varchar(20) NOT NULL,
      password varchar(50) NOT NULL,
      RolId int,
      created_at datetime,
    PRIMARY KEY (UserId),
    FOREIGN KEY (RolId) REFERENCES Rol(RolId)
);

INSERT INTO Rol (name, created_at)
VALUES ('Administrador', NOW());
INSERT INTO Rol (name, created_at)
VALUES ('Operativo', NOW());
INSERT INTO Rol (name, created_at)
VALUES ('General', NOW());


INSERT INTO User (name, password, RolId, created_at)
VALUES ('MatyAdmin', 'MatyAdmin', 1, NOW());
INSERT INTO User (name, password, RolId, created_at)
VALUES ('MatyOPerativo', 'MatyOPerativo', 2, NOW());
INSERT INTO User (name, password, RolId, created_at)
VALUES ('MatyGeneral', 'MatyGeneral', 3, NOW());
```

Diagrama entidad relación de la base de datos



Conclusión

Al realizar la instalación y creación de bases de datos en MySQL en Ubuntu, he adquirido conocimientos sobre cómo configurar y administrar un entorno de base de datos, cómo crear estructuras de datos y cómo interactuar con los datos utilizando comandos SQL básicos. Estas habilidades me permitirán comenzar a desarrollar aplicaciones y sistemas que requieran almacenamiento y manipulación de datos.

Actividad 8. Proyecto integrador. Etapa 2

Objetivo

El objetivo de la segunda parte del Proyecto integrador es establecer la conexión desde una aplicación web a la base de datos y utilizar el equipo cliente para visualizar la aplicación como un cliente Web, completando de éste modo la estructura del sistema distribuido planteado en la Etapa 1 de tu Proyecto integrador.

¿Qué hacer?

1. Utilizando el IDE de desarrollo de tu preferencia (IDE's de desarrollo sugeridos: Visual Studio, Eclipse, NetBeans), realiza una aplicación web de inicio de sesión, con los siguientes requerimientos:

Pantalla de inicio de sesión:

a) Campo usuario (cuadro de texto para captura)

b) Campo contraseña (cuadro de texto para captura)

c) Campo rol (lista desplegable con los roles definidos)

La aplicación deberá conectarse a la base de datos creada en la Etapa 1 del proyecto integrador (de ser necesario realiza ajustes en tu base de datos) y realizar las validaciones correspondientes para realizar el inicio de sesión.

Envío de mensajes de error correspondientes, cuando los datos sean incorrectos.

2. Accede a la aplicación web desarrollada desde un navegador en el equipo Cliente y realiza las pruebas respectivas con los datos de tu base de datos.

3. Los entregables que formarán parte de tu trabajo son los siguientes:

a) Explicación del proceso realizado

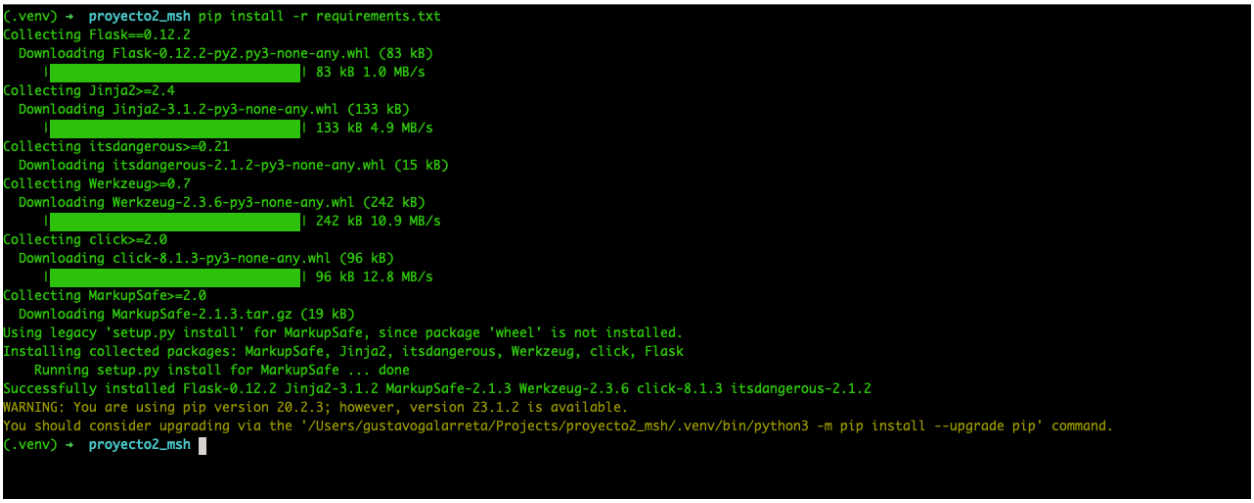
b) Capturas de pantalla que evidencien el desarrollo

c) Análisis del sistema distribuido desarrollado

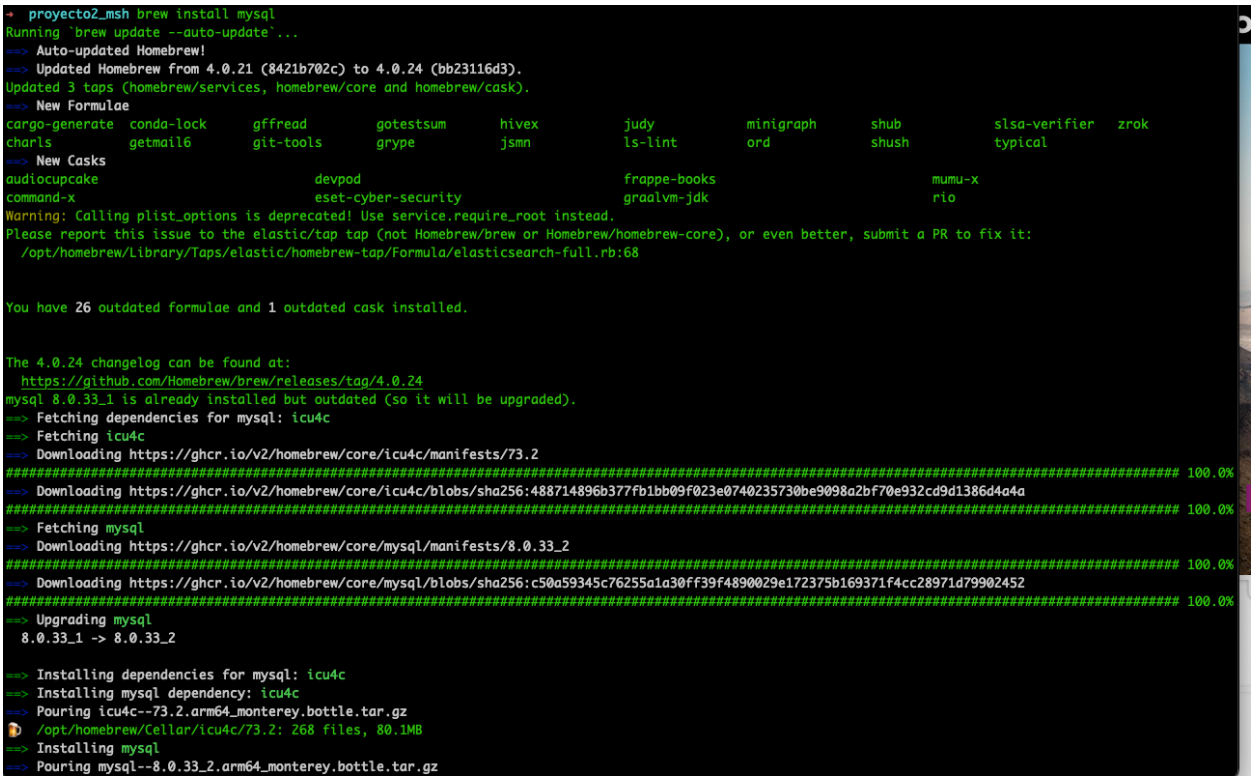
d) Esquema de Arquitectura del sistema distribuido realizado. Incluye explicación del funcionamiento de cada uno de los elementos.

Desarrollo

Iniciamos con la instalación de Flask y sus requerimientos.

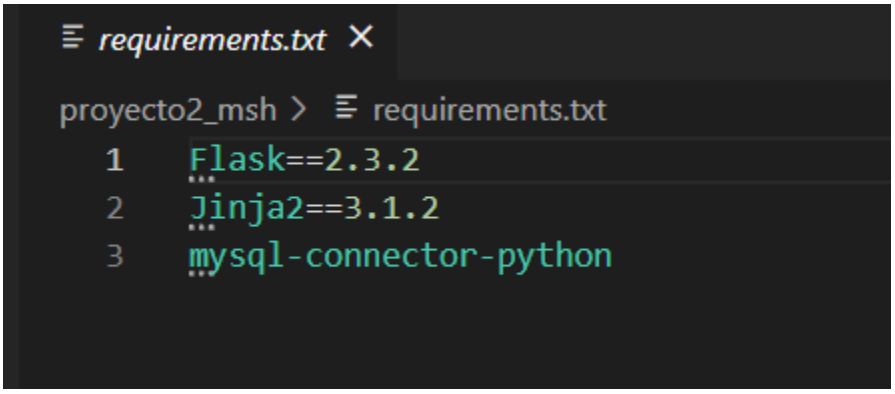


Instalamos Mysql



El siguiente paso es crear el código usando Visual Studio Code con el lenguaje Python.

- Indicamos los requerimientos.



- Creamos los controladores del login

login_controller.py

proyecto2_msh > controllers > login_controller.py > ...

```python
from db.db_connector import DbConnector

class LoginController:
    def __init__(self, request):
        self.request = request

    # public methods

    def login(self):
        if self.__valid_login():
            return True
        else:
            return False

    # private methods

    def __valid_login(self):
        db = DbConnector()
        users = db.find_user(self.__username(), self.__password(), self.__rol())
        if users:
            return True
        else:
            return False

    def __username(self):
        return self.request.form['username']

    def __password(self):
        return self.request.form['password']

    def __rol(self):
        return self.request.form['rol']

```

- Controladores para Flask

app.py 1

proyecto2_msh > app.py > login

```python
from flask import Flask, render_template, request
from controllers.login_controller import LoginController

# The default folder name should be "templates" else need to mention custom folder name
app = Flask(__name__, template_folder='views', static_folder='static_files')

@app.route("/")
def main():
    return render_template('home.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        login_controller = LoginController(request)
        if login_controller.login():
            return render_template('login_success.html')
        else:
            return render_template('login_try_again.html')

    return render_template('login.html')

```

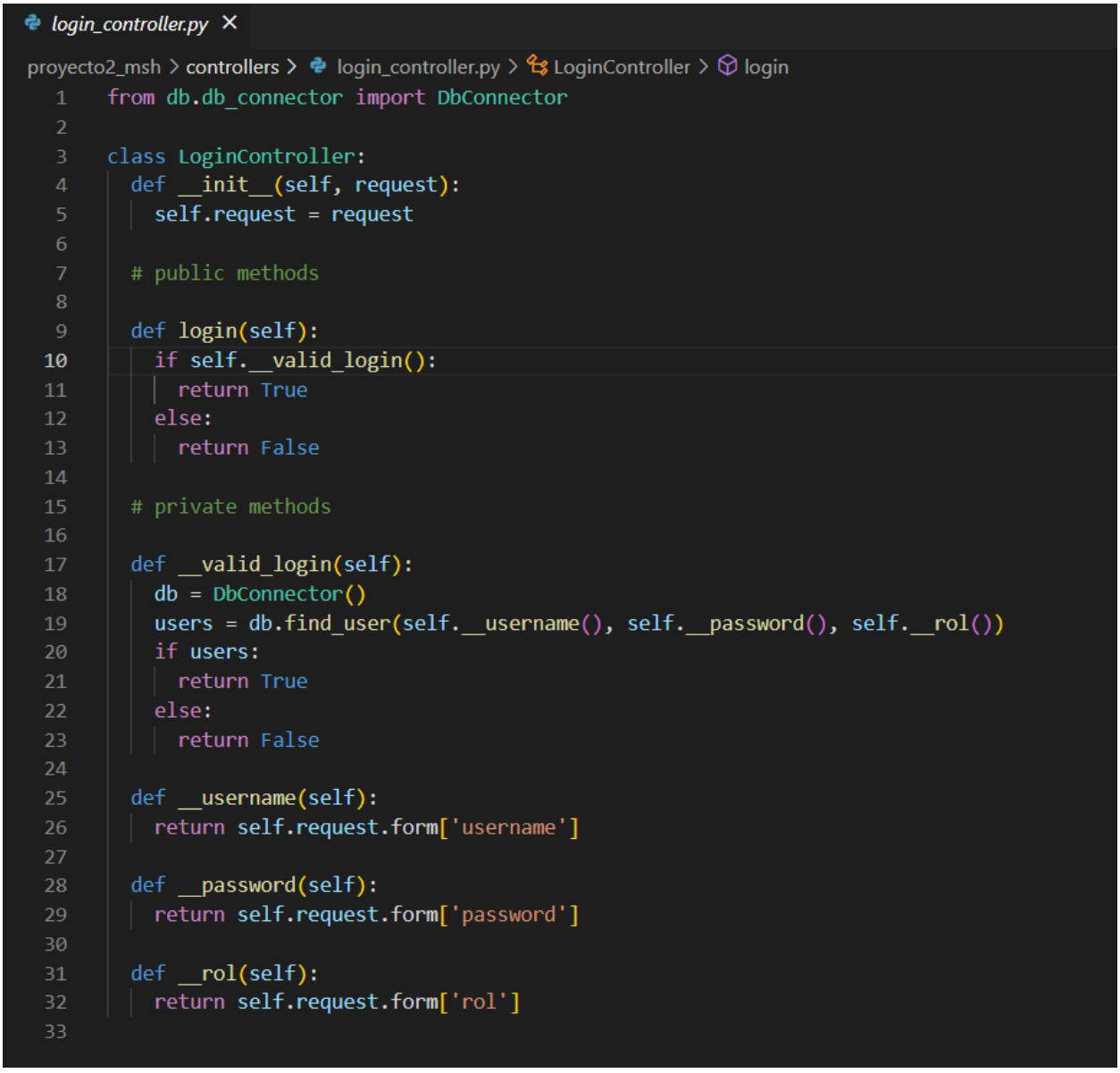
- Asignamos Ajax para login.

login.js

proyecto2_msh > static_files > login.js > ...

```javascript
$.ajax({
    type: "POST",
    contentType: 'application/json',
    url: "/login",
    data: {},
    dataType: "json",
    success: function(result) {
        console.log("it works!!");
        document.write(response);
    }
});
```

- Asignamos Usuario, contraseña y roles.

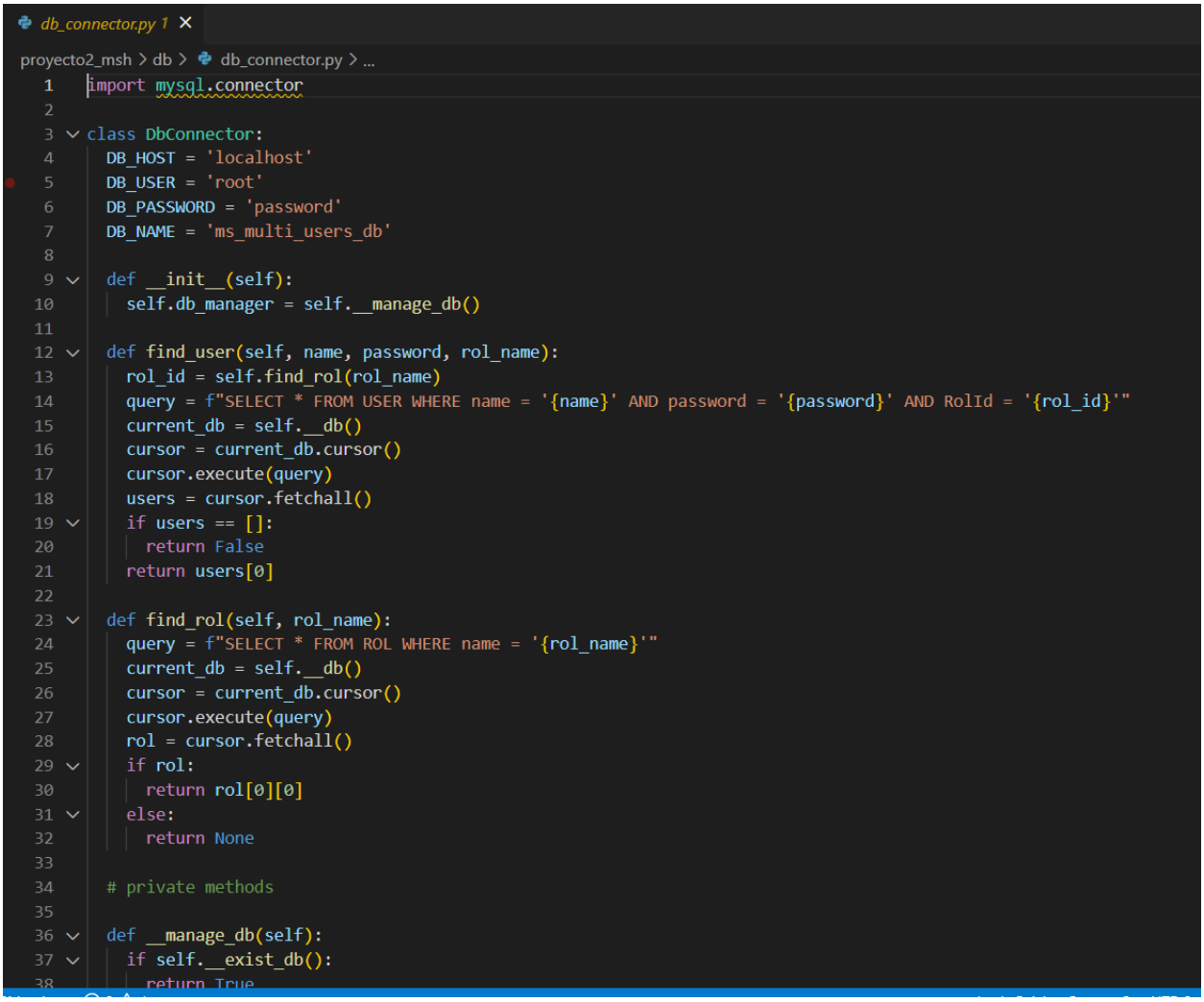


login_controller.py

proyecto2_msh > controllers > login_controller.py > LoginController > login

```python
from db.db_connector import DbConnector

class LoginController:
    def __init__(self, request):
        self.request = request

    # public methods

    def login(self):
        if self.__valid_login():
            return True
        else:
            return False

    # private methods

    def __valid_login(self):
        db = DbConnector()
        users = db.find_user(self.__username(), self.__password(), self.__rol())
        if users:
            return True
        else:
            return False

    def __username(self):
        return self.request.form['username']

    def __password(self):
        return self.request.form['password']

    def __rol(self):
        return self.request.form['rol']

```

- Creación del servidor web:



db_connector.py 1

proyecto2_msh > db > db_connector.py > ...

```python
import mysql.connector

class DbConnector:
    DB_HOST = 'localhost'
    DB_USER = 'root'
    DB_PASSWORD = 'password'
    DB_NAME = 'ms_multi_users_db'

    def __init__(self):
        self.db_manager = self.__manage_db()

    def find_user(self, name, password, rol_name):
        rol_id = self.find_rol(rol_name)
        query = f"SELECT * FROM USER WHERE name = '{name}' AND password = '{password}' AND RolId = '{rol_id}'"
        current_db = self.__db()
        cursor = current_db.cursor()
        cursor.execute(query)
        users = cursor.fetchall()
        if users == []:
            return False
        return users[0]

    def find_rol(self, rol_name):
        query = f"SELECT * FROM ROL WHERE name = '{rol_name}'"
        current_db = self.__db()
        cursor = current_db.cursor()
        cursor.execute(query)
        rol = cursor.fetchall()
        if rol:
            return rol[0][0]
        else:
            return None

    # private methods

    def __manage_db(self):
        if self.__exist_db():
            return True
```

db_connector.py 1

proyecto2_msh > db > db_connector.py > ...

```
38        return True
39      else:
40        self.__create_db()
41
42    def __exist_db(self):
43      current_db = self.__connector()
44      cursor = current_db.cursor()
45      cursor.execute("SHOW DATABASES")
46      databases = cursor.fetchall()
47      if databases == None:
48        return False
49      for db_name in databases:
50        if db_name[0] == self.DB_NAME:
51          return True
52      return False
53
54    def __connector(self):
55      db = mysql.connector.connect(
56        host = self.DB_HOST,
57        user = self.DB_USER,
58        password = self.DB_PASSWORD
59      )
60
61      return db
62
63    def __create_db(self):
64      current_db =  self.__connector()
65      cursor = current_db.cursor()
66      query = f"CREATE DATABASE {self.DB_NAME}"
67      cursor.execute(query)
68
69      current_db = self.__db()
70      cursor = current_db.cursor()
71      self.__create_tables(cursor)
72      self.__insert_rol(current_db, cursor)
73      self.__insert_users(current_db, cursor)
74      return True
```

db_connector.py 1

proyecto2_msh > db > db_connector.py > DbConnector > __connector

```python
        cursor = current_db.cursor()
        query = f"CREATE DATABASE {self.DB_NAME}"
        cursor.execute(query)

        current_db = self.__db()
        cursor = current_db.cursor()
        self.__create_tables(cursor)
        self.__insert_rol(current_db, cursor)
        self.__insert_users(current_db, cursor)
        return True

    def __create_tables(self, cursor):
        cursor.execute(
            "CREATE TABLE Rol (RolId int NOT NULL AUTO_INCREMENT, name varchar
        )
        cursor.execute(
            "CREATE TABLE User (UserId int NOT NULL AUTO_INCREMENT, name varch
        )

    def __insert_rol(self, db, cursor):
        query = "INSERT INTO Rol (name, created_at) VALUES (%s, %s)"
        values = [
            ('Administrador', None),
            ('Operativo', None),
            ('General', None)
        ]
        cursor.executemany(query, values)
        db.commit()

    def __insert_users(self, db, cursor):
        query = "INSERT INTO User (name, password, RolId, created_at) VALUES
        values = [
            ('MatyAdmin','MatyAdmin', self.find_rol('Administrador'), None),
            ('MatyOPerativo','MatyOPerativo', self.find_rol('Operativo'), None
            ('MatyGeneral', 'MatyGeneral', self.find_rol('General'),  None)
        ]
        cursor.executemany(query, values)
        db.commit()

    def __db(self):
        db = mysql.connector.connect(
            host = self.DB_HOST,
            user = self.DB_USER,
            password = self.DB_PASSWORD,
            database = self.DB_NAME
        )
        return db
```

Agregamos las vistas para:

Home

home.html

proyecto2_msh > views > home.html > html > head > style

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lato">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<style>
body,h1,h2,h3,h4,h5,h6 {font-family: "Lato", sans-serif;}
body, html {
  height: 100%;
  color: #777;
  line-height: 1.8;
}

/* Create a Parallax Effect */
.bgimg-1, .bgimg-2, .bgimg-3 {
  background-attachment: fixed;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}

/* First image (Logo. Full height) */
.bgimg-1 {
  background-image: url('/w3images/parallax1.jpg');
  min-height: 100%;
}

.w3-wide {letter-spacing: 10px;}
.w3-hover-opacity {cursor: pointer;}

/* Turn off parallax scrolling for tablets and phones */
@media only screen and (max-device-width: 1600px) {
  .bgimg-1, .bgimg-2, .bgimg-3 {
    background-attachment: scroll;
    min-height: 400px;
  }
}
</style>
</head>
<body>


<div class="bgimg-1 w3-display-container w3-opacity-min" id="home">
  <div class="w3-display-middle" style="white-space:nowrap;">
    <a href="login" target="_blank" class="w3-center w3-padding-large w3-black w3-xlarge w3-wide w3-animate-opacity"> LOGIN</a>
  </div>
</div>

<!-- Footer -->
<footer class="w3-center w3-black w3-padding-64 w3-opacity w3-hover-opacity-off">
  <div class="w3-xlarge w3-section">
    <i class="fa fa-twitter w3-hover-opacity"></i>
    <i class="fa fa-linkedin w3-hover-opacity"></i>
  </div>
</footer>

</body>
</html>
```

Inicio de pantalla exitoso:

home.html | login_success.html

proyecto2_msh > views > login_success.html > ...

```
<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <body class="">
  <h2>Tu Login ha sido exitoso</h2>
    <div class="w3-panel w3-pale-green w3-border">
      <h3>Felicidades!</h3>
      <p>Tus datos de inicio de session son correctos.</p>
    </div>
  </body>
</html>

<a href="login" target="_blank">Volver al Login</a>
```

Inicio de pantalla error:

home.html | login_try_again.html

proyecto2_msh > views > login_try_again.html > ...

```
<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <body class="">
  <h2>Tu Login no ha sido exitoso</h2>
    <div class="w3-panel w3-pale-red w3-border">
      <h3>Error!</h3>
      <p>Tus datos de inicio de session no correctos.</p>
    </div>
  </body>
</html>

<a href="login" target="_blank">Volver al Login</a>
```

Inicio de sesión:

home.html | login.html

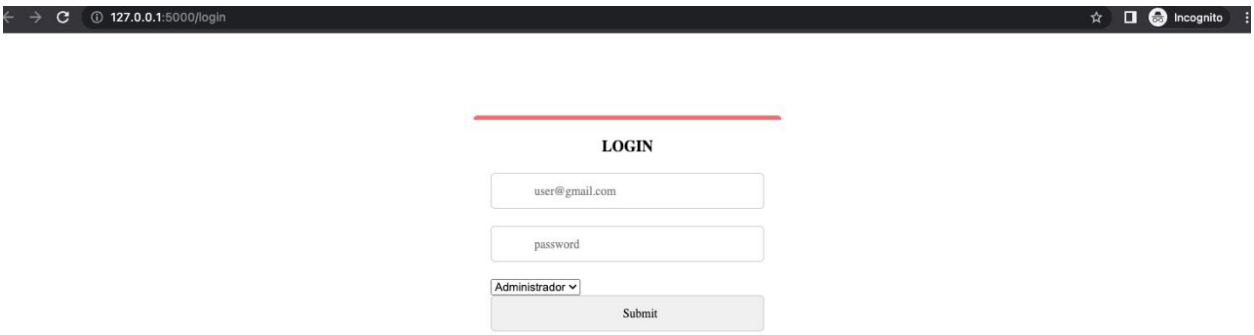
proyecto2_msh > views > login.html > ...

```
1  <!--  how to use css in python_ flask-->
2  <!DOCTYPE html>
3  <html lang="en">
4    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
5    <head>
6      <!--  Load CSS in Flask with html-->
7        <link rel="stylesheet" href='/static_files/main.css' />
8      <!--  Load JavaScript in Flask with html-->
9    </head>
10   <body>
11     <div class="logo"></div>
12     <div class="login-block">
13       <form action="login" method="post">
14         <h1>Login</h1>
15         <input type="text" value="" placeholder="user@gmail.com" name="username" id="username" />
16         <input type="password" value="" placeholder="password" name="password" id="password" />
17         <select name="rol" id="rol">
18           <option value="Administrador">Administrador</option>
19           <option value="Operativo">Operativo</option>
20           <option value="General">General</option>
21         </select>
22         <input type="submit" value="Submit">
23       </form>
24     </div>
25   </body>
26 </html>
```

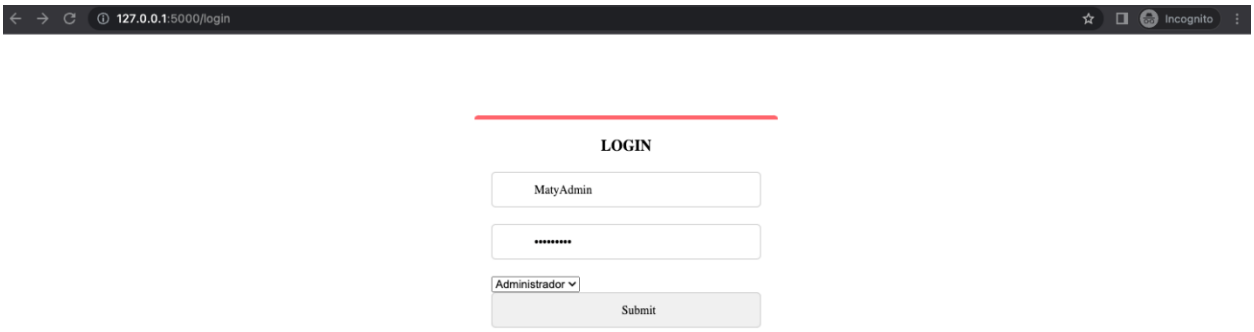
Pantalla de inicio de sesión:

LOGIN

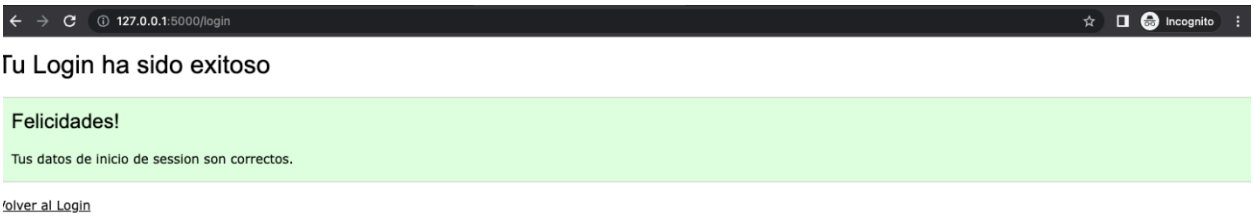
Ingresamos al inicio de sesión:



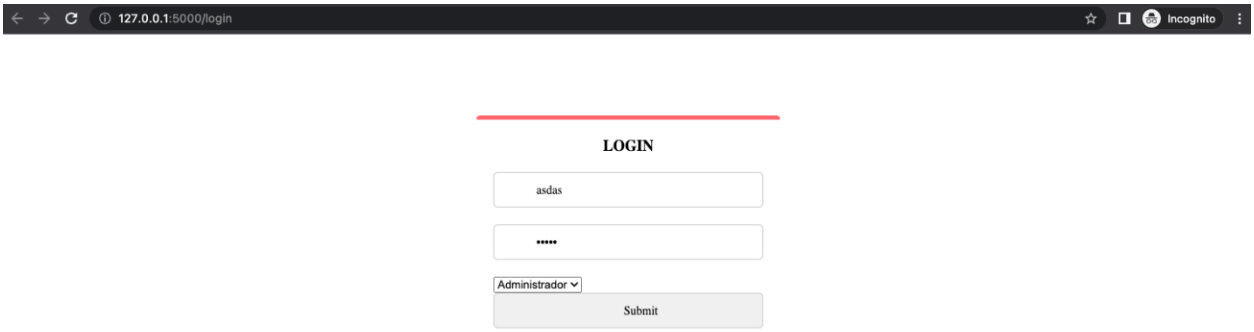
Ingresamos con el usuario y contraseña asignados en la base de datos:



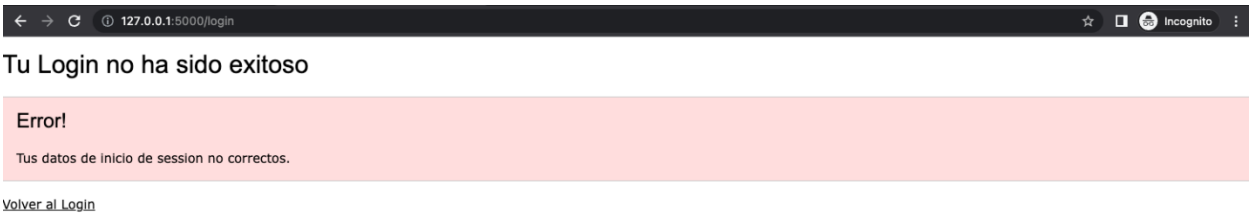
Comprobamos que es correcto y regresamos al inicio:



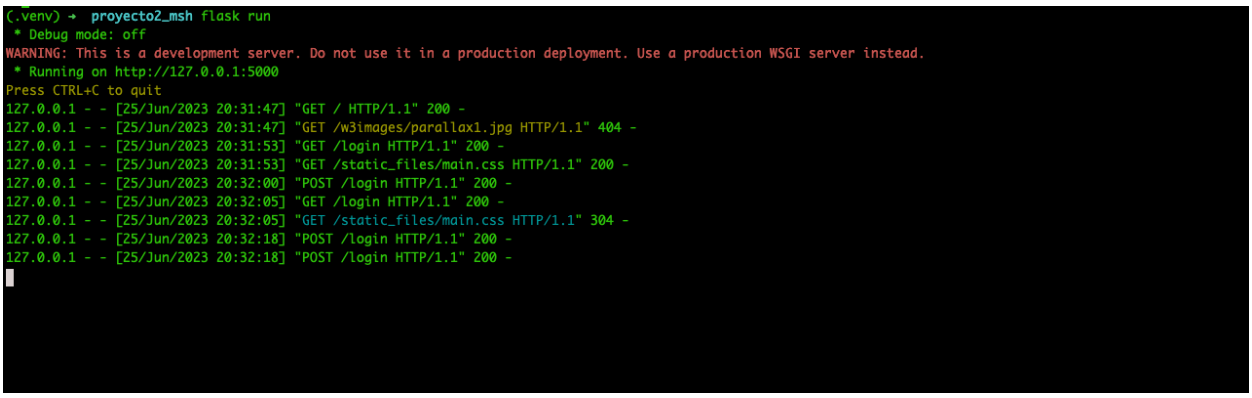
Intentamos ingresar con un usuario no existente en la base de datos:



Recibimos un error y regresamos al inicio:



Corroboramos de que no existen errores realizados en el inicio de sesión:



Durante el proyecto tuve varios errores, como lazy load el cual estaba ejecutando en segundo plano el cursor ya que al principio lo había duplicado, otro error fue callable, no se ejecutaba el cursor.

La definición de lo roles en la creación de las tablas también me encontré con errores al igual que al agregar a los usuarios.

La creación del front fue en parte con ayuda de paginas externas, ya que no estoy muy familiarizada con HTML.

Conclusión

El proyecto fue un reto, tenia conocimiento de bases de datos lo que me facilito realizar la primera parte del proyecto, pero la segunda parte fue mas difícil, porque fue un código largo en el cual tuve que enlazar la base de datos con el servidor para que el cliente tenga visibilidad. El usar Flask y realizar las conexiones entre Mysql y Python fueron todo un reto, también se tiene que asegurar de que el usuario tenga los permisos adecuados para acceder a la base de datos y ejecutar las consultas necesarias.

Es importante revisar y depurar cuidadosamente el código para identificar y corregir posibles errores.

Referencias

School, W. (s.f.). W3 School. Obtenido de Python MySQL Create Database: https://www.w3schools.com/python/python_mysql_create_db.asp

Microsoft Azure (2017). Procesamiento por lotes Haga clic para ver más opciones [sitio web.] Recuperado de https://docs.microsoft.com/es-es/azure/architecture/data-guide/big-data/batch-processing

Yuson, P. (2017). Types of Mainframe Processing: Batch and Online Haga clic para ver más opciones [sitio web]. Recuperado de http://conceptsolutionsbc.com/wordpress/types-of-mainframe-processing-batch-and-online/