

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

SISTEMAS OPERATIVOS I

SECCIÓN N



INFORME Y VIDEO DE CONFIGURACIONES

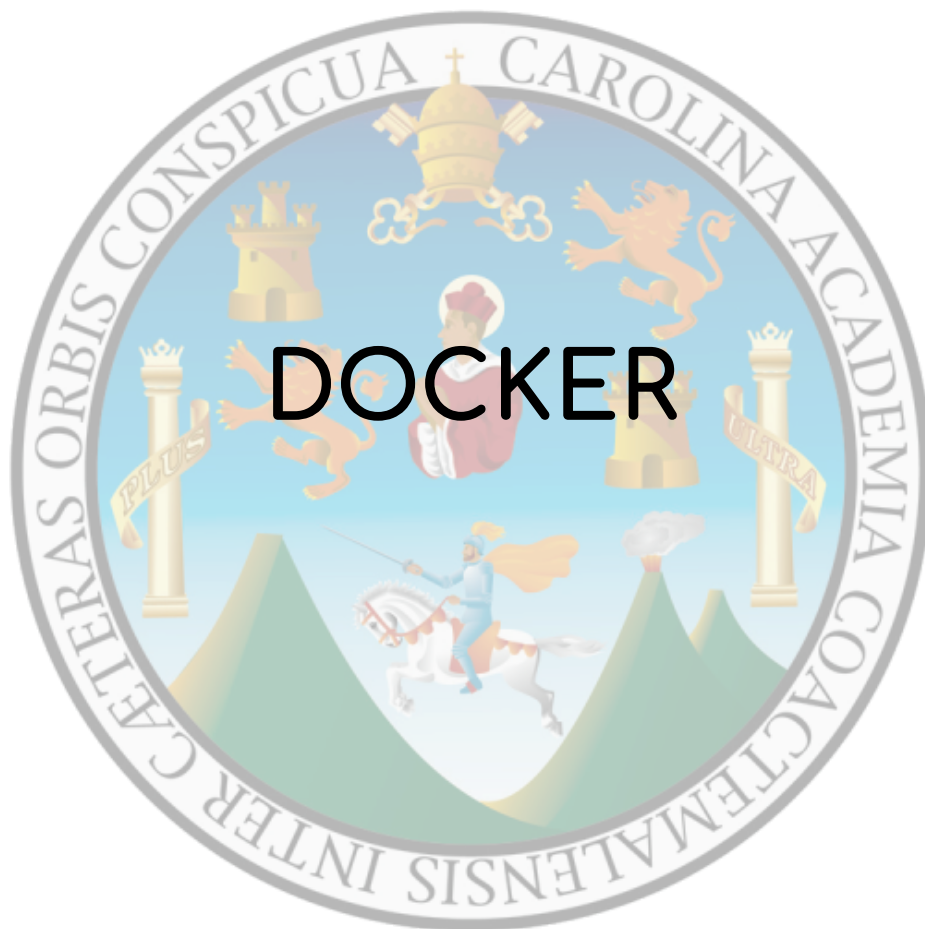
Estudiante	Carné No.
Harry Aaron Gómez Sanic	202103718

Guatemala, 29 de Diciembre de 2023

VIDEO DE CONFIGURACIONES

<https://youtu.be/UuUZQArItAg>



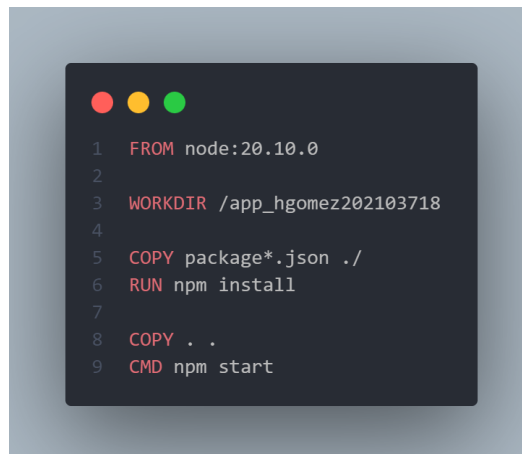


Se utilizó docker compose para almacenar los dos servicios siendo estos una base de datos y un backend con express

1. INSTALACIÓN DE DOCKER

- Descargar Docker Desktop para Windows desde el sitio web oficial:
Docker Desktop for Windows
- Ejecutar el instalador y seguir los pasos de instalación.
- Durante la instalación, seleccionar la opción para usar contenedores de Windows, si es aplicable.
- Después de la instalación, reiniciar el sistema si llega a pedirlo.

2. CONFIGURACIÓN DE ARCHIVO *DOCKERFILE*



- **FROM node:20.10.0:** Indica que esta imagen se basará en la imagen oficial de Node.js en la versión 20.10.0.
- **WORKDIR /app_hgomez202103718:** Establece el directorio de trabajo actual dentro del contenedor como "/app_hgomez202103718".
- **COPY package.json ./*** Copia los archivos package.json y package-lock.json (si existen) del directorio local al directorio de trabajo del contenedor.
- **RUN npm install:** Ejecuta el comando npm install dentro del contenedor. Este comando instala las dependencias especificadas en el archivo package.json en el directorio de trabajo.
- **COPY . .:** Copia todos los archivos del directorio local al directorio de trabajo del contenedor. Esto incluye el código fuente de la aplicación.
- **CMD npm start:** Especifica el comando por defecto que se ejecutará cuando el contenedor se inicie. En este caso, se ejecuta npm start, que probablemente esté definido en el archivo package.json como el script de inicio de la aplicación.

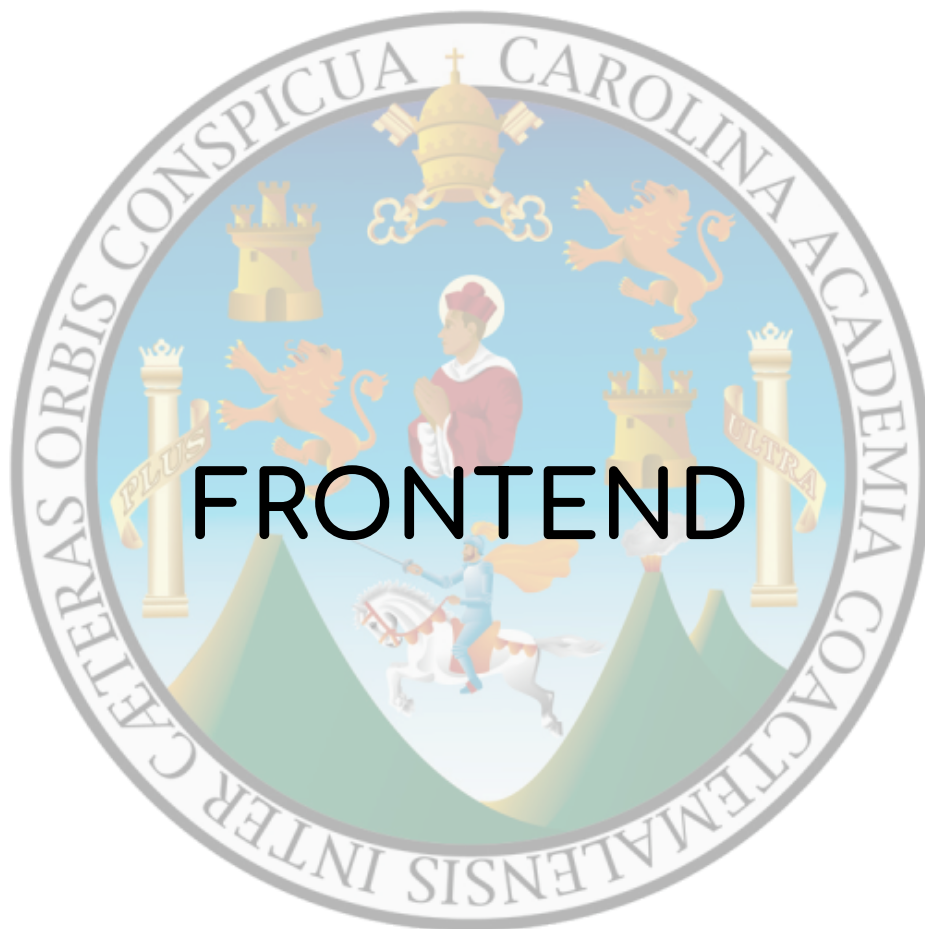
3. CONFIGURACIÓN DE ARCHIVO *DOCKER-COMPOSE.YML*

Acá se colocan los servicios que se usarán en el contenedor, en este caso solo se usarán dos servicios.

- **Mysql:** Se utilizó una imagen de docker para la base de datos
- **NodeJs:** Se construye a partir del archivo Dockerfile, cargando el comando build y sus dependencias que se refiere a la base de datos ya que hay una conexión entre los dos contenedores.



```
1  version: '3.8'
2
3  services:
4    mysql_hgomez202103718:
5      image: mysql
6      environment:
7        - MYSQL_ROOT_PASSWORD=12345
8        - MYSQL_DATABASE=hgomez202103718
9      ports:
10       - 3307:3306 #PUERTO_LOCAL:PUERTO_CONTENEDOR
11
12    app_hgomez202103718:
13      build: .
14      command: npm start
15      depends_on:
16        - mysql_hgomez202103718
17      ports:
18        - 3000:3000
```



FRONTEND

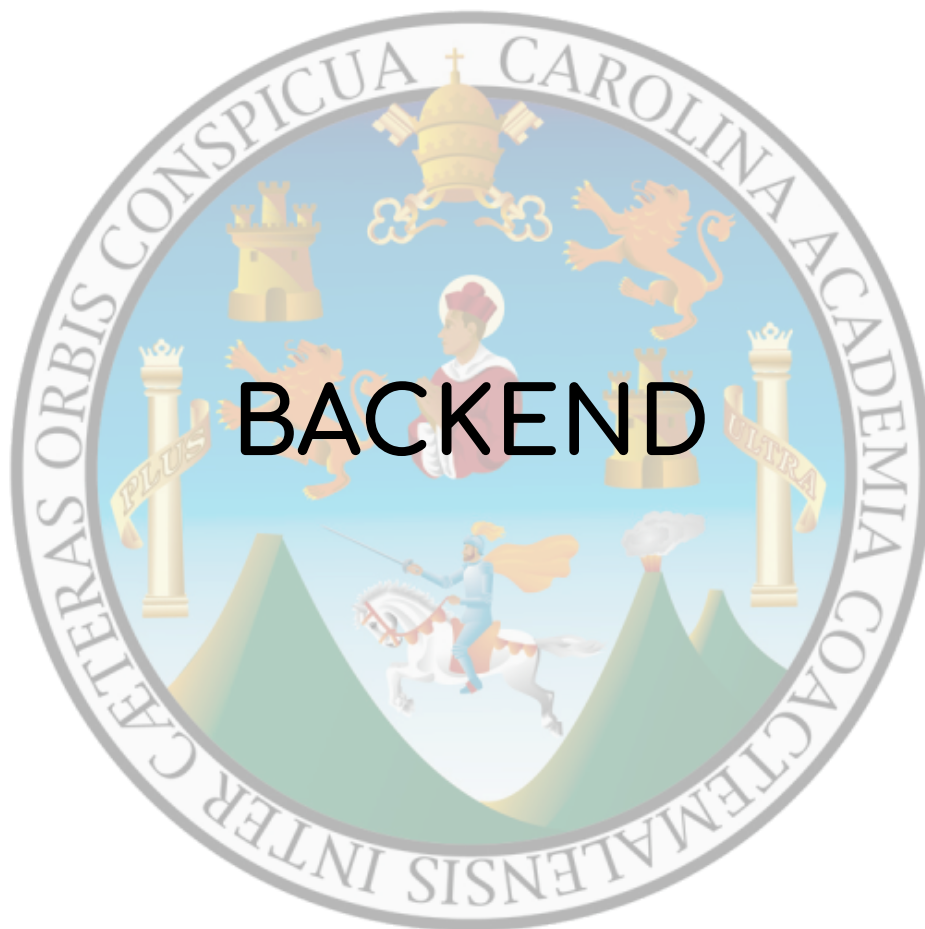
Se utilizó un archivo html y bootstrap para la creación de un login básico

1. Importación de bootstrap5 al documento, dentro de la etiqueta head.

```
1 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QgpjLIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOMLASJC" crossorigin="anonymous">
2 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-Mrcw6ZMFY1zcLA8N1+NtUvF0sA7MsXsP1UyJoMp4YLEUUNSFAP+JcXn/tWTIaxVXM" crossorigin="anonymous"></script>
3 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoLX15PILFhosVnuBq5LC7Qb9DXgDA91+tQ8Z33iWAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous"></script>
4 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWLc2A14u+LmgxFKTRICfu01TxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF" crossorigin="anonymous"></script>
```

2. Creación de form con su método post y nombre de la ruta a enviar la información.


```
1 <form action="/login" method="post" class="d-flex flex-column justify-content-center align-items-center">
2   <div class="container mt-4">
3     <label for="username" class="text-light">Username</label>
4     <input id="username" name="username" class="form-control mt-3 h-75"/>
5   </div>
6
7   <div class="container mt-5">
8     <label for="password" class="text-light">Password</label>
9     <input type="password" id="password" name="password" class="form-control mt-3 h-75"/>
10  </div>
11
12  <button type="submit" name="loginButton" class="btn btn-primary mt-5 mb-4">Login</button>
13 </form>
```

Se utilizó Express para realizar las peticiones del archivo html hacia el backend y también createPool para la conexión hacia la base de datos y un path para mostrar el frontend en el puerto 3000.

1. Conexión a la base de datos

- Importacion: createPool de “mysql2/promise”
- host: Nombre del contenedor que almacena la base de datos
- database: Nombre asignado a la base de datos
- port: puerto donde se aloja el servicio.
- connectionLimit: Cantidad de conexiones permitidas a la vez.



```
1  const pool = createPool({
2    host: "mysqldb_hgomez202103718",
3    user: "root",
4    password: "12345",
5    database: "hgomez202103718",
6    port: 3306, // Se coloca el puerto del contenedor
7    connectionLimit: 10,
8  });
```

2. Visualización de frontend por medio de puerto 3000



```
1  const app = express();
2  app.use(express.urlencoded({ extended: true }));
3
4  // Ruta a la carpeta 'frontend'
5  const frontendPath = path.join(__dirname, 'frontend');
6  // Configuración para servir archivos estáticos desde la carpeta 'frontend'
7  app.use(express.static(frontendPath));
8
9  // Ruta para el archivo index.html
10 app.get('/', (req, res) => {
11   res.sendFile(path.join(frontendPath, 'index.html'));
12 });
```

3. Creación de ruta “/login” para obtener los datos desde el archivo index.html y realizar la consulta a la base de datos.

- Línea 3: Obtiene el cuerpo de la petición en dos constantes
- Línea 9: hace la consulta para verificar si existe ese username y password y almacena en la constante result

```
1 // Ruta para el login
2 app.post('/login', async (req, res) => {
3   const { username, password } = req.body;
4
5   console.log(">>>+> Datos recibidos: ");
6   console.log(username, password);
7
8   try {
9     const [result] = await pool.query(
10       'SELECT * FROM users WHERE username = ? AND password = ?',
11       [username, password]
12     );
13
14     if (result.length > 0) {
15       res.send("Login correcto!");
16     } else {
17       res.send("Login incorrecto!");
18     }
19   } catch (error) {
20     res.send("Ocurrió un error!");
21   }
22 });
```