UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
SISTEMAS OPERATIVOS 1
SECCIÓN “N”
ING. JOAQUIN ADOLFO GUERRERO
AUX. RANDY FERNANDO JUÁREZ
SEGUNDO SEMESTRE 2023

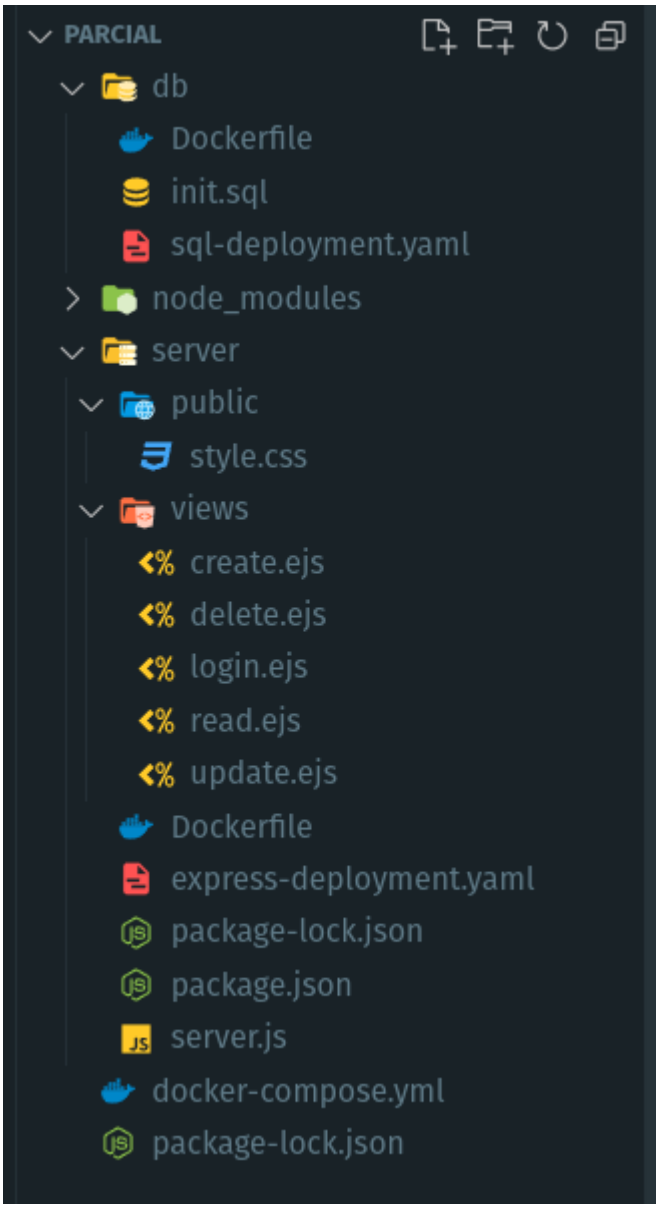
Examen Final

Josue Rolando Gramajo Roldan
202000895
02/01/2024

Serie I (9 Pts.). Realizar una aplicación web básica que permita gestionar usuarios y contraseñas: crear, modificar, eliminar, buscar y realizar logins a la aplicación. Almacenar los datos de usuario en una base de datos relacional (SQL Server, MySQL, etc.)

Separar la arquitectura de aplicación y base de datos en Kubernetes a través del uso de nodos (contenedores).

Estructura del proyecto:



NodeJS:

Configuración **express-deployment.yaml**:

server > express-deployment.yaml

```yaml
# app-deployment-service.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: node-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: node-app
  template:
    metadata:
      labels:
        app: node-app
    spec:
      containers:
      - name: node-app
        image: ijosuer/express-app:latest
        ports:
        - containerPort: 5000

---
apiVersion: v1
kind: Service
metadata:
  name: node-app-service
spec:
  selector:
    app: node-app
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
  type: NodePort
```

Código main:

```
server > server.js > ...
const express = require('express');
const mysql = require('mysql2');
const bodyParser = require('body-parser');

const app = express();
const port = 5000;
app.use(bodyParser.urlencoded({ extended: true }), express.json());
app.use(express.static('public'));
app.set('view engine', 'ejs');

> app.get('/', (req, res) => {…
  });

> app.post('/login', (req, res) => {…
  });

> app.get('/create', (req, res) => {…
  });

> app.post('/create', (req, res) => {…
  });

> app.get('/update', (req, res) => {…
  });

> app.post('/update', (req, res) => {…
  });

> app.get('/delete', (req, res) => {…
  });

> app.post('/delete', (req, res) => {…
  });


> app.listen(port, () => {…
  });
```

Petición **Create** user:

```
app.post('/create', (req, res) => {
  const { username, password } = req.body;

   const db = mysql.createConnection({
    host: 'mysql-service',
    user: 'root',
    password: 'josue',
    database: 'jgramajo'
  });

  db.on('connect', () => {
    console.log('Connected to MySQL database');
  });

> db.on('error', (err) => {…
  });

  const query = 'INSERT INTO users (username, password) VALUES (?, ?)';
  db.query(query, [username, password], (err, result) => {
    if (err) {
      res.json({ success: false, message: 'Error creating user.' });
    } else {
      res.json({ success: true, message: 'User created successfully.' });
    }
  });
});
```

Petición **read** user:

```
app.post('/login', (req, res) => {
  const { username, password } = req.body;

  const db = mysql.createConnection({
    host: 'mysql-service',
    user: 'root',
    password: 'josue',
    database: 'jgramajo'
  });

  db.on('connect', () => {
    console.log('Connected to MySQL database');
  });

  db.on('error', (err) => {…
  });

  const query = 'SELECT * FROM users WHERE username = ? AND password = ?';
  console.log(req.body);

  db.query(query, [username, password], (err, results) => {
    if (err) throw err;

    if (results.length > 0) {
      res.json({ success: true, message: 'Login successful' });
    } else {
      res.json({ success: false, message: 'Login failed. Please check your username and password.' });
    }

    // Close the connection after the query is executed
    db.end((err) => {…
    });
  });
});
```

Petición **Update** user:

```
app.post('/update', (req, res) => {
  const { username, oldPassword, newPassword } = req.body;
  console.log("body: ",req.body);

   // Move database connection setup inside the login endpoint
   const db = mysql.createConnection({
    host: 'mysql-service',
    user: 'root',
    password: 'josue',
    database: 'jgramajo'
  });

  db.on('connect', () => {…
  });

  db.on('error', (err) => {…
  });

  // Consulta SQL de actualización
  const updateQuery = `UPDATE users
                       SET password = ?
                       WHERE username = ? AND password = ?`;
  // Ejecuta la consulta
  db.query(updateQuery, [newPassword, username, oldPassword], (err, results) => {
    console.log('user:',username)
    if (err) {
      console.error(err);
      res.status(500).send('Error al actualizar el usuario.');
    } else {
      if (results.affectedRows > 0) {
        res.json({ message: 'Usuario actualizado correctamente.' });
      } else {
        res.json({ error: 'Usuario no encontrado o la contraseña anterior es incorrecta.' });
      }
    }
  });
});
```

Petición **Delete** user:

```
app.post('/delete', (req, res) => {
  const { username } = req.body;

   const db = mysql.createConnection({
    host: 'mysql-service',
    user: 'root',
    password: 'josue',
    database: 'jgramajo'
  });

  db.on('connect', () => {…
  });

  db.on('error', (err) => {…
  });

  // Consulta SQL para eliminar un usuario
  const query = 'DELETE FROM users WHERE username = ?';

  db.query(query, [username], (err, result) => {
    if (err) {
      res.json({ success: false, message: 'Error eliminado el usuario.' });
    } else {
      res.json({ success: true, message: 'User eliminado correctamente.' });
    }
  });
});
```

MySQL (Base de datos):

Configuración **sql-deployment.yaml**:

```
db > sql-deployment.yaml
    selector:
      matchLabels:
        app: mysql
    template:
      metadata:
        labels:
          app: mysql
      spec:
        containers:
        - name: mysql
          image: ijosuer/mysql-app:latest
          env:
          - name: MYSQL_ROOT_PASSWORD
            value: josue

          ports:
          - containerPort: 3306
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
  labels:
    app: mysql
spec:
  selector:
    app: mysql
  ports:
  - name: mysql
    protocol: TCP
    port: 3306
    targetPort: 3306
```

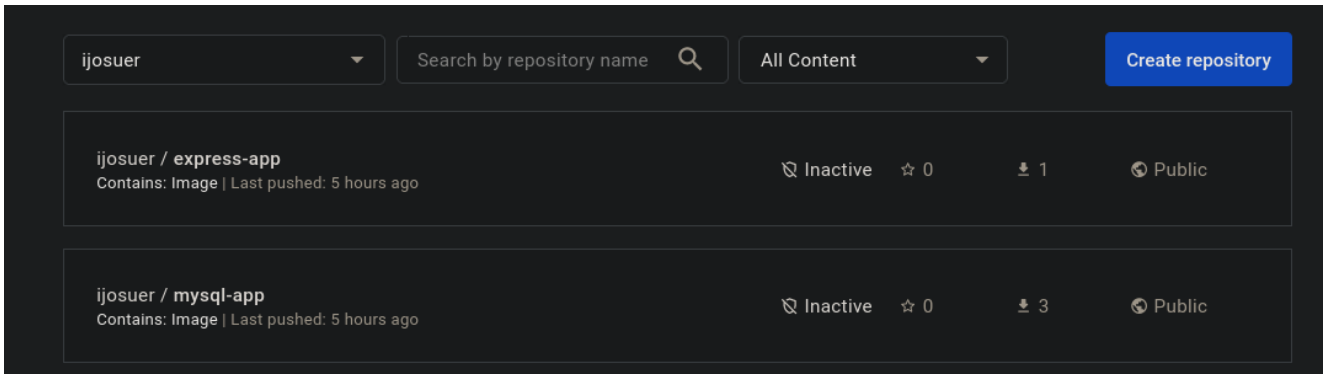
Configuración **init.sql**:

```sql
db > init.sql
CREATE DATABASE jgramajo;

USE jgramajo;

CREATE TABLE IF NOT EXISTS users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL
);

INSERT INTO users (username, password) VALUES ('usuario1', 'contra1');
```

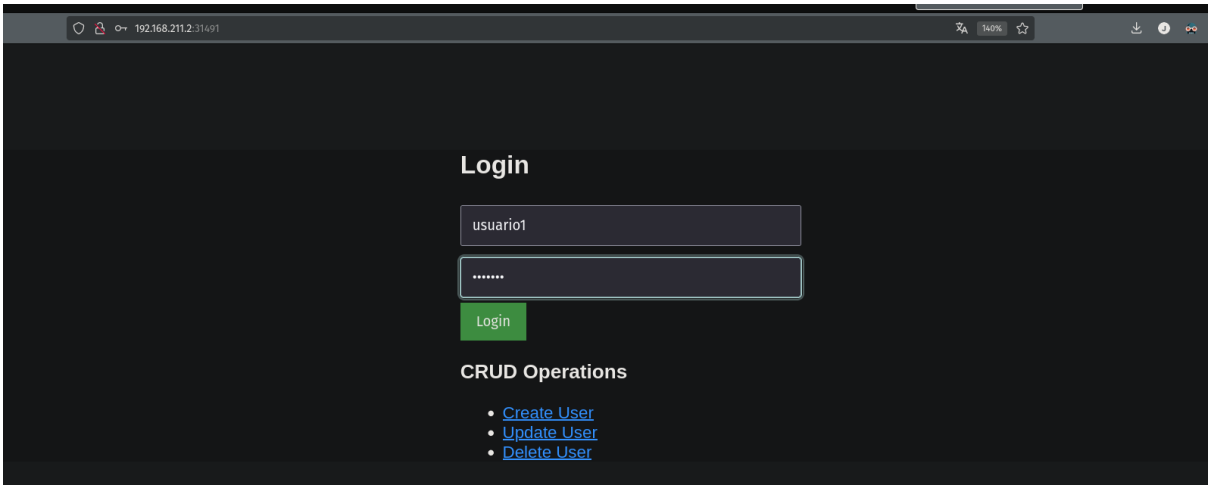
Configuración **Dockerfile**:

```dockerfile
db > Dockerfile
# Dockerfile

# Utiliza la imagen oficial de MySQL como base
FROM mysql:8.0

ENV MYSQL_ROOT_PASSWORD=josue

# Copia el script de inicialización a la carpeta /docker-entrypoint-initdb.d/
COPY ./init.sql /docker-entrypoint-initdb.d/
```

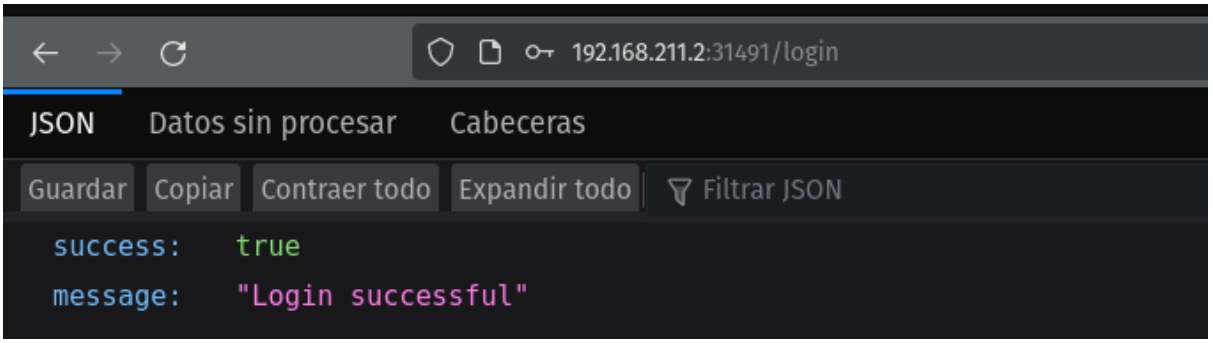
Dockerhub (Contenedores):



EJECUCIÓN:



User: usuario1
Pwd: contra1



LINK VIDEO

https://youtu.be/J7-haZMKAn4