

Tecnología ANÁLISIS DESARROLLO SOFTWARE

Ficha ADSO: 2758368

Aprendices:

Nelly Llasmin Aguilar Arbeláez

Wilson Armando Puentes Padilla

**Definir estándares de codificación de acuerdo a plataforma de desarrollo
elegida**

Instructor

Ricardo Alfonso González Vargas

SERVICIO NACIONAL DE APRENDIZAJE

Centro de Comercio y Turismo - Regional Quindío

Agosto 2024

INTRODUCCIÓN

La codificación es el proceso de usar lenguajes de programación para dar instrucciones a una computadora; los programadores utilizan éstos, para crear aplicaciones, juegos, sitios web y demás aplicaciones que las personas usan a diario.

OBJETIVO

- Mostrar los avances en la codificación del proyecto software formativo para la miscelánea “A y F”.

MÓDULOS SOFTWARE A IMPLEMENTAR

Codificación del “main” y del “logo” de la miscelánea:

```
# stilos.css x index.html
C: > Users > Administrador > Downloads > maquetacion_Proyecto > # stilos.css > ...

1 |
2 | .main{
3 |     width: 100%;
4 |     background: url(Ima-p3.jpg);
5 |     background-position: center;
6 |     background-size: cover;
7 |     height: 100vh;
8 |     justify-content: center;
9 | }
10 |
11 |
12 | .nabvar{
13 |     width: 100%;
14 |     height: 75px;
15 |     margin: auto;
16 | }
17 |
18 |
19 | .icon{
20 |
21 |     width: 100%;
22 |     float: left;
23 |     height: 120px;
24 | }
25 |
26 |
27 | .logo{
28 |
29 |     color: rgba(19, 3, 88, 0.733);
30 |     font-size: 100px;
31 |     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
```

Codificación del “menú” y “login” de la página:

```
# stilos.css x index.html
C: > Users > Administrador > Downloads > ma

35 |     margin-top: 0%;
36 | }
37 |
38 | /* parametros lista */
39 |
40 | .menu ul {
41 |     list-style: none;
42 |     margin: 0;
43 |     padding: 0;
44 | }
45 |
46 | .menu ul li {
47 |     display: inline-block;
48 |     margin-right: 20px;
49 | }
50 |
51 | .menu ul li:last-child {
52 |     margin-right: 0;
53 | }
54 |
55 | .menu ul li a {
56 |     color: white;
57 |     text-decoration: none;
58 | }
59 |
60 |
61 | /* menu de login */
62 |
63 | .login-container {
64 |     width: 300px;
65 |     margin: 0 auto;
```

```
# stilos.css x index.html
C: > Users > Administrador > Downloads > maquetacion_Proyecto > # stilos.css
59
60
61 /* menu de login */
62
63 .login-container {
64     width: 300px;
65     margin: 0 auto;
66     text-align: center;
67 }
68
69 .login-container h2 {
70     color: #333; /* Color de texto */
71     font-size: 24px; /* Tamaño de letra */
72 }
73
74 .input-group {
75     margin-bottom: 15px;
76     text-align: left;
77 }
78
79 .input-group label {
80     display: block;
81     margin-bottom: 5px;
82     color: #333; /* Color de texto */
83 }
84
85 .input-group input {
86     width: 100%;
87     padding: 10px;
88     border: 1px solid #ccc;
89     border-radius: 4px;
```

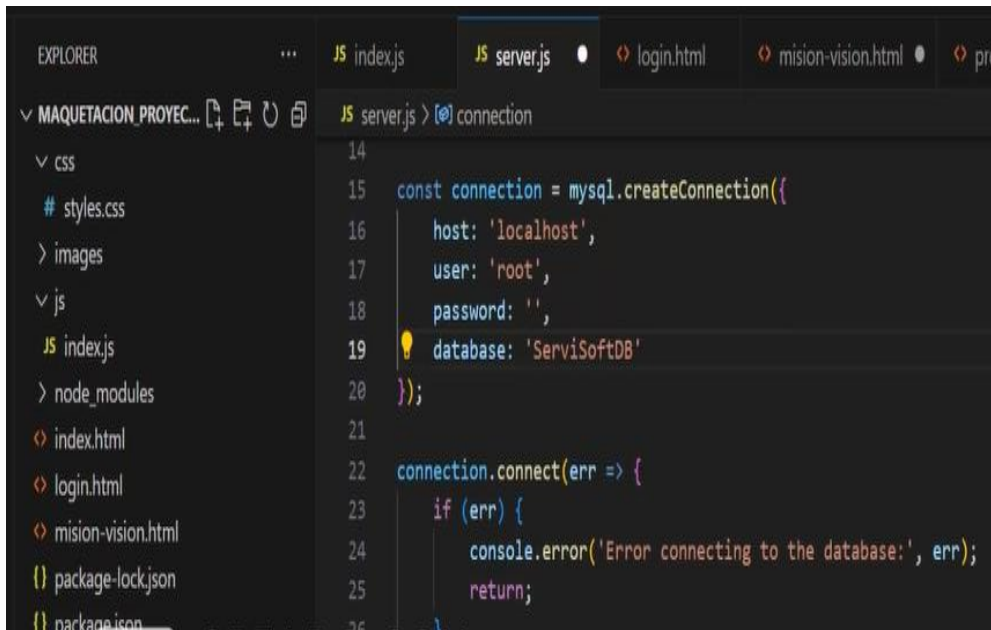
```
# stilos.css x index.html
C: > Users > Administrador > Downloads > maquetacion_Proyecto > # stilos.css
78
79 .input-group label {
80     display: block;
81     margin-bottom: 5px;
82     color: #333; /* Color de texto */
83 }
84
85 .input-group input {
86     width: 100%;
87     padding: 10px;
88     border: 1px solid #ccc;
89     border-radius: 4px;
90     font-size: 16px; /* Tamaño de letra */
91 }
92
93 button {
94     width: 100%;
95     padding: 10px;
96     border: none;
97     border-radius: 4px;
98     background-color: #007bff;
99     color: white; /* Color de texto */
100    font-size: 16px; /* Tamaño de letra */
101    cursor: pointer;
102    transition: background-color 0.3s;
103 }
104
105 button:hover {
106     background-color: #0056b3;
107 }
```

Codificación de conexión a base de datos de “Mysql”:

```
EXPLORER JS index.js JS server.js login.html mision-vision.html
MAQUETACION_PROYEC...
  css
  # styles.css
  > images
  > js
  JS index.js
  > node_modules
  login.html
  mision-vision.html
  package-lock.json
  > server.js

JS server.js > connection
1  const express = require('express');
2  const mysql = require('mysql');
3  const bodyParser = require('body-parser');
4  const cors = require('cors');
5
6  const app = express();
7  const port = 3000;
8
9
10 app.use(cors());
11 app.use(bodyParser.urlencoded({ extended: true }));
12 app.use(bodyParser.json());
```

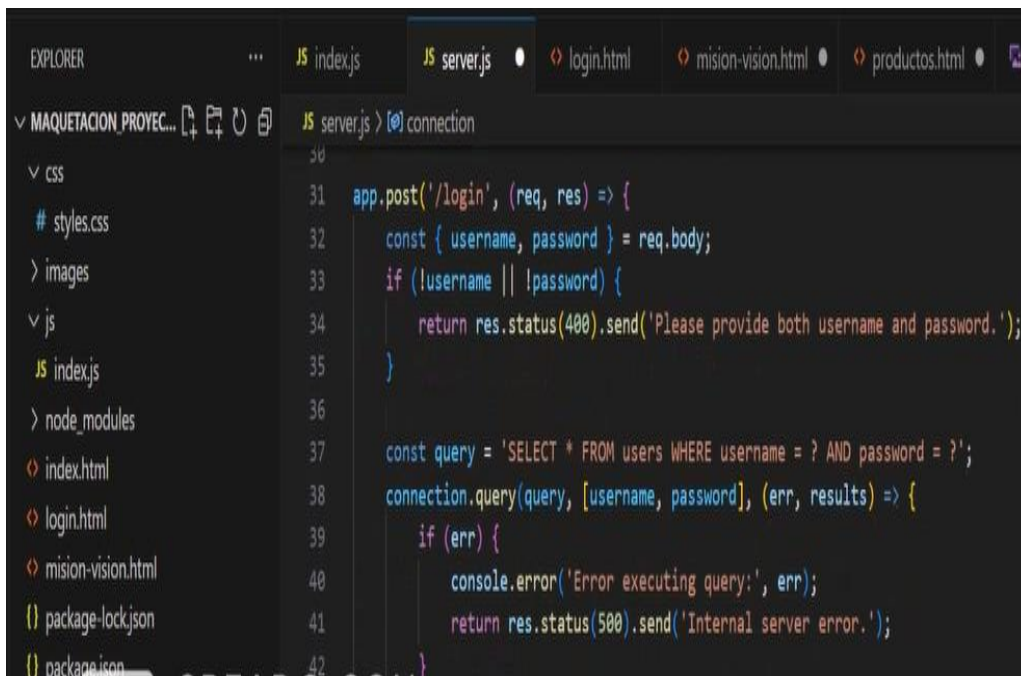
Codificación a bases de datos y generación de “error conectando a base de datos”:



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure for 'MAQUETACION_PROYEC...'. The file explorer includes folders for 'css', 'images', and 'js', and files like 'index.html', 'login.html', 'mision-vision.html', 'package-lock.json', and 'package.json'. The main editor window is open to 'server.js', showing a MySQL connection configuration. The code defines a connection object with host 'localhost', user 'root', an empty password, and database 'ServiSoftDB'. It then attempts to connect and handles errors by logging a message to the console.

```
14
15 const connection = mysql.createConnection({
16   host: 'localhost',
17   user: 'root',
18   password: '',
19   database: 'ServiSoftDB'
20 });
21
22 connection.connect(err => {
23   if (err) {
24     console.error('Error connecting to the database:', err);
25     return;
26   }
27 })
```

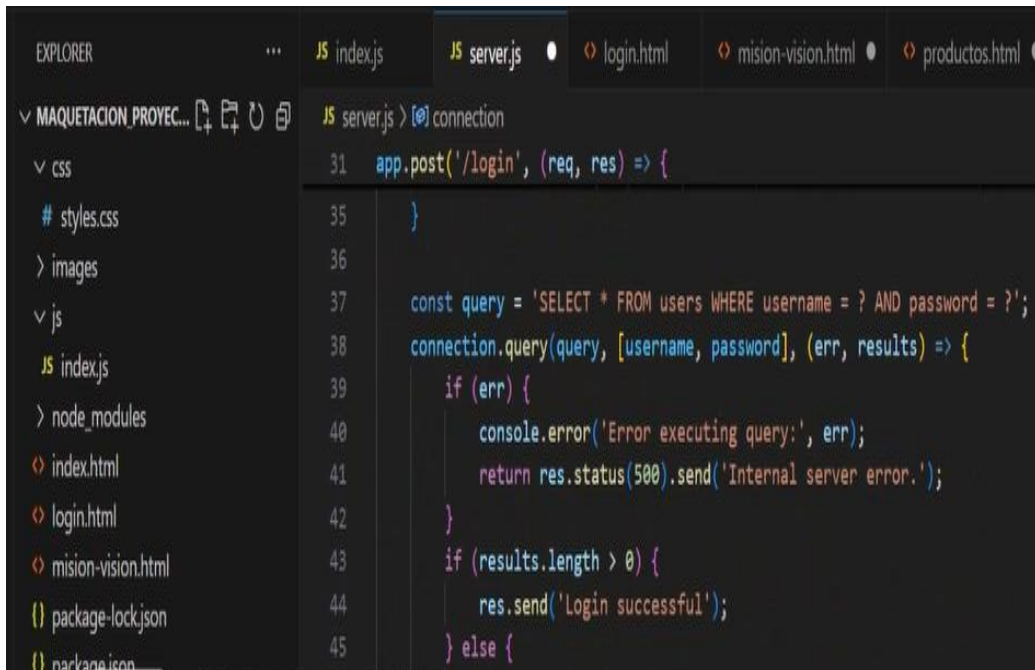
Configuración del “usuario” y “contraseña” de ingreso:



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the same project structure. The main editor window is open to 'server.js', showing the configuration for the login endpoint. The code uses Express.js to handle a POST request to '/login'. It checks if the username and password are provided in the request body. If not, it returns a 400 status with a message. If provided, it executes a SQL query to find the user by username and password. If the query fails, it returns a 500 status with an internal server error message. If successful, it would typically return the user details, but the code is partially obscured at the bottom.

```
31 app.post('/login', (req, res) => {
32   const { username, password } = req.body;
33   if (!username || !password) {
34     return res.status(400).send('Please provide both username and password.');
```

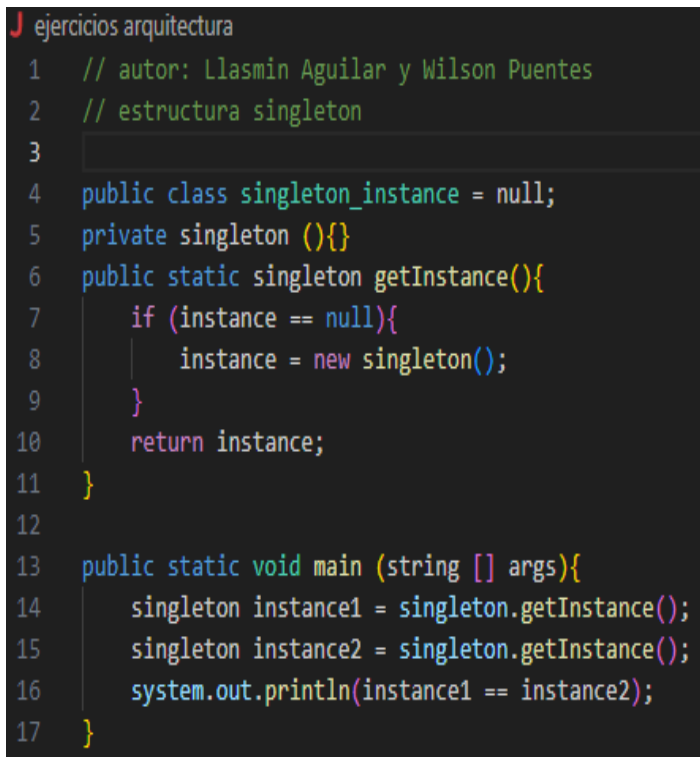
Codificación de “error de ejecución”:



```
EXPLORER    ...    JS index.js    JS server.js    login.html    mision-vision.html    productos.html

MAQUETACION_PROYEC...  JS server.js > connection
31  app.post('/login', (req, res) => {
35      }
36
37  const query = 'SELECT * FROM users WHERE username = ? AND password = ?';
38  connection.query(query, [username, password], (err, results) => {
39      if (err) {
40          console.error('Error executing query:', err);
41          return res.status(500).send('Internal server error.');
```

PATRÓN DE CREACIÓN - SINGLETON



```
J ejercicios arquitectura
1  // autor: Llasmin Aguilar y Wilson Puentes
2  // estructura singleton
3
4  public class singleton_instance = null;
5  private singleton (){}
6  public static singleton getInstance(){
7      if (instance == null){
8          instance = new singleton();
9      }
10     return instance;
11 }
12
13 public static void main (string [] args){
14     singleton instance1 = singleton.getInstance();
15     singleton instance2 = singleton.getInstance();
16     system.out.println(instance1 == instance2);
17 }
```

PATRÓN DE ESTRUCTURA - ADAPTER

```
ejercicio adapter
1 // autor: Llasmin Aguilar y Wilson Puentes
2 // estructura adapter
3
4 public class Rectangle {
5     public double calculateArea (double length, double width) {
6         return length * width;
7     }
8 }
9
10 public interface shape {
11     double calculateArea (double side);
12 }
13
14 public class SquateAdapter implements Shape {
15     private Rectangle rectangle;
16     public SquateAdapter (Rectangle rectangle){
17         this.rectangle = rectangle;
18     }
19     public double calculateArea (double side){
20         // El área de un cuadrado es igual al área de un rectángulo con longitud
21         return rectangle.calculateArea(side,side);
22     }
23 }
```

PATRÓN DE COMPORTAMIENTO - OBSERVER

```
ejercicio observer
1 // autor: Llasmin Aguilar y Wilson Puentes
2 // estructura observer
3
4 public interface observer {
5     void update (String message);
6 }
7
8 public class concreteobserver implements observer {
9     private string name;
10     public concreteobserver(string name) {
11         this.name = name;
12     }
13     public void update(string message){
14         system.out.println(name + "received message:" + message);
15     }
16 }
17 // ArrayList
18
19 public class concretesubject implements subject {
20     private list<observer> observer = new ArrayList<>();
21
22     public void registerobserver(observer observer) {
23         observers.add(observer);
24     }
25     public void removeobserver(observer observer){
26         observer.remove(observer);
27     }
28     public void notifyobservers(string message){
29         for (observer observer : observers){
30             observer.update(message);
31         }
32     }
33 }
```

CONCLUSIÓN

Aprender los conceptos básicos de la codificación nos permite entender los componentes del creciente panorama tecnológico, además de obtener una perspectiva completamente nueva sobre el papel trascendental que juega el uso y desarrollo de las nuevas tecnologías en nuestra vida cotidiana; unido a una mejor apreciación de cómo todo se vincula.

BIBLIOGRAFÍA

- Sesión virtual, instructor Ricardo Alfonso González Vargas
- Motor de búsqueda "google"
- Herramienta tic "Visual Studio Code"