# TECNOLÓGICO NACIONAL DE MÉXICO

# INSTITUTO TECNOLÓGICO DE LEÓN

## *Programación Web*

**Trabajo:**

*Proyecto Final*

**Presenta:**

*Daniel Irad Maldonado Villanueva*

*Cervantes Torres Armando*

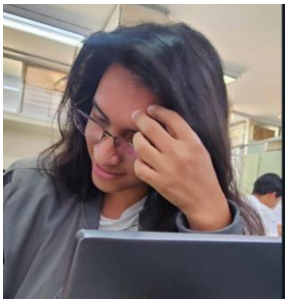
*Nájera Saul*

*García Méndez Angel Saul*

*Preciado Eduardo*







## CUADRO COMPARATIVO DE LOS DISTINTOS TIPOS DE LENGUAJE DEL LADO DEL SERVIDOR

| Lenguaje | Costos de Software | Funcionalidad | Seguridad |
|---|---|---|---|
| PHP | Gratuito (open source) | Amplia comunidad, fácil integración con bases de datos (MySQL, PostgreSQL). | Buena seguridad con configuraciones adecuadas, aunque puede ser vulnerable si no se siguen buenas prácticas. |
| Python | Gratuito (open source) | Potente, escalable, ideal para frameworks como Django y Flask. | Ofrece soporte para protocolos modernos y bibliotecas como bcrypt para encriptación. |
| Node.js (JavaScript) | Gratuito (open source) | Ideal para aplicaciones en tiempo real, alta velocidad, arquitectura no bloqueante. | Seguridad fuerte si se implementan módulos y prácticas correctas (helmet.js, OWASP). |
| Ruby | Gratuito (open source) | Usado en aplicaciones web con Rails, excelente para prototipos rápidos. | Buen soporte para seguridad, aunque puede ser menos eficiente que otros lenguajes en grandes escalas. |
| Java | Gratuito o Licencia | Ideal para aplicaciones empresariales, altamente robusto y multiplataforma. | Seguridad sólida con características incorporadas como autentificación y encriptación avanzadas. |
| C# (.NET) | Pago (Microsoft) | Integración con servicios de Windows, ideal para aplicaciones corporativas. | Muy seguro, soporta protocolos avanzados, pero depende del ecosistema de Microsoft. |

## SERVICIOS DISPONIBLES EN LA WEB, BENEFICIOS, COSTOS Y ACCESO REMOTO

| Servicio | Descripción | Beneficios | Costos | Posibilidades de Acceso Remoto |
|---|---|---|---|---|
| **Alojamiento Web** | Proveedores como AWS, DigitalOcean, Bluehost, HostGator | Escalabilidad, soporte 24/7, backups automáticos | Desde $3-$100 al mes | Acceso completo desde cualquier lugar |
| **Bases de Datos** | MySQL, PostgreSQL, MongoDB, SQLite | Facilidad de uso, opciones gratuitas, alto rendimiento | Desde gratuitos hasta $10+ al mes | Acceso remoto mediante herramientas como phpMyAdmin, etc. |
| **Pasarelas de Pago** | PayPal, Stripe, MercadoPago, Razorpay | Facilita pagos online, seguridad y soporte a múltiples monedas | Tarifa por transacción (0.5% - 3%) | Acceso remoto desde el dashboard de la pasarela |
| **Autenticación de Usuarios** | Auth0, Firebase Authentication, Passport.js | Autenticación rápida y segura, integración sencilla | Gratuito para proyectos pequeños | Acceso remoto por API RESTful |
| **Servicios de Envíos** | SendGrid, Mailgun, Amazon SES | Envío de correos masivos, integración fácil | Desde gratuitos hasta $20+ al mes | Acceso completo mediante API |

## Protocolos de Seguridad y Estándares que Aplican al Proyecto

SSL/TLS (Secure Sockets Layer / Transport Layer Security):

SSL/TLS encripta los datos entre el cliente y el servidor, garantizando la confidencialidad e integridad de la información. Este protocolo es vital para proteger las credenciales de los usuarios y las transacciones de compra, reduciendo el riesgo de interceptación por terceros.

Autenticación y Autorización:

El protocolo OAuth 2.0 permite inicios de sesión seguros mediante plataformas de terceros como Google o Facebook, evitando almacenar contraseñas en el sistema. Los JWT (JSON Web Tokens) permiten gestionar la autenticación de usuarios en aplicaciones SPA, y la autenticación de dos factores (2FA) añade una capa extra de seguridad, requiriendo un segundo paso de verificación.

Prevención de Vulnerabilidades Comunes:

Para prevenir ataques comunes como XSS (Cross-Site Scripting), CSRF (CrossSite Request Forgery) y SQL Injection, es esencial usar sanitización de entradas, tokens de autenticación en formularios y consultas preparadas para proteger la base de datos.

Protocolos de Comunicación Segura:

Es necesario utilizar HTTPS para encriptar todas las comunicaciones entre el cliente y el servidor. Además, CORS (Cross-Origin Resource Sharing) controla las solicitudes HTTP de diferentes dominios, protegiendo contra solicitudes maliciosas.

Normas de Seguridad para el Almacenamiento de Datos:

Para proteger los datos sensibles, se deben seguir normas como PCI-DSS (para pagos con tarjeta de crédito) y GDPR (para proteger los datos personales de usuarios europeos).

Backup y Recuperación:

Implementar copias de seguridad automáticas de bases de datos y archivos importantes es crucial. Esto garantiza la recuperación de datos en caso de ataques o fallos del sistema, asegurando la continuidad del servicio.

## Interoperabilidad que Aplicaría al Proyecto:

**Uso de APIs RESTful**:

Permiten la comunicación entre diferentes sistemas utilizando el protocolo HTTP. Facilitan la integración con servicios externos, como pasarelas de pago (PayPal, Stripe), sistemas de autenticación (OAuth 2.0) y servicios de envío de correos electrónicos (SendGrid, Mailgun). Estándar para intercambiar datos en formatos como JSON o XML.

**Formatos de intercambio de datos (JSON y XML)**:

Son formatos comúnmente utilizados para el intercambio de datos entre aplicaciones. Permiten que los datos enviados desde el servidor sean comprensibles y procesados por otros sistemas, lo que facilita la integración con plataformas externas.

**Uso de CORS (Cross-Origin Resource Sharing)**:

Permite que nuestro servidor acepte solicitudes de otros dominios de manera controlada y segura. Es útil cuando se trabaja con servicios y recursos alojados en diferentes servidores o se tienen aplicaciones en diferentes dominios.

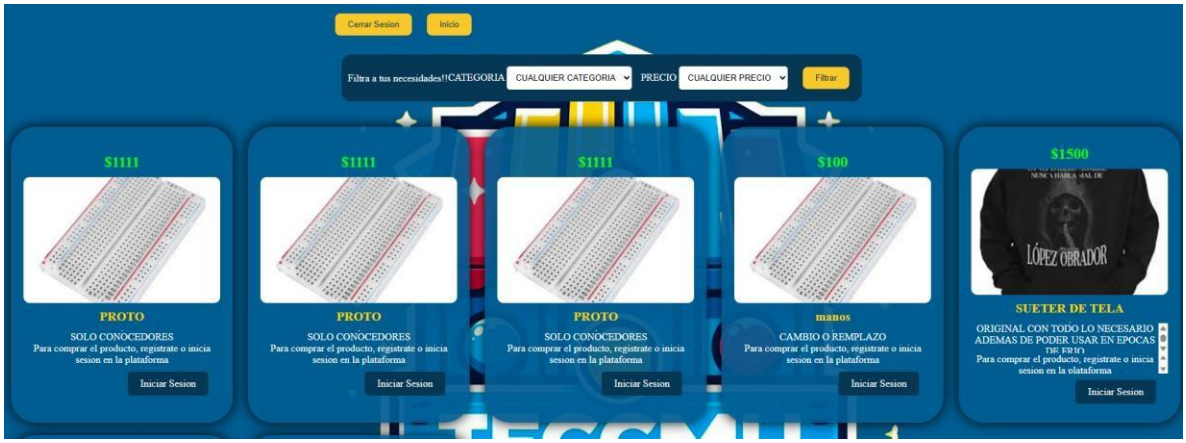
**Integración con Microservicios**:

Si el proyecto requiere integrar varios servicios independientes, el uso de tecnologías como **GraphQL** puede facilitar la interacción con múltiples microservicios a través de una única API. **GraphQL** permite obtener datos desde diversas fuentes de manera eficiente y optimizada.

**Compatibilidad con Navegadores y Dispositivos**:

Asegura que el sitio web funcione correctamente en distintos navegadores (Chrome, Firefox, Safari) y dispositivos (móviles, tabletas, computadoras de escritorio). Mejora la experiencia del usuario al garantizar que todos los visitantes tengan acceso a una funcionalidad consistente sin importar el dispositivo o navegador que utilicen.


Como actualización se habilito el modo visitante



El cual se puso como texto “Para comprar el producto, registrate o inicia sesion en la plataforma” y de esta manera es que se avisa para comprar inicia sesión  si le da click al botón lo redirige para el inicio


Usuario Administrador ->

User: admin

Password: admin

Usuario visitante ->

User: visitante

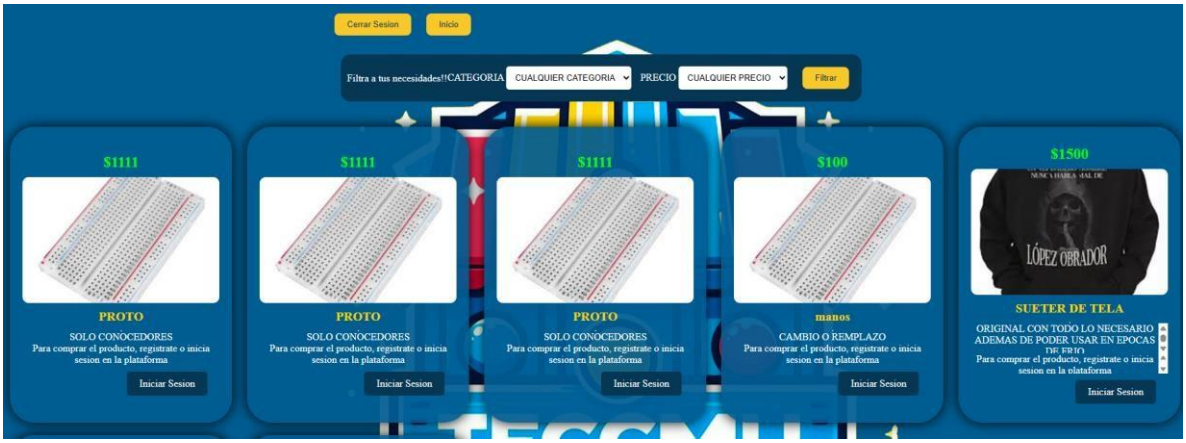
Password: visitante

Usuario con sesión normal

User: 20241251@leon.tecnm.mx

Password: 1234

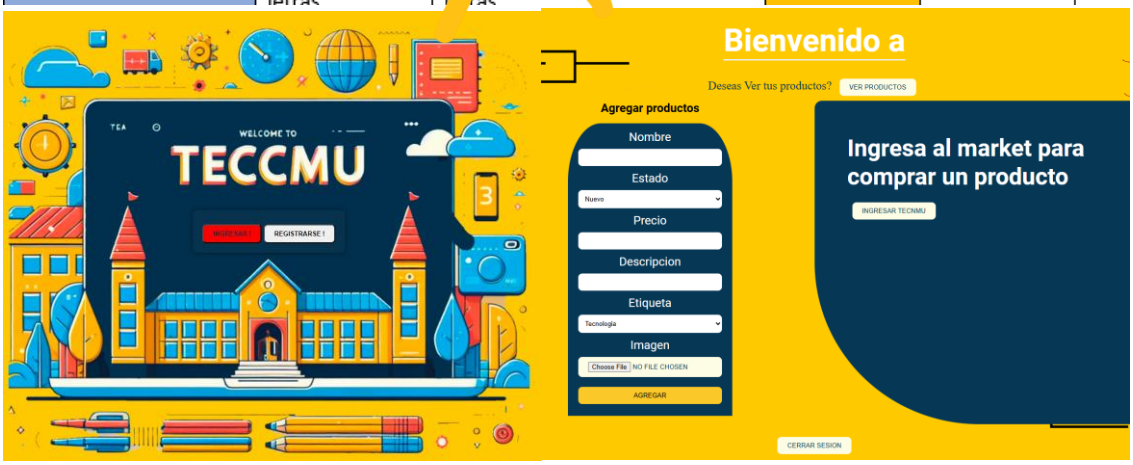
| Criterio | Excelente | Bueno | Regular | Suficiente | No Suficiente |
|---|---|---|---|---|---|
| **Front End Maquetación 10%** | esenta el di o de l maqu on con una distrib ci de com onente re onsiva | Presenta el diseño de la maquetación con una buena distribución de componentes | Presenta el diseño de la maquetación. | **Presenta el diseño mínimo** | **No hay diseño** |





Para movil tanto como para escritorio es totalmente responsiva y se ajusta a la pantalla permitiendo la navegación y demás sobre los componentes esta es la más difícil ya que es dinámica la cantidad de productos que tiene o puede existir en el

| Criterio | Excelente | Bueno | Regular | Suficiente | No Suficiente |
|---|---|---|---|---|---|
| **Front End Aspecto visual 10%** | Utiliza adecuadamente la combinacion de imágenes, de colores y fuentes de letras | Utiliza adecuadamente la combinacion de imágenes o colores o fuentes de letras | Utiliza imágenes, colores y fuentes de letras, que no combinan | **No mantiene el mismo aspecto visual en todas las páginas** | **El aspecto visual no tiene los suficientes elementos** |




Intentamos poner un diseño moderno y con colores llamativos tomando en cuenta colores del Tec tratando de tener la mejor forma de colores y tomando en cuenta la mejor manera el uso de colores junto con diseño lo cual fue un problema grande al encontrar la mejor manera visual de los componentes para verse llamativo

| Criterio | Excelente | Bueno | Regular | Suficiente | No Suficiente |
|---|---|---|---|---|---|
| Criterio | Excelente | Bueno | Regular | Suficiente | No Suficiente |
| **Front End Navegabilidad y Seguridad 10%** | Utiliza protocolo https. Usa manejo de sesiones para el control de acceso a las páginas. Autenticación simple. Tiene una excelente navegabilidad | Usa manejo de sesiones para el control de acceso a las páginas. Autenticación simple. Tiene una buena navegabilidad. | Tiene una buena navegabilidad. | **Tiene algunos herrores en la navegación y tiene algunas fallas en la seguridad** | **No maneja seguridad. La navegabilidad es inconsistente.** |

La autenticación es simple sin el uso de http

```php
<?php
include ('../../servidor.php');

$data = json_decode(file_get_contents("php://input"));
$correo = $data->correo;
$contrasena = $data->password;

if ($correo === 'admin' && $contrasena === 'admin') {
    echo json_encode('admin');
    return;
}
if ($correo === 'visitante' && $contrasena === 'visitante') {
    echo json_encode('visitante');
    return;
}

$sql = "SELECT * FROM usuario WHERE correo = '$correo' AND contraseña =
'$contrasena'";
$result = $db->query($sql);


if ($result->num_rows > 0) {
    session_start();
    $row = $result->fetch_assoc();
    $_SESSION['id'] = $row['id_usuario'];
    $_SESSION['usuario'] = $row['nombre'];

    echo json_encode('exito');
} else {

    echo json_encode('error');
}
//cierra la conexión a la base de datos
$db->close();
?>
```

Utiliza una forma de autenticación simple solo comparando el usuario y sabiendo si es un admin o es un visitante regresando su status sobre que es el usuario danto como prioridad admin o visitante el final la sesión se inicia por medio de phpSesion()

El cual es una herramienta nativa de php para poder obtener el usuario activo teniendo la información de el presente cada que lo requieras y consultando sus productos y su nombre tanto como numero y demás

| Criterio | Excelente | Bueno | Regular | Suficiente | No Suficiente |
|---|---|---|---|---|---|
| **Front End Manejo del DOM !0%** | Domina javaScript para manipular el DOM | Utiliza javaScript para manipular el DOM de manera adecuada | Utiliza javaScript para manipular el DOM de manera parcial | No Utiliza javaScript para manipular el DOM de manera adecuada | **No utiliza javaScript** |

Para nosotros lo mas difícil fue el tratar de usar tanto el filtro de la tabla como el control de altas nuevas

```
 filtro_productos.addEventListener('submit', event => {
        event.preventDefault();
        filtro_categoria = document.getElementById('categoria').value;
        filtro_precio = document.getElementById('precio').value;
        console.log(`Filtrando por categoría: ${filtro_categoria} y
precio: ${filtro_precio}`);
        fetch('../productosGUI/php/producto.php', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                filtro_categoria: filtro_categoria,
                filtro_precio: filtro_precio
            })
        })
            .then(response => response.json())
            .then(data => {
                if (data.error) {
                    console.error('Error:', data.error);
                    productos.innerHTML = `<p>Error: ${data.error}</p>`;
                } else {
                    productos.innerHTML = ''; // Clear the container
                    data.forEach(producto => {
                        const div = document.createElement('div');
                        div.classList.add('product');
```

```
                        div.innerHTML = `
                        <span
class="product__price">$${producto.precio}</span>
                        <img class="product__image"
src="${producto.imagen}"
alt="${producto.modelo}"/>
                        <h1
class="product__title">${producto.modelo}</h1>
                        <hr/>
                        <p>${producto.descripcion}</p>
                        <a class="btn product__btn" data-
id="${producto.id}" href="#">COMPRAR</a>
                    `;

                        productos.appendChild(div);
                    });

                    document.querySelectorAll('.product__btn').forEach(b
utton => {
                        button.addEventListener('click', function
(event) {
                            event.preventDefault();
                            const productId = this.getAttribute('data-
id');
                            buyProduct(productId);
                        });
                    });
                }
            })
    }
    );
```

Tenemos en este apartado el uso de filtros para poder mostrar solo los necesario o lo que pide el filtro obteniendo

La etiqueta de clasificación como el precio y eso toma para poder saber que productos mostrar en el dom

```
 fetch('../productosGUI/php/producto.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            filtro_categoria: filtro_categoria,
            filtro_precio: filtro_precio
        })
    })
        .then(response => response.json())
        .then(data => {
            if (data.error) {
                console.error('Error:', data.error);
                productos.innerHTML = `<p>Error: ${data.error}</p>`;
            } else {
                productos.innerHTML = ''; // Clear the container
```

```
                data.forEach(producto => {
                    const div = document.createElement('div');
                    div.classList.add('product');

                    div.innerHTML = `
                    <span
class="product__price">$${producto.precio}</span>
                    <img class="product__image"
src="${producto.imagen}"
alt="${producto.modelo}"/>
                    <h1 class="product__title">${producto.modelo}</h1>
                    <hr/>
                    <p>${producto.descripcion}</p>
                    <a class="btn product__btn" data-id="${producto.id}"
href="#">COMPRAR</a>
                `;

                    productos.appendChild(div);
                });

                document.querySelectorAll('.product__btn').forEach(butto
n => {
                    button.addEventListener('click', function (event) {
                        event.preventDefault();
                        const productId = this.getAttribute('data-id');
                        buyProduct(productId);
                    });
                });
            }
        })
```

Y este código es sobre mostrar el producto en el DOM tanto los escucha de eventos para compra y en caso de visitante inicio de sesión para mejoramiento

| Criterio | Excelente | Bueno | Regular | Suficiente | No Suficiente |
|---|---|---|---|---|---|
| **Back End Arquitectura y Diseño**<br><br>**15%** | Patrones arquitecturales:<br><br>En capas. MVC.<br><br>Patrones de diseño:<br><br>Utiliza por lo menos dos patrones de diseño<br><br>Elabora los diagramas UML correspondientes. | Uso de MVC<br><br>Autenticación simple<br><br>Utiliza por lo menos un patrón de diseño<br><br>Elabora algunos de los diagramas UML correspondientes. | Uso de MVC<br><br>Autenticación incompleta<br><br>Utiliza por lo menos un patrón de diseño | Uso de MVC<br><br>No hay autenticación | No utiliza MVC ni en capas, no hay autenticación, solamente está en servidor local |

Como modelo vista y controlador tenemos el uso de JavaScrip junto con el front

Para manupulacion del doom

```
    }
    filtro_productos.addEventListener('submit', event => {···
    );
    fetch('../productosGUI/php/producto.php', {···
    }).then(response => response.json())
        .then(data => {···
        })
        .catch(error => {
            console.error('Error:', error);
        });


    function buyProduct(id) {···
    }

    function mostrarFormularioCalificacion(id) {···
    }
    cerrar_sesion.addEventListener('click', () => {···
    });
    inicio.addEventListener('click',()=>{
        console.log('inicio');
        window.location.href = '../usuarioGUI/usuario.html';
    });
});
```

Solo mostrando el uso del de productos que de misma manera en productos de cualquier ventana


Al igual tenemos para visitante

Aun que no puedes comprar si no existe una sesión de un usuario real

| Criterio | Excelente | Bueno | Regular | Suficiente | No Suficiente |
|---|---|---|---|---|---|
| **Back end Orientación a objetos**<br><br>**15%** | Implementa el diseño y la programación Orientada a objetos | Implementa el diseño y la programación Orientada a objetos parcialmente | Implementa el diseño y la programación Orientada a objetos con errores de conceptos | Implementa el diseño y la programación Orientada a objetos inconsistente | No utiliza Orientada a objetos |

Para compra de producto tenemos

```
function buyProduct(id) {···
}
```

El cual lo que hace pasas el ID y ese producto se compra o hace el proceso de compra

```
function mostrarFormularioCalificacion(id) {
        // Crear el formulario de calificación
        const formularioCalificacion = `
        <h2>Calificar producto</h2>
        <div>
            <p>Por favor, califique este producto:</p>
            <fieldset class="rating">
                <input type="radio" id="star5" name="rating" value="5"
/><label for="star5" title="Excelente">5</label>
                <input type="radio" id="star4" name="rating" value="4"
/><label for="star4" title="Muy bueno">4</label>
                <input type="radio" id="star3" name="rating" value="3"
/><label for="star3" title="Bueno">3</label>
```

```
                <input type="radio" id="star2" name="rating" value="2"
/><label for="star2" title="Regular">2</label>
                <input type="radio" id="star1" name="rating" value="1"
/><label for="star1" title="Malo">1</label>
            </fieldset>
            <textarea id="comentario" placeholder="Escribe tu comentario
aquí"></textarea>
            <button id="enviarCalificacionBtn">Enviar
calificación</button>
        </div>
        `;

        // Mostrar el formulario en algún lugar del documento HTML
        const formularioDiv = document.getElementById('formulario-
calificacion');
        formularioDiv.innerHTML = formularioCalificacion;

        // Asignar evento onclick al botón de enviar calificación
        const enviarCalificacionBtn =
document.getElementById('enviarCalificacionBtn');
        enviarCalificacionBtn.addEventListener('click', function() {
            enviarCalificacion(id);
        });
    }
```

Igual el mostrar calificación usa orientado  aobjetos el cual envia el ID del producto para darle la calificación

El cual al final también se le da a

```
function enviarCalificacion(id) {
        const calificacion =
document.querySelector('input[name="rating"]:checked').value;
        const comentario = document.getElementById('comentario').value;

        fetch('../productosGUI/php/calificar_producto.php', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/x-www-form-urlencoded'
            },
            body:
`id=${id}&calificacion=${calificacion}&comentario=${comentario}`
        })
            .then(response => response.json())
            .then(data => {
                if (data === 'exito') {
                    alert(`¡Producto calificado con éxito! Gracias por
tu opinión.`);
                    document.getElementById('formulario-
calificacion').classList.add('ocultar');
                } else {
                    alert('Error al calificar el producto.');
```

```
            }
        })
        .catch(error => console.error('Error:', error));
    }
```

| | | | | | | |
|---|---|---|---|---|---|---|
| **Back end Interoperabilidad 5%** | Diseña y crea por lo menos dos servicios | Diseña y crea por lo menos un servicio | Consume por lo menos un servicio externo | Presenta fallas en el consumo un servicio externo | No crea ni consume ningún servicio | |
| **Back end Persistencia 15%** | Manejar por lo menos 5 tablas en la base de datos | Manejar por lo menos 4 tablas en la base de datos | Manejar por lo menos 3 tablas en la base de datos | Manejar por lo menos 2 tablas en la base de datos | No tiene persistencia | |
| **Despliegue de la aplicación 10%** | Colocado en un servidor externo y funciona adecuadamente | Colocado en un servidor externo y funciona parcialmente | Colocado en el servidor local y funciona bien | Colocado en el servidor local y funciona parcialmente | No funciona | |

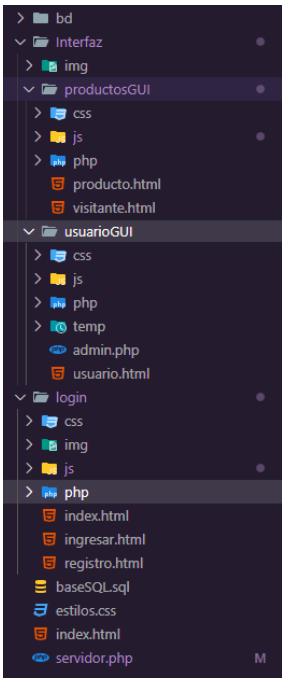
Como interoperabilidad

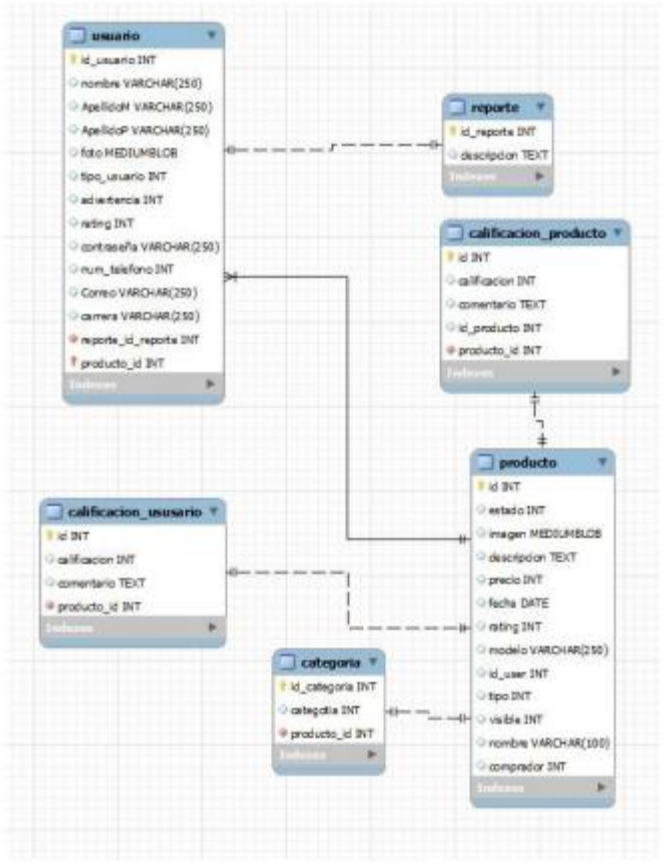
```
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>    <style>
```

Tengo el uso de Bootstrap y demás

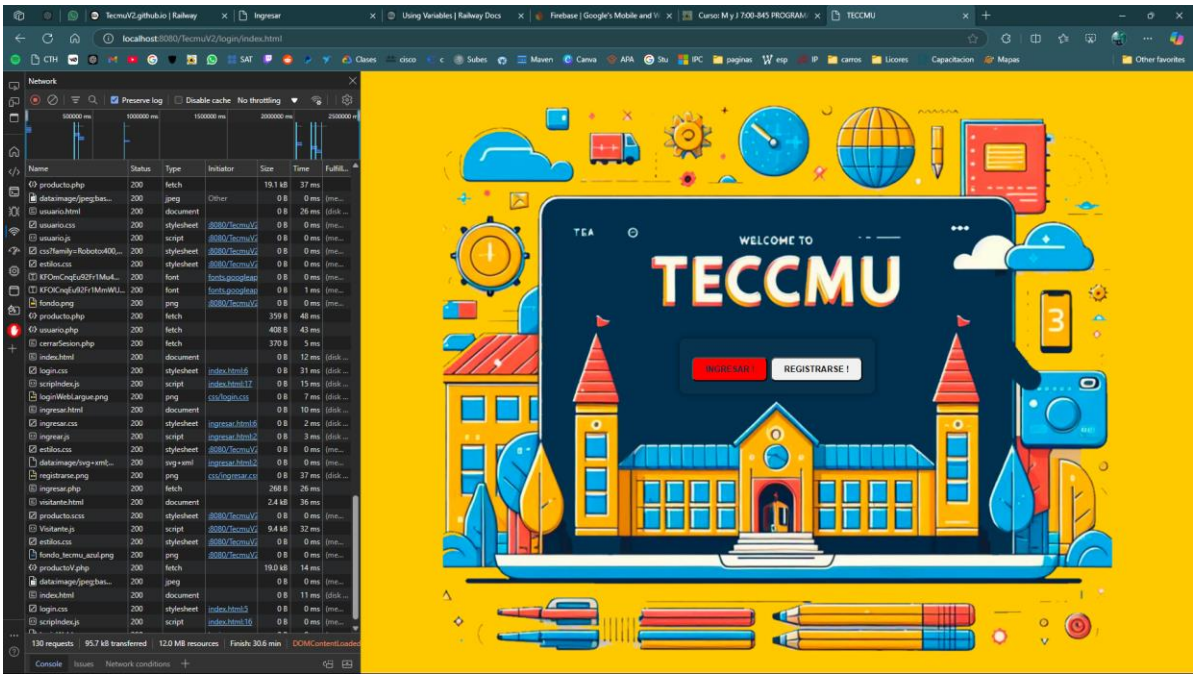
Tiene además el uso de css propio

Mostrando un acomodo bueno para expansión respetando carpetas de cada vista






Uso de mas de 5 tablas




función de manera local y excelente funcionamiento