

Xeishop

(Por Eloy Pérez)

Índice

Portada	1
Índice	2
P1	3-4
P2	5-7
P3	11
P4	15-18
P5	19-20
M1	8
M2	12
M3	15-18
M4	21-23
D1	9-10
D2	13-14
D3	15-18
Parte Visual	24-28
WebGrafía	29

P1: Examinar la relación entre una API y un kit de desarrollo de software (SDK)

Relación entre API y SDK

Una API (interfaz de programación de aplicaciones) es un conjunto de definiciones y protocolos que permiten que dos programas de computadora se comuniquen entre sí. Un SDK (kit de desarrollo de software) es un conjunto de herramientas y bibliotecas que ayudan a los desarrolladores a crear aplicaciones que utilizan una API.

La principal diferencia entre una API y un SDK es que una API define cómo los programas deben comunicarse entre sí, mientras que un SDK proporciona las herramientas y bibliotecas necesarias para implementar esa comunicación.

Una API puede ser utilizada por cualquier programa que cumpla con los protocolos definidos por la API. Un SDK, por otro lado, está diseñado para un lenguaje de programación específico o una plataforma de desarrollo específica.

Cómo se utilizan juntos

Una API y un SDK se utilizan juntos para crear aplicaciones que se comunican con otros programas o servicios. El SDK proporciona las herramientas y bibliotecas necesarias para que los desarrolladores utilicen la API.

Por ejemplo, un desarrollador que quiera crear una aplicación que utilice la API de Google Maps necesitará utilizar el SDK de Google Maps. El SDK de Google Maps proporciona a los desarrolladores las herramientas y bibliotecas necesarias para conectarse a la API de Google Maps y obtener datos de mapas.

Ejemplos

Algunos ejemplos de API y SDK incluyen:

- API de Google Maps: Proporciona acceso a los datos de mapas de Google Maps. El SDK de Google Maps proporciona las herramientas y bibliotecas necesarias para conectarse a la API de Google Maps y obtener datos de mapas.
- API de PayPal: Permite aceptar pagos con PayPal. El SDK de PayPal proporciona las herramientas y bibliotecas necesarias para implementar la API de PayPal en sus aplicaciones.
- SDK de Android: Proporciona las herramientas y bibliotecas necesarias para crear aplicaciones para Android.
- SDK de iOS: Proporciona las herramientas y bibliotecas necesarias para crear aplicaciones para iOS.

P2. Revisión de una serie de API para diferentes Plataformas

En esta sección, se revisarán una serie de API para diferentes plataformas. Se evaluarán las API en función de los siguientes criterios:

- Requisitos de la aplicación: La API debe cumplir con los requisitos de la aplicación, como el tipo de datos que necesita acceder o las funciones que necesita proporcionar.
- Facilidad de uso: La API debe ser fácil de usar y comprender, incluso para desarrolladores sin experiencia.
- Seguridad: La API debe ser segura y proteger los datos confidenciales.
- Documentación: La API debe estar bien documentada para que los desarrolladores puedan entender cómo usarla.

API de Google Maps

La API de Google Maps es una API de geolocalización que proporciona a los desarrolladores acceso a los datos de mapas de Google Maps. La API es gratuita y está disponible para una amplia gama de plataformas, incluidos Android, iOS, web y escritorio.

Requisitos de la aplicación: La API de Google Maps puede ser utilizada por cualquier aplicación que necesite acceder a los datos de mapas de Google Maps.

Facilidad de uso: La API de Google Maps es relativamente fácil de usar. La documentación es clara y concisa, y hay una gran cantidad de ejemplos y tutoriales disponibles en línea.

Seguridad: La API de Google Maps utiliza una serie de medidas de seguridad para proteger los datos confidenciales, como la codificación SSL y la autenticación.

Documentación: La documentación de la API de Google Maps es completa y detallada. Cubre todos los aspectos de la API, desde la instalación hasta el uso avanzado.

API de PayPal

La API de PayPal es una API de pago que permite a los desarrolladores aceptar pagos con PayPal. La API es gratuita y está disponible para una amplia gama de plataformas, incluidos Android, iOS, web y escritorio.

Requisitos de la aplicación: La API de PayPal puede ser utilizada por cualquier aplicación que necesite aceptar pagos con PayPal.

Facilidad de uso: La API de PayPal es relativamente fácil de usar. La documentación es clara y concisa, y hay una gran cantidad de ejemplos y tutoriales disponibles en línea.

Seguridad: La API de PayPal utiliza una serie de medidas de seguridad para proteger los datos confidenciales, como la codificación SSL y la autenticación.

Documentación: La documentación de la API de PayPal es completa y detallada. Cubre todos los aspectos de la API, desde la instalación hasta el uso avanzado.



API de Stripe

La API de Stripe es otra API de pago que permite a los desarrolladores aceptar pagos con Stripe. La API es gratuita y está disponible para una amplia gama de plataformas, incluidos Android, iOS, web y escritorio.

Requisitos de la aplicación: La API de Stripe puede ser utilizada por cualquier aplicación que necesite aceptar pagos con Stripe.

Facilidad de uso: La API de Stripe es relativamente fácil de usar. La documentación es clara y concisa, y hay una gran cantidad de ejemplos y tutoriales disponibles en línea.

Seguridad: La API de Stripe utiliza una serie de medidas de seguridad para proteger los datos confidenciales, como la codificación SSL y la autenticación.

Documentación: La documentación de la API de Stripe es completa y detallada. Cubre todos los aspectos de la API, desde la instalación hasta el uso avanzado.

The Stripe logo, consisting of the word "stripe" in a bold, blue, sans-serif font. The letter 'i' is stylized with a small blue square above it.

M1. Evaluación de una gama de APIs que cubra una variedad de usos, adecuados para un escenario dado

Según el escenario dado, las mejores APIs que creo que podemos utilizar, son la de google maps, en caso de que queramos consultar una ubicación o poner la de nuestra tienda física, y la más importante, la API de PayPal, ya que al tener que hacer una pasarela de pago, nos va a venir muy bien.

Las características de las APIs son las mismas que en el p2, pero ahora incluiré una tabla con las características:

API	Requisitos del escenario	Facilidad de uso	Seguridad	Documentación
Google Maps	Sí	Sí	Sí	Sí
PayPal	Sí	Sí	Sí	Sí
Stripe	Sí	Sí	Sí	Sí

D1. Evaluación de una API seleccionada para un escenario determinado, incluidos los posibles problemas de seguridad

En esta sección, se evaluará la API de Google Maps para el escenario de un sistema de compras en línea, incluidos los posibles problemas de seguridad.

Requisitos del escenario

La API de Google Maps es adecuada para el escenario de un sistema de compras en línea porque permite a los usuarios encontrar tiendas cercanas.

Facilidad de uso

La API de Google Maps es relativamente fácil de usar. La documentación es clara y concisa, y hay una gran cantidad de ejemplos y tutoriales disponibles en línea.

Seguridad

La API de Google Maps utiliza una serie de medidas de seguridad para proteger los datos confidenciales, como la codificación SSL y la autenticación.

Posibles problemas de seguridad

A pesar de las medidas de seguridad que utiliza la API de Google Maps, existen algunos posibles problemas de seguridad que deben tenerse en cuenta.

- **Inyección de código:** Los atacantes podrían inyectar código malicioso en una solicitud a la API de Google Maps. Esto podría permitirles tomar el control del sistema o robar datos confidenciales.
- **Robo de datos:** Los atacantes podrían robar datos confidenciales, como las coordenadas GPS de los usuarios, a través de la API de Google Maps.
- **Negación de servicio:** Los atacantes podrían utilizar la API de Google Maps para sobrecargar el sistema, lo que podría causar un tiempo de inactividad.

Medidas de mitigación

Para mitigar estos riesgos, se deben implementar las siguientes medidas:

- Usar una **versión actualizada** de la API de Google Maps. Las versiones más recientes de la API incluyen medidas de seguridad mejoradas.
- **Validar** todas las **entradas** a la API de Google Maps. Esto ayudará a prevenir la inyección de código.
- Usar una **conexión segura** (SSL) al llamar a la API de Google Maps. Esto ayudará a proteger los datos confidenciales.
- Implementar un **sistema de detección de ataques**. Esto ayudará a detectar y responder a los ataques.

Conclusión

La API de Google Maps es una API segura, pero es importante ser consciente de los posibles problemas de seguridad y tomar medidas para mitigarlos.

P3. Investigar una aplicación que podría ser ampliado con un API.

App: *App para Tomar Notas y Organizar Cosas (la de IOS por ejemplo)*

Mejora con API: *API de Traducción (cualquiera (Google Cloud Translation API o Microsoft Translator API)).*

Razón:

Si nuestra app de notas mejoraría aún más agregándole una API de traducción. Eso significa que los usuarios podrían traducir sus notas a cualquier idioma de manera fácil y rápida. Esto sería muy útil, ya que sería para aquellos que toman notas en un idioma y luego quieren compartirlas o revisarlas en otro idioma. Además, facilita que personas que hablan diferentes idiomas colaboren entre sí, ya que las notas compartidas se traducirían automáticamente.

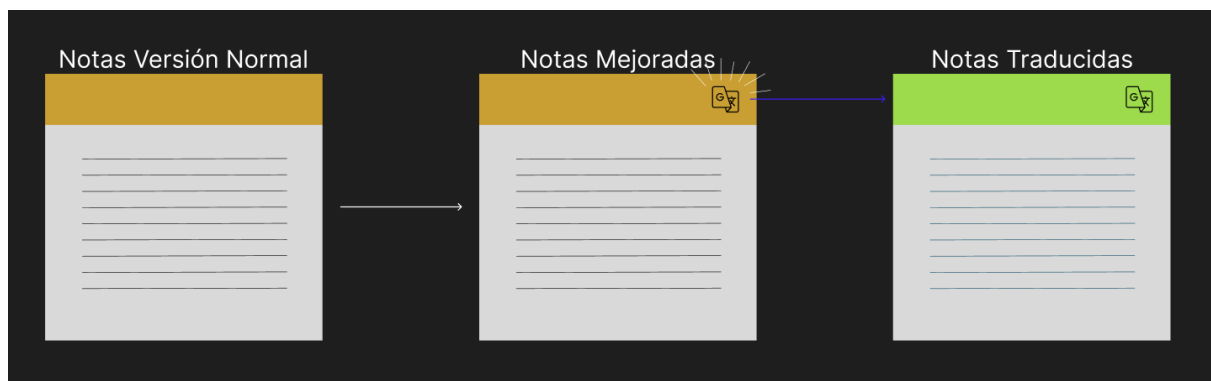
Con la integración de esta API, nuestra app se vuelve más “poderosa”, haciéndola útil para más gente y mejorando la experiencia de usuario en situaciones donde se hablan varios idiomas. Sería hacer que nuestras notas sean aún más útiles y accesibles para todo el mundo.

M2. Diseñar una solución que amplíe el utilizar una API para un fin determinado.

La solución que propongo es la siguiente:

Un botón en las notas, que cuando le damos, nos traduce la nota al idioma seleccionado. De esta forma sería muy sencillo su uso, además de la comunicación con otras lenguas. Como usaríamos la API de Google, por ejemplo, tendríamos comprendidos todos los idiomas que tenga entendidos Google, por lo que podríamos comunicarnos prácticamente con todo el mundo.

Además tiene una interfaz visual sencilla, por lo que desde los más pequeños hasta los más mayores, podrían entenderlo.



D2. Revisar críticamente la solución diseñada para informar mejoras, incluyendo la utilización de una gama de APIs.

Dentro de mi idea, hay algunos aspectos que podrían mejorarse y considerar la utilización de una gama de APIs para enriquecer la funcionalidad de la aplicación:

1. Flexibilidad de API: En lugar de depender exclusivamente de una sola API, podríamos explorar la posibilidad de ofrecer a los usuarios la opción de elegir entre varias APIs de traducción. Esto podría proporcionar redundancia y flexibilidad en caso de que una API específica experimente problemas o limitaciones.

2. Idiomas Adicionales: Aunque menciono la utilización de la API de Google para abarcar una amplia gama de idiomas, podría ser beneficioso explorar APIs adicionales que ofrezcan soporte para idiomas específicos o características particulares de traducción. Esto permitiría una adaptación más fina a las necesidades de los usuarios.

3. Personalización de Traducción: Consideramos la posibilidad de permitir a los usuarios personalizar las configuraciones de traducción, como el tono de la traducción, el nivel de formalidad, o la preferencia de términos específicos. Esto podría mejorar la experiencia del usuario al adaptarse a sus preferencias individuales.

4. Interfaz de Usuario Intuitiva: Nos aseguraremos de que la interfaz de usuario sea intuitiva y fácil de entender para usuarios de todas las edades. Iconos claros y mensajes informativos nos pueden ayudar a guiar a los usuarios a través del proceso de traducción de manera efectiva.

5. Seguridad y Privacidad: Dada la naturaleza de la traducción de texto, es importantísimo abordar las preocupaciones de seguridad y privacidad. Nos tendremos que asegurar de cumplir con los estándares de seguridad de datos y considerar la posibilidad de ofrecer opciones para que los usuarios gestionen la privacidad de sus datos durante el proceso de traducción.

6. Pruebas de Usabilidad: Antes de implementar la solución, realizaremos pruebas de usabilidad con un grupo diverso de usuarios, para así identificar posibles obstáculos y recibir un feedback para ajustar la experiencia de usuario.

Integrando estas mejoras, la aplicación no solo sería una herramienta útil para la traducción de notas, sino que también proporcionaría una experiencia más personalizada y adaptada a las necesidades individuales de los usuarios.

P4, M3, D3. Construir sobre un marco de aplicación existente para implementar una API.

Yo he creado una aplicación con nodejs y express con la siguiente finalidad:

La finalidad de esta aplicación hipotética es permitir a los usuarios agregar tareas a una lista y obtener traducciones de estas tareas a diferentes idiomas. En este caso, la aplicación se centra en la gestión de tareas y la capacidad de traducir el contenido de esas tareas a idiomas específicos.

Cómo se usa:

Agregar una Tarea:

- Un usuario puede agregar una tarea haciendo una solicitud POST al endpoint `/api/tasks` con el texto de la tarea y el idioma al que desea traducirla. Por ejemplo:

```
POST /api/tasks
Content-Type: application/json

{
  "taskText": "Write a report",
  "targetLanguage": "es"
}
```

Obtener Traducción de una Tarea:

- Un usuario puede obtener la traducción de una tarea específica haciendo una solicitud GET al endpoint `/api/translate/:taskId`, donde `:taskId` es el identificador único de la tarea. Por ejemplo:

```
GET /api/translate/1
```

- La aplicación busca la tarea en la lista y devuelve el texto original de la tarea junto con su traducción en la respuesta.

Es un ejemplo básico y la aplicación real podría tener más características, como autenticación de usuarios, persistencia de datos en una base de datos, gestión de usuarios, entre otras.

Code: Se runeará haciendo “npm install” y luego “npm start” (Puedes hacer solicitudes a la API utilizando herramientas como Postman o curl.)

Estructura:

```
> controllers
> node_modules
> routes
▼ services
  JS translationService.js
JS index.js
{} package-lock.json
{} package.json
```

index.js:

```
// index.js
const express = require('express');
const bodyParser = require('body-parser');
const tasksRoutes = require('./routes/tasks');

const app = express();

app.use(bodyParser.json());
app.use(tasksRoutes);

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

routes/tasks.js:

```
// routes/tasks.js
const express = require('express');
const router = express.Router();
const translationController = require('../controllers/translationController');

// Agregar una tarea
router.post('/api/tasks', translationController.addTask);

// Obtener traducción de una tarea
router.get('/api/translate/:taskId', translationController.getTranslation);

module.exports = router;
```


controllers/translationController.js:

```
// controllers/translationController.js
const translationService = require('../services/translationService');

// Agregar una tarea y obtener traducción
exports.addTask = async (req, res) => {
  const { taskText, targetLanguage } = req.body;

  // Lógica para agregar tarea a la base de datos (supongamos que usas una base de datos)

  // Lógica para obtener traducción
  const translatedText = await translationService.translate(taskText, targetLanguage);

  res.json({ taskText, translatedText });
};

// Obtener traducción de una tarea
exports.getTranslation = async (req, res) => {
  const taskId = req.params.taskId;

  // Lógica para obtener la tarea de la base de datos (supongamos que usas una base de datos)

  // Lógica para obtener traducción
  const translatedText = await translationService.translate(taskText, targetLanguage);

  res.json({ taskText, translatedText });
};
```

services/translationService.js:

```
// services/translationService.js
const translate = require('google-translate-api');

exports.translate = async (text, targetLanguage) => {
  // Lógica para traducir el texto usando google-translate-api
  const translation = await translate(text, { to: targetLanguage });
  return translation.text;
};
```

package.json:

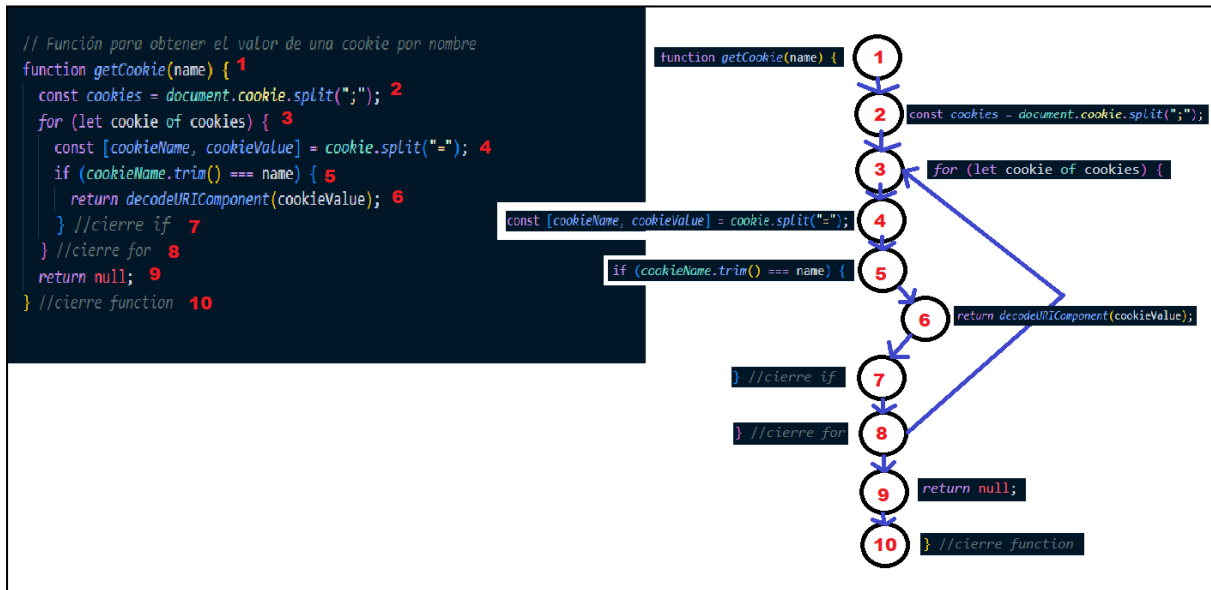
```
{
  "name": "task-translation-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "express": "^4.17.1",
    "body-parser": "^1.19.0",
    "google-translate-api": "^2.3.0"
  }
}
```

La conclusión final del ejemplo es que es un **ejemplo básico** y la aplicación real podría tener más características, como **autenticación** de usuarios, **persistencia** de datos en una base de datos, **gestión de usuarios**, entre otras. Además, es importante tener en cuenta que, en un entorno de producción real, se deberían implementar medidas de **seguridad**, como la gestión de **tokens** de autenticación y **validación** de datos de entrada.

P5. Caja Blanca y Caja Negra

Caja Blanca para una parte de code:

(Se ve un poco mal ya que está hecha en paint, pero si se amplía, se lee perfectamente)



Caja Negra para un Login:

2. En cierto lenguaje de programación las palabras definidas por el usuario deben construirse conforme a las siguientes reglas:

Tienen como máximo 30 caracteres.

El juego de caracteres utilizable es:

- Letras (mayúsculas o minúsculas).
- Dígitos (0–9).
- guión (–).

En los nombres se diferencian mayúsculas de minúsculas.

El guión no puede estar ni al principio ni al final de la palabra, pero puede haber varios consecutivos.

Deben contener al menos un carácter alfabético.

No puede ser una de las palabras reservadas del lenguaje (if, data, real, . . .).

Realizar una batería de pruebas para validar un programa que recibe como entrada la palabra del usuario y proporciona como salida un mensaje de error, si la palabra no es correcta, o la palabra, si es correcta.

Condiciones Entrada	Clases de equivalencia	Clases de equivalencia NO válidos
Caracteres = 1 o <=30	1<=carácter<=30 1	1. carácter<1 2. 30<carácter 9
Utiliza letra minúscula	"c" 2	"C" 10
Utiliza letra mayúscula	"C" 3	"c" 11
Utiliza algún dígito comprendido entre el 0 - 9	"1-2-3-4-5-6-7-8-9" 4	1. num<0 2. num>9 3. num no entero 4. No num 12
Usa un guión: "-"	"-" 5	"NO UTILIZA" 13
El guión no puede estar ni al principio ni al final	Letra – Letra 6	1. " – Letra " 2. " Letra – " 3. " – Letra – " 14
Mínimo un carácter alfabético	"c" 7	"73532" 15
Palabra != "if", "data", "real"	"cualquiera" 8	1. "if" 2. "data" 3. "real" 16

Ejemplo	Cumple	Resultado
Hasta-Los-Huevos-69 (19)	1,2,3,4,5,6,7,8	OK
CaminantenohaY-caminosehacecaminoalAndar12 (+30)	2,3,4,5,6,7,8,9	Error
hola-buenastardes12ja (21)	1,2,4,5,6,7,8,10	Error
HOLA-BUENASTARDES12JA (21)	1,3,4,5,6,7,8,11	Error
PUeSun-ejemplo-SinDigitos (24)	1,2,3,5,7,8,12	Error
EsteEjemplo23No (15)	1,2,3,4,6,7,8,13,14	Error
-MalEjemplo1 (12)	1,2,3,4,5,7,8,14	Error
MalEjemplo1- (12)	1,2,3,4,5,7,8,14	Error
131-5313 (8)	1,2,3,4,5,6,8,15	Error
Imagine-if-12 (13)	1,2,3,4,5,6,7,16	Error
Imagine-data-12 (15)	1,2,3,4,5,6,7,16	Error
Imagine-real-12 (15)	1,2,3,4,5,6,7,16	Error

M4. Perfeccionar la solicitud basándose en los resultados de pruebas.

No quiero meter cambios en mi aplicación, ya que la considero bastante delicada aparte de propensa a la rotura. Pero te puedo proporcionar el código de mejora que sería para mi aplicación, en este caso, para el login (caja negra):

El formulario nos quedaría así:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Login</title>
</head>
<body>
  <form id="loginForm">
    <label for="username">Usuario:</label>
    <input type="text" id="username" name="username" required>
    <label for="password">Contraseña:</label>
    <input type="password" id="password" name="password" required>
    <button type="button" onclick="validateLogin()">Iniciar
sesión</button>
  </form>
  <script src="rutaDelArchivoJs"></script>
</body>
</html>
```

Este código HTML crea una página de inicio de sesión con un formulario que incluye campos para el nombre de usuario y la contraseña. Utiliza un script JavaScript externo referenciado con `src="rutaDelArchivoJs"`, para implementar la lógica de validación del formulario. Además, se establece la codificación del documento como UTF-8 y se incluyen metaetiquetas para la escala de vista y el título de la página.

Mientras que el script que necesitaríamos, sería de la siguiente manera:

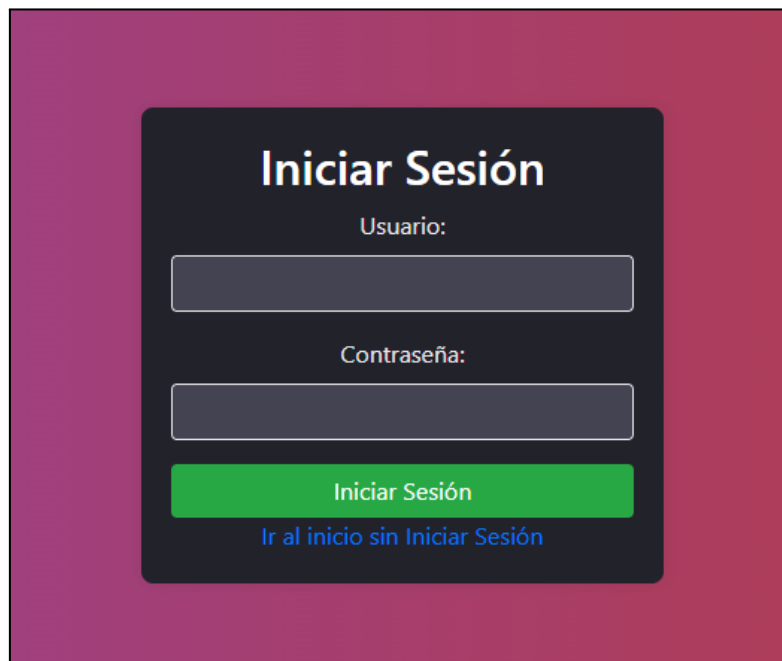
```
function validateLogin() {  
    // Obtener valores del formulario  
    var username = document.getElementById("username").value;  
    var password = document.getElementById("password").value;  
    // Verificar longitud máxima  
    if (username.length > 30 || password.length > 30) {  
        alert("El usuario y la contraseña deben tener como  
máximo 30 caracteres.");  
        return;  
    }  
    // Verificar caracteres permitidos  
    var regex = /^[a-zA-Z0-9-]+$/;  
    if (!regex.test(username) || !regex.test(password)) {  
        alert("El usuario y la contraseña solo pueden contener  
letras (mayúsculas o minúsculas), dígitos y guiones (-).");  
        return;  
    }  
    // Verificar guión no al principio ni al final  
    if (username.startsWith("-") || username.endsWith("-") ||  
password.startsWith("-") || password.endsWith("-")) {  
        alert("El guión no puede estar al principio ni al final  
del usuario o la contraseña.");  
        return;  
    }  
    // Verificar al menos un carácter alfabético  
    if (!/[a-zA-Z]/.test(username) ||  
![a-zA-Z]/.test(password)) {  
        alert("El usuario y la contraseña deben contener al  
menos un carácter alfabético.");  
        return;  
    }  
    // Verificar palabras reservadas  
    var reservedWords = ["if", "data", "real", "admin",  
"null"];  
    if (reservedWords.includes(username.toLowerCase()) ||  
reservedWords.includes(password.toLowerCase())) {  
        alert("El usuario o la contraseña no pueden contener  
palabras reservadas.");  
        return;  
    }  
    // Si todas las validaciones pasan, se puede proceder al  
inicio de sesión  
    alert("Inicio de sesión exitoso");  
}
```

La función `validateLogin()` en el Js se encarga de validar la información ingresada en un formulario de inicio de sesión. Primero, obtiene los valores del nombre de usuario y la contraseña del formulario. Luego, realiza varias validaciones: verifica que tanto el nombre de usuario como la contraseña no excedan los 30 caracteres, que sólo contengan caracteres permitidos (letras mayúsculas y minúsculas, dígitos y guiones), que el guión no esté al principio ni al final, que ambos contengan al menos un carácter alfabético, y que no coincidan con palabras reservadas como "if", "data", "real", "admin" o "null". Si alguna de estas condiciones no se cumple, muestra una alerta correspondiente. Si todas las validaciones son exitosas, muestra una alerta indicando un "Inicio de sesión exitoso".

Es importante señalar que el código asume que las funciones `startsWith` y `endsWith` son compatibles con la versión de JavaScript utilizada. Además, se espera que el código HTML incluya elementos con los IDs "username" y "password" para recoger la entrada del usuario.

Mi aplicación, Xeishop, e-commerce solicitado por el cliente. (Parte visual)

Nada más entrar, nos encontramos con un LogIn, donde nos pide Usuario y Contraseña. También hay un botón para entrar sin iniciar sesión (entrar como invitado), esto nos hará que a la hora de ver el perfil no nos salga nuestro nombre, ya que hemos entrado como invitados.



Iniciar Sesión

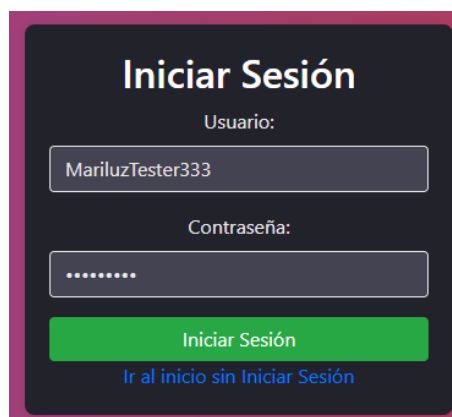
Usuario:

Contraseña:

Iniciar Sesión

[Ir al inicio sin Iniciar Sesión](#)

Si iniciamos sesión correctamente, nos redirigirá directamente a nuestro perfil.



Iniciar Sesión

Usuario:

Contraseña:

Iniciar Sesión

[Ir al inicio sin Iniciar Sesión](#)

Perfil de: MariluzTester333

¡Bienvenido de nuevo, MariluzTester333!

[Explorar la tienda](#)

Si le damos a “Explorar la tienda”, nos llevará a nuestro inicio.

Vemos como es un inicio bastante bonito visualmente, sin muchos elementos, para de esta forma, centrar la atención del cliente en lo que nosotros queramos.



Contamos con una barra de navegación bastante bonita también, con los elementos necesarios para que el usuario se mueva por la página.

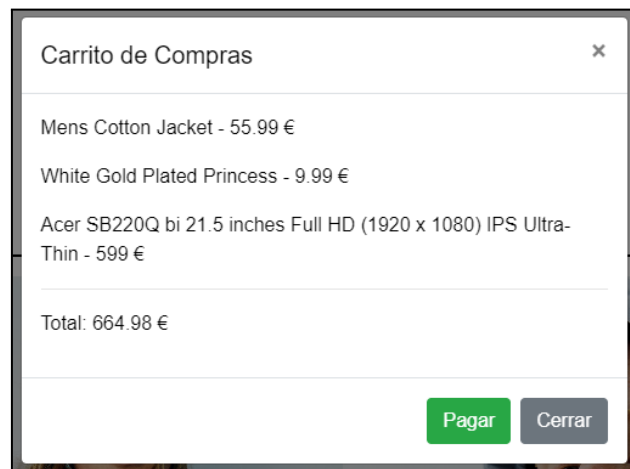


El Log In nos redirige a Iniciar Sesión de nuevo. About Me, Mapa y Tienda, nos bajan la ventana hasta el punto donde está lo que queremos, si es el Mapa, nos baja hasta el mapa, si es la Tienda, nos baja hasta la tienda. Esto es gracias al atributo id de HTML y a la redirección. Si le damos a Mi perfil, nos lleva al apartado anteriormente visto, donde recuperamos el nombre de usuario de una cookie que hemos creado al iniciar sesión. El carrito nos mostrará los elementos agregados al carrito. La persistencia se logra gracias a un token de sesión creado cuando se inicia sesión. Gracias a Math.random, este token es único para cada usuario.

	Name	Value
https://www.google.com	username	MariluzTester333
IndexedDB		
Web SQL		
Cookies		
http://localhost:3000		

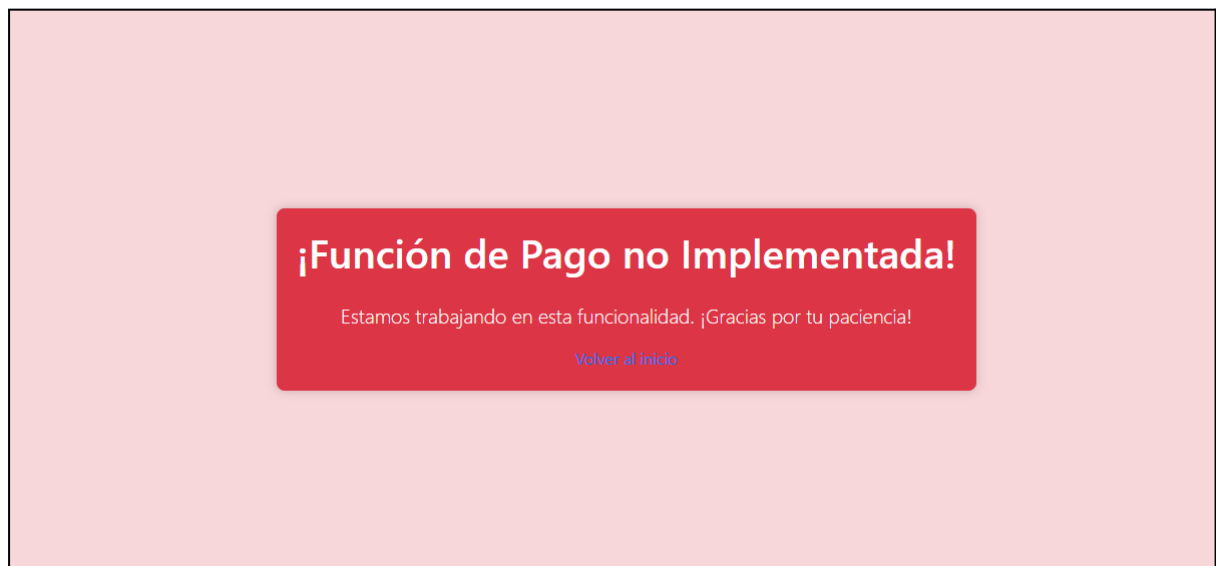
	Key	Value
Local storage		
Session storage		
http://localhost:3000	token	TWFyaWx1eIRlc3RlcjMzMzAuOTA0OTM3...
https://www.google.com	cart	[{"name":"Mens Cotton Jacket","price":55...

Si añado por ejemplo 3 productos aleatorios al carrito, luego los podré ver cuando le de al icono del cart.



Si le damos a Pagar, el botón verde (para llamar la atención del usuario), nos redirigirá a nuestra pasarela de pago.

En mi caso, aún no está implementada, ya que no está pensada para lanzarlo a lo industrial.



Poco más se puede comentar de la parte visual, destacar el AboutUs y el Mapa, con ubicación personalizada, que los encontraremos bajando un poco la página.

About Us

En Xeishop, nos entusiasma tenerte aquí y te ofrecemos más que una simple tienda en línea; somos un equipo apasionado comprometido a proporcionar una experiencia de compra única y auténtica. Desde nuestros inicios, hemos trabajado incansablemente para seleccionar cuidadosamente una amplia gama de productos que reflejen diversidad y calidad en cada rincón de nuestra tienda en línea. En Xeishop, no buscamos ser los más grandes, sino los mejores en lo que hacemos. Valoramos cada compra, nos esforzamos por superar expectativas con productos de alta calidad y precios justos, y comprometemos un servicio al cliente excepcional para cada cliente especial. Nuestro equipo está formado por personas reales con historias y pasiones genuinas, compartiendo un amor por productos únicos. Creemos en la transparencia, honestidad y autenticidad. Agradecemos ser parte de tu historia, ansiamos crecer y aprender contigo, ofreciéndote lo mejor en cada compra. En Xeishop, no solo somos una tienda en línea; somos tu destino para descubrir, explorar y encontrar tesoros que enriquecerán tu vida. ¡Gracias por elegir Xeishop!




También destacar la tienda con su buscador, aunque visualmente me gusta menos. La tienda es el resultado del consumo de una API, concretamente, la de fakestore

Nuestros productos

Aquí podrás consultar los mejores productos disponibles a la venta.
¡No lo dudes y disfruta de nuestra gran variedad!

Buscar




Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops

Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday

Price: \$109.95

Comprar




Mens Casual Premium Slim Fit T-Shirts

Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for breathable and comfortable wearing. And Solid stitched shirts with round neck made for durability and a great fit for casual fashion wear and diehard baseball fans. The Henley style round neckline includes a three-button placket.

Price: \$22.3

Comprar



Mens Cotton Jacket

great outerwear jackets for Spring/Autumn/Winter, suitable for many occasions, such as working, hiking, camping, mountain/rock climbing, cycling, traveling or other outdoors. Good gift choice for you or your family member. A warm hearted love to Father, husband or son in this thanksgiving or Christmas Day.

Price: \$55.99

Comprar

Poco más a comentar sobre la parte visual de la aplicación. Si queremos comentarla más en profundidad, tendríamos que meternos al code.

Hecho por Eloy Pérez Gómez

WebGrafía:

<https://fakestoreapi.com/products>

https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch

<https://developer.mozilla.org/es/docs/Web/API/Document/cookie>

<https://blogprog.gonzalolopez.es/articulos/sesiones-en-javascript.html>