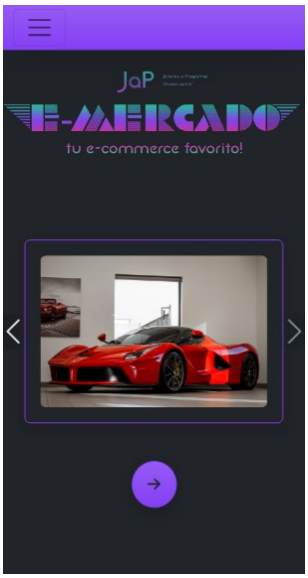
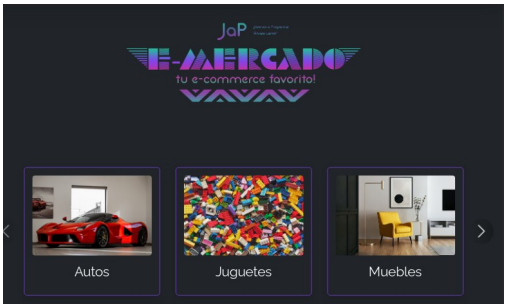
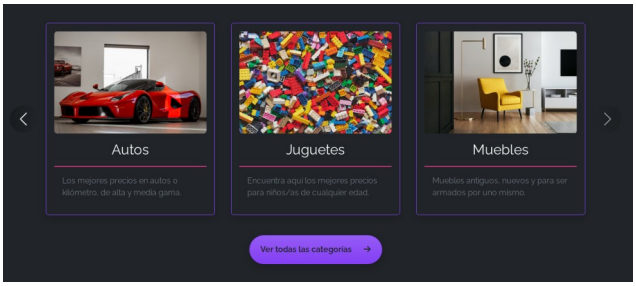
Documentación de “e-mercado”

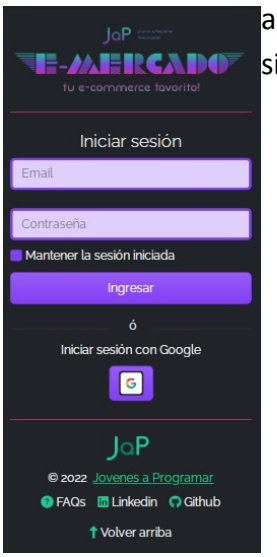
A continuación se presentan algunas de las características y funcionalidades del sitio web de comercio electrónico realizado como parte del curso de desarrollo web de Jóvenes a Programar (Ceibal).







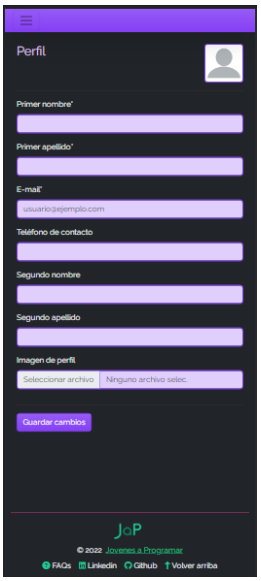
Con un diseño inspirado en las tendencias “retro” de la época actual, totalmente responsive, e-mercado se encarga de ofrecerle al usuario una buena experiencia a la hora de navegar por el sitio, agregar productos al carrito, comprar, personalizar su perfil y vender, sin importar el dispositivo desde el que acceda.



Funcionalidades*:

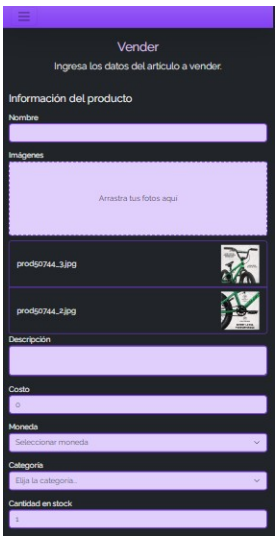
- Login: Para hacer uso de la página es necesario que el usuario se registre, siendo este un requisito para navegar por ella.
  - Si es un usuario nuevo, sus datos se guardan en una base de datos simulada y se almacena de forma local el ID con el que quedó registrado para luego realizar las solicitudes necesarias.

- Si es un usuario existente, se comprueba que sus datos, como la contraseña, sean correctos, y se almacena de forma local el ID en base a la respuesta a la solicitud.



- Mi perfil: El usuario puede ingresar a modificar sus datos personales, tal como sus nombres, apellidos, teléfono, correo, así como elegir una foto de perfil. Estos datos se almacenan junto con los datos de inicio de sesión en la ya mencionada base de datos.
  - Las solicitudes se realizan de acuerdo al identificador de ese usuario específico en la base de datos.

- Mi carrito: El usuario puede optar por agregar sus productos favoritos al carrito, el cual es guardado junto con su información de inicio de sesión.
- Comentarios: El usuario puede dejar comentarios en cada producto, los cuales serán almacenados en la base de datos junto a los comentarios de otros usuarios en el mismo producto.
- Búsqueda de productos: estando en la página de productos, el usuario puede realizar búsquedas para encontrar un producto por su nombre o descripción.
- Compra: Dentro de su carrito, si el usuario llena el formulario correspondiente, puede concretar la compra de los productos elegidos. Una vez realizada la compra, los productos se eliminan de la base de datos.



- Venta: El usuario puede simular la publicación de productos. A través de drag and drop puede arrastrar imágenes desde su dispositivo.

*El backend se encuentra en desarrollo, por lo tanto se utilizan otros medios para simularlo (mockapi.io, API json, almacenamiento local).

APIs utilizadas:

- E-mercado API: se trata de una API json para obtener el listado de categorías y productos del sitio. Se utilizan los siguientes endpoints:
  - https://japceibal.github.io/emercado-api/cats/cat.json
  - https://japceibal.github.io/emercado-api/cats_products/:id.json
  - https://japceibal.github.io/emercado-api/products/:id.json
- Mockapi.io: una API simulada utilizada para almacenar los datos de los usuarios, sus carritos personales y los comentarios. Se le realizan solicitudes de tipo GET, POST y PUT de acuerdo al caso. Se utilizan los siguientes endpoints:

- https://SECRET.mockapi.io/users/
- https://SECRET.mockapi.io/users/:id
- https://SECRET.mockapi.io/comments/
- https://SECRET.mockapi.io/comments/:id

- ExchangeRates API: se utiliza para obtener el valor del dólar en moneda uruguaya para realizar las conversiones correspondientes y mostrar los precios totales en dólares. Se le realiza una solicitud de tipo GET una vez al día, algo controlado a través de una cookie que se almacena con el valor de ese día. Una vez hecha la solicitud en el día, se utiliza el valor almacenado en cookies para evitar el constante envío de peticiones.
- API de inicio de sesión con Google: Se utiliza para brindarle acceso al usuario sin necesitar registrarse en la página. A través de una librería que decodifica JWT, se verifica que se trate de un e-mail verificado, de ser así, se registra al usuario en la base de datos y se lo redirige a la portada.

Uso de almacenamiento local:

LocalStorage:

- Si el usuario elige mantener iniciada su sesión, se guarda allí el correo con el que inicia sesión. Esto se utiliza para verificar el inicio de sesión. De no encontrarse ese dato en local ni en session, se lo redirige a la pantalla de inicio de sesión.
- Cuando el usuario inicia sesión, se almacena el ID con el que figura en la base de datos. Este dato se utiliza para realizar las peticiones necesarias (al modificar su perfil, al añadir productos al carrito, al realizar una compra, etc).

Key	Value
user	usuario@gmail.com
userID	1

- Cuando el usuario hace click en un producto, su ID se almacena.

SessionStorage:

- Mismo procedimiento que para localStorage, pero se utiliza este almacenamiento si el usuario elige no mantener iniciada su sesión.

Key	Value
cookieConsent	true
user	usuario@gmail.com

- Se almacena aquí el consentimiento de uso de cookies. Esto ocurre una vez el usuario cierra el modal de aviso, algo que ocurre en la página de inicio, a la cual es redirigido luego de iniciar sesión o registrarse.

Cookies:

- Se almacena el valor del dólar en moneda uruguaya para el presente día, obtenido con una petición a ExchangeRates API. Tiene una fecha de expiración de 1 día, luego de lo cual se realiza nuevamente la petición.

Name	Value	Domain	P...	Expires / Max-Age
currencyRate	39.807182	127.0.0.1	/	2022-11-13T21:18:09.000Z

Construcción del sitio:

El sitio está construido en base a HTML, CSS, Javascript puro y Bootstrap personalizado con Sass, instalado a través de npm. También se utiliza Font Awesome para hacer uso de los íconos.

Funciones principales:

Las siguientes funciones, definidas en init.js, se utilizan a lo largo de todo el sitio

- showSpinner() / hideSpinner(): utilizadas para mostrar/ocultar el spinner de carga cuando se realicen solicitudes.
- setProductID(id): utilizada cuando se clickea un elemento para almacenar en localStorage el ID de cada producto clickeado.
- setCatID(id): utilizada cuando se clickea un elemento para almacenar en localStorage el ID de cada categoría.
- getUser(): obtiene y retorna el correo del usuario almacenado de forma local para determinar si ha iniciado sesión. Si no hay usuario, redirecciona al login para que inicie sesión.
- showUser(): Muestra al usuario en la barra de navegación en un dropdown junto a las opciones de carrito, perfil y cierre de sesión. A esta última opción le agrega una escucha de eventos de tipo click que remueve la información almacenada de forma local que identifica al usuario. Esta función es utilizada cuando carga el documento en todas las páginas del sitio y hace uso de getUser() para obtener el usuario.
- cookiesAlert(): muestra una alerta de uso de cookies en forma de un modal, que tras ser cerrado almacena en sessionStorage el consentimiento del usuario. Se llama en el index.
- createBSAlert(message, type): crea y muestra una alerta de Bootstrap con un mensaje que recibe por parámetro y con un estilo definido por el tipo también recibido (danger, warning, success, etc). Es utilizada a través de todo el sitio luego de realizar peticiones.
- setCookie(cName, cValue, exDays): almacena una cookie con nombre, valor y días en los que va a expirar. Se utiliza para almacenar el valor del dólar durante un día luego de realizar la petición.
- getCookie(cName): obtiene una cookie con el nombre recibido por parámetro. Se utiliza para obtener el valor del dólar si ese día ya se ha realizado la petición a la API.
- USDConversion(cost): calcula y retorna la conversión de pesos a dólares haciendo uso de la variable global exchangeRate y del costo que recibe por parámetro.

FUNCIONES ASÍNCRONAS

- getJSONData(): utilizada para obtener los datos estáticos de la API json (productos y categorías)
- getCurrencyRate(): realiza la petición a la API ExchangeRates y almacena una cookie con el valor obtenido que expira luego de 1 día. La petición se realiza sólo si esa cookie no existe. Se utiliza en las páginas que muestran los precios de los productos y en el carrito.
- getInfo(url): realiza una petición de tipo GET a la url especificada. Se utiliza al momento de iniciar sesión para verificar la existencia del usuario (y verificar sus datos) y a los comentarios del producto actual.

- getSpecificInfo(id, url): realiza una petición de tipo GET al endpoint especificado. Se utiliza para acceder a la información del usuario (datos del perfil y carrito).
- postInfo(info, url): realiza una petición de tipo POST para almacenar la información de un nuevo usuario.
- putInfo(info, url): realiza una petición de tipo PUT para modificar los datos del perfil del usuario, agregar elementos a un carrito y eliminarlos así como para agregar comentarios a un producto haciendo uso de la variable global currentProductDB_ID que tiene el valor que identifica al producto actual en el endpoint de comentarios.

Sitio realizado por Melina Montelongo
https://github.com/melinamontelongo