| Mostrando Marca_Txurdinaga_Color_H_SUP_01_01.jpg | Curso / *Kurtsoa* | Fecha / *Data* | Nivel / *Maila* | Eval. /*Ebal.* |
| --- | --- | --- | --- | --- |
| 2023-24 | 08/02/2024 | 2º | 1ª FINAL |
| Módulo / *Modulua* | *Kodea / Código* | *U.Didak /Unid didác* | *tipo* | Calificación/*Kalifikazioa* |
| DWC | 2DW3 | 1, 2, 3, 4, 5, 6, Vue.js | P |  |
| Nombre Alumno/a / Ikaslearen izena |  | | | |

**EXAMEN**

* Este examen es de tipo procedimental.
* Tiene un valor total de 12,5 puntos.
* Si la propuesta del examen se resuelve de forma correcta se le aplica el valor especificado en cada apartado.
* Descargar el scaffolding desde Github Classroom. <https://classroom.github.com/a/VMFfrep6> en vuestro equipo, haced los commits necesarios. Cuando quieras entregar sube el repositorio a Github.

**Ejercicio 1. Javascript: (2,5 ptos)**

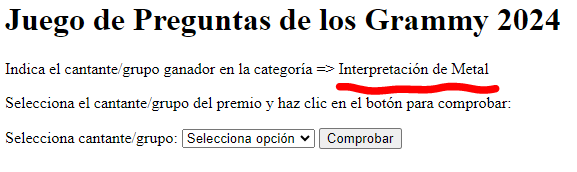
Diseña un programa en Javascript que permita comprobar si conoces los premiados en los Grammy 2024.

El programa toma como punto de partida un array de objetos grammys con la información de las categorías premiadas y su ganador/a.

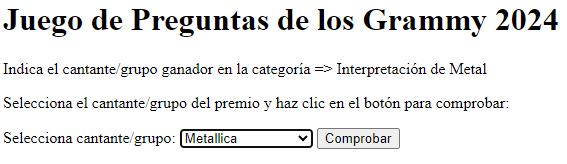
El programa nos mostrará una pregunta aleatoria con la categoría premiada, la pregunta se mostrará en el div reservado del fichero html y nosotros tendremos que responder quién ha sido el artista premiado utilizando la información de un campo desplegable <select>.

Si la respuesta es correcta generará de manera dinámica un nuevo elemento DOM de tipo <li> en la lista <ul> con la información correcta. Si la respuesta es incorrecta nos mostrará un mensaje de error, pero no generará el elemento <li>. Todo el proceso se repite 5 veces

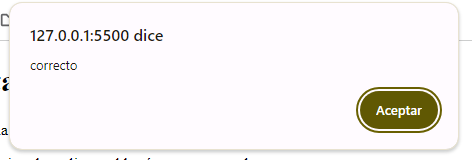
1. Al inicio nos hace una pregunta



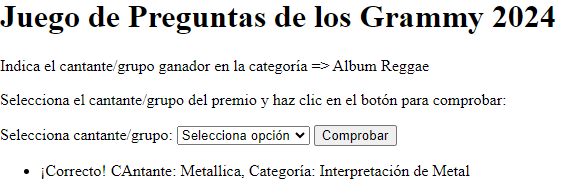
1. Seleccionamos la opción que creemos correcta



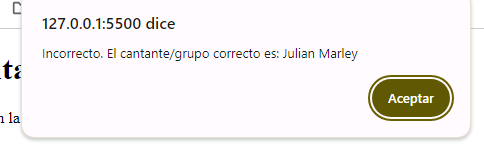
1. Si es correcta muestra mensaje



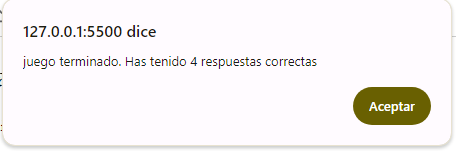
1. Añade <li>, muestra la siguiente pregunta, y reinicia el select.



1. Si está incorrecto. Muestra mensaje, y reinicia el select.



1. Cuando se repite el proceso 5 veces da un mensaje indicando las respuestas correctas y se acaba el juego.



| **Total ejercicio - 1** | **2,50** |
| --- | --- |
| * 1. Al iniciar genera una pregunta al azar con una categoría del array grammys | 0,25 |
| * 1. Recoge el dato del select, verifica si es correcto y crea mensaje correcto | 0,50 |
| * 1. Genera y añade un nuevo elemento <li> con la info a la <ul> (no es válido usar innerHTML) | 0,75 |
| * 1. Reinicia el campo select | 0,25 |
| * 1. Si opción incorrecta genera mensaje, y reinicia campo select | 0,25 |
| * 1. El proceso se repite 5 veces | 0,25 |
| * 1. En el mensaje de fin indica número de respuestas correctas | 0,25 |

**Ejercicio 2. Vue.js: (10 ptos)**

El ejercicio tratará de simular una tienda online (Txurdi Vue Store) en la que podremos loguearnos y realizar compras que irán a un carrito (Pinia). Al salir/cerrar nuestra cuenta todos los datos de la compra de ese usuario desaparecerán.

Para ello la app dispondrá de diferentes rutas (Vue-Router) que estarán accesibles en unas determinadas condiciones. Las diferentes vistas (Views) nos permitirán navegar por la SPA (Single Page Application)

**En este ejercicio vamos a tomar como base el proyecto ya creado y que podrás descargar del repositorio de Github Classroom.**

Para la carga de datos utilizaremos **Axios** (ya incluido en package.json) y los ficheros json (categories.json, products.json, users.json) que están en el directorio **/json**, que cuelga de **public**

Como veis, este proyecto ya incluye 2 rutas que apuntan a las vistas ***HomeView.vue*** (en la que se carga el componente ***Inicio.vue***), y la vista ***AboutView.vue***.

Tendréis que crear la vistas ***LoginView.vue, ArticulosView.vue, CarritoView.vue y LogOutView.vue*** , para conseguir la siguiente app.

1. **Home**

Cuando se inicia la app se visualizarán **SOLAMENTE** las rutas Home | Login | About



1. **LoginView**

Al entrar en el enlace Login, nos aparecerá un formulario de acceso.



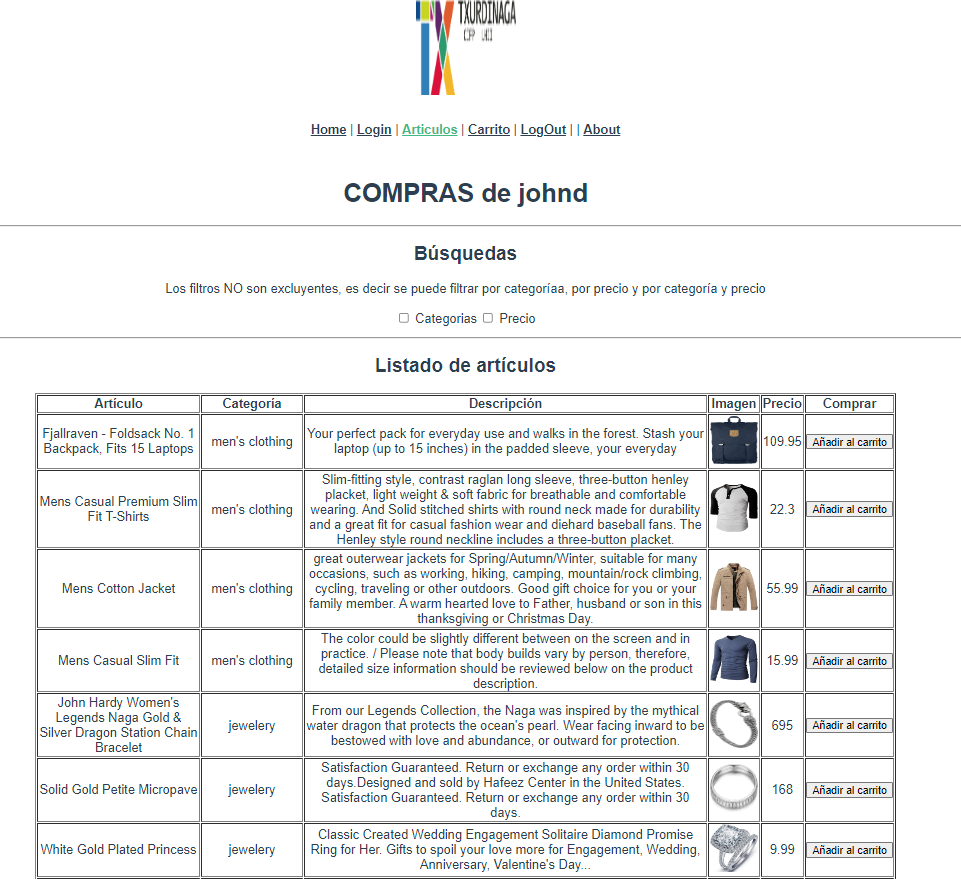
El usuario introducirá su nombre y contraseña. Estos datos se validan con el fichero json /json/users.json.

* Si el usuario y la contraseña NO coinciden con ningún usuario y contraseña del fichero json nos saldrá un mensaje de usuario y/o contraseña incorrectos, y no podremos avanzar.
* Si los datos son correctos (*se comprobará usuario correcto mediante el botón y la tecla enter en el campo password*)
  + Nos mostrará un alert de usuario correcto
  + Se mostrarán el resto de rutas para el usuario autentificado
  + El botón de “Login Usuario” se deshabilitará



1. **ArticulosView**

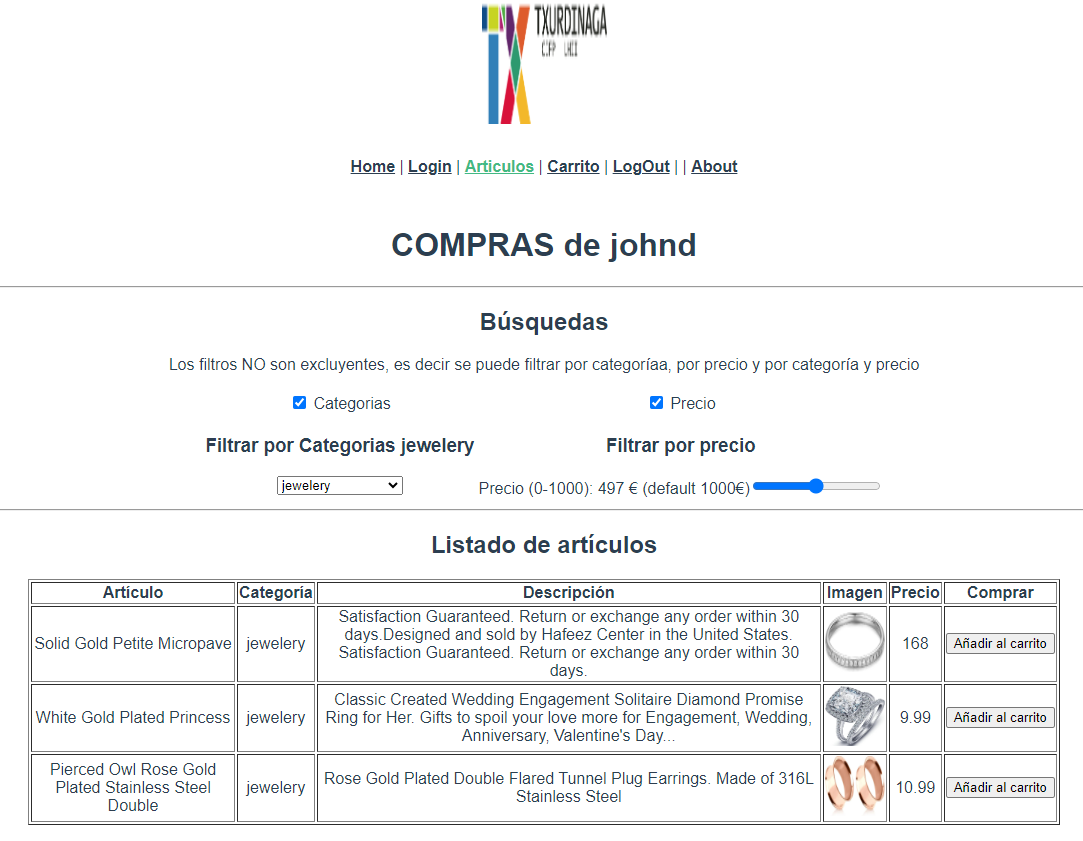
Al entrar en la ruta Artículos, se nos mostrará el título COMPRAS de “usuario”, un apartado con dos checkbox, para filtrar las búsquedas por categorías y precio; y debajo un listado inicial con todos los productos (todas las categorías y precio por defecto inferior a 1000€)



* Al hacer click sobre categorías
  + Se verá un campo select con todas las categorías (datos recogidos desde el fichero /json/categories.json Al seleccionar una categoría, se filtrarán todos los artículos de esa categoría



* Al hacer click sobre precio
  + Se mostrará un campo tipo range, para poder filtrar los artículos por precio



Hay que tener en cuenta que los dos filtros NO SON EXCLUYENTES, es decir, se puede filtrar por categoría, por precio, y por categoría y precio.

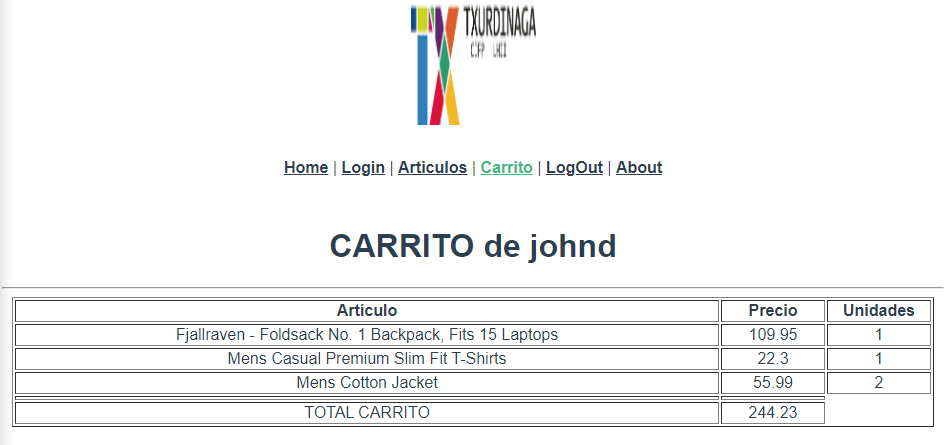
A la derecha de cada producto aparecerá también un botón. Cada vez que se pulsa el botón, aparecerá un mensaje indicando que se ha añadido dicho producto al carrito y se añadirá dicho artículo a un carrito de la compra (carrito que se almacenará en un store de Pinia). La información que se guardará en el carrito del store será: título del artículo, precio y unidades compradas.



1. **CarritoView**

Al entrar en la ruta Carrito, nos mostrará la información que está almacenada en el store.

* Nos aparecerá un título con la información CARRITO de ‘usuario’
* Nos aparecerá una tabla con la información del carrito. Una fila por cada artículo (tituloArticulo, precioArticulo, unidadesCompradas)
* En la última fila de la tabla, nos dará el precio total de todos los artículos del carrito. El formato será un número con 2 decimales.



1. **LogOutView**

Al pinchar en la ruta LogOut,

* Inicialmente nos mostrará un texto como en la imagen.



* Al pulsar el botón. Reiniciará todos los datos del store, nos mostrará un mensaje de despedida, y posteriormente, después de 3 segundos, nos redireccionará a la vista Home, y se dejarán de ver las rutas de usuario logueado. Para acceder a otra ruta podemos utilizar la instrucción this.$router.push('/')



**Valoraciones:**

| **Total ejercicio - 2** | **10** |
| --- | --- |
| 1. **Home** |  |
| * 1. Se han creado las 4 rutas propuestas y cargan los componentes correctamente. | 0,50 |
| 1. **LoginView.** |  |
| * 1. La carga de datos de usuarios con axios para su uso posterior es correcta | 0,50 |
| * 1. Al pulsar el botón se comprueba si el usuario es correcto y se muestran los mensajes (user correcto/incorrecto) | 0,75 |
| * 1. La comprobación también funciona al pulsar enter en el campo password | 0,25 |
| * 1. Se guarda el nombre de usuario en el store para usarlo posteriormente | 0,50 |
| * 1. Se habilitan/muestran el resto de rutas para el usuario (Artículos, Carrito, LogOut) | 0,50 |
| * 1. Se deshabilita el botón “Login Usuario” | 0,25 |
| 1. **ArticulosView** |  |
| * 1. Aparece el mensaje ‘COMPRAS de usuario’ | 0,25 |
| * 1. Se renderiza al inicio la tabla con los artículos y el botón de añadir al carrito. | 1,00 |
| * 1. Al pinchar los checkboxes se visualizan o esconden los filtros | 0,25 |
| * 1. El filtro de categorías funciona correctamente | 0,50 |
| * 1. El filtro de precio funciona correctamente | 0,50 |
| * 1. Los dos filtros funcionan a la vez | 0,25 |
| * 1. Al pulsar el botón se añaden los datos al array del store | 1,00 |
| 1. **CarritoView** |  |
| * 1. Se muestra correctamente el carrito | 0,50 |
| * 1. Se acumulan más de una unidad de un artículo en concreto | 0,50 |
| * 1. Se obtiene el total de las compras correctamente | 0,50 |
| 1. **LogOutView** |  |
| * 1. Al pulsar el botón se reinicia el store. | 0,50 |
| * 1. Se esconden las rutas de usuario logueado | 0,50 |
| * 1. Se redirecciona automáticamente después de 3 segundos a la vista / (HomeView) | 0.50 |