| Mostrando Marca_Txurdinaga_Color_H_SUP_01_01.jpg | Curso / *Kurtsoa* | Fecha / *Data* | Nivel / *Maila* | Eval. /*Ebal.* |
| --- | --- | --- | --- | --- |
| 2022-23 | 14/02/2023 | 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 1ª FINAL**

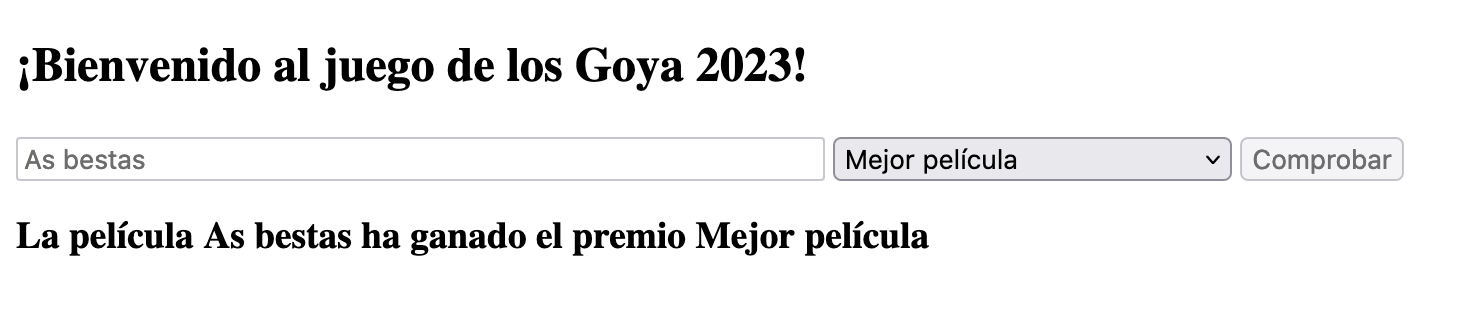
* Este examen es de tipo procedimental.
* Tiene un valor total de 10.
* Si la propuesta del examen se resuelve de forma correcta se le aplica el valor especificado en cada apartado.
* Crear cada ejercicio en un directorio diferente con el nombre Ejercicio1, Ejercicio2... Dentro de cada directorio meter todos los archivos que necesitéis para hacer el ejercicio.
* Cuando terminéis, subir los archivos comprimidos en el drive que tenéis a vuestro nombre (cada ejercicio en su carpeta), con el nombre EVA2\_Apellidos\_Nombre.

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

Debes crear un pequeño juego en el que el usuario pueda ponerse a prueba con las películas de los Goya en esta última edición 2023.

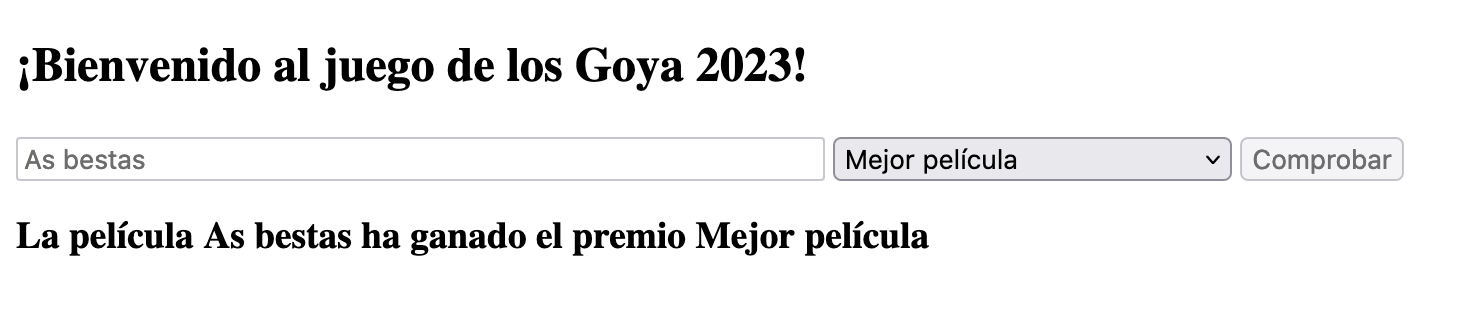
Para ello dispones de un código en HTML que contiene:

* Un campo input deshabilitado para que no pueda editarse.
* Un desplegable con una lista de premios.
* Un botón.
* Un encabezado h3 vacío.

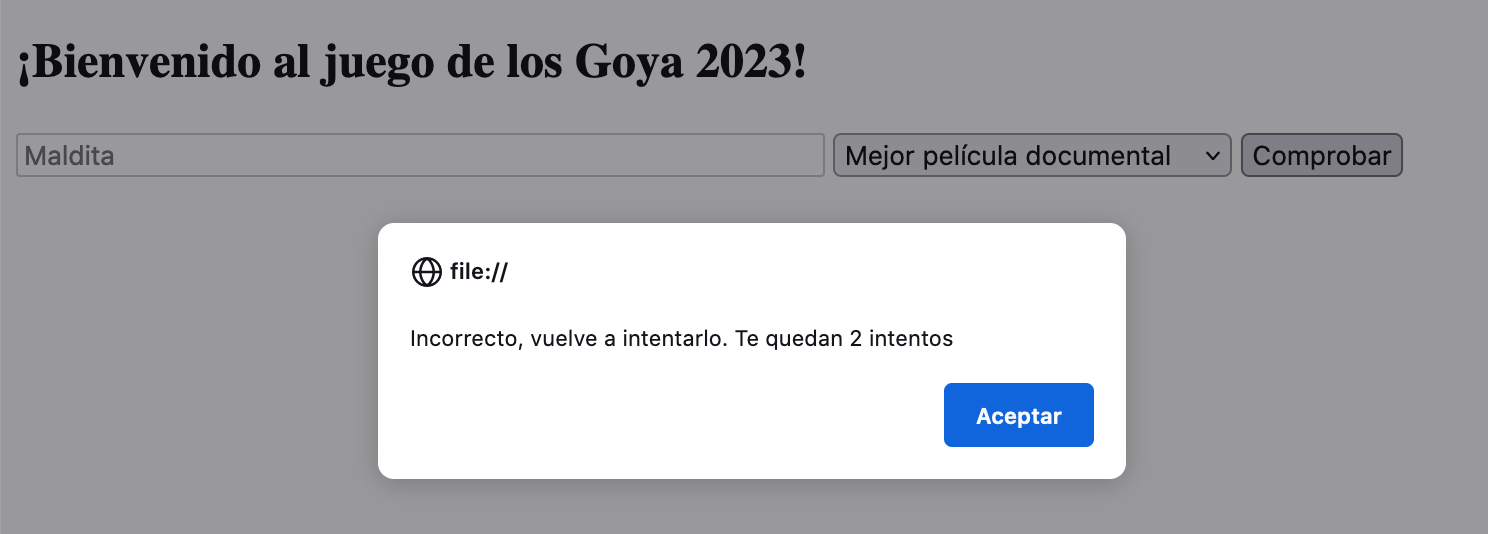


El juego consistirá en lo siguiente:

* Al cargar la página, automáticamente, se introduce una película en el campo input de manera aleatoria.
* El usuario deberá elegir un elemento del desplegable.
* Al pulsar el botón, en caso de que se corresponda la película con el premio, se mostrará un mensaje en la pantalla, en el espacio del texto h3.



* En caso contrario, se mostrará un mensaje con un alert en el que ponga que la respuesta es incorrecta, y el número de intentos que le quedan, partiendo de tres.



El proceso internamente debe ser el siguiente:

* Cuando el usuario pulsa el botón, se comprueba que la película y el premio coinciden, teniendo en cuenta que se corresponden en orden (el primer elemento del array con la primera opción del select, el segundo con la segunda y así sucesivamente).
* Si el usuario acierta:
  + Se creará un objeto con dos atributos (pelicula, premio) para lo cual se habrá definido previamente una clase que incluya esos dos atributos, sus métodos get y set (no usar el modo getPelicula sino get película…), y un método adicional "toString" que devuelve el texto: "La película <pelicula> ha ganado el premio <premio>".
  + Este objeto, además, se añadirá al array de objetos llamado "premiosPeliculas".
  + Por último el texto se mostrará en la página debajo del botón.
  + El botón se deshabilitará.
* Si el usuario falla:
  + Se mostrará un mensaje “Incorrecto, inténtalo de nuevo. Tienes X intentos”, teniendo en cuenta que el usuario puede intentar acertar 3 veces como máximo.
  + El campo select capturará el foco.
* Si el usuario falla tres veces seguidas:
  + Se mostrará un mensaje “FIN DEL JUEGO”.
  + El botón se deshabilitará.

Se podrá obtener más puntuación en el examen si se sustituye uno o más de los elementos del formulario por su creación dinámica utilizando DOM.

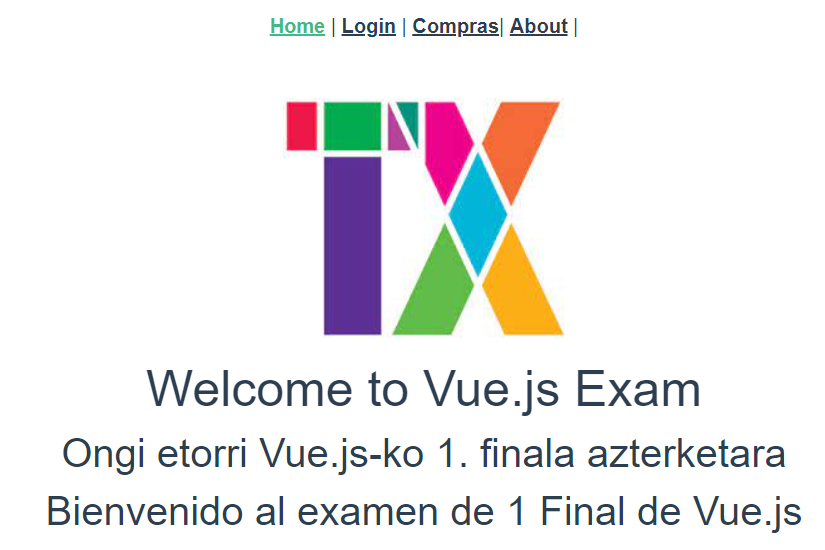
| **Total ejercicio - 1** | **2,5** |
| --- | --- |
| El campo input se carga aleatoriamente con los datos del array. | 0,5 |
| Se realiza la comprobación de la correspondencia entre película y premio correctamente. | 0,3 |
| La clase está correctamente definida con sus método get, set y toString. | 0,2 |
| El objeto se ha creado correctamente. | 0,1 |
| Se ha insertado el objeto en el array. | 0,1 |
| El select capta el foco si el usuario falla el intento. | 0,1 |
| El botón se deshabilita si el usuario acierta o si el usuario termina el juego. | 0,1 |
| Se controla el número de veces que se falla. | 0,1 |
| Los dos mensajes en caso de que el usuario falla se comprueban correctamente. | 0,1 |
| Se ha creado el elemento input con DOM e incluye todos los atributos (disabled, id, size) | 0,4 |
| Se ha creado el elemento select con DOM (id, options con value) | 0,5 |

**Ejercicio 2. Vue.js: (7,5 ptos)**

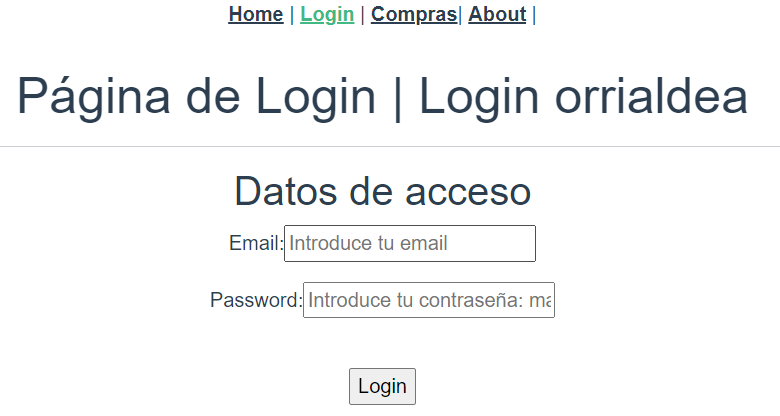
En este ejercicio vamos a tomar como base el proyecto ya creado y que está en la carpeta.

Como veis en este proyecto tenemos 2 rutas que apuntan a las vistas ***HomeView.vue***(en la que se carga el componente HelloWorld.vue), y la vista ***AboutView.vue***. Tendréis que hacer lo siguiente. Crear dos nuevas rutas a las ya existentes ***LoginView.vue*** y ***ComprasView.vue***, La primera de ellas nos permitirá acceder a partir de un usuario existente o a partir de invitado, y la segunda de ella nos permitirá hacer alguna compra de un almacén y nos mostrará el precio final

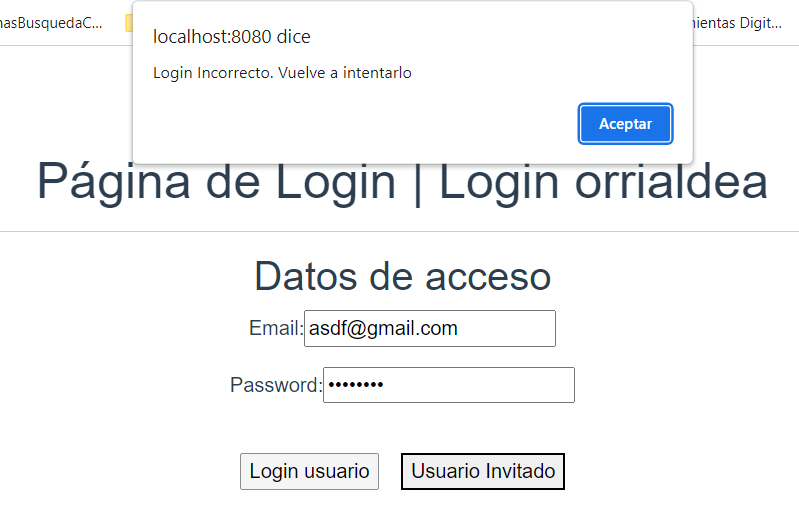
1. Modificar Home para que se vea lo siguiente. El logo *TX.jpg*, lo tenéis en la carpeta ***src/assets***



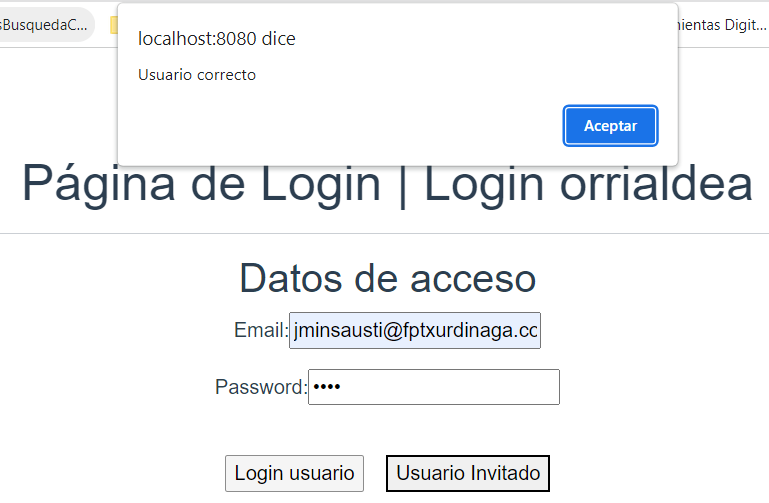
1. Crear una nueva vista ***LoginView.vue*** y en ella hacer lo siguiente:

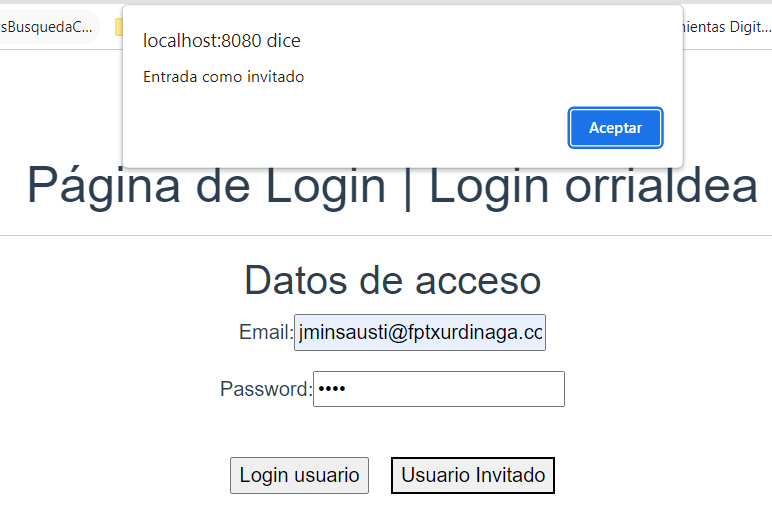


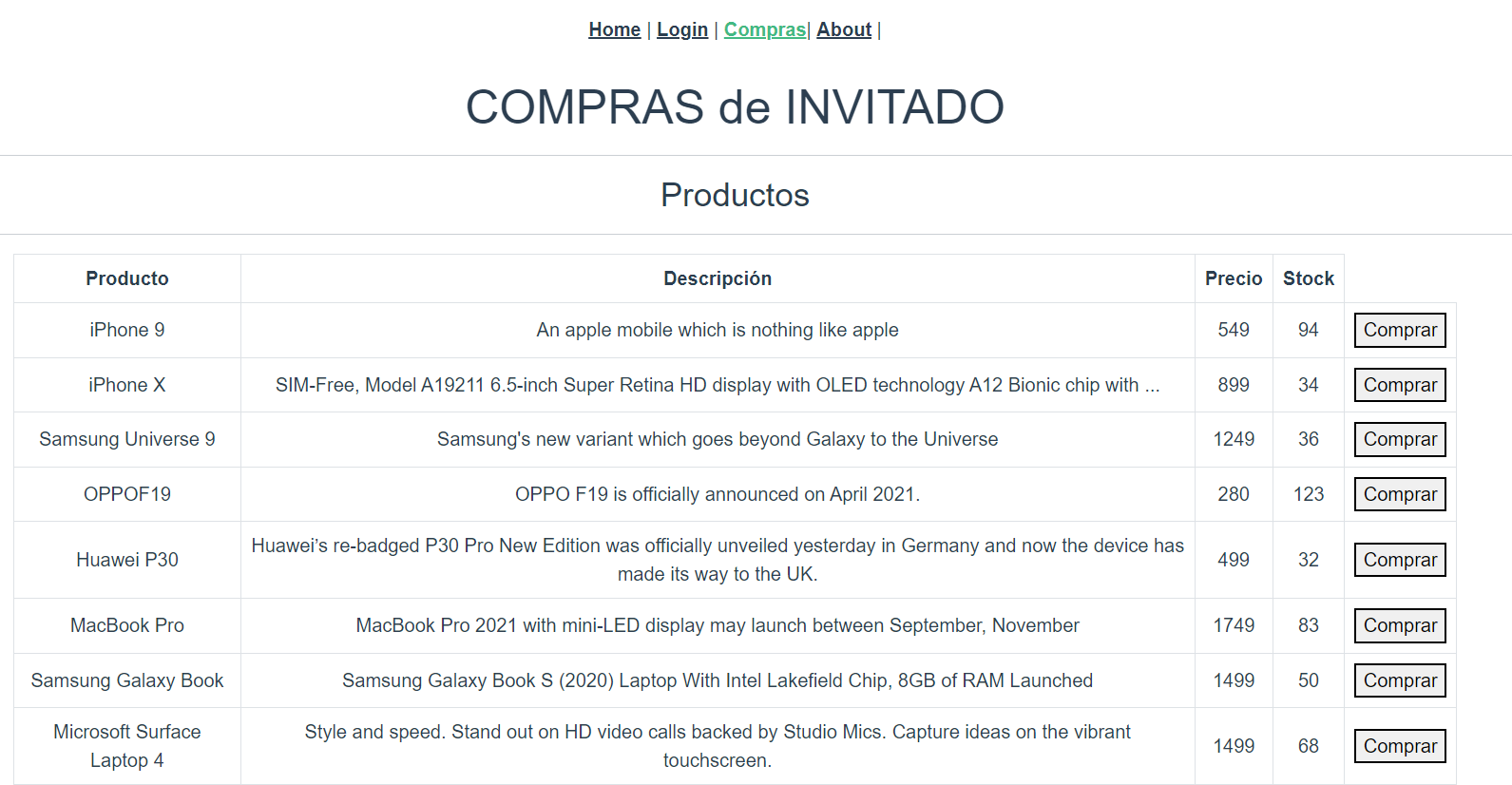
* 1. Según se carga la vista se importará a memoria (mediante axios) los datos de usuarios que se encuentran en el fichero ***public/json/Usuarios.json***.
  2. Una vez tengamos los datos en memoria, podemos acceder mediante usuario registrado
     1. Si el **login NO es correcto** nos muestra un mensaje, para volver a intentarlo.



* + 1. Si el **login SI es correcto.**
       1. Nos muestra el mensaje de usuario correcto

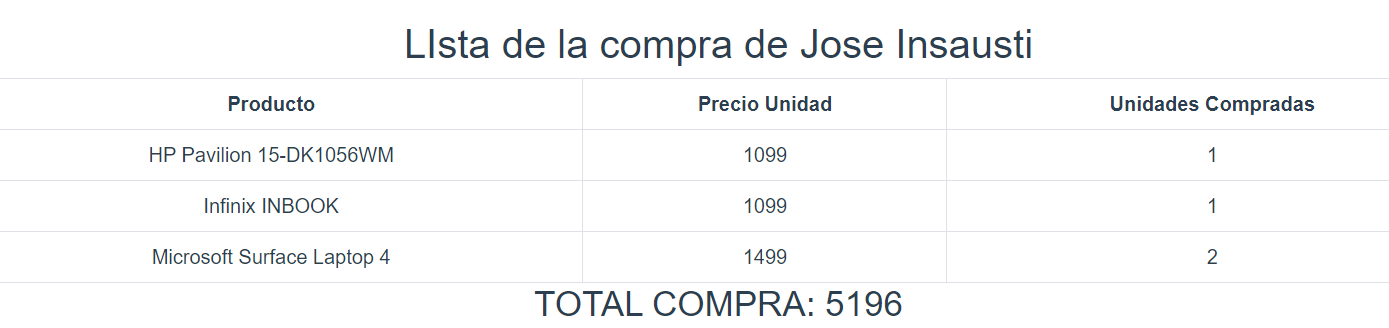


* + - 1. Nos guarda en **sessionStorage** el nombre del usuario logueado y nos lleva a la vista de compras (***window.location.href = '/compras'***).
  1. Si accedemos desde el botón de ‘**Usuario invitado**’, nos muestra un mensaje diciendo que accedemos como invitado, y nos lleva también a la página de compras.
  2. Al acceder a la página de compras mirará el Storage, y si hay un dato guardado nos indicará que hacemos la compra con ese nombre y si no tiene nada guardado nos indicará que hacemos la compra como invitado.





* 1. Nos muestra un listado de productos, que los recogerá del fichero ***public/json/products.json.***
  2. Podremos ir añadiendo elementos a nuestro carrito pulsando el botón comprar.



* 1. Y en un listado que aparecerá en la parte inferior nos muestra los elementos comprados y el total de la compra.
  2. Los productos no se repiten. Se incrementa el número de unidades compradas.

| **Total ejercicio - 2** | **7,5** |
| --- | --- |
| Modificar Home para mostrar logo TX y mensajes | 0,10 |
| Crear vistas LoginView y ComprasView y que sean accesible desde las rutas | 0,15 |
| Cargar datos de usuarios con axios | 0,5 |
| Comprobar usuario correcto y mensaje de usuario correcto | 1,25 |
| Guardar nombreUsuario en sessionStorage | 1 |
| Acceso mediante invitado => redirecciona a vista comprasView | 0,25 |
| Al entrar a comprasView detecta sessionStorage o Invitado y muestra mensajes de entrada | 1,25 |
| Carga datos de productos con axios y los muestra en la tabla | 0,75 |
| Añade a cada fila de la tabla el botón comprar | 0,25 |
| Al pulsar comprar añade el elemento a tabla inferior | 0,5 |
| Si elemento repetido incrementa unidades | 1 |
| Saca el coste total | 0,5 |
|  |  |