UT05.4: Ejemplo Manager

Contenido

1. MVC del Manager

Vamos a crear el MVC del *Manager*.

1.1. Objetos iniciales

Necesitamos 4 ficheros:

- *manager.js*: El modelo que ya tenemos creado.
- *managerView.js*: Vista de la App.
- *managerController.js*: Controlador de la App.
- *managerApp.js*: Necesario para instanciar la App.

1.1.1. Modelo

Como ya hemos comentado está ya implementado y no debemos hacer cambios si hemos verificado que la funcionalidad es correcta.

1.1.2. Vista

Creamos la clase para implementar la vista declarando como propiedades del objeto los elementos de la página que vamos a utilizar más habitualmente. Exportamos la clase por defecto al estar trabajando con módulos.

```
class ManagerView {
  constructor() {
    this.main = document.getElementsByTagName('main')[0];
    this.categories = document.getElementById('categories');
    this.menu = document.querySelector('.navbar-nav');
  }
}

export default ManagerView;
```

1.1.3. Controlador

En el *Controlador*, creamos las constantes Symbol que nos servirán de campos privados. En el constructor asignamos la *Vista* y el *Modelo* recibidos como argumentos a estos campos y exportamos la clase por defecto.

```
const MODEL = Symbol('ShoppingCartModel');
const VIEW = Symbol('ShoppingCartView');

class ManagerController {
  constructor(model, view) {
    this[MODEL] = model;
    this[VIEW] = view;
  }
}

export default ManagerController;
```

1.1.4. Aplicación

Para crear la *Aplicación* vamos a instanciar un *Controlador* a partir del *Modelo* y la *Vista*. Previamente debemos haber importado ambas clases. Para permitir hacer la carga inicial de datos vamos a importar las clases de las entidades que están disponibles en el *Modelo*. Por último, tenemos que exportar la instancia de la aplicación.

```
import Manager from './manager.js';
import ManagerController from './managerController.js';
import ManagerView from './managerView.js';

const ManagerApp = new ManagerController(Manager.getInstance(), new
ManagerView());

export default ManagerApp;
```

La instancia de la *Aplicación* la tenemos que importar en *mitienda.js* como parte del módulo que estamos utilizando en la página.

```
import ManagerApp from './manager/managerApp.js';
```

1.2. Carga inicial de datos

Necesitamos hacer una carga inicial de los datos para dotar de contenido a la página, para ello necesitamos instanciar una serie de objetos. Vamos a crear un método privado para instanciar los objetos. Desde el *Controlador* primero creamos un `Symbol` por seguir la línea de elementos privados.

```
const LOAD_MANAGER_OBJECTS = Symbol('Load Manager Objects');
```

Añadimos el método en la clase. Como podemos ver, hacemos uso de las factorías que habíamos implementado, por lo que no es necesario importar las clases.

```
[LOAD_MANAGER_OBJECTS]() {
  const category1 = this[MODEL].getCategory('Apple',
'img/brands/apple.png');
  const category2 = this[MODEL].getCategory('HP', 'img/brands/HP.png');
  const category3 = this[MODEL].getCategory('Microsoft',
'img/brands/microsoft.png');
  const category4 = this[MODEL].getCategory('Samsung',
'img/brands/samsung.png');
  category1.description = 'Think Different.';
  category2.description = 'HP makes technology work for you.';
  category3.description = 'Be what\'s next.';
  category4.description = 'Designed For.';

  this[MODEL].addCategory(category1, category2, category3, category4);

  const product1 = this[MODEL].getProduct('1', 'Apple', 'Laptop Model1',
1100, 'Laptop');
  const product2 = this[MODEL].getProduct('2', 'Apple', 'Camera Model2',
1200, 'Camera');
  const product3 = this[MODEL].getProduct('3', 'Apple', 'Smartphone
Model3', 1300, 'Smartphone');
  const product4 = this[MODEL].getProduct('4', 'Apple', 'Tablet Model4',
1400, 'Tablet');
  const product5 = this[MODEL].getProduct('5', 'Apple', 'Laptop Model5',
1500, 'Laptop');
  const product6 = this[MODEL].getProduct('6', 'HP', 'Laptop Model1',
2100, 'Laptop');
  const product7 = this[MODEL].getProduct('7', 'HP', 'Camera Model2',
2200, 'Camera');
  const product8 = this[MODEL].getProduct('8', 'HP', 'Tablet Model3',
2300, 'Tablet');
  const product9 = this[MODEL].getProduct('9', 'HP', 'Smartphone Model4',
2400, 'Smartphone');
```

```
  const product10 = this[MODEL].getProduct('10', 'HP', 'Laptop Model5',
2500, 'Laptop');
  const product11 = this[MODEL].getProduct('11', 'Microsoft', 'Laptop
Model1', 3100, 'Laptop');
  const product12 = this[MODEL].getProduct('12', 'Microsoft', 'Camera
Model2', 3200, 'Camera');
  const product13 = this[MODEL].getProduct('13', 'Microsoft', 'Tablet
Model3', 3300, 'Tablet');
  const product14 = this[MODEL].getProduct('14', 'Microsoft', 'Smartphone
Model4', 3400, 'Smartphone');
  const product15 = this[MODEL].getProduct('15', 'Microsoft', 'Laptop
Model5', 3500, 'Laptop');
  const product16 = this[MODEL].getProduct('16', 'Samsung', 'Laptop
Model1', 4100, 'Laptop');
  const product17 = this[MODEL].getProduct('17', 'Samsung', 'Camera
Model2', 4200, 'Camera');
  const product18 = this[MODEL].getProduct('18', 'Samsung', 'Tablet
Model3', 4300, 'Tablet');
  const product19 = this[MODEL].getProduct('19', 'Samsung', 'Tablet
Model4', 4400, 'Tablet');
  const product20 = this[MODEL].getProduct('20', 'Samsung', 'Laptop
Model5', 4500, 'Laptop');

  product1.url =
`https://via.placeholder.com/258x172.jpg?text=${product1.brand}+${product
1.model}`;
  product2.url =
`https://via.placeholder.com/258x172.jpg?text=${product2.brand}+${product
2.model}`;
  product3.url =
`https://via.placeholder.com/258x172.jpg?text=${product3.brand}+${product
3.model}`;
  product4.url =
`https://via.placeholder.com/258x172.jpg?text=${product4.brand}+${product
4.model}`;
  product5.url =
`https://via.placeholder.com/258x172.jpg?text=${product5.brand}+${product
5.model}`;
  product6.url =
`https://via.placeholder.com/258x172.jpg?text=${product6.brand}+${product
6.model}`;
  product7.url =
`https://via.placeholder.com/258x172.jpg?text=${product7.brand}+${product
7.model}`;
```

```
  product8.url =
`https://via.placeholder.com/258x172.jpg?text=${product8.brand}+${product
8.model}`;
  product9.url =
`https://via.placeholder.com/258x172.jpg?text=${product9.brand}+${product
9.model}`;
  product10.url =
`https://via.placeholder.com/258x172.jpg?text=${product10.brand}+${produc
t10.model}`;
  product11.url =
`https://via.placeholder.com/258x172.jpg?text=${product11.brand}+${produc
t11.model}`;
  product12.url =
`https://via.placeholder.com/258x172.jpg?text=${product12.brand}+${produc
t12.model}`;
  product13.url =
`https://via.placeholder.com/258x172.jpg?text=${product13.brand}+${produc
t13.model}`;
  product14.url =
`https://via.placeholder.com/258x172.jpg?text=${product14.brand}+${produc
t14.model}`;
  product15.url =
`https://via.placeholder.com/258x172.jpg?text=${product15.brand}+${produc
t15.model}`;
  product16.url =
`https://via.placeholder.com/258x172.jpg?text=${product16.brand}+${produc
t16.model}`;
  product17.url =
`https://via.placeholder.com/258x172.jpg?text=${product17.brand}+${produc
t17.model}`;
  product18.url =
`https://via.placeholder.com/258x172.jpg?text=${product18.brand}+${produc
t18.model}`;
  product19.url =
`https://via.placeholder.com/258x172.jpg?text=${product19.brand}+${produc
t19.model}`;
  product20.url =
`https://via.placeholder.com/258x172.jpg?text=${product20.brand}+${produc
t20.model}`;
  product1.description = `Descripción ${product1.model}`;
  product2.description = `Descripción ${product2.model}`;
  product3.description = `Descripción ${product3.model}`;
  product4.description = `Descripción ${product4.model}`;
  product5.description = `Descripción ${product5.model}`;
  product6.description = `Descripción ${product6.model}`;
  product7.description = `Descripción ${product7.model}`;
```

```
  product8.description = `Descripción ${product8.model}`;
  product9.description = `Descripción ${product9.model}`;
  product10.description = `Descripción ${product10.model}`;
  product11.description = `Descripción ${product11.model}`;
  product12.description = `Descripción ${product12.model}`;
  product13.description = `Descripción ${product13.model}`;
  product14.description = `Descripción ${product14.model}`;
  product15.description = `Descripción ${product15.model}`;
  product16.description = `Descripción ${product16.model}`;
  product17.description = `Descripción ${product17.model}`;
  product18.description = `Descripción ${product18.model}`;
  product19.description = `Descripción ${product19.model}`;
  product20.description = `Descripción ${product20.model}`;

  this[MODEL].addProductInCategory(category1, product1, product2,
product3, product4, product5);
  this[MODEL].addProductInCategory(category2, product6, product7,
product8, product9, product10);
  this[MODEL].addProductInCategory(category3, product11, product12,
product13, product14, product15);
  this[MODEL].addProductInCategory(category4, product16, product17,
product18, product19, product20);
}
```

Vamos a crear un evento de aplicación que permita invocar el método anterior.

```
onLoad = () => {
  this[LOAD_MANAGER_OBJECTS]();
};
```

Este evento solamente se invocará en el inicio de la aplicación, por lo que lo lanzamos en el constructor.

```
this.onLoad();
```

1.3. Vistas iniciales

En el evento `onInit()` del *Controlador* vamos a añadir las invocaciones de los métodos de la *Vista* para el inicio de la aplicación. La diferencia con el evento `onload()` es que éste último solo se ejecutará cada vez en la carga de la página, y `onInit()` lo hará en respuesta a una petición del usuario de reiniciar el estado de la aplicación.

Creamos el evento y el manejador.

```
onInit = () => {
}

handleInit = () => {
  this.onInit();
}
```

En la *Vista* creamos el bind para los enlaces de inicio y del logo.

```
bindInit(handler) {
  document.getElementById('init').addEventListener('click', (event) => {
    handler();
  });
  document.getElementById('logo').addEventListener('click', (event) => {
    handler();
  });
}
```

En el constructor del *Controlador* invocamos el evento y asignamos el manejador al bind. Tenemos que recordad que primero generaremos el contenido con el evento, ya que enlazará con la *Vista*, y luego debemos enlazar las acciones que se pueden ejecutar con los *bind*.

```
this.onInit();
this[VIEW].bindInit(this.handleInit);
```

1.3.1. Mostrar tipos de productos

Eliminamos el contenido de la capa *categories* ya que lo vamos a generar desde la *Vista*. Creamos un método para renderizar los tipos de productos, añadiendo atributos personalizado `data-type` en cada enlace para obtener el tipo de productos que queremos buscar.

```
showProductTypes() {
  this.categories.replaceChildren();
  this.categories.insertAdjacentHTML('beforeend', `<div class="row"
id="type-list">
    <div class="col-lg-3 col-md-6"><a href="#product-list" data-
type="Camera">
        <div class="cat-list-image"><img alt="Categoría cámaras"
src="img/catcamara.jpg" />
        </div>
        <div class="cat-list-text">
          <h3>Cámaras</h3>
          <div>Digitales y reflex</div>
        </div>
      </a>
    </div>
    <div class="col-lg-3 col-md-6"><a href="#product-list" data-
type="Smartphone">
        <div class="cat-list-image"><img alt="Categoría móviles"
src="img/catmovi.jpg" />
        </div>
        <div class="cat-list-text">
          <h3>Móviles</h3>
          <div>Modelos exclusivos</div>
        </div>
      </a>
```

```
    </div>
    <div class="col-lg-3 col-md-6"><a href="#product-list" data-
type="Laptop">
        <div class="cat-list-image"><img alt="Categoría portátiles"
src="img/catpportatil.jpg" />
        </div>
        <div class="cat-list-text">
          <h3>Portátiles</h3>
          <div>Intel y AMD</div>
        </div>
      </a>
    </div>
    <div class="col-lg-3 col-md-6"><a href="#product-list" data-
type="Tablet">
        <div class="cat-list-image"><img alt="Categoría Tablets"
src="img/cattablet.jpg" />
        </div>
        <div class="cat-list-text">
          <h3>Tablets</h3>
          <div>Android y iPad</div>
        </div>
      </a>
    </div>
  </div>`);
}
```

Este método lo invocamos desde el evento `onLoad()` ya que será estático a lo largo de la vida de la aplicación. Lo invocaremos antes de lanzar un evento.

```
this[VIEW].showProductTypes();
```

1.3.2. Mostrar categorías

Vamos a iterar sobre las categorías recogidas en el *Modelo* para mostrarlas desde la *Vista*. Este método en la *Vista* recibe un iterador con todas las categorías. Es importante prestar atención al detalle del atributo personalizado `data-category` que hemos creado en el enlace, porque lo utilizaremos para comunicarnos con el *Controlador*.

```
showCategories(categories) {
  if (this.categories.children.length > 1)
this.categories.children[1].remove();
  const container = document.createElement('div');
  container.id = 'category-list';
  container.classList.add('row');
  for (const category of categories) {
    container.insertAdjacentHTML('beforeend', `<div class="col-lg-3 col-
md-6"><a data-category="${category.title}" href="#product-list">
      <div class="cat-list-image"><img alt="${category.title}"
src="${category.url}" />
      </div>
      <div class="cat-list-text">
        <h3>${category.title}</h3>
        <div>${category.description}</div>
      </div>
    </a>
  </div>`);
  }
  this.categories.append(container);
}
```

Las categorías desaparecerán de la página según vayamos navegando, por lo que añadimos en el evento *onInit()* la invocación del método para que pueda ser restaurado cuando lo necesitemos. Pasamos como argumento el iterador de categorías.

```
this[VIEW].showCategories(this[MODEL].categories);
```

1.3.3. Categorías en el menú de navegación

Por último, creamos una nueva vista para generar en el menú el listado de categorías. Cada enlace debe tener el atributo personalizado `data-category`.

```
showCategoriesInMenu(categories) {
  const li = document.createElement('li');
  li.classList.add('nav-item');
  li.classList.add('dropdown');
  li.insertAdjacentHTML('beforeend', `<a class="nav-link dropdown-toggle"
href="#" id="navCats" role="button"
    data-bs-toggle="dropdown" aria-expanded="false">Categorías</a>`);
  const container = document.createElement('ul');
  container.classList.add('dropdown-menu');

  for (const category of categories) {
    console.log(category);
    container.insertAdjacentHTML('beforeend', `<li><a data-
category="${category.title}" class="dropdown-item" href="#product-
list">${category.title}</a></li>`);
  }
  li.append(container);
  this.menu.append(li);
}
```

Creamos un evento *onAddCategory()* en el *Controlador* que invoque el método anterior. El evento lo utilizaremos para ser invocado cuando añadamos una nueva categoría en futuras unidades.

```
onAddCategory = () => {
  this[VIEW].showCategoriesInMenu(this[MODEL].categories);
};
```

Hacemos la invocación del evento en el *onLoad()* para hacer la carga de las categorías. Al no alterar el menú con la navegación, lo hacemos al cargar la página, y no lo repetimos hasta que tengamos una nueva categoría.

```
this.onAddCategory();
```

1.4. Eliminar categoría durante la navegación

Queremos modificar la *Vista* para que se borre el listado de categorías, y volver a restaurarlas al cliquear en el enlace de inicio.

En el método *showShoppingCart()* en la *Vista* del módulo *ShoppingCart* genera la tabla resumen del carrito, además de eliminar el contenido del elemento `main`, eliminamos la fila con las categorías.

```
if (this.categories.children.length > 1)
  this.categories.children[1].remove();
```

Tenemos que tener la propiedad *categories* previamente asignada en el constructor.

```
this.categories = document.getElementById('categories');
```

1.5. Listado de productos

Al cliquear en cualquiera de los enlaces de las categorías, debemos mostrar el listado de sus productos. Podemos obtenemos el título de la categoría a partir del atributo personalizado que hemos generado para hacer la búsqueda.

Primero vamos a añadir varios selectores de CSS para generar el diseño del listado.

```css
/* Listado de productos */
.card-product-list,
.card-product-grid {
    margin-bottom: 0
}

.card {
    position: relative;
    display: -webkit-box;
    display: -ms-flexbox;
    display: flex;
    -webkit-box-orient: vertical;
    -webkit-box-direction: normal;
    -ms-flex-direction: column;
    flex-direction: column;
    min-width: 0;
    word-wrap: break-word;
    background-color: #fff;
    background-clip: border-box;
    border: 1px solid rgba(0, 0, 0, 0.1);
    border-radius: 0.10rem;
    margin-top: 10px
}

.card-product-grid:hover {
    -webkit-box-shadow: 0 4px 15px rgba(153, 153, 153, 0.3);
    box-shadow: 0 4px 15px rgba(153, 153, 153, 0.3);
    -webkit-transition: .3s;
    transition: .3s
}

.card-product-grid .img-wrap {
    border-radius: 0.2rem 0.2rem 0 0;
    height: 220px
}

.card .img-wrap {
    overflow: hidden
}
```

```
.card-lg .img-wrap {
    height: 280px
}

.card-product-grid .img-wrap {
    border-radius: 0.2rem 0.2rem 0 0;
    height: 228px;
    padding: 16px
}

[class*='card-product'] .img-wrap img {
    height: 100%;
    max-width: 100%;
    width: auto;
    display: inline-block;
    -o-object-fit: cover;
    object-fit: cover
}

.img-wrap {
    text-align: center;
    display: block
}

.card-product-grid .info-wrap {
    overflow: hidden;
    padding: 18px 20px
}

[class*='card-product'] a.title {
    color: #faa541;
    transition: 0.3s;
    display: block
}

[class*='card-product'] a.title:hover,
[class*='card-product'] a.title:focus  {
    color: #e88006;
    text-decoration: none;
}

.rating-stars {
    display: inline-block;
    vertical-align: middle;
    list-style: none;
    margin: 0;
```

```
    padding: 0;
    position: relative;
    white-space: nowrap;
    clear: both
}

.rating-stars li.stars-active {
    z-index: 2;
    position: absolute;
    top: 0;
    left: 0;
    overflow: hidden
}

.rating-stars li {
    display: block;
    text-overflow: clip;
    white-space: nowrap;
    z-index: 1
}

.card-product-grid .bottom-wrap {
    padding: 18px;
    border-top: 1px solid #e4e4e4
}

.btn {
    display: inline-block;
    font-weight: 600;
    color: #343a40;
    text-align: center;
    vertical-align: middle;
    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    background-color: transparent;
    border: 1px solid transparent;
    padding: 0.45rem 0.85rem;
    font-size: 1rem;
    line-height: 1.5;
    border-radius: 0.2rem
}

.btn-primary {
    color: #fff;
```

```
    background-color: #faa541;
    border-color: #e88006;
}
.btn-primary:hover,
.btn-primary:active,
.btn-primary:focus  {
    border-color: #e88006 !important;
    background-color: #e88006 !important;
}

.bi {
    color: #faa541;
}

.Laptop-style {
    border:2px solid red;
}
.Camera-style {
    border:2px solid green;
}
.Smartphone-style {
    border:2px solid blue;
}
.Tablet-style {
    border:2px solid #faa541;
}
```

Creamos un método para generar la vista con el listado de productos, recibiendo como argumento un iterador con los productos y un título del listado de productos que estamos generando. Por cada producto añadimos un borde diferenciado en función del tipo de producto.

```
listProducts(products, title) {
  this.main.replaceChildren();
  if (this.categories.children.length > 1)
this.categories.children[1].remove();
  const container = document.createElement('div');
  container.id = 'product-list';
  container.classList.add('container');
  container.classList.add('my-3');
  container.insertAdjacentHTML('beforeend', '<div class="row"> </div>');

  for (const product of products) {
    const div = document.createElement('div');
    div.classList.add('col-md-4');
    div.insertAdjacentHTML('beforeend', `<figure class="card card-
product-grid card-lg"> <a data-serial="${product.serial}" href="#single-
product" class="img-wrap"><img class="${product.constructor.name}-style"
src="${product.url}"></a>
        <figcaption class="info-wrap">
          <div class="row">
            <div class="col-md-8"> <a data-serial="${product.serial}"
href="#single-product" class="title">${product.brand} -
${product.model}</a> </div>
            <div class="col-md-4">
              <div class="rating text-right"> <i class="bi bi-star-
fill"></i> <i class="bi bi-star-fill"></i> <i class="bi bi-star-
fill"></i> </div>
            </div>
          </div>
        </figcaption>
        <div class="bottom-wrap">
          <a href="#" data-serial="${product.serial}" class="btn btn-
primary float-end"> Comprar </a>
          <div><span class="price h5">${product.price.toLocaleString('es-
ES', { style: 'currency', currency: 'EUR' })}</span> <br> <small
class="text-success">Free shipping</small></div>
        </div>
      </figure>`);
    container.children[0].append(div);
  }
  container.insertAdjacentHTML('afterbegin', `<h1>${title}</h1>`);
  this.main.append(container);
}
```

En el *Controlador* generamos un manejador que reciba el título de una categoría, obtenga el iterador de productos de la categoría e invoque el método de la vista anterior.

```
handleProductsCategoryList = (title) => {
  const category = this[MODEL].getCategory(title);
  this[VIEW].listProducts(this[MODEL].getCategoryProducts(category),
category.title);
};
```

Vamos a crear dos métodos *bind* que nos permita enlazar el *manejador* con los elementos de la página, uno para el listado de categorías principal y otro para las categorías del menú. Como argumento para pasar al manejador, obtenemos el título de la categoría recuperado del atributo personalizado `data.category`.

```
bindProductsCategoryList(handler) {
  const categoryList = document.getElementById('category-list');
  const links = categoryList.querySelectorAll('a');
  for (const link of links) {
    link.addEventListener('click', (event) => {
      handler(event.currentTarget.dataset.category);
    });
  }
}

bindProductsCategoryListInMenu(handler) {
  const navCats = document.getElementById('navCats');
  const links = navCats.nextSibling.querySelectorAll('a');
  for (const link of links) {
    link.addEventListener('click', (event) => {
      handler(event.currentTarget.dataset.category);
    });
  }
}
```

Invocamos el enlazado desde el evento `onInit()` del *Controlador* para las categorías principales. La razón es porque hay que rehacer el enlace en la vista cada vez que volvamos a añadir las categorías al árbol DOM de la página.

```
this[VIEW].bindProductsCategoryList(
  this.handleProductsCategoryList,
);
```

El enlazado para el menú lo invocamos en el evento *onAddCategory()* para rehacer el enlace cada vez que añadamos o borramos una nueva categoría.

```
this[VIEW].bindProductsCategoryListInMenu(
  this.handleProductsCategoryList,
);
```

1.5.1. Listado de productos por tipo

Para mostrar los productos por tipo reutilizamos el método `listProducts()` porque el resultado es el mismo. Necesitamos un iterador con los productos y un título para mostrarlo como titular.

Creamos un manejador que recibe el tipo de producto que queremos mostrar. Recogemos el tipo para obtener el iterador de productos.

En lugar de utilizar un `switch` para elegir el tipo de instancia, lo refactorizamos en un objeto literal para acceder al tipo de instancia con la notación corchete.

```
handleProductsTypeList = (type) => {
  const instance = {
    Laptop,
    Camera,
    Smartphone,
    Tablet,
  };
  if (instance[type]) {
    this[VIEW].listProducts(this[MODEL].getTypeProducts(instance[type]),
type);
  } else {
    throw new Error(`${type} isn't a type of Product.`);
  }
};
```

Al hacer uso de las funciones constructoras para buscar por tipo de instancia, es necesario importarlas.

```
import {
  Laptop, Camera, Smartphone, Tablet,
} from './manager.js';
```

Asociamos en los enlaces de tipos el manejador anterior en la *Vista*. Al manejador le pasamos como argumento el atributo personalizado `data-type`.

```
bindProductsTypeList(handler) {
  const typeList = document.getElementById('type-list');
  const links = typeList.querySelectorAll('a');
  for (const link of links) {
    link.addEventListener('click', (event) => {
      handler(event.currentTarget.dataset.type);
    });
  }
}
```

Invocamos el método desde el evento `onLoad()` porque los enlaces quedan estáticos en la navegación en la página.

```
this[VIEW].bindProductsTypeList(this.handleProductsTypeList);
```

1.6. Mostrar producto individual

En este último apartado tenemos que mostrar un producto individual. Primero vamos a añadir los selectores CSS que vamos a utilizar.

```css
/* Producto individual */
.card {
    border: none
}

.product {
    background-color: #eee
}

.brand {
    font-size: 13px
}

.act-price {
    color:  #faa541;
    font-weight: 700
}

.dis-price {
    text-decoration: line-through
}

.about {
    font-size: 14px
}

.color {
    margin-bottom: 10px
}

label.radio {
    cursor: pointer
}

label.radio input {
    position: absolute;
    top: 0;
    left: 0;
    visibility: hidden;
    pointer-events: none
}
```

```
label.radio span {
    padding: 2px 9px;
    border: 2px solid #faa541;
    display: inline-block;
    color:  #faa541;
    border-radius: 3px;
    text-transform: uppercase
}

label.radio input:checked+span {
    border-color:  #faa541;
    background-color: #faa541;
    color: #fff
}

.cart i {
    margin-right: 10px
}

#main-image {
    width:100%
}
```

Diseñamos el método que genera la vista para mostrar un producto. El método recibe un producto y un posible mensaje de error como argumentos, este último utilizado en caso de que el producto no exista.

```
showProduct(product, message) {
  this.main.replaceChildren();
  if (this.categories.children.length > 1)
this.categories.children[1].remove();
  const container = document.createElement('div');
  container.classList.add('container');
  container.classList.add('mt-5');
  container.classList.add('mb-5');

  if (product) {
    container.id = 'single-product';
    container.classList.add(`${product.constructor.name}-style`);
    container.insertAdjacentHTML('beforeend', `<div class="row d-flex
justify-content-center">
      <div class="col-md-10">
        <div class="card">
          <div class="row">
            <div class="col-md-6">
              <div class="images p-3">
```

```
                <div class="text-center p-4"> <img id="main-image"
src="${product.url}"/> </div>
              </div>
            </div>
            <div class="col-md-6">
              <div class="product p-4">
                <div class="mt-4 mb-3"> <span class="text-uppercase
brand">${product.brand}</span>
                  <h5 class="text-uppercase">${product.model}</h5>
                  <div class="price d-flex flex-row align-items-center">
                    <span class="act-
price">${product.price.toLocaleString('es-ES', { style: 'currency',
currency: 'EUR' })}</span>
                  </div>
                </div>
                <p class="about">${product.description}</p>
                <div class="sizes mt-5">
                  <h6 class="text-uppercase">Características</h6>
                </div>
                <div class="cart mt-4 align-items-center"> <button data-
serial="${product.serial}" class="btn btn-primary text-uppercase mr-2 px-
4">Comprar</button> </div>
              </div>
            </div>
          </div>
        </div>
      </div>`);
    const characteristics = container.querySelector('h6');
    characteristics.insertAdjacentHTML('afterend',
this.instance[product.constructor.name](product));
  } else {
    container.insertAdjacentHTML(
      'beforeend',
      `<div class="row d-flex justify-content-center">
      ${message}
    </div>`,
    );
  }
  this.main.append(container);
}
```

El método está preparado para mostrar características diferentes en función del tipo de producto, ya que cada uno tiene sus propias propiedades. Para ello tenemos la siguiente propiedad que enumera los métodos privados que personalizan las características.

```
instance = {
  Laptop: this.LaptopCharacteristics,
  Camera: this.CameraCharacteristics,
  Smartphone: this.SmartphoneCharacteristics,
  Tablet: this.TabletCharacteristics,
};

LaptopCharacteristics(product) {
  return (`<div>Características de portátil. ${product.brand}
${product.model}</div>`);
}

CameraCharacteristics(product) {
  return (`<div>Características de cámara. ${product.brand}
${product.model}</div>`);
}

SmartphoneCharacteristics(product) {
  return (`<div>Características de teléfono. ${product.brand}
${product.model}</div>`);
}

TabletCharacteristics(product) {
  return (`<div>Características de tablet. ${product.brand}
${product.model}</div>`);
}
```

En el *Controlador* añadimos un manejador para obtener el objeto `Product`. Para seleccionarlo recibimos el número de serie.

```
handleShowProduct = (serial) => {
  try {
    const product = this[MODEL].getProduct(serial);
    this[VIEW].showProduct(product);
  } catch (error) {
    this[VIEW].showProduct(null, 'No existe este producto en la
página.');
  }
};
```

En la *Vista* creamos el método que enlace los eventos del usuario con el manejador del *Controlador*. Tenemos eventos para acceder desde los enlaces y desde las imágenes de cada producto del listado.

```
bindShowProduct(handler) {
  const productList = document.getElementById('product-list');
  const links = productList.querySelectorAll('a.img-wrap');
  for (const link of links) {
    link.addEventListener('click', (event) => {
      handler(event.currentTarget.dataset.serial);
    });
  }
  const images = productList.querySelectorAll('figcaption a');
  for (const link of links) {
    link.addEventListener('click', (event) => {
      handler(event.currentTarget.dataset.serial);
    });
  }
}
```

Por último, tenemos que invocar el método *bind* anterior. Los tenemos que hacer en dos métodos una vez invocado el método de la vista. Primero en el manejador `handleProductsCategoryList()` que genera el listado de productos a partir de una categoría, y en el manejador `handleProductsTypeList()` para obtener el listado en función del tipo.

```
this[VIEW].bindShowProduct(this.handleShowProduct);
```