

0. Instalar Angular Material

<https://material.angular.io/guide/getting-started>

- 0.1. `ng add @angular/material`

elijo el tema indigo/pink
y le doy a YES para las otras dos preguntas que me hacen

- 0.2. Vamos a crear un módulo para importar en él, todas las cosas de Angular Material más frecuentemente usadas para tenerlas a mano ya preparadas

`ng g m modules/angular-material`

- 0.3. Añadimos a `angular-material.module.ts` lo siguiente:

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MatButtonModule } from '@angular/material/button';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatInputModule } from '@angular/material/input';
import { MatProgressSpinnerModule } from
'@angular/material/progress-spinner';
import { MatCardModule } from '@angular/material/card';
import { MatMenuModule } from '@angular/material/menu';
import { MatIconModule } from '@angular/material/icon';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatListModule } from '@angular/material/list';
import { MatProgressBarModule } from '@angular/material/progress-bar';

@NgModule({
  declarations: [],
  imports: [MatProgressBarModule, MatListModule, MatSidenavModule,
MatButtonModule,
MatCheckboxModule,MatToolbarModule,MatInputModule,MatProgressSpinnerMod
ule,MatCardModule,MatMenuModule,MatIconModule, CommonModule],
  exports: [MatProgressBarModule, MatListModule, MatSidenavModule,
MatButtonModule,
MatCheckboxModule,MatToolbarModule,MatInputModule,MatProgressSpinnerMod
ule,MatCardModule,MatMenuModule,MatIconModule, CommonModule]
})
export class AngularMaterialModule { }
```

- 0.4. Y ahora importamos este módulo, en el módulo principal de app.module.ts

```
imports: [  
  BrowserModule,  
  BrowserAnimationsModule,  
  AngularMaterialModule  
,
```

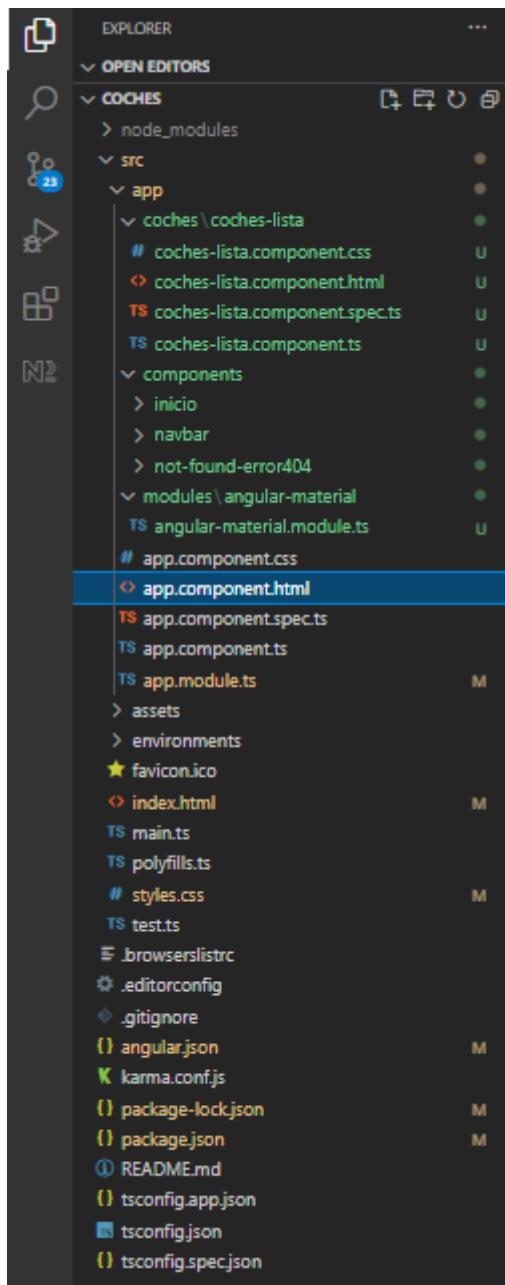
- Ya está listo Angular Material para usarse !!

<https://material.angular.io/components/categories>

1. Routing: Introducción y configuración básica

- 1.1. Creamos los componentes:

- 1.1.1. ng g c components/navbar
- 1.1.2. ng g c components/inicio
- 1.1.3. ng g c components/not-found-error404
- 1.1.4. ng g c coches/coches-lista



- 1.2. Vamos a montar un navbar con Angular Material

1.2.1. Vamos a components/navbar.component.html

1.2.2. Tomamos de ejemplo el primer toolbar que nos encontramos y lo pegamos en nuestro navbar.component.html

<https://material.angular.io/components/toolbar/examples>

```
<!-- <p>navbar works!</p> -->

<mat-toolbar> <!-- color="primary"-->
  <mat-toolbar-row>
    <button mat-icon-button>
      <mat-icon (click)="sidenav.toggle();">menu</mat-icon>
    </button>

    <h1>Angular Material Sidenav</h1>

    <span class="example-spacer"></span>

    <button mat-icon-button class="example-icon favorite-icon">
      <mat-icon>favorite</mat-icon>
    </button>

    <button mat-icon-button class="example-icon">
      <mat-icon>share</mat-icon>
    </button>
  </mat-toolbar-row>
</mat-toolbar>

<mat-sidenav-container>
  <mat-sidenav #sidenav opened="false" mode="side">
    <mat-nav-list>
      <a mat-list-item>Inicio</a>
      <a mat-list-item>Coches</a>
    </mat-nav-list>
  </mat-sidenav>
  <mat-sidenav-content>
    <div style="height: 88vh;">
      <router-outlet></router-outlet>
    </div>
  </mat-sidenav-content>
</mat-sidenav-container>

<!-- https://www.youtube.com/watch?v=IRzGphrefsY -->
```

Nota: Aquí me da error el tag de <router-outlet>, pero no pasa nada, es porque al crear el proyecto, no se nos ha creado predeterminadamente el archivo app-routing.module.ts, en el cual deberemos de importar y exportar el RouterModule, así que, de momento, ignoramos el error

1.2.3. Y en el navbar.component.css ponemos...

```
.example-spacer {  
    flex: 1 1 auto;  
}  
  
.active-link {  
    color: orange;  
}
```

1.2.7. El siguiente paso, será el de ir al app.component.html y borrarlo todo, para poner:

```
<app-navbar></app-navbar>  
  
<router-outlet></router-outlet>
```

- 1.3. En este punto, desarrollaremos el componente coches-lista.
Para ello, previamente crearemos el modelo de datos CochesModel (coches.model.ts) y un archivo mock data (mocks.ts) con un conjunto de coches de prueba.

1.3.1. Creamos la carpeta models

Dentro de ella, creamos el la interface coches-model.ts

ng g interface interfaces/coches-model

```
export interface CochesModel {  
    id: number;  
    marca: string;  
    modelo: string;  
    color: string;  
    cv: number;  
    descripcion: string;  
    mensaje: string;  
    equipamiento: {  
        starStop: string,  
        sensoresLucesLluvia: string,  
    }  
}
```

```

        puertosUSB: string,
        navegador: string,
        ledsDiurnas: string,
        lunasTintadas: string,
        frenosABS: string,
        llantasAleacion: string,
        climatizador: string,
        camaraTrasera: string,
        bluetooth: string,
        direccionAsistida: string,
    } ;
    ranking: number;
    imagen: string;
    logo: string;
    precio: number;
}

```

1.3.2. Creamos la carpeta mocks

Dentro de ella, creamos el archivo videojuego-mock.ts

```

import { CochesModel } from "../interfaces/coches-model";

export const COCHES: CochesModel[] = [
    {
        id: 0,
        marca: "BMW",
        modelo: "SERIE 7 730D",
        color: "Gris",
        cv: 260,
        descripcion: "*TAE: 9,57%. TIN: 8,75%. Importe a financiar: 46.950,00 €. 36 meses. 35 cuotas de 634,97 €/mes. Importe total adeudado: 58.276,86 €. Precio total a plazos financiando con BMW Bank: 86.446,47 € (cumpliendo condiciones). Condiciones válidas para operaciones pagadas hasta 31/12/2021. Precio de venta al contado BMW Serie 7 730d: 68.500 € (descuento e impuestos incluidos)",
        mensaje: "Apenas sale a la carretera, el BMW Serie 3 deja todo atrás, tanto las convenciones como las dudas. Este ícono muestra una vez más cómo reinventarse completamente. Con su elegante lenguaje de diseño representa el comienzo de una nueva era.",
        equipamiento: {
            starStop: "Start & Stop",
            sensoresLucesLluvia: "Sensores de Luces y Lluvia",
            puertosUSB: "Puertos USB",
        }
    }
]

```

```
navegador: "Navegador",
ledsDiurnas: "Luces Leds Diurnas",
lunasTintadas: "Lunas Traseras Tintadas",
frenosABS: "Frenos ABS",
llantasAleacion: "Llantas de Aleación",
climatizador: "Climatizador",
camaraTrasera: "Cámara Trasera",
bluetooth: "Bluetoooh",
direccionAsistida: "Dirección Asistida",
},
ranking: 7,
imagen: "../../assets/BMW.jpg",
logo: "../../assets/BMWLogo.jpg",
precio: 87920000,
},
{
id: 1,
marca: "AUDI",
modelo: "A6 ALLROAD 3.0 TDI S-TRONIC 5P",
color: "Blanco",
cv: 313,
descripcion: "*TAE: 9,57%. TIN: 8,75%. Importe a financiar: 46.950,00 €. 36 meses. 35 cuotas de 634,97 €/mes. Importe total adeudado: 58.276,86 €. Precio total a plazos financiando con BMW Bank: 86.446,47 € (cumpliendo condiciones). Condiciones válidas para operaciones pagadas hasta 31/12/2021. Precio de venta al contado BMW Serie 7 730d: 68.500 € (descuento e impuestos incluidos)",
mensaje: "Cuando el diseño se une a la eficiencia y la deportividad al estilo. Cuando la innovación y la precisión van de la mano y la elegancia se puede sentir en cada línea. Cuando la forma y la función son una declaración de intenciones y el progreso está por encima de todo. Así es el nuevo referente de la clase Business: el Audi A6.",
equipamiento: {
starStop: "Start & Stop",
sensoresLucesLluvia: "Sensores de Luces y Lluvia",
puertosUSB: "Puertos USB",
navegador: "Navegador",
ledsDiurnas: "Luces Leds Diurnas",
lunasTintadas: "Lunas Traseras Tintadas",
frenosABS: "Frenos ABS",
llantasAleacion: "Llantas de Aleación",
climatizador: "Climatizador",
```

```
        camaraTrasera: "Cámara Trasera",
        bluetoooh: "Bluetoooh",
        direccionAsistida: "Dirección Asistida",
    },
    ranking: 10,
    imagen: "../../assets/Audi.jpg",
    logo: "../../assets/AudiLogo.jpg",
    precio: 90000000,
},
{
    id: 2,
    marca: "MERCEDES",
    modelo: "CLASE E 220 COUPE MANUAL 2.1",
    color: "Negro",
    cv: 170,
    descripcion: "*TAE: 9,57%. TIN: 8,75%. Importe a financiar: 46.950,00 €. 36 meses. 35 cuotas de 634,97 €/mes. Importe total adeudado: 58.276,86 €. Precio total a plazos financiando con BMW Bank: 86.446,47 € (cumpliendo condiciones). Condiciones válidas para operaciones pagadas hasta 31/12/2021. Precio de venta al contado BMW Serie 7 730d: 68.500 € (descuento e impuestos incluidos)",

    mensaje: "La nueva Clase E Estate es una opción idónea para afrontar con máxima elegancia los desafíos de la conducción a diario. Con un diseño más dinámico en todos sus detalles, un puesto de conducción nuevo, de manejo intuitivo, y un espacio de carga más generoso que nunca.",

    equipamiento: {
        starStop: "Start & Stop",
        sensoresLucesLluvia: "Sensores de Luces y Lluvia",
        puertosUSB: "Puertos USB",
        navegador: "Navegador",
        ledsDiurnas: "Luces Leds Diurnas",
        lunasTintadas: "Lunas Traseras Tintadas",
        frenosABS: "Frenos ABS",
        llantasAleacion: "Llantas de Aleación",
        climatizador: "Climatizador",
        camaraTrasera: "Cámara Trasera",
        bluetoooh: "Bluetoooh",
        direccionAsistida: "Dirección Asistida",
    },
    ranking: 8,
    imagen: "../../assets/Mercedes.jpg",
    logo: "../../assets/MercedesLogo.jpg",
```

```
        precio: 90000000,  
    },  
]
```

- 1.4. Vamos coches-lista.component.ts

- 1.4.1. Creamos una propiedad llamada coches que es un Array del CochesModel
- 1.4.2. En el ngOnInit() llamamos a this.coches = COCHES;
- para que al iniciarse la app, siempre se despliegue la lista de coches que hemos definido en el coches-mock.ts

```
export class CochesListaComponent implements OnInit {  
  
    coches: CochesModel[] = [];  
  
    constructor() { }  
  
    ngOnInit(): void {  
        this.coches = COCHES;  
    }  
}
```

- 1.4.3. Ahora en el coches-lista.component.html vamos a copiar y pegar un modelo de Card desde Angular Material, para desplegar con la directiva *ngFor todos los coches:

```
<p>coches-lista works!</p>  
  
<div class="container">  
    <ngIf="coches != null || coches != undefined">  
        <mat-card class="example-card" *ngFor="let coche of coches">  
            <mat-card-header>  
                <!-- <div mat-card-avatar class="example-header-image"></div>  
            -->  
                  
  
            <mat-card-content>
```

```

<mat-card-subtitle>{{ coche.mensaje }}</mat-card-subtitle>
</mat-card-content>

<mat-card-actions>
    <button mat-button>ID: {{ coche.id }}</button>
    <button mat-button>Ranking: {{ coche.ranking }}</button>
    <button mat-button>LIKE</button>
    <button mat-button [routerLink]="['/coches', coche.id]" routerLinkActive="router-link-active">BUY</button>
</mat-card-actions>
</mat-card>
</div>

```

1.4.4. Y en coches.component.css añadimos esto...

```

.example-card {
    max-width: 350px;
    margin: 15px;
}

/* .example-header-image {
    background-image:
url('https://material.angular.io/assets/img/examples/shiba1.jpg');
    background-size: cover;
} */

.container {
    padding: 1px;
    display: flex;
    flex-wrap: wrap;
}

```

- 1.5. Una vez creados los componentes, pasaremos a realizar la configuración del servicio Router, y para ello vamos al app-routing.module.ts para crear nuestras primeras rutas a nuestros componentes:

Nota: Cuando creé el proyecto, no se me creó el app-routing.module.ts, así que tengo que crearlo yo mismo ahora ejecutando el comando [ng g m app-routing --flat -m app]

```

import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

```

```

import { CochesListaComponent } from
'./coches/coches-lista/coches-lista.component';
import { InicioComponent } from './components/inicio/inicio.component';
import { NotFoundError404Component } from
'./components/not-found-error404/not-found-error404.component';
// import { CommonModule } from '@angular/common';

const routes: Routes = [
  {
    path: 'inicio',
    component: InicioComponent
  },
  {
    path: 'coches',
    component: CochesListaComponent
  },
  {
    path: '',
    redirectTo: '/inicio',
    pathMatch: 'full'
  },
  {
    path: '**',
    component: NotFoundError404Component
  }
];

@NgModule({
  declarations: [],
  imports: [
    // CommonModule
    RouterModule.forRoot(routes)
  ],
  exports: [
    RouterModule
  ]
})
export class AppRoutingModule { }

```

Nota: Ahora que en el NgModule del app-routing.module.ts ya hemos importado y exportado el RouterModule, ya sí que Angular puede reconocer el tag de <router-outlet> y por tanto ya no nos da error.

2. Routing: RouterLinks

- 2.1. Volvemos al navbar.component.html para retocar nuestro sidebar:

```
<mat-sidenav-container>
  <mat-sidenav #sidenav opened="false" mode="side">
    <mat-nav-list>
      <a mat-list-item routerLinkActive="active-link"
        routerLink="/inicio">Inicio</a>
      <a mat-list-item routerLinkActive="active-link"
        routerLink="/coches">Coches</a>
    </mat-nav-list>
  </mat-sidenav>
  <mat-sidenav-content>
    <div style="height: 88vh;">
      <router-outlet></router-outlet>
    </div>
  </mat-sidenav-content>
</mat-sidenav-container>
```

- 2.2. Llegados a este punto, el componente de coches-lista debería verse así:

The screenshot displays a mobile application interface for a car listing. At the top, there's a blue header bar with the text "Coches Angular App". Below the header, there are three cards, each representing a different car model:

- BMW SERIE 7 730d**: A silver BMW 7 Series sedan. Below the car image, there is small text about financing terms: "*TAE: 9,57%. TIN: 8,75%. Importe a financiar: 46.950,00 €. 36 meses. 35 cuotas de 634,97 €/mes. Importe total adeudado: 58.276,86 €. Precio total a plazos financiando con BMW Bank: 86.446,47 € (cumpliendo condiciones). Condiciones válidas para operaciones pagadas hasta 31/12/2021. Precio de venta al contado BMW Serie 7 730d: 68.500 € (descuento e impuestos incluidos)". At the bottom of the card, there are buttons for "ID: 0", "Ranking: 7", "LIKE", and "SHARE".
- AUDI A6 ALLROAD 3.0 TDI S-TRONIC 5P**: A white Audi A6 Allroad. Below the car image, there is small text: "*TAE: 9,57%. TIN: 8,75%. Importe a financiar: 46.950,00 €. 36 meses. 35 cuotas de 634,97 €/mes. Importe total adeudado: 58.276,86 €. Precio total a plazos financiando con BMW Bank: 86.446,47 € (cumpliendo condiciones). Condiciones válidas para operaciones pagadas hasta 31/12/2021. Precio de venta al contado BMW Serie 7 730d: 68.500 € (descuento e impuestos incluidos)". At the bottom of the card, there are buttons for "ID: 1", "Ranking: 10", "LIKE", and "SHARE".
- MERCEDES CLASE E 220 COUPE MANUAL 2.1**: A dark grey Mercedes-Benz E-Class Coupe. Below the car image, there is small text: "*TAE: 9,57%. TIN: 8,75%. Importe a financiar: 46.950,00 €. 36 meses. 35 cuotas de 634,97 €/mes. Importe total adeudado: 58.276,86 €. Precio total a plazos financiando con BMW Bank: 86.446,47 € (cumpliendo condiciones). Condiciones válidas para operaciones pagadas hasta 31/12/2021. Precio de venta al contado BMW Serie 7 730d: 68.500 € (descuento e impuestos incluidos)". At the bottom of the card, there are buttons for "ID: 2", "Ranking: 8", "LIKE", and "SHARE".

3. Routing: Rutas con parámetros y ActivatedRoute

- 3.1. Creamos el componente de coches-detalles
ng g c coches/coches-detalles
- 3.2. Añadimos una ruta para este componente

```
const routes: Routes = [
  {
    path: 'inicio',
    component: InicioComponent
  },
  {
    path: 'coches',
    component: CochesListaComponent
  },
  {
    path: 'coches/:id',
    component: CochesDetallesComponent
  },
  {
    path: '',
    redirectTo: '/inicio',
    pathMatch: 'full'
  },
  {
    path: '**',
    component: NotFoundError404Component
  }
];
```

En el componente coches-detalles añadiremos el código para primero obtener el identificador del coche (parámetro “:id” del URL) y luego para leer el detalle de ese coche.

- 3.3. Vamos al coches-detalles.component.ts para:
 - 3.3.1. Crear una propiedad llamada coche que siga el CocheModel
 - 3.3.2. Inyectamos la ActivatedRoute en el constructor
 - 3.3.3. Para obtener los parámetros de la ruta, ActivateRoute nos proporciona el observable paramMap. Para obtenerlos, simplemente nos suscribimos al observable a la espera de recibirlas. De esta manera recibiremos el identificador de coche.

```
export class CochesDetallesComponent implements OnInit {
```

```

coche: CochesModel | undefined;

constructor(
  private activatedRoute: ActivatedRoute
) { }

ngOnInit(): void {
  this.activatedRoute.paramMap
    .subscribe((paramMaps: ParamMap) => {
      let id = Number(paramMaps.get('id'));
      this.coche = COCHES[id];
      this.coche = COCHES.find((item) => item.id === id);
    })
}
}

```

- 3.4. En el template de coches-detalles.component.html añadiremos el código para visualizar el detalle del coche

```

<p>coches-detalles works!</p>

<div class="superContainer">

  <div class="container" *ngIf="coche != null || coche != undefined">

    <mat-card class="example-card">
      <mat-card-subtitle>ID: {{ coche.id }}</mat-card-subtitle>
      <mat-card-title>{{ coche.marca }}</mat-card-title>
      <mat-card-title>{{ coche.modelo }}</mat-card-title>

      <mat-card-content>
        <p>{{ coche.descripcion }}</p>
      </mat-card-content>

      <mat-divider inset></mat-divider>

    <mat-card class="example-card">
        <mat-card-title>Equipamiento:</mat-card-title>
        <mat-divider inset></mat-divider>
        <br>

        <mat-card-subtitle>{{ coche.equipamiento.starStop }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.sensoresLucesLluvia }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.navegador }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.ledsDiurnas }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.lunasTintadas }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.frenosABS }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.llantasAleacion }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.climatizador }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.camaraTrasera }}</mat-card-subtitle>
        <mat-card-subtitle>{{ coche.equipamiento.bluetooth }}</mat-card-subtitle>

        <mat-divider inset></mat-divider>
        <br>
        <mat-card-title>Precio: {{ coche.precio }}</mat-card-title>

        <mat-card-actions>
            <button mat-button>LIKE</button>
            <button mat-button>SHARE</button>
            <button mat-button>Rank: {{ coche.ranking }}</button>
        </mat-card-actions>

        <mat-card-footer>
            <mat-progress-bar mode="indeterminate"></mat-progress-bar>
```

```
        <mat-progress-bar  
mode="indeterminate"></mat-progress-bar>  
      </mat-card-footer>  
  
    </mat-card>  
  </div>  
  
</div>
```

- 3.5. Y, finalmente, añadiremos un enlace o link al componente coches-detalles para cada uno de los coches dentro del coches-lista.component.html:

```
<a [routerLink]=["/coches", coche.id]">  
  

  <div class="container" *ngIf="coche != null || coche != undefined">

    <mat-card class="example-card">
      <mat-card-subtitle>ID: {{ coche.id }}</mat-card-subtitle>
      <mat-card-title>{{ coche.marca }}</mat-card-title>
      <mat-card-title>{{ coche.modelo }}</mat-card-title>

      <mat-card-content>
        <p>{{ coche.descripcion }}</p>
      </mat-card-content>

      <mat-divider inset></mat-divider>

    </mat-card>
  </div>

  <div class="container" *ngIf="coche != null || coche != undefined">

    <mat-card class="example-card">
      <mat-card-title>Equipamiento:</mat-card-title>
      <mat-divider inset></mat-divider>
      <br>

      <mat-card-subtitle>{{ coche.equipamiento.starStop }}&#10004;</mat-card-subtitle>
    
```

```
        <mat-card-subtitle>{{ coche.equipamiento.sensoresLucesLluvia }} &#10004;</mat-card-subtitle>
            <mat-card-subtitle>{{ coche.equipamiento.navegador }} &#10004;</mat-card-subtitle>
                <mat-card-subtitle>{{ coche.equipamiento.ledsDiurnas }} &#10004;</mat-card-subtitle>
                    <mat-card-subtitle>{{ coche.equipamiento.lunasTintadas }} &#10004;</mat-card-subtitle>
                        <mat-card-subtitle>{{ coche.equipamiento.frenosABS }} &#10004;</mat-card-subtitle>
                            <mat-card-subtitle>{{ coche.equipamiento.llantasAleacion }} &#10004;</mat-card-subtitle>
                                <mat-card-subtitle>{{ coche.equipamiento.climatizador }} &#10004;</mat-card-subtitle>
                                    <mat-card-subtitle>{{ coche.equipamiento.camaraTrasera }} &#10004;</mat-card-subtitle>
                                        <mat-card-subtitle>{{ coche.equipamiento.bluetooth }} &#10004;</mat-card-subtitle>

            <mat-divider inset></mat-divider>
            <br>
            <mat-card-title>Precio: {{ coche.precio }}</mat-card-title>

            <mat-card-actions>
                <button mat-button>LIKE</button>
                <button mat-button>SHARE</button>
                <button mat-button>Rank: {{ coche.ranking }}</button>
            </mat-card-actions>

            <mat-card-footer>
                <mat-progress-bar mode="indeterminate"></mat-progress-bar>
                <mat-progress-bar mode="indeterminate"></mat-progress-bar>
            </mat-card-footer>

        </mat-card>
    </div>

    <mat-selection-list class="mat-list" #videojuegos [multiple]="false">
        <mat-list-option routerLinkActive="active-link" routerLink='imagenes'>
```

```

    Imágenes
    </mat-list-option>

    <mat-list-option routerLinkActive="active-link"
routerLink='opiniones'>
    Opiniones
    </mat-list-option>

    <div class="router-outlet">
        <router-outlet></router-outlet>
    </div>

    </mat-selection-list>

</div>

```

4.4. A continuación, añadiremos el código necesario en coches-imagenes para cargar el identificador de coche que serviría al componente para cargar las imágenes relacionadas.

Para ello, haremos uso de ActivatedRoute primero para acceder al ActivatedRoute parent, y luego para obtener el valor del parámetro.

4.4.1. Creamos la propiedad idCoche de tipo number, y lo ponemos como undefined

4.4.2. Inyectamos el ActivatedRoute en el constructor

4.4.3. Obtenemos el id de cada coche con ActivatedRoute

```

export class CochesImagenesComponent implements OnInit {

    idCoche: number | undefined;

    constructor(
        private activatedRoute: ActivatedRoute
    ) { }

    ngOnInit(): void {
        this.activatedRoute.parent?. paramMap
            .subscribe((paramMaps: ParamMap) => {
                this.idCoche = Number(paramMaps.get('id'));
            })
    }
}

```

- 4.5. Y en coches-imagenes.component.html ponemos...

```
<!-- <p>coches-imagenes works!</p> -->

<p>
    (Imagenes del libro con identificador: {{idCoche}})
</p>
```

5. Extras

- 5.1. Vamos a conseguir mostrar las imágenes de cada coche
- 5.2. Vamos al coche-imagenes.component.ts para añadir:

- 5.2.1. Una propiedad llamada coche que es un objeto del CochesModel indefinido
- 5.2.2. Obtenemos el id de ese coche

```
export class CochesImagenesComponent implements OnInit {

  idCoche: number | undefined;

  coche: CochesModel | undefined;

  constructor(
    private activatedRoute: ActivatedRoute
  ) { }

  ngOnInit(): void {
    this.activatedRoute.parent?. paramMap
      .subscribe((paramMaps: ParamMap) => {
        this.idCoche = Number(paramMaps.get('id'));

        let idCoche = Number(paramMaps.get('id'));
        this.coche = COCHES[idCoche];
      })
  }
}
```

5.2.3. En el coches-imagenes.component.html añadimos lo siguiente para mostrar la portada de cada coche

```
<!-- <p>coches-imagenes works!</p> -->

<p>
    (Imagenes del libro con identificador: {{idCoche}})
</p>

<div class="container mt-3" *ngIf="coche != null || coche != undefined">
    
</div>
```

5.2.3. En el coches-imagenes.component.css añadimos lo siguiente para mostrar un poco mejor la imagen de cada coche

```
.coches-imagenes-img {
    width: 500px;
    height: 400px;
    display: flex;
    flex-wrap: wrap;
}
```

The screenshot shows a car listing for a BMW Serie 7 730D. On the left, there's a large image of the silver sedan. To its right, the car's name 'BMW SERIE 7 730D' is displayed in bold. Below the name, there's a financing offer: 'Importe a financiar: 46.950,00 €, 36 meses, 35 cuotas de 634,97 €/mes. Precio total a plazos financiando con BMW Bank: 86.446,47 € (cumpliendo condiciones). Precio de venta al contado BMW Serie 7 730d: 68.500 € (descuento e impuestos incluidos)'.

On the right side of the screen, there are two columns. The first column, titled 'Equipamiento:', lists various car features with checkmarks: Start & Stop ✓, Sensores de Luces y Lluvia ✓, Navegador ✓, Luces Leds Diurnas ✓, Lunas Traseras Tintadas ✓, Frenos ABS ✓, Llantas de Aleación ✓, Climatizador ✓, Cámara Trasera ✓, and Bluetooth ✓. Below this list, the price 'Precio: 87920000' is shown, followed by 'LIKE', 'SHARE', and 'Rank: 7'.

The second column, titled 'Imágenes', contains the text '(Imagenes del libro con identificador: 0)' and a placeholder image of the same silver BMW sedan.

- 5.3. Vamos a cambiar el componente de las imágenes para poner un efecto a modo de “slider rotatorio en 3D”

https://www.youtube.com/watch?v=WNhpXJw_730&list=PL5e68IK9hEzfTVV5L4q-tf3OhJ6vDtZ6a&index=8

5.3.1. En el coches-imagenes.component.html creamos esta estructura

```
<!-- <p>coches-imagenes works!</p> -->

<div class="container">
  <p>
    (Imagenes del libro con identificador: {{idCoche}})
  </p>

  <!-- <div *ngIf="coche != null || coche != undefined">
    
  </div> -->

  <div class="carousel" *ngIf="coche != null || coche != undefined">
    <div class="slideshow">
      <div class="slide">
        
        
        
        
        
        
            
            
            
             {
        let id = Number(paramMaps.get('id'));
        this.coche = COCHES[id];
      })
    // una vez que hemos definido el método para las opiniones, lo
    llamamos aquí
    this.opinionesDeCadaCoche();
  }

  opinionesDeCadaCoche(): void {

```

```

        this.opiniones = OPINIONES.filter(
            (item) => item.idCoche === this.coche?.id
        );
    }
}

```

5.4.5. Luego vamos al coches-opiniones.component.html para insertar esto:

```

<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"
      integrity="sha512-iBBXm8fW90+nuLcSKlbmrPcLa0OT92x01BIzZ+ywDWZCvqsWgccV3gFoRBv0z+8dLJgyAHIhR35VZc2oM/gI1w=="
      crossorigin="anonymous"
      referrerPolicy="no-referrer" />

<!-- <p>coches-opiniones works!</p> -->

<div class="container" *ngIf="opiniones != null || opinione != undefined">

    <div class="slide-container active" *ngFor="let opinion of
opiniones">
        <div class="slide">
            <div class="fas fa-quote-right icon"></div>
            <div class="user">
                
                <div class="user-info">
                    <h3>{{ opinion.titulo }}</h3>
                    <h5>{{ opinion.nombre }}</h5>
                    <h6><i>Nickname:</i> {{ opinion.nickname }}</h6>
                    <h6>Puntuación: {{ opinion.puntuacion }}</h6>
                    <div class="stars">
                        <i class="fas fa-star"></i>
                        <i class="fas fa-star"></i>
                        <i class="fas fa-star"></i>
                        <i class="fas fa-star"></i>
                        <i class="far fa-star"></i>
                    </div>
                </div>
            </div>
        </div>
        <p class="text">{{ opinion.descripcion }}</p>
    
```

```
        </div>
    </div>

</div>
```

5.4.6. Por último, vamos al coches-opiniones.component.css para insertar esto:

```
.container {
    position: relative;
    perspective: 1000px;
}

.container .slide-container .slide {
    border-radius: 5px;
    background: whitesmoke;
    box-shadow: 0 5px 10px #3337;
    width: 450px;
    padding: 12.5px;
    margin: 10px;
    position: relative;
    transform-style: preserve-3d;
}

.container .slide-container .slide .icon {
    position: absolute;
    top: 10px;
    right: 15px;
    font-size: 70px;
    color: #3498db;
}

.container .slide-container .slide .user {
    display: flex;
    align-items: center;
}

.container .slide-container .slide img {
    width: 100px;
    height: 100px;
    border-radius: 50%;
```

```
object-fit: cover;
margin-right: 10px;
}

.container .slide-container .slide .user .user-info h3 {
    color: #333;
    font-size: 20px;
}

.container .slide-container .slide .user .user-info h5,
.container .slide-container .slide .user .user-info h6 {
    color: #333;
    font-size: 14px;
    margin-top: -12px;
}

.container .slide-container .slide .user .user-info .stars i {
    color: #3498db;
    font-size: 15px;
}

.container .slide-container .slide .text {
    color: #333;
    font-size: 14px;
    padding-top: 5px;
    font-style: italic;
}

.container .slide-container {
    display: none;
}

.container .slide-container.active {
    display: block;
}

.container #next,
.container #prev {
    position: absolute;
    top: 50%;
    transform: translateY(-50%);
    height: 50px;
    width: 50px;
}
```

```

        line-height: 50px;
        text-align: center;
        font-size: 20px;
        background: #fff;
        color: #333;
        cursor: pointer;
        border-radius: 50%;
        box-shadow: 0 5px 10px #3337;
    }

.container #next {
    right: -70px;
}

.container #prev {
    left: -70px;
}

.container #next:hover,
.container #prev:hover {
    background: #333;
    color: #fff;
}

```

≡ Coches Angular App Heart Share

ID: 0

BMW SERIE 7 730D

Importe a financiar: 46.950,00 €. 36 meses. 35 cuotas de 634,97 €/mes. Precio total a plazos financiando con BMW Bank: 86.446,47 € (cumpliendo condiciones). Precio de venta al contado BMW Serie 7 730d: 68.500 € (descuento e impuestos incluidos)



Equipamiento:

- Start & Stop ✓
- Sensores de Luces y Lluvia ✓
- Navegador ✓
- Luces Leds Diurnas ✓
- Lunas Traseras Tintadas ✓
- Frenos ABS ✓
- Llantas de Aleación ✓
- Climatizador ✓
- Cámara Trasera ✓
- Bluetooth ✓

Precio: 87920000

LIKE SHARE Rank: 7

Imágenes

Opiniones

Me encanta este BMW



Álvaro Esmoris
Nickname: VaroMoris
Puntuación: 9

★★★★★

Cuando sea programador senior, me comprará uno como éste

Salen muy bueno este modelo



D. José Mª García
Nickname: Polavieja
Puntuación: 7

★★★★★

Tenía uno como este y me duró muchos años

6. Elegir opciones de Equipamientos Extras

Ahora vamos a añadir un par de nuevas interfaces (componentes) más.

Uno será para que el usuario elija el color/motor/xxx, aumentando así la base imponible del precio del coche en x%; y también crearemos una última interfaz para el resumen de la posible futura compra.

<https://www.youtube.com/watch?v=855KrFfF9-w>

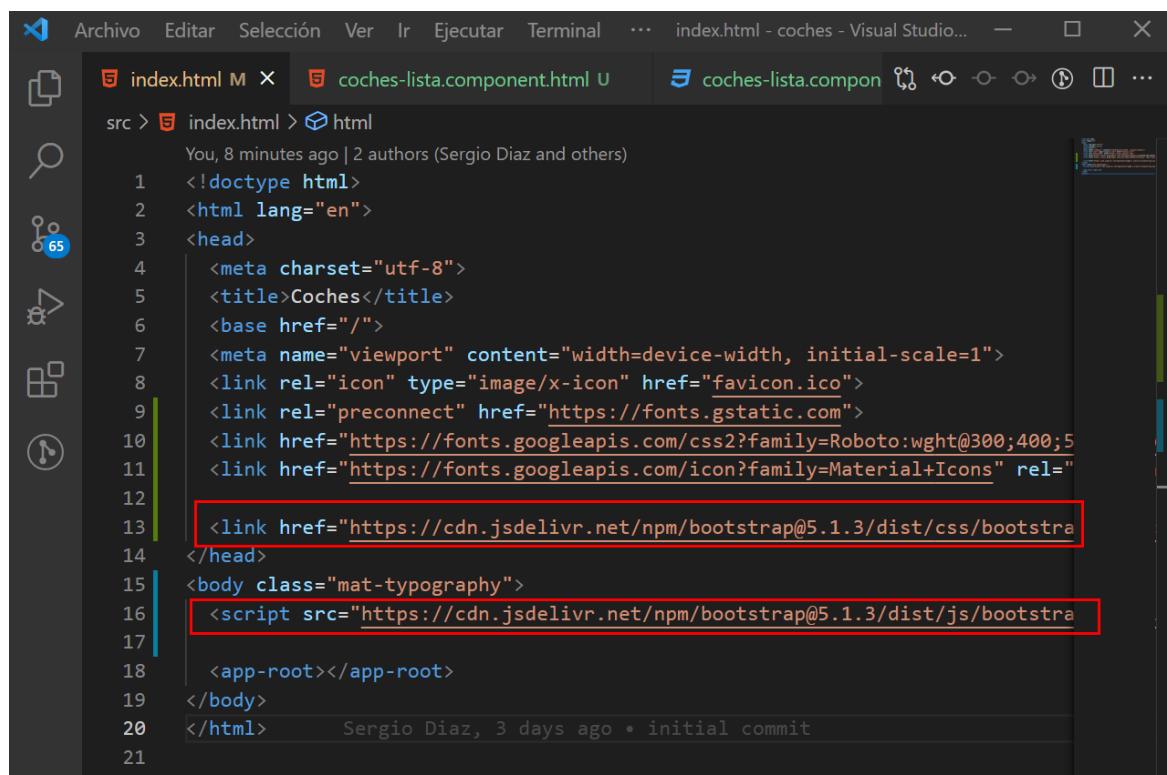
- 6.1. Vamos a la web de Bootstrap para copiar sus CDNs, tanto el del css

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFlvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jlW3" crossorigin="anonymous">
```

... como el del js ...

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MIQnikT1wXgYsOg+OMhuP+lRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>
```

... y ambos, los pegamos en el index.html en estos, el del css antes de que acabe el tag </head>, y el del js al principio del tag <body>



```
index.html M X coches-lista.component.html U coches-lista.compon 65
src > index.html > html
      You, 8 minutes ago | 2 authors (Sergio Diaz and others)
1   <!doctype html>
2   <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Coches</title>
6     <base href="/">
7     <meta name="viewport" content="width=device-width, initial-scale=1">
8     <link rel="icon" type="image/x-icon" href="favicon.ico">
9     <link rel="preconnect" href="https://fonts.gstatic.com">
10    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap" rel="stylesheet">
11    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
12
13    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
14  </head>
15  <body class="mat-typography">
16    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MIQnikT1wXgYsOg+OMhuP+lRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>
17
18  <app-root></app-root>
19 </body>
20 </html> Sergio Diaz, 3 days ago • initial commit
21
```

Nota: A pesar de haber usado CDNs, nosotros sabemos perfectamente instalar Bootstrap o NgBootstrap con npm ... y sabemos insertar las direcciones hacia tales archivos en el Angular.json, en su sección de styles: []

Nota: Al hacer esto, he podido comprobar que bootstrap estaba cogiendo las clases de .container que había en el coches-lista.component.html y coches-detalles.component.html, así que, en el coches-lista.component.html he tenido que renombrar la clase container a superContainer, y quitar el botón de Like de la card

Por otro lado, en el coches-detalles.component.html he renombrado también container a superContainer.

- 6.2. Ponemos también en el index.html, el CDN del kit fontawesome

```
<head>
  <meta charset="utf-8">
  <title>Coches</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link
    href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&
    display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
    rel="stylesheet">

  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.m
    in.css" rel="stylesheet"
    integrity="sha384-1BmE4kWBq78iYhF1dvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoq
    yl2QvZ6jIW3" crossorigin="anonymous">
    <script src="https://kit.fontawesome.com/b7926f3752.js"
    crossorigin="anonymous"></script>
</head>
<body class="mat-typography">
  <script
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bun
    dle.min.js"
    integrity="sha384-ka7Sk0Gln4gmtz2M1QnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q
    +8nbTov4+lP" crossorigin="anonymous"></script>

  <app-root></app-root>
</body>
</html>
```

- 6.3. Creamos el componente para seleccionar los equipamientos.
ng g c components/equipamientos
- 6.4. Creamos otro componente para el cart
ng g c components/cart
- 6.5. Vamos al navbar.component.html para añadir un input con el icono de una lupa a su derecha, para permitir buscar los futuros equipamientos.

```
<mat-toolbar color="primary">
  <mat-toolbar-row>
    <button mat-icon-button>
      <mat-icon (click)="sidenav.toggle();">menu</mat-icon>
    </button>

    <h1>Coches Angular App</h1>

    <div class="form-group">
      <input
        type="text"
        placeholder="search products"
        class="form-control"
      >
      <span class="fas fa-search search-icon"></span>
    </div>

    <span class="example-spacer"></span>

    <button mat-icon-button class="example-icon favorite-icon">
      <mat-icon>favorite</mat-icon>
    </button>

    <button mat-icon-button class="example-icon">
      <mat-icon>share</mat-icon>
    </button>
  </mat-toolbar-row>
</mat-toolbar>
```

```
.form-control {
```

```

border-radius: 5px;
/* width: 100%; */
margin-left: 20px;
display: flex;
flex-wrap: wrap;
}

.search-icon {
  position: absolute;
  z-index: 10;
  top: 20px;
  margin-left: 220px;
  display: flex;
  flex-wrap: wrap;
  color: blue;
}

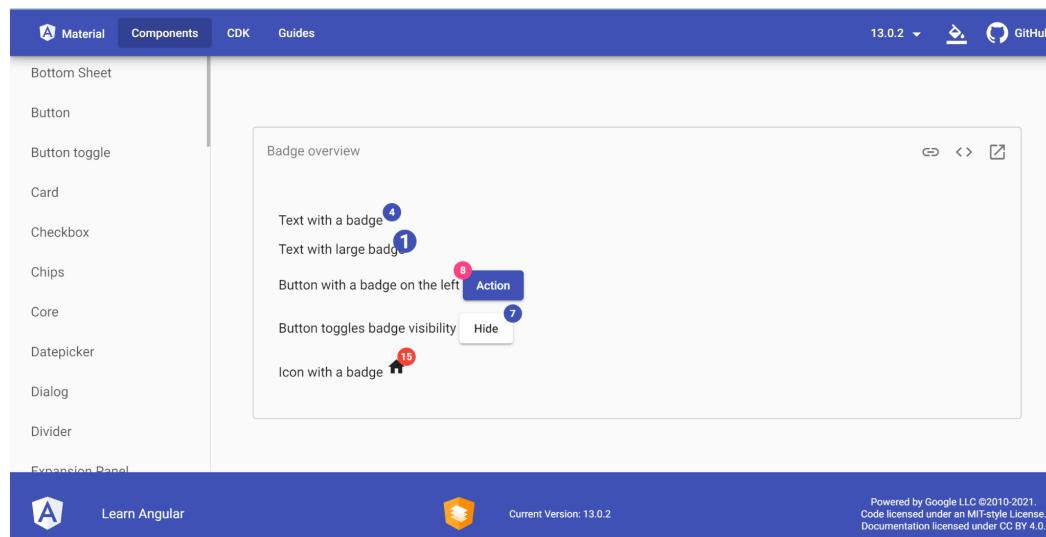
```

6.5.1. También vamos a añadir un icono de un carrito con un badge a modo contador de los equipamientos seleccionados

Para ello, recurriremos a [Icon | Angular Material](#) , y a un ejemplo de badge en [Angular Material](#)

<https://www.angularjswiki.com/angular/angular-material-icons-list-mat-icon-list/>

<https://material.angular.io/components/badge/examples>



Nota: No me funciona el Badge de Angular Material ... pero de todos modos, decir que este sería el código:

```
<button mat-icon-button class="example-icon">
  <mat-icon matBadge="1"
matBadgeColor="warn">add_shopping_cart</mat-icon>
</button>
```

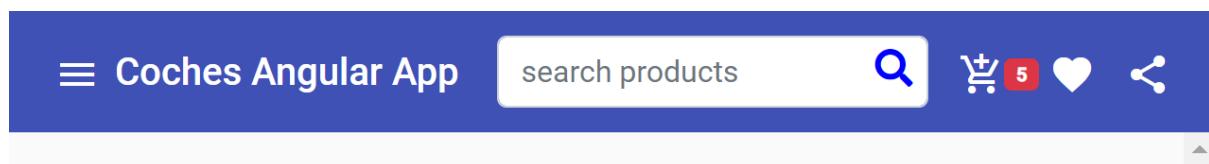
6.5.1. También vamos a añadir un icono de un carrito con un badge a modo contador de los equipamientos seleccionados

Una primera posibilidad que encontré fue esta:

```
<button class="cart-button btn btn-primary">
  <i style="font-size: 20px;" class="fas fa-cart-plus"></i>
  <div style="font-size: 10px;" class="badge bg-danger">5</div>
</button>
```

Pero finalmente, me he decidido por adaptar esta posibilidad a Angular Material, y el resultado ha sido este:

```
<button mat-icon-button class="example-icon favorite-icon">
  <mat-icon>add_shopping_cart</mat-icon>
  <div style="font-size: 10px;" class="badge bg-danger">5</div>
</button>
```



- 6.6. Añadimos una nueva ruta para el componente de comprar

```
const routes: Routes = [
  {
    path: 'inicio',
    component: InicioComponent
  },
  {
    path: 'coches',
    component: CochesListaComponent
  },
  {
    path: 'comprar/:id',
    component: CompraComponent
  }
]
```

```
{
  path: 'coches/:id',
  component: CochesDetallesComponent,
  children: [
    {
      path: 'imagenes',
      component: CochesImagenesComponent
    },
    {
      path: 'opiniones',
      component: CochesOpinionesComponent
    },
    {
      path: '',
      redirectTo: 'imagenes',
      pathMatch: 'full'
    },
    {
      path: '',
      component: NotFoundError404Component
    }
  ]
},
{
  path: 'equipamientos',
  component: EquipamientosComponent
},
{
  path: '',
  redirectTo: '/inicio',
  pathMatch: 'full'
},
{
  path: '**',
  component: NotFoundError404Component
}
];
```

- 6.7. En su botón del navbar añadimos el acceso a su ruta:

```
<button mat-icon-button class="example-icon favorite-icon"
routerLinkActive="active-link" routerLink="/equipamientos">
    <mat-icon>add_shopping_cart</mat-icon>
    <div style="font-size: 10px;" class="badge bg-danger">5</div>
</button>
```

- 6.7. Equipamientos era solo para probar que funcionaba equipamientos, así que ahora ponemos el cart efectivamente

```
<button mat-icon-button class="example-icon favorite-icon"
routerLinkActive="active-link" routerLink="/cart">
    <mat-icon>add_shopping_cart</mat-icon>
    <div style="font-size: 10px;" class="badge bg-danger">5</div>
</button>
```

- 6.8. Ahora, en el equipamientos.component.html añadiremos esta estructura para empezar

```
<div class="card-top container-fluid">
    <div class="container d-flex">
        <div class="item">
            <a>
                
                <h6>Todo</h6>
            </a>
        </div>

        <div class="item">
            <a>
                
                <h6>Pinturas</h6>
            </a>
        </div>

        <div class="item">
            <a>
                
                <h6>Motor</h6>
            </a>
        </div>
    </div>
</div>
```

```

        </div>

        <div class="item">
            <a>
                
                <h6>Combustible</h6>
            </a>
        </div>
    </div>
</div>

```

- 6.9. Y para el css, de momento esto...

```

.card-top {
    position: relative;
    display: flex;
    flex-direction: column;
    min-width: 0;
    word-wrap: break-word;
    background: #fff;
    background-clip: border-box;
    border: 1px solid rgba(0, 0, 0, 0.2);
    border-radius: 0.25rem;
}

.item {
    margin: 0 15px;
    text-align: center;
}

.img-pinturas {
    width: 80px;
    height: 60px;
    margin-top: 10px;
}

.img-motores {
    width: 100px;
    height: 60px;
    margin-top: 10px;
}

```

}

- 6.10. Ahora tenemos que crear un model y un mock para los equipamientos (productos)

6.10.1. Empezamos creando el EquipamientoModel

```
export interface EquipamientoModel {  
    idCoche: number;  
    id: number;  
    nombre: string;  
    descripcion: string;  
    color: string;  
    motor: number;  
    combustible: string;  
    precio: number;  
    imagen: string;  
}
```

6.10.2. Despu s creamos el equipamientos-mock.ts

```
import { EquipamientoModel } from "../interfaces/equipamiento-model";

export const EQUIPAMIENTOS: EquipamientoModel[] = [
/****** COCHE 0 - PINTURAS ***** */
{
    idCochе: 0,
    id: 0,
    nombre: "Pintura Azul",
    descripcion: "Phytonic Blau Metallic",
    precio: 200,
    color: "Azul",
    motor: 0,
    combustible: "",
    imagen: "../../assets/pinturas/BMW/BMW-azul.png",
},
{
    idCochе: 0,
    id: 1,
    nombre: "Pintura Blanca",
    descripcion: "Mineralweiss metalizado",
}
```

```
        precio: 100,
        color: "",
        motor: 0,
        combustible: "",
        imagen: "../../assets/pinturas/BMW/BMW-blanco.png",
    },
    {
        idCoche: 0,
        id: 2,
        nombre: "Pintura Naranja",
        descripcion: "Sunset Orange",
        precio: 200,
        color: "Naranja",
        motor: 0,
        combustible: "",
        imagen: "../../assets/pinturas/BMW/BMW-naranja.png",
    },
    {
        idCoche: 0,
        id: 3,
        nombre: "Pintura Negra",
        descripcion: "Saphirschwarz",
        precio: 100,
        color: "Negro",
        motor: 0,
        combustible: "",
        imagen: "../../assets/pinturas/BMW/BMW-negro.png",
    },
    {
        idCoche: 0,
        id: 4,
        nombre: "Pintura Roja",
        descripcion: "Melbourne Rot metalizado",
        precio: 200,
        color: "Rojo",
        motor: 0,
        combustible: "",
        imagen: "../../assets/pinturas/BMW/BMW-rojo.png",
    },
    /**
     * ***** COCHE 0 - MOTORES **** */
{
    idCoche: 0,
    id: 5,
```

```
        nombre: "Motor Nivel 1",
        descripcion: "135 kW (184 CV)",
        precio: 1000,
        color: "",
        motor: 135,
        combustible: "",
        imagen: "../../assets/motores/BMW-motor.png",
    },
{
    idCoche: 0,
    id: 6,
    nombre: "Motor Nivel 2",
    descripcion: "150 kW (210 CV)",
    precio: 2000,
    color: "",
    motor: 150,
    combustible: "",
    imagen: "../../assets/motores/BMW-motor.png",
},
{
    idCoche: 0,
    id: 7,
    nombre: "Motor Nivel 3",
    descripcion: "175 kW (250 CV)",
    precio: 3000,
    color: "",
    motor: 175,
    combustible: "",
    imagen: "../../assets/motores/BMW-motor.png",
},
/******************************** COCHE 0 - COMBUSTIBLES *****/
{
    idCoche: 0,
    id: 8,
    nombre: "Gasolina",
    descripcion: "Gasolina",
    precio: 1500,
    color: "",
    motor: 0,
    combustible: "Gasolina",
    imagen: "../../assets/combustibles/gasolina.png",
},
{
```

```

        idCoche: 0,
        id: 9,
        nombre: "Diesel",
        descripcion: "Diesel",
        precio: 1250,
        color: "",
        motor: 0,
        combustible: "Diesel",
        imagen: "../../assets/combustibles/diesel.png",
    },
    {
        idCoche: 0,
        id: 10,
        nombre: "Híbrido",
        descripcion: "Híbrido",
        precio: 1750,
        color: "",
        motor: 0,
        combustible: "Híbrido",
        imagen: "../../assets/combustibles/hibrido.png",
    },
];

```

Nota: En vez de crear los datos en un mock y proyectarlos enviándolos todos de golpe, vamos a probar a crear un JSON y a pedir los datos a través de HttpClient (get/post)

6.10. Vamos a la carpeta de Assets para crear el archivo equipamientos.json

```

[
{
    "idCoche": 0,
    "id": 0,
    "nombre": "Pintura Azul",
    "categoria": "pintura",
    "descripcion": "Phytonic Blau Metallic",
    "precio": 200,
    "color": "Azul",
    "motor": 0,
    "combustible": "",
    "imagen": "/assets/pinturas/BMW/BMW-azul.png",
    "avatar": "/assets/BMWLogo.jpg"
},
{

```

```
        "idCoche": 0,
        "id": 1,
        "nombre": "Pintura Blanca",
        "categoria": "pintura",
        "descripcion": "Mineralweiss metalizado",
        "precio": 100,
        "color": "",
        "motor": 0,
        "combustible": "",
        "imagen": "/assets/pinturas/BMW/BMW-blanco.png",
        "avatar": "/assets/BMWLogo.jpg"
    },
    {
        "idCoche": 0,
        "id": 2,
        "nombre": "Pintura Naranja",
        "categoria": "pintura",
        "descripcion": "Sunset Orange",
        "precio": 200,
        "color": "Naranja",
        "motor": 0,
        "combustible": "",
        "imagen": "/assets/pinturas/BMW/BMW-naranja.png",
        "avatar": "/assets/BMWLogo.jpg"
    },
    {
        "idCoche": 0,
        "id": 3,
        "nombre": "Pintura Negra",
        "categoria": "pintura",
        "descripcion": "Saphirschwarz",
        "precio": 100,
        "color": "Negro",
        "motor": 0,
        "combustible": "",
        "imagen": "/assets/pinturas/BMW/BMW-negro.png",
        "avatar": "/assets/BMWLogo.jpg"
    },
    {
        "idCoche": 0,
        "id": 4,
        "nombre": "Pintura Roja",
        "categoria": "pintura",
```

```
        "descripcion": "Melbourne Rot metalizado",
        "precio": 200,
        "color": "Rojo",
        "motor": 0,
        "combustible": "",
        "imagen": "/assets/pinturas/BMW/BMW-rojo.png",
        "avatar": "/assets/BMWLogo.jpg"
    },
}

{
    "idCochе": 0,
    "id": 5,
    "nombre": "Motor Nivel 1",
    "categoria": "motor",
    "descripcion": "135 kW (184 CV)",
    "precio": 1000,
    "color": "",
    "motor": 135,
    "combustible": "",
    "imagen": "/assets/motores/BMW-motor.png",
    "avatar": "/assets/BMWLogo.jpg"
},
{
    "idCochе": 0,
    "id": 6,
    "nombre": "Motor Nivel 2",
    "categoria": "motor",
    "descripcion": "150 kW (210 CV)",
    "precio": 2000,
    "color": "",
    "motor": 150,
    "combustible": "",
    "imagen": "/assets/motores/BMW-motor.png",
    "avatar": "/assets/BMWLogo.jpg"
},
{
    "idCochе": 0,
    "id": 7,
    "nombre": "Motor Nivel 3",
    "categoria": "motor",
    "descripcion": "175 kW (250 CV)",
```

```
        "precio": 3000,
        "color": "",
        "motor": 175,
        "combustible": "",
        "imagen": "/assets/motores/BMW-motor.png",
        "avatar": "/assets/BMWLogo.jpg"
    },
}

{
    "idCoche": 0,
    "id": 8,
    "nombre": "Gasolina",
    "categoria": "combustible",
    "descripcion": "Gasolina",
    "precio": 1500,
    "color": "",
    "motor": 0,
    "combustible": "Gasolina",
    "imagen": "/assets/combustibles/gasolina.png",
    "avatar": "/assets/BMWLogo.jpg"
},
{
    "idCoche": 0,
    "id": 9,
    "nombre": "Diesel",
    "categoria": "combustible",
    "descripcion": "Diesel",
    "precio": 1250,
    "color": "",
    "motor": 0,
    "combustible": "Diesel",
    "imagen": "/assets/combustibles/diesel.png",
    "avatar": "/assets/BMWLogo.jpg"
},
{
    "idCoche": 0,
    "id": 10,
    "nombre": "Híbrido",
    "categoria": "combustible",
    "descripcion": "Híbrido",
    "precio": 1750,
```

```

        "color": "",
        "motor": 0,
        "combustible": "Híbrido",
        "imagen": "/assets/combustibles/hibrido.png",
        "avatar": "/assets/BMWLogo.jpg"
    }
]

```

6.11. Vamos a crear un servicio para obtener los equipamientos del equipamientos.json que habíamos creado

ng g s services/equipamiento

6.11.1. Lo primero, será importar el HttpClientModule en el app.module.ts

```

import { HttpClientModule } from '@angular/common/http';

imports: [
  BrowserModule,
  BrowserAnimationsModule,
  AngularMaterialModule,
  AppRoutingModule,
  SwiperModule,
  HttpClientModule
],

```

6.11.2. Ahora, en el equipamiento.service.ts, inyectamos el servicio de HttpClient

6.11.3. Creamos el método getProduct()

```

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { map } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})

export class EquipamientoService {

  constructor(

```

```

    private httpClient: HttpClient
) { }

getProduct() {
    return this.httpClient.get<any>("../assets/equipamientos.json")
        .pipe(map((res: any) => {
            return res;
        }))
}

}

```

6.12. Ahora vamos al equipamientos.component.ts para inyectar al servicio que acabamos de crear, y en el ngOnInit() llamar a su método getProduct()

```

export class EquipamientosComponent implements OnInit {

    public listaEquipamientos: any;

    constructor(
        private equipamientoService: EquipamientoService
    ) { }

    ngOnInit(): void {
        this.equipamientoService.getProduct()
            .subscribe((res) => {
                this.listaEquipamientos = res;
            })
    }
}

```

6.13. Ahora, en el equipamientos.component.html, añadimos debajo de lo que ya teníamos de antes ...

```

<div class="superContainer" *ngIf="equipmentsList != null ||
equipmentsList != undefined">
    <mat-card class="example-card" *ngFor="let equipment of
equipmentsList">
        <mat-card-header>
            

```

```

        <mat-card-subtitle>ID: {{ equipment.id
} }</mat-card-subtitle>
        <mat-card-title>{{ equipment.nombre }}</mat-card-title>
        <mat-card-subtitle>Categoria: {{ equipment.categoria
} }</mat-card-subtitle>
    </mat-card-header>

    <mat-card-content>
        <mat-card-subtitle>{{ equipment.descripcion
} }</mat-card-subtitle>
        <mat-card-title>{{ equipment.precio }}€</mat-card-title>
    </mat-card-content>

    <mat-divider inset></mat-divider>

    <mat-card-actions>
        <!-- <button mat-raised-button color="accent">LIKE</button>
-->
        <button mat-raised-button color="primary">ADD TO
CART</button>
    </mat-card-actions>
</mat-card>
</div>

```

6.14. Ahora, en el `equipamientos.component.css`, añadimos debajo de lo que ya teníamos de antes ...

```

/***** /
.example-card {
    max-width: 250px;
    max-height: 400px;
    margin: 5px;
}

.superContainer {
    padding: 1px;
    display: flex;

```

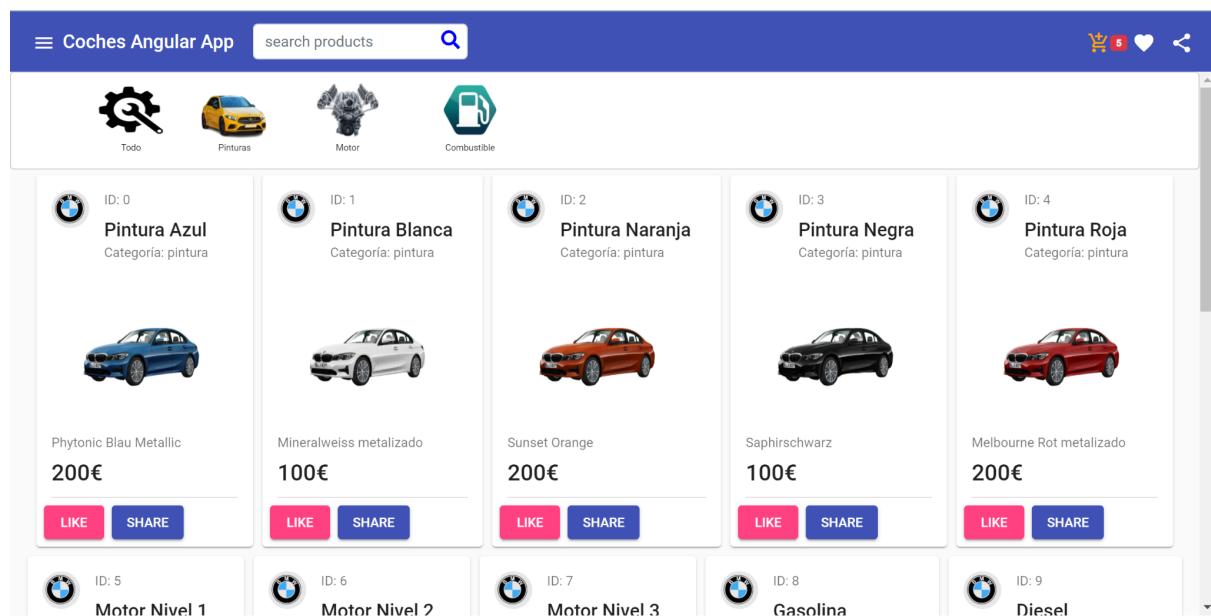
```

flex-wrap: wrap;
align-items: center;
justify-content: center;
}

.card-img {
  width: 230px;
  height: 150px;
  transition: 0.3s ease-in-out;
}

.card-img:hover {
  transition: 0.3s ease-in-out;
  transform: scale(1.3);
}

```



6.15. Vamos a añadir nuevas rutas para este nuevo componente

6.15.1. En el app-routing.module.ts

```

const routes: Routes = [
  {
    path: 'inicio',
    component: InicioComponent
  },
  {
    path: 'motor',
    component: MotorComponent
  }
]

```

```
        path: 'coches',
        component: CochesListaComponent
    },
{
    path: 'coches/:id',
    component: CochesDetallesComponent,
    children: [
        {
            path: 'imagenes',
            component: CochesImagenesComponent
        },
        {
            path: 'opiniones',
            component: CochesOpinionesComponent
        },
        {
            path: '',
            redirectTo: 'imagenes',
            pathMatch: 'full'
        },
        {
            path: '',
            component: NotFoundError404Component
        }
    ]
},
{
    path: 'equipamientos',
    component: EquipamientosComponent
},
{
    path: 'equipamientos/:id',
    component: EquipamientosComponent
},
{
    path: 'cart',
    component: CartComponent
},
{
    path: '',
    redirectTo: '/inicio',
    pathMatch: 'full'
},
```

```
{
  path: '**',
  component: NotFoundError404Component
}
];
```

6.15.2. En el coches-lista.component.html

```
<mat-card-actions>
  <button mat-button>ID: {{ coche.id }}</button>
  <button mat-button>Rank: {{ coche.ranking }}</button>
  <button mat-button [routerLink]=["/coches", coche.id]>
    routerLinkActive="router-link-active">INFO</button>
    <button mat-button [routerLink]=["/equipamientos",
      coche.id]" routerLinkActive="router-link-active">BUY</button>
</mat-card-actions>
```

6.15.3. Y en el coches-detalles.component.html

```
<mat-card-actions>
  <button mat-button>LIKE</button>
  <button mat-button>Rank: {{ coche.ranking }}</button>
  <button mat-button [routerLink]=["/equipamientos",
    coche.id"]>BUY</button>
</mat-card-actions>
```

- 7. Vamos a desarrollar ahora el CartComponent

7.1. Al igual que hicimos con el EquipamientosComponent, vamos a crear un servicio para recibir los equipamientos seleccionados

ng g s services/cart

7.2. En él, tenemos que hacer un CRUD

```
export class CartService {  
  
  public cartEquipmentList: any = [];  
  
  public equipmentList = new BehaviorSubject<any>([]);  
  
  constructor() { }  
  
  getEquipments() {  
    return this.equipmentList.asObservable();  
  }  
  
  setEquipment(equipment: any) {  
    this.cartEquipmentList.push(...equipment);  
    this.equipmentList.next(equipment);  
  }  
  
  addToCart(equipment: any) {  
    this.cartEquipmentList.push(equipment);  
    this.equipmentList.next(this.cartEquipmentList);  
    this.getTotalPrice();  
  
    console.log(this.cartEquipmentList);  
  }  
  
  getTotalPrice(): number {  
    let finalTotal = 0;  
    this.cartEquipmentList.map((equipment: any) => {  
      finalTotal += equipment.total;  
    })  
  
    return finalTotal;  
  }  
  
  removeCartEquipment(equipment: any) {
```

```

        this.cartEquipmentList.map((item: any, index: any) => {
            if (equipment.id === item.id) {
                this.cartEquipmentList.splice(index, 1);
            }
        })
    }

removeAllCart() {
    this.cartEquipmentList = [];
    this.equipmentList.next(this.cartEquipmentList);
}
}

```

7.4. Pero no olvidemos que en el EquipamientosComponent, tenemos un botón para añadir el equipamiento al carrito, sobre el cual, tendremos que añadir tal método al botón en el .html, y en el .ts, tenemos que inyectar el CartService y crear tal método.

```

<mat-card-actions>
    <!-- <button mat-raised-button color="accent">LIKE</button>
-->
    <button mat-raised-button color="primary"
(click)=addToCart(equipment)
>ADD TO CART</button>
</mat-card-actions>

```

```

export class EquipamientosComponent implements OnInit {

    public equipmentsList: any;

    constructor(
        private equipamientoService: EquipamientoService,
        private cartService: CartService
    ) { }

    ngOnInit(): void {
        this.equipamientoService.getProduct()
            .subscribe((res) => {
                this.equipmentsList = res;
            })
    }

    addToCart(equipment: any) {
        this.cartService.addToCart(equipment);
    }
}

```

```
    }  
}  
}
```

7.5. Ahora, tenemos que conseguir que cuando pulsemos el botón de AddToCart, efectivamente guardemos un objeto tipo Array con las propiedades del objeto en cuestión.

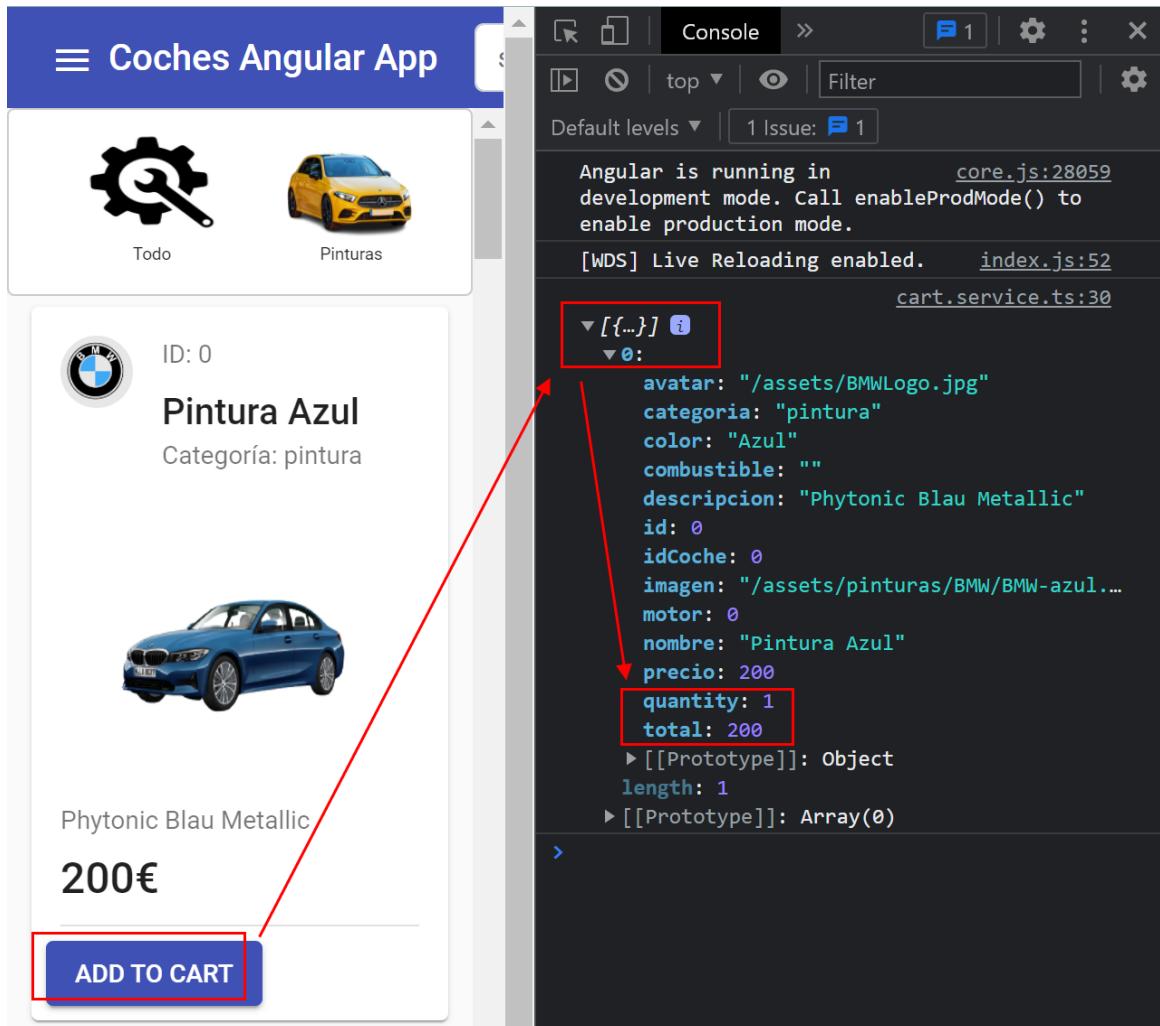
7.5.1. En el ngOnInit() del equipamientos.component.ts añadimos ...

```
ngOnInit(): void {  
    this.equipamientoService.getProduct()  
        .subscribe((res) => {  
            this.equipmentsList = res;  
  
            this.equipmentsList.forEach((equipment: any) => {  
                Object.assign(equipment, { quantity: 1, total:  
                    equipment.precio });  
            });  
        })  
}
```

7.5.2. Y en el addToCart() del CartService ponemos un console.log() para comprobar que funciona a través del inspeccionar de Chrome ...

```
addToCart(equipment: any) {  
    this.cartEquipmentList.push(equipment);  
    this.equipmentList.next(this.cartEquipmentList);  
    this.getTotalPrice();  
  
    console.log(this.cartEquipmentList);  
}
```

7.5.3. y, si pulso en el AddToCart del primer producto ... funciona !!



7.6. Ahora vamos a actualizar el botón del carrito en el navbar para que el badge vaya cambiando según la cantidad de equipamientos añadidos

7.6.1. Vamos al navbar.component.ts para añadir una propiedad a modo contador, y para añadir al ngOnInit() lo siguiente:

```
export class NavbarComponent implements OnInit {

  public totalEquipment: number = 0;

  constructor(
    private cartService: CartService
  ) { }

  ngOnInit(): void {
    this.cartService.getEquipments()
```

```

        .subscribe((res) => {
            this.totalEquipment = res.length;
        })
    }

}

```

7.6.2. Luego, en el navbar.component.html, debemos interpolar ...

```

<button mat-icon-button class="example-icon favorite-icon"
routerLinkActive="active-link" routerLink="/cart">
    <mat-icon>add_shopping_cart</mat-icon>
    <div style="font-size: 10px;" class="badge bg-danger">{ {
totalEquipment } }</div>
</button>

```

7.7. Es hora de traer los equipamientos seleccionados al carrito !!

7.7.1. Vamos al cart.component.ts para añadir un par de propiedades (un objeto array, y un “finalTotal”), también vamos a crear los dos métodos de eliminar equipamientos, y además añadiremos al ngOnInit() lo siguiente:

```

export class CartComponent implements OnInit {

    public equipments: any = [];

    public finalTotal: number | undefined;

    constructor(
        private cartService: CartService
    ) { }

    ngOnInit(): void {
        this.cartService.getEquipments()
            .subscribe((res) => {
                this.equipments = res;
                this.finalTotal = this.cartService.getTotalPrice();
            })
    }

    removeEquipment(equipment: any) {

```

```

        this.cartService.removeCartEquipment(equipment);
    }

emptyCart() {
    this.cartService.removeAllCart();
}

}

```

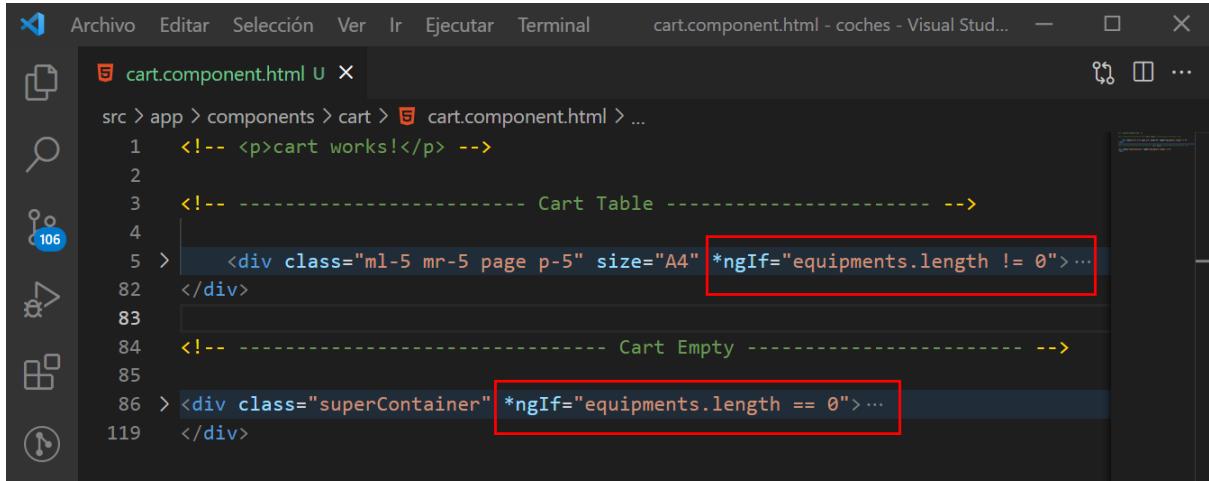
7.7.2. Y en el .html realizaremos las siguientes modificaciones para reproducir todos los equipamientos seleccionados con un *ngFor, e interpolamos ...

```

<tbody>
    <tr *ngFor="let equipment of equipments; let i =
index">
        <td>
            <div class="media">
                
                <div class="media-body">
                    {{ i + 1 }}
                    <p class="mt-0 title">{{
equipment.nombre }}</p>
                    {{ equipment.descripcion }}
                </div>
            </div>
        </td>
        <td>{{ equipment.precio }}</td>
        <td>{{ equipment.quantity }}</td>
        <td>{{ equipment.total }}€</td>
        <td>
            <button mat-icon-button class="example-icon"
(click)="removeEquipment(equipment)">
                <mat-icon
class="delete-icon">delete_forever</mat-icon>
            </button>
        </td>
    </tr>
</tbody>

```

7.7.3. Y ahora es el momento de cambiar las condiciones de los *ngIf tanto del Cart Table, como del Cart Empty



```
src > app > components > cart > cart.component.html > ...
1   <!-- <p>cart works!</p> -->
2
3   <!-- ----- Cart Table ----- -->
4
5 >   <div class="ml-5 mr-5 page p-5" size="A4" *ngIf="equipments.length != 0">...
82 </div>
83
84 <!-- ----- Cart Empty ----- -->
85
86 > <div class="superContainer" *ngIf="equipments.length == 0">...
119 </div>
```

7.7.4. Dentro de la parte del Cart Empty, ponemos un routerLink en el botón de SHOP NOW

```
<mat-card-actions>
    <button mat-raised-button color="primary"
routerLink="/equipamientos">SHOP NOW</button>
</mat-card-actions>
```

7.8. Encontramos un pequeño defecto en el badge del botón del shopping cart del navbar, y es que, cuando vamos eliminando equipamientos del carrito, el contador de este badge no disminuye ...

Simplemente tenemos que ir al cart.service.ts y añadir una línea al método de removeCartEquipment()

```
removeCartEquipment(equipment: any) {
  this.cartEquipmentList.map((item: any, index: any) => {
    if (equipment.id === item.id) {
      this.cartEquipmentList.splice(index, 1);
    }
  })

  this.equipmentList.next(this.cartEquipmentList);
}
```

7.9. Ahora, vamos a añadir unos botones para las acciones de los métodos que ya definimos en el CartService pero que aún no hemos probado

```
<tbody>
    <tr *ngFor="let equipment of equipments; let i =
index">
        <td>
            <div class="media">
                
                <div class="media-body">
                    {{ i + 1 }}
                    <p class="mt-0 title">{{
equipment.nombre }}</p>
                    {{ equipment.descripcion }}
                </div>
            </div>
        </td>
        <td>{{ equipment.precio }}</td>
        <td>{{ equipment.quantity }}</td>
        <td>{{ equipment.total }}€</td>
        <td>
            <button mat-icon-button class="example-icon"
(click)="removeEquipment(equipment)">
                <mat-icon
class="delete-icon">delete_forever</mat-icon>
            </button>
        </td>
    </tr>

    <tr>
        <td colspan="4"></td>
        <td class="final-buttons">
            <button class="final-button btn
btn-danger">Empty Cart</button>
            <button class="final-button btn
btn-primary">Shop More</button>
            <button class="final-button btn
btn-success">CheckOut</button>
        </td>
    </tr>
</tbody>
```

```

.final-buttons {
  display: flex;
  flex-wrap: wrap;
  align-items: flex-end;
  justify-content: right;
}

.final-button {
  margin-left: 10px;
  margin-right: 10px;
  margin-top: 5px;
  margin-bottom: 5px;
}

```

Item Description	Price	Quantity	Total	Action
1 Pintura Blanca Mineralweiss metalizado	100	1	100€	

Note:
Lorem ipsum, dolor sit amet consectetur adipisicing elit. Natus, suscipit. Minus inventore quas nihil ut!

SubTotal	800€
Tax:	15€
Deliver	10€
Total:	825€

Empty Cart **Shop More** **CheckOut**

7.10. Vamos a añadir las funcionalidades a tales botones

```

<tr>
  <td colspan="4"></td>
  <td class="final-buttons">
    <button class="final-button btn btn-danger"
(click)="emptyCart()">Empty Cart</button>
    <button class="final-button btn btn-primary"
routerLink="/equipamientos">Shop More</button>
    <button class="final-button btn
btn-success">CheckOut</button>
  </td>
</tr>

```

7.11. Ahora vamos a tratar de modificar el método de `getTotalPrice()` para que sea la suma del Subtotal + IVA 21% + Envío 10%, y también vamos a crear los métodos para obtener el Subtotal, el IVA 21% y el Envío 10%

7.11.1 En el `CartService` añadimos los siguientes métodos y modificamos el `getTotalPrice()`

```
export class CartService {

    public cartEquipmentList: any = [];

    public equipmentList = new BehaviorSubject<any>([]);

    constructor() { }

    getEquipments() {
        return this.equipmentList.asObservable();
    }

    setEquipment(equipment: any) {
        this.cartEquipmentList.push(...equipment);
        this.equipmentList.next(equipment);
    }

    addToCart(equipment: any) {
        this.cartEquipmentList.push(equipment);
        this.equipmentList.next(this.cartEquipmentList);
        this.getTotalPrice();

        console.log(this.cartEquipmentList);
    }

    getSubtotal(): number {
        let subtotal = 0;
        this.cartEquipmentList.map((equipment: any) => {
            subtotal += equipment.total;
        })

        return subtotal;
    }

    getTax(subtotal: number): number {
        let tax = 0.21;
        this.cartEquipmentList.map((equipment: any) => {
```

```
        tax = subtotal * 0.21;
    } )

    return tax;
}

getDeliver(subtotal: number): number {
    let deliver = 0.1;
    this.cartEquipmentList.map((equipment: any) => {
        deliver = subtotal * 0.1;
    } )

    return deliver;
}

getTotalPrice(): number {
    let finalTotal = 0;
    this.cartEquipmentList.map((equipment: any) => {
        finalTotal += equipment.total + (equipment.total * 0.21) +
(equipment.total * 0.1);
    } )

    return finalTotal;
}

removeCartEquipment(equipment: any) {
    this.cartEquipmentList.map((item: any, index: any) => {
        if (equipment.id === item.id) {
            this.cartEquipmentList.splice(index, 1);
        }
    } )

    this.equipmentList.next(this.cartEquipmentList);
}

removeAllCart() {
    this.cartEquipmentList = [];
    this.equipmentList.next(this.cartEquipmentList);
}
}
```

7.11.2. En el cart.component.ts añadimos las siguientes propiedades y lo siguiente al ngOnInit()

```
export class CartComponent implements OnInit {

    public equipments: any = [];

    public subtotal: number | undefined;
    public tax: number | undefined;
    public deliver: number | undefined;
    public finalTotal: number | undefined;

    constructor(
        private cartService: CartService
    ) { }

    ngOnInit(): void {
        this.cartService.getEquipments()
            .subscribe((res) => {
                this.equipments = res;
                this.subtotal = this.cartService.getSubtotal();
                this.tax = this.cartService.getTax(this.subtotal);
                this.deliver = this.cartService.getDeliver(this.subtotal);
                this.finalTotal = this.cartService.getTotalPrice();
            })
    }

    removeEquipment(equipment: any) {
        this.cartService.removeCartEquipment(equipment);
    }

    emptyCart() {
        this.cartService.removeAllCart();
    }
}
```

7.11.3. Y por último en el cart.component.html interpolamos los nuevos datos en sección del Balance Info, es decir, el resumen de la factura debajo de la lista de equipamientos añadidos al carrito

```
<table class="table border-0 table-hover">
  <tr>
    <td>SubTotal</td>
    <td>{{ subtotal }}€</td>
  </tr>
  <tr>
    <td>Tax:</td>
    <td>{{ tax }}€</td>
  </tr>
  <tr>
    <td>Deliver</td>
    <td>{{ deliver }}€</td>
  </tr>

  <tfoot>
    <tr>
      <td>Total:</td>
      <td>{{ finalTotal }}€</td>
    </tr>
  </tfoot>
</table>
```

The screenshot shows the user interface of the "Coches Angular App". At the top, there is a navigation bar with a menu icon, the app name, a search bar containing "search products" and a magnifying glass icon, a shopping cart icon with a red notification badge (2), a heart icon, and a share icon.

The main content area displays a table of items in the cart:

	Item Description	Price	Quantity	Total	Action
1	Pintura Azul Phytonic Blau Metallic	200	1	200€	
2	Pintura Blanca Mineralweiss metalizado	100	1	100€	

At the bottom of the page, there are three buttons: "Empty Cart" (red), "Shop More" (blue), and "CheckOut" (green).

A note at the bottom left says: "Note: Lorem ipsum, dolor sit amet consectetur adipisicing elit. Natus, suscipit. Minus inventore quas nihil ut!"

A summary table at the bottom right provides the breakdown of the total cost:

SubTotal	300€
Tax:	63€
Deliver	30€
Total:	393€

8. Filtros de búsqueda y ordenación por categorías (sorting)

<https://www.youtube.com/watch?v=Bi2iYOH89og>

- 8.1. Esta parte la vamos hacer utilizando pipes

Para ello, puede ser útil revisar esta doc de angular:

<https://angular.io/guide/pipes>

- 8.2. Creamos una carpeta llamada shared y dentro de ella crearemos un pipe

ng g pipe shared/filter

Nota: recordar que cuando creamos diversas clases aparte de los simples componentes, en caso de error, debemos comprobar el app.module.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'filter'
})

export class FilterPipe implements PipeTransform {

  // transform(value: unknown, ...args: unknown[]): unknown {
  transform(value: any[], filterString: string, propertyName: string): any[] {
    const result: any = [];

    if (!value || filterString === '' || propertyName === '') {
      return value;
    }

    value.forEach((i: any) => {
      if
(i[propertyName].trim().toLowerCase().includes(filterString.toLowerCase()))
      {
        result.push(i);
      }
    });
  }

  return result;
  // return null;
}
```

```
}
```

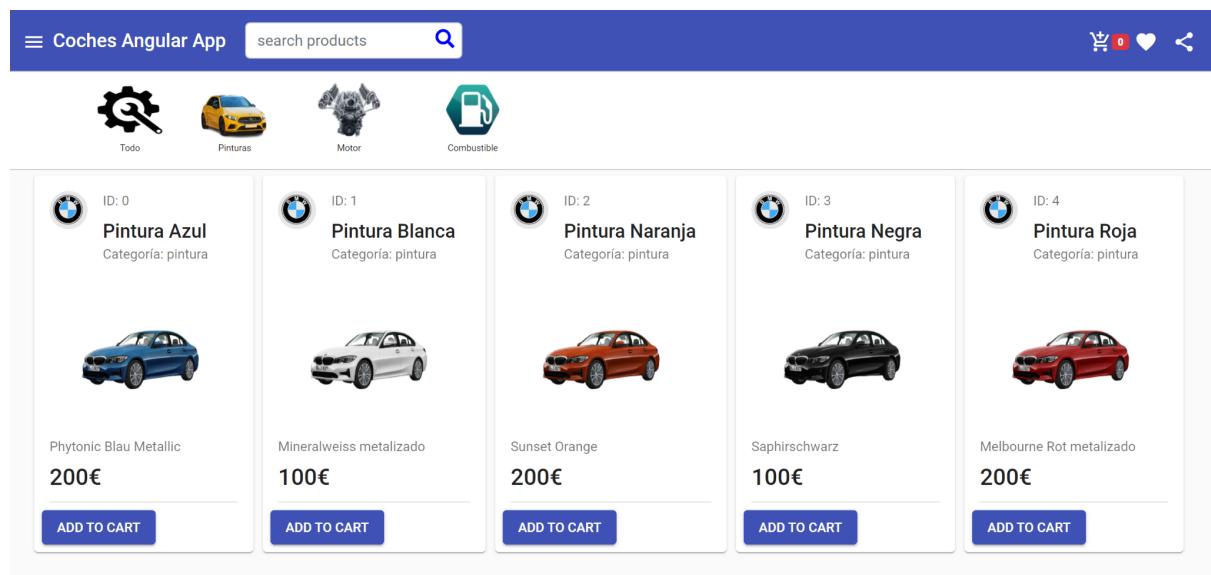
- 8.3. Ahora vamos al equipamientos.component.ts para crear la propiedad que usaremos en el html

```
searchKey: string = "";
```

8.3.1. Vamos hacer una prueba para ver si está funcionando correctamente, así que en el equipamientos.component.html, añadimos el pipe al *ngFor que despliega los equipamientos

```
<mat-card class="example-card" *ngFor="let equipment of equipmentsList  
| filter:'pintura':'nombre'">
```

8.3.2. Si en vez de nuestra propiedad (searchKey) ponemos por ejemplo 'pintura', veremos que se filtra correctamente:



8.3.3. Volvemos a dejar en el html la propiedad de searchKey

- 8.4. Ahora vamos a nuestro navbar.component.html para en el input de la búsqueda, insertar una directiva ngModel y un método que crearemos:

```
<div class="form-group">  
  <input  
    type="text"  
    placeholder="search products"  
    class="form-control"
```

```

[ (ngModel)]="searchStringMatch"
  (keyup)="search($event)"
>
<span class="fas fa-search search-icon"></span>
</div>

```

- 8.5. Volvemos al equipamientos.component.ts para crear una nueva propiedad para el [(ngModel)] del html, y un método para rastrear las coincidencias de lo que se escribe en el input de búsqueda, el search(\$event) del html

```

export class NavbarComponent implements OnInit {

  public totalEquipment: number = 0;
  public searchStringMatch: string | undefined;

  constructor(
    private cartService: CartService
  ) { }

  ngOnInit(): void {
    this.cartService.getEquipments()
      .subscribe((res) => {
        this.totalEquipment = res.length;
      })
  }

  search(event: any) {
    this.searchStringMatch = (event.target as HTMLInputElement).value;
    console.log(this.searchStringMatch);
  }
}

```

Nota: si al codificar esta parte, el servidor me diese el error de:

error NG8002: Can't bind to 'ngModel' since it isn't a known property of 'input'.
 16 [(ngModel)]="searchTerm"

error TS2322: Type 'Event' is not assignable to type 'string'.
 16 [(ngModel)]="searchTerm"

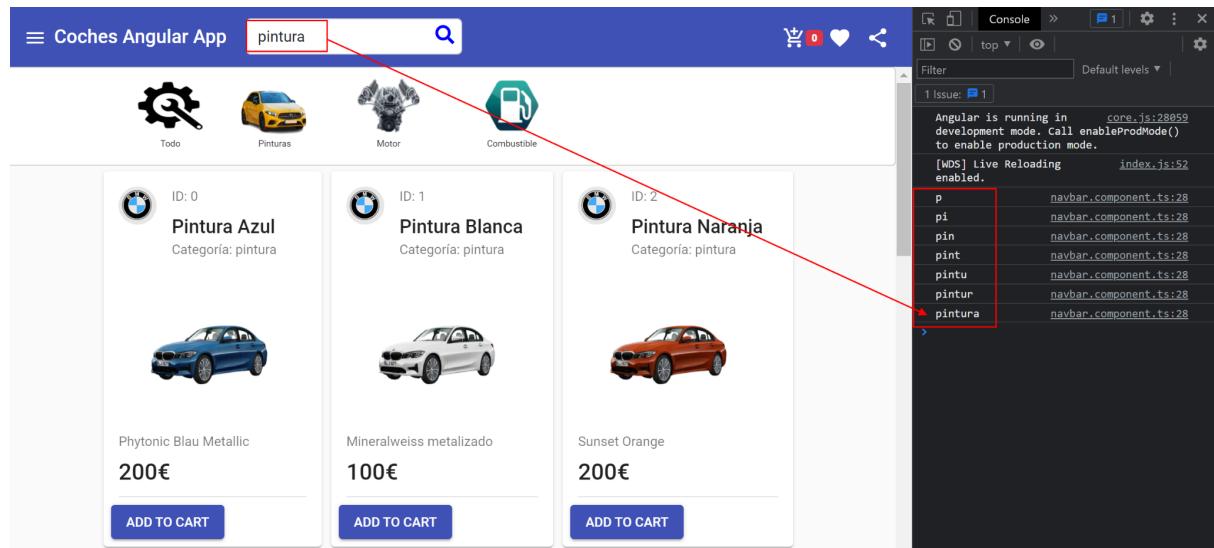
Lo que tenemos que hacer es importar manualmente en el app.module.ts el FormsModule y el ReactiveFormsModule

```

imports: [
  BrowserModule,
  BrowserAnimationsModule,
  AngularMaterialModule,
  AppRoutingModule,
  SwiperModule,
  HttpClientModule,
  FormsModule,
  ReactiveFormsModule
],

```

- 8.6. Con el inspeccionar del Chrome podemos ver que efectivamente funciona, que se actualiza el string de búsqueda cada vez que escribimos una nueva letra



- 8.7. Vamos un momento a nuestro CartService para introducir una nueva propiedad, que será la búsqueda que el navbar enviará al EquipmentComponent

```
public search = new BehaviorSubject<string>("") ;
```

- 8.8. Volviendo al navbar.component.ts, añadimos esta sentencia al método search(event: any)

```

search(event: any) {
  this.searchStringMatch = (event.target as HTMLInputElement).value;
  console.log(this.searchStringMatch);

  this.cartService.search.next(this.searchStringMatch);
}

```

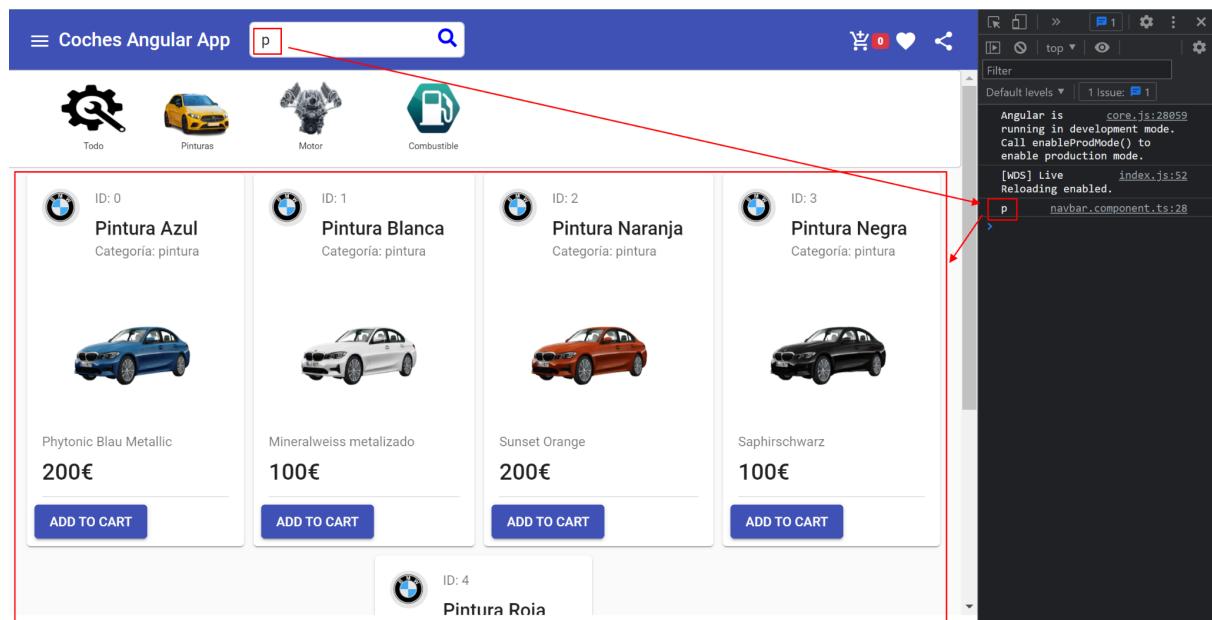
```
}
```

- 8.9. Regresando al equipamientos.component.ts para conectar el valor (nombre) que busca el método search, con el valor coincidente de cada equipamiento (equipamiento.nombre)... de modo que en el ngOnInit() añadimos lo siguiente:

```
ngOnInit(): void {
    this.equipamientoService.getProduct()
        .subscribe((res) => {
            this.equipmentsList = res;

            this.equipmentsList.forEach((equipment: any) => {
                Object.assign(equipment, { quantity: 1, total: equipment.precio });
            });
        });
}

this.cartService.search
    .subscribe((value: any) => {
        this.searchKey = value;
    });
}
```



≡ Coches Angular App

m

Console

Default levels

Angular is running in core.js:28059 development mode. Call enableProdMode() to enable production mode.

[WDS] Live Reloading enabled. index.js:52 navbar.component.ts:28

Todo Pinturas Motor Combustible

ID: 5 Motor Nivel 1 Categoría: motor

ID: 6 Motor Nivel 2 Categoría: motor

ID: 7 Motor Nivel 3 Categoría: motor

135 kW (184 CV) 150 kW (210 CV) 175 kW (250 CV)

1000€ 2000€ 3000€

ADD TO CART ADD TO CART ADD TO CART

The screenshot shows a search interface where the user has typed 'm' into the search bar. The results are displayed below, with three items highlighted: 'Motor Nivel 1', 'Motor Nivel 2', and 'Motor Nivel 3'. Each item is shown with its ID, name, category, power rating, price, and an 'ADD TO CART' button. The Angular developer tools console on the right shows the application is in development mode and live reloading is enabled.

≡ Coches Angular App

g

Console

Default levels

Angular is running in core.js:28059 development mode. Call enableProdMode() to enable production mode.

[WDS] Live Reloading enabled. index.js:52 navbar.component.ts:28

Todo Pinturas Motor Combustible

ID: 3 Pintura Negra Categoría: pintura

ID: 8 Gasolina Categoría: combustible

Saphirschwarz Gasolina

100€ 1500€

ADD TO CART ADD TO CART

The screenshot shows a search interface where the user has typed 'g' into the search bar. The results are displayed below, with two items from each category highlighted: 'Pintura Negra' and 'Gasolina'. Each item is shown with its ID, name, category, color or fuel type, price, and an 'ADD TO CART' button. The Angular developer tools console on the right shows the application is in development mode and live reloading is enabled.

- 8.10. Ahora toca desarrollar los botones para filtrar los equipamientos por las categorías de Todos, Pinturas, Motores, Combustibles.
Así que volvemos a `equipamientos.component.ts` para añadir una nueva propiedad llamada `filterCategory`, la cual también debe obtener la respuesta del `HttpClient` en el método `ngOnInit()`, y para la cual, también crearemos un método llamado `categoryFilter()` que buscará las coincidencias entre las categorías de los equipamientos

```
export class EquipamientosComponent implements OnInit {

  public equipmentsList: any;

  searchKey: string = "";

  public filterCategory: any;

  constructor(
    private equipamientoService: EquipamientoService,
    private cartService: CartService
  ) { }

  ngOnInit(): void {
    this.equipamientoService.getProduct()
      .subscribe((res) => {
        this.equipmentsList = res;
        this.filterCategory = res;

        this.equipmentsList.forEach((equipment: any) => {
          if (equipment.categoría === "pintura") {
            equipment.categoría = "pinturas";
          }
          else if (equipment.categoría === "motor") {
            equipment.categoría = "motores";
          }
          else if (equipment.categoría === "combustible") {
            equipment.categoría = "combustibles";
          }

          Object.assign(equipment, { quantity: 1, total:
            equipment.precio });
        });
        console.log(this.equipmentsList);
      });
  }
}
```

```

    this.cartService.search
      .subscribe((value: any) => {
        this.searchKey = value;
      });
  }

  addToCart(equipment: any) {
    this.cartService.addToCart(equipment);
  }

  categoryFilter(category: string) {
    this.filterCategory = this.equipmentsList
      .filter((i: any) => {
        if (i.categoria == category || category == '') {
          return i;
        }
      })
  }
}

```

- 8.11. Y ahora en equipamientos.compoennt.html, cambiamos en el *ngFor el equipmentList por la nueva propiedad de filterCategory

```

<div class="superContainer" *ngIf="equipmentsList != null ||
equipmentsList != undefined">
  <!-- <mat-card class="example-card" *ngFor="let equipment of
equipmentsList | filter:searchKey:'nombre'"> -->
  <mat-card class="example-card" *ngFor="let equipment of
filterCategory | filter:searchKey:'nombre'">

```

- 8.12. Comprobamos con el inspeccionar que efectivamente las categorías de los equipamientos han cambiado

```

Angular is running in development mode. Call enableProdMode() to enable production mode. core.js:28059
equipamientos.component.ts:43
    ▶ (11) [{}]
      ▶ 0: {idCoche: 0, id: 0, nombre: 'Pintura Azul', categoria: 'pinturas', descripcion: 'Phytonic Blau Metallico', ...}
      ▶ 1: {idCoche: 0, id: 1, nombre: 'Pintura Blanca', categoria: 'pinturas', descripcion: 'Mineralweiss metalizado', ...}
      ▶ 2: {idCoche: 0, id: 2, nombre: 'Pintura Naranja', categoria: 'pinturas', descripcion: 'Sunset Orange', ...}
      ▶ 3: {idCoche: 0, id: 3, nombre: 'Pintura Negra', categoria: 'pinturas', descripcion: 'Saphirschwarz', ...}
      ▶ 4: {idCoche: 0, id: 4, nombre: 'Pintura Roja', categoria: 'pinturas', descripcion: 'Melbourne Rot metallic', ...}
      ▶ 5: {idCoche: 0, id: 5, nombre: 'Motor Nivel 1', categoria: 'motores', descripcion: '135 KW (184 CV)', ...}
      ▶ 6: {idCoche: 0, id: 6, nombre: 'Motor Nivel 2', categoria: 'motores', descripcion: '150 KW (210 CV)', ...}
      ▶ 7: {idCoche: 0, id: 7, nombre: 'Motor Nivel 3', categoria: 'motores', descripcion: '175 KW (250 CV)', ...}
      ▶ 8: {idCoche: 0, id: 8, nombre: 'Gasolina', categoria: 'combustibles', descripcion: 'Gasolina', ...}
      ▶ 9: {idCoche: 0, id: 9, nombre: 'Diesel', categoria: 'combustibles', descripcion: 'Diesel', ...}
      ▶ 10: {idCoche: 0, id: 10, nombre: 'Hibrido', categoria: 'combustibles', descripcion: 'Hibrido', ...}
    > [prototype]: Array(10)
  
```

- 8.13. Por último, en el equopamientos.component.html, debemos agregar el nuevo método de categoryFilter() a cada <a> de cada categoría

```

<div class="card-top container-fluid">
  <div class="container d-flex">
    <div class="item">
      <a (click)="categoryFilter('')">
        
        <h6>Todo</h6>
      </a>
    </div>

    <div class="item">
      <a (click)="categoryFilter('pinturas')">
        
        <h6>Pinturas</h6>
      </a>
    </div>

    <div class="item">
      <a (click)="categoryFilter('motores')">
        
        <h6>Motor</h6>
      </a>
    </div>
  </div>
</div>

```

```

        </a>
    </div>

    <div class="item">
        <a (click)="categoryFilter('combustibles')">
            
            <h6>Combustible</h6>
        </a>
    </div>
</div>

```

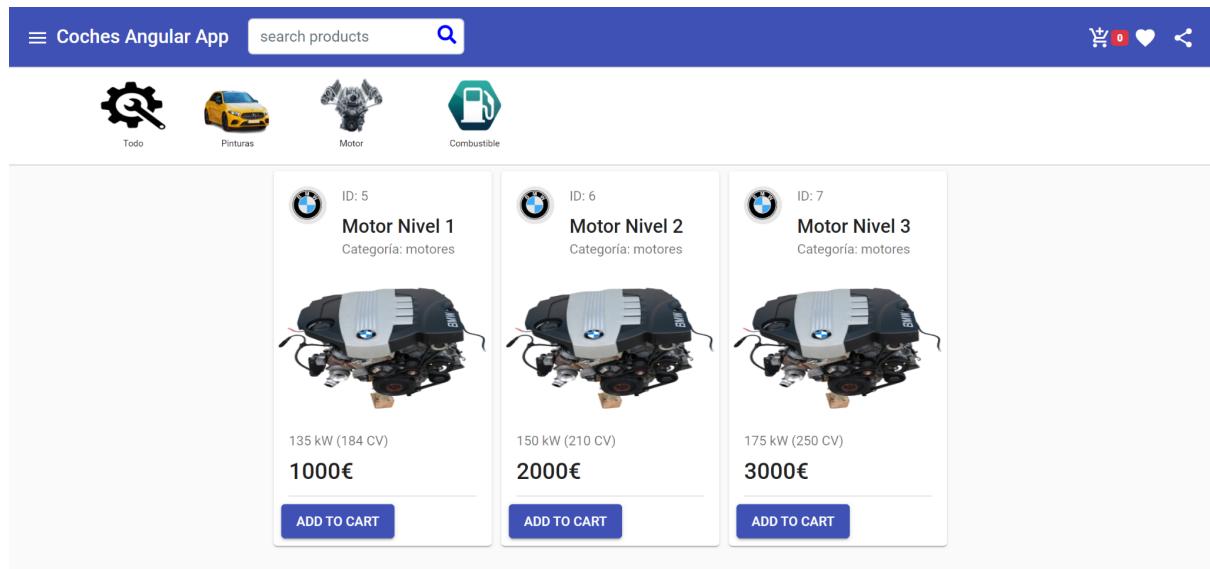
- 8.14. Cómo último detalle de esta parte, sería conveniente, por estética, que las secciones sean clickables, así que en el equipamientos.component.css pongamos en la clase del .item a:hover{} el cursor:pointer

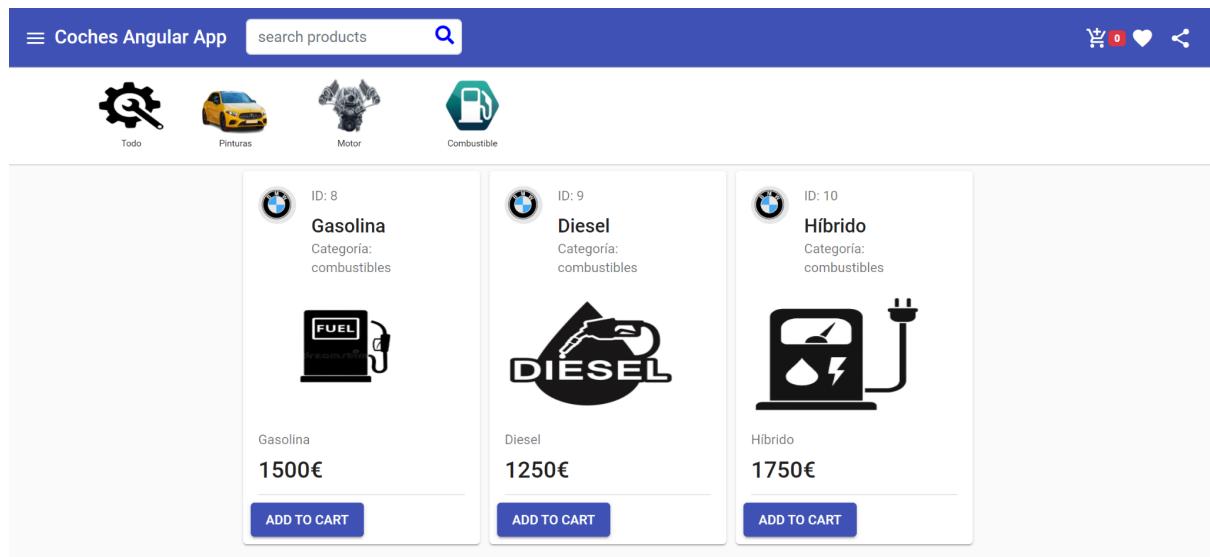
```

.item a:hover {
    color: blue;
    cursor: pointer;
}

```

- 8.15. Y comprobamos en el navegador que filtra perfectamente !!





9. La última pincelada

- 9.1. En el equipamientos.json debemos copiar y pegar todo el trozo que hicimos para BMW (idCoche: 0) hasta dos veces, y modificarlo ahora para Mercedes y Audi
- 9.2. Vamos a Angular Material, al componente Tabs, y cogemos el html del ejemplo básico llamado “Tab Group Animation” → “Very Slow Animation” (en este ejemplo vienen los milisegundos que dura la animación y podremos jugar con ello)
- 9.3. En el equipamientos.component.html, cortamos toda la estructura que teníamos, de la carta del equipamiento, para pegar primero el ejemplo básico del MatTabs que hemos copiado, y poner en cada uno de sus tres contenidos, la carta que ya estaba antes y habíamos copiado.
- 9.4. A priori nos dará error, porque tenemos que ir al angular-material.module.ts para importar el MatTabsModule

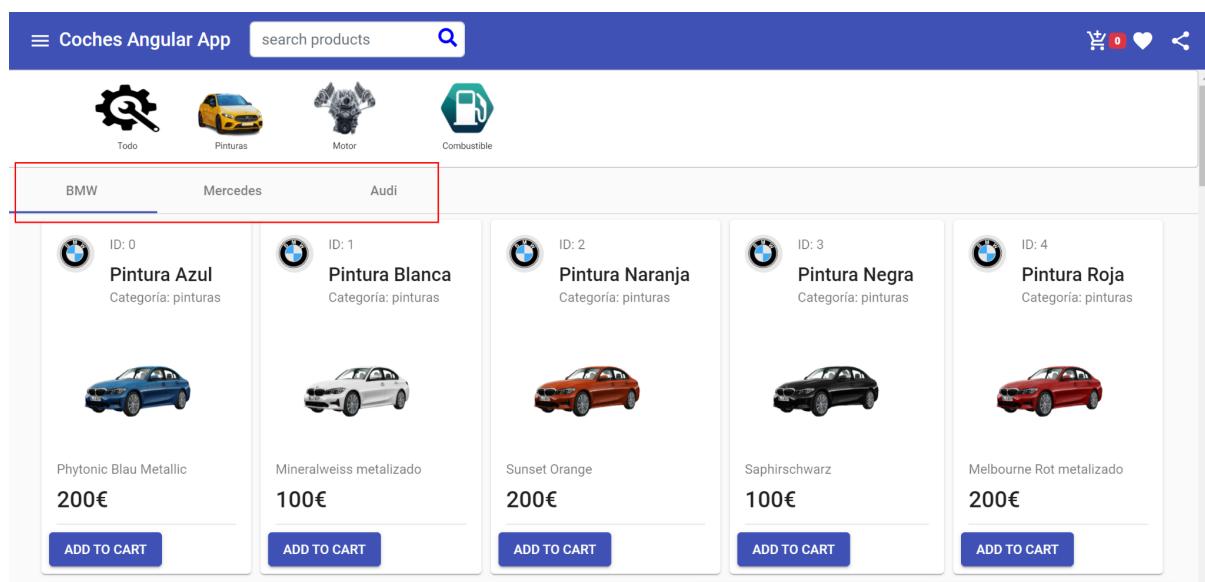
```
import { MatTabsModule } from '@angular/material/tabs';

@NgModule({
  declarations: [],
  imports: [MatTabsModule, MatProgressBarModule, MatListModule,
    MatSidenavModule, MatButtonModule,
    MatCheckboxModule, MatToolbarModule, MatInputModule, MatProgressSpinnerModule,
    MatCardModule, MatMenuModule, MatIconModule, CommonModule],
  exports: [MatTabsModule, MatProgressBarModule, MatListModule,
    MatSidenavModule, MatButtonModule,
```

```

MatCheckboxModule, MatToolbarModule, MatInputModule, MatProgressSpinnerModule, MatCardModule, MatMenuModule, MatIconModule, CommonModule]
} )

```



- 9.4. Llegados a este punto, vamos a pararnos a pensar el qué queremos conseguir, para poder llegar al cómo lo vamos a hacer.

Nosotros queremos separar los datos (equipamientos) según la marca de coche de la cual se trate.

Sin embargo, tenemos todos los datos en un mismo json, al cual se manda una única solicitud HttpClient, y se obtiene una única respuesta ...

Por aquí van los tiros, es decir, necesitamos separar los datos, para así poder obtener diferentes respuestas.

Vamos a empezar creando en assets, 3 nuevos json, los cuales serán:

- equipamientosBMW.json
- equipamientosMercedes.json
- equipamientosAudi.json

Acudiremos al equipamientos.json para copiar la parte de datos de cada marca, para pegarlo en el json correspondiente de cada marca

- 9.5. Ahora, debemos ir al equipamientos.service.ts para copiar y modificar tres veces la función que ya teníamos de getproduct() para obtener los equipamientos, por un lado de BMW, por otro lado de Mercedes, y por último de Audi

```

getProduct() {
  return this.httpClient.get<any>("../assets/equipamientos.json")
    .pipe(map((res: any) => {
      return res;
    })
}

```

```

        } ) );
    };

getBMWequipments() {
    return
this.httpClient.get<any>("../assets/equipamientosBMW.json")
    .pipe(map((resBMW: any) => {
        return resBMW;
    }));
};

getMercedesEquipments() {
    return
this.httpClient.get<any>("../assets/equipamientosMercedes.json")
    .pipe(map((resMercedes: any) => {
        return resMercedes;
    }));
};

getAudiEquipments() {
    return
this.httpClient.get<any>("../assets/equipamientosAudi.json")
    .pipe(map((resAudi: any) => {
        return resAudi;
    }));
};

```

- 9.6. Ahora vamos al equipamientos.component.ts para agregar 3 nuevas propiedades correspondientes a recibir los datos de cada una de las nuevas funciones que hemos incorporado justo antes en el equipamientos.service.ts.

También deberemos copiar y pegar hasta tres veces la parte del ngOnInit() para adaptarla a las diferentes marcas.

Y por último, copiaremos y pegaremos hasta tres veces el método de categoryFilter() para adaptarlo a cada una de las marcas. (previamente también habremos creado tres nuevas propiedades de filterCategoryBMW, filterCategoryMercedes, filterCategoryAudi)

El código final de equipamientos.component.ts sería el siguiente:

```
export class EquipamientosComponent implements OnInit {

  public equipmentsList: any;

  public equipmentsListBMW: any;
  public equipmentsListMercedes: any;
  public equipmentsListAudi: any;

  searchKey: string = "";

  public filterCategory: any;

  public filterCategoryBMW: any;
  public filterCategoryMercedes: any;
  public filterCategoryAudi: any;

  constructor(
    private equipamientoService: EquipamientoService,
    private cartService: CartService
  ) { }

  ngOnInit(): void {
    this.equipamientoService.getProduct()
      .subscribe((res) => {
        this.equipmentsList = res;
        this.filterCategory = res;

        this.equipmentsList.forEach((equipment: any) => {
          if (equipment.categoría === "pintura") {
            equipment.categoría = "pinturas";
          }
          else if (equipment.categoría === "motor") {
            equipment.categoría = "motores";
          }
          else if (equipment.categoría === "combustible") {
            equipment.categoría = "combustibles";
          }

          Object.assign(equipment, { quantity: 1, total: equipment.precio });
        });
        // console.log(this.equipmentsList);
      });
  }
}
```

```
this.equipamientoService.getBMWequipments()
.subscribe((resBMW) => {
    this.equipmentsListBMW = resBMW;
    this.filterCategoryBMW = resBMW;

    this.equipmentsListBMW.forEach((equipment: any) => {
        if (equipment.categoría === "pintura") {
            equipment.categoría = "pinturas";
        }
        else if (equipment.categoría === "motor") {
            equipment.categoría = "motores";
        }
        else if (equipment.categoría === "combustible") {
            equipment.categoría = "combustibles";
        }

        Object.assign(equipment, { quantity: 1, total:
equipment.precio });
    });
    // console.log(this.equipmentsList);
});

this.equipamientoService.getMercedesEquipments()
.subscribe((resMercedes) => {
    this.equipmentsListMercedes = resMercedes;
    this.filterCategoryMercedes = resMercedes;

    this.equipmentsListMercedes.forEach((equipment: any) => {
        if (equipment.categoría === "pintura") {
            equipment.categoría = "pinturas";
        }
        else if (equipment.categoría === "motor") {
            equipment.categoría = "motores";
        }
        else if (equipment.categoría === "combustible") {
            equipment.categoría = "combustibles";
        }

        Object.assign(equipment, { quantity: 1, total:
equipment.precio });
    });
    // console.log(this.equipmentsList);
});
```

```
}) ;

this.equipamientoService.getAudiEquipments()
.subscribe((resAudi) => {
    this.equipmentsListAudi = resAudi;
    this.filterCategoryAudi = resAudi;

    this.equipmentsListAudi.forEach((equipment: any) => {
        if (equipment.categoría === "pintura") {
            equipment.categoría = "pinturas";
        }
        else if (equipment.categoría === "motor") {
            equipment.categoría = "motores";
        }
        else if (equipment.categoría === "combustible") {
            equipment.categoría = "combustibles";
        }

        Object.assign(equipment, { quantity: 1, total:
equipment.precio });
    });
    // console.log(this.equipmentsList);
}) ;

this.cartService.search
.subscribe((value: any) => {
    this.searchKey = value;
}) ;
};

addToCart(equipment: any) {
    this.cartService.addToCart(equipment);
};

categoryFilter(category: string) {
    this.filterCategory = this.equipmentsList
    .filter((i: any) => {
        if (i.categoría === category || category === '') {
            return i;
        }
    }) ;
};
```

```

categoryFilterBMW(category: string) {
  this.filterCategoryBMW = this.equipmentsListBMW
    .filter((i: any) => {
      if (i.categoria == category || category == '') {
        return i;
      }
    }) ;
} ;

categoryFilterMercedes(category: string) {
  this.filterCategoryMercedes = this.equipmentsListMercedes
    .filter((i: any) => {
      if (i.categoria == category || category == '') {
        return i;
      }
    }) ;
} ;

categoryFilterAudi(category: string) {
  this.filterCategoryAudi = this.equipmentsListAudi
    .filter((i: any) => {
      if (i.categoria == category || category == '') {
        return i;
      }
    }) ;
} ;
}

```

- 9.7. Y el código final para `equipamientos.component.html`, con todas las modificaciones hechas, sería el siguiente:

```
<!-- <p>equipments works!</p> -->





###### Todo





###### Pinturas





###### Motor


```

```
        
        <h6>Combustible</h6>
        </a>
    </div>
</div>

</div>

<!-- ----- Equipment Card
----- -->

<mat-tab-group animationDuration="1500ms">

    <mat-tab label="BMW">
        <div class="superContainer" *ngIf="equipmentsListBMW != null ||
equipmentsListBMW != undefined">
            <mat-card class="example-card" *ngFor="let equipment of
filterCategoryBMW | filter:searchKey:'nombre'">

                <mat-card-header>
                    

                    <mat-card-subtitle>ID: {{ equipment.id
}}</mat-card-subtitle>
                    <mat-card-title>{{ equipment.nombre
}}</mat-card-title>
                    <mat-card-subtitle>Categoria: {{ equipment.categoria }}</mat-card-subtitle>
                </mat-card-header>

                <mat-card-content>
                    <mat-card-subtitle>{{ equipment.descripcion
}}</mat-card-subtitle>
                    <mat-card-title>{{ equipment.precio
}}€</mat-card-title>
                </mat-card-content>

                <mat-divider inset></mat-divider>
            </mat-card>
        </div>
    </mat-tab>
</mat-tab-group>
```

```
        <mat-card-actions>
            <button mat-raised-button color="primary"
(click)=addToCart(equipment)>ADD TO CART</button>
        </mat-card-actions>
    </mat-card>
</div>
</mat-tab>

<mat-tab label="Mercedes">
    <div class="superContainer" *ngIf="equipmentsListMercedes != null || equipmentsListMercedes != undefined">
        <mat-card class="example-card" *ngFor="let equipment of filterCategoryMercedes | filter:searchKey:'nombre'">

            <mat-card-header>
                

                <mat-card-subtitle>ID: {{ equipment.id }}</mat-card-subtitle>
                <mat-card-title>{{ equipment.nombre }}</mat-card-title>
                <mat-card-subtitle>Categoría: {{ equipment.categoria }}</mat-card-subtitle>
            </mat-card-header>

            <mat-card-content>
                <mat-card-subtitle>{{ equipment.descripcion }}</mat-card-subtitle>
                <mat-card-title>{{ equipment.precio }}€</mat-card-title>
            </mat-card-content>

            <mat-divider inset></mat-divider>

            <mat-card-actions>
                <button mat-raised-button color="primary"
(click)=addToCart(equipment)>ADD TO CART</button>
            </mat-card-actions>
        </mat-card>
    </div>
</mat-tab>
```

```
</div>
</mat-tab>

<mat-tab label="Audi">
    <div class="superContainer" *ngIf="equipmentsListAudi != null
|| equipmentsListAudi != undefined">
        <mat-card class="example-card" *ngFor="let equipment of
filterCategoryAudi | filter:searchKey:'nombre'">

            <mat-card-header>
                

                <mat-card-subtitle>ID: {{ equipment.id
}}</mat-card-subtitle>
                <mat-card-title>{{ equipment.nombre
}}</mat-card-title>
                <mat-card-subtitle>Categoría: {{ equipment.categoria
 }}</mat-card-subtitle>
            </mat-card-header>

            ADD TO CART</button>
            </mat-card-actions>
        </mat-card>
    </div>
</mat-tab>

</mat-tab-group>
```

≡ Coches Angular App

Todo Pinturas Motor Combustible

BMW Audi Mercedes

ID: 0 Pintura Azul Categoría: pinturas Phytonic Blau Metallico 200€ ADD TO CART	ID: 1 Pintura Blanca Categoría: pinturas Mineralweiss metalizado 100€ ADD TO CART	ID: 2 Pintura Naranja Categoría: pinturas Sunset Orange 200€ ADD TO CART	ID: 3 Pintura Negra Categoría: pinturas Saphirschwarz 100€ ADD TO CART	ID: 4 Pintura Roja Categoría: pinturas Melbourne Rot metalizado 200€ ADD TO CART
--	--	---	---	---

≡ Coches Angular App

Todo Pinturas Motor Combustible

BMW Audi Mercedes

ID: 0 Pintura Azul Categoría: pinturas Azul Ultra Metalizado 200€ ADD TO CART	ID: 1 Pintura Gris Categoría: pinturas Plata Florete Metalizado 100€ ADD TO CART	ID: 2 Pintura Azul Categoría: pinturas Arablau Kristalleffekt 200€ ADD TO CART	ID: 3 Pintura Negra Categoría: pinturas Negro Mito Metalizado 100€ ADD TO CART	ID: 4 Pintura Roja Categoría: pinturas Rojo Tango Metalizado 200€ ADD TO CART
--	---	---	---	--

≡ Coches Angular App

Todo Pinturas Motor Combustible

BMW Audi Mercedes

ID: 0 Pintura Negra Categoría: pinturas Negro 200€ ADD TO CART	ID: 1 Pintura Roja Categoría: pinturas MANUFAKTUR rojo jacinto metalizado 100€ ADD TO CART	ID: 2 Pintura Blanca Categoría: pinturas Blanco polar 200€ ADD TO CART	ID: 3 Pintura Azul Categoría: pinturas Azul náutico metalizado 100€ ADD TO CART	ID: 4 Pintura Gris Categoría: pinturas MANUFAKTUR gris selenita magno 200€ ADD TO CART
---	---	---	--	---

10. What's Next ??

Aquí concluye este ejercicio/práctica de Angular.

El componente de Inicio y el componente del 404, cada uno se busca el suyo propio !!

Para que el ejercicio quede más bonito, puedes añadirle más marcas de coches.

La idea de la nueva parte de este proyecto, no iba más lejos que la idea de hacer un Shopping Cart y adaptarlo a nuestro modelo de app de tipo Concesionario.

Sería interesante añadir alguna comprobación con if() para que el usuario sólo pudiese añadir al carrito una pintura de una sola marca, un motor de una sola marca, y un combustible de una sola marca, es decir, que todo lo que añada al carrito sea de la misma marca.

Hay que tener en cuenta, que la forma en que hemos pasado el el precio, la foto, y otros datos del coche desde EquipamientosComponent hacia CartComponent, es lo que deberíamos hacer para poder hacer el ejercicio lo más parecido posible a las webs de coches de verdad, porque en las de verdad, a la hora de comprar el coche, vas eligiendo los equipamientos, cada uno por separado, es decir, un componente para las pinturas, otro componente para el motor, y así sucesivamente hasta el resumen/factura final, pero siempre manteniendo abajo un <div> con position: fixed para que un pequeño cuadro resumen con el precio total acumulado vaya informando al usuario del dinero que lleva acumulado eligiendo sus equipamientos...

Por último decir que, este ejercicio continuaría con una API para realizar el pago, en la que el usuario pone sus datos bancarios/tarjeta y sus datos personales... y realizaría el pedido de su coche... un pedido que también podría verse desde otra API tipo Admin Dashboard ... pero eso ya será para otro ejercicio en el futuro !!