

# Ejercicio 1

Descargar este enunciado [pdf](#) o [html](#)

Partiendo del proyecto **Vehiculos** que contiene ya ciertos módulos y clases que nos van a ser útiles para la realización del examen y entre los cuales **podemos destacar**:

En el paquete `com.vahiculos.data` :

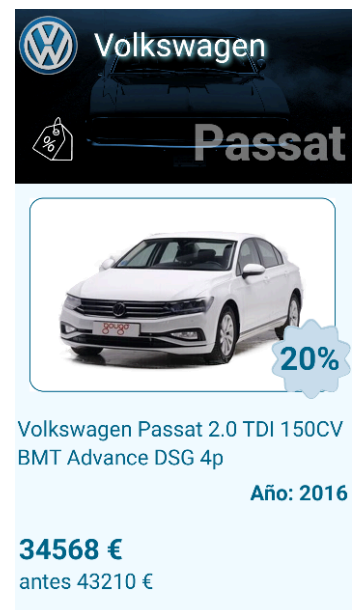
- En el paquete `.coche` están `CocheMock.kt` y `CocheDaoMock.kt` que contiene un listado de coches que vamos a utilizar para la primera carga de datos en la base de datos de Room.

En el paquete `com.vehiculos.ui.composables` :

- Dispones de las clases utilizadas en los materiales del curso denominadas `FieldTextCommon.kt` e `FilterChipCommon.kt` que puedes utilizar para algunos de los apartados del examen.

En el paquete `com.vehiculos.ui.features` :

- Dispones del data class `CocheUiState.kt` que vamos a utilizar para guardar el estado de la ficha de un coche y de los items de la lista.
- En el paquete `.fichacoche` dispones de `FichaCocheScreen.kt` que contiene la implementación de la UI de la ficha de un coche y que deberás completar según las especificaciones.
- En el paquete `.galeriacoches` dispones de `ItemListaCoches.kt` que contiene la definición de la UI para ver un coche en la lista a modo de galería de coches y que puedes ver en la imagen de ejemplo justo abajo.



En el paquete `com.vehiculos.utilities` :

- Dispones de diferentes clases de utilidad utilizadas en los materiales del curso y que puedes utilizar si lo consideras necesario de forma opcional.

# Especificaciones

1. Crea un BD con Room para almacenar todos los datos de nuestro modelo denominada "**coches.db**". La BD debe contener una única tabla denominada **coches** donde la clave primaria será el campo **id** de tipo **Int** que tienes en el modelo. Los campos de las tabla deben ir en **snake\_casing**



## Nota

Fíjate bien el los datos del mock para definir **CocheEntity** y fíjate que la foto es de tipo **String?** que contiene la URL de la imagen o null si no tiene.

Las operaciones sobre la tablas serán todas asíncronas y debes definir las siguientes:

- Operaciones básicas **insert** , **delete** , **update** y **count**
- **get()** : Devuelve todos los coches de la base de datos.
- **get(Int)** : Devuelve un coche de la base de datos a partir de su id.
- **getOertas()** : Devuelve todos los coches de la base de datos que tengan un porcentaje de descuento mayor que 0.  

```
SELECT * FROM coches WHERE porcentaje_descuento > 0
```
- **getOrdenadosPrecio()** : Devuelve todos los coches de la base de datos ordenados por precio de menor a mayor.  

```
SELECT * FROM coches ORDER BY precio ASC
```
- **getOertasOrdenadasPrecio()** : Devuelve todos los coches de la base de datos que tengan un porcentaje de descuento mayor que 0 ordenados por precio de menor a mayor.

2. Prepara los '*providers*' que consideres necesarios de las instancias de Room para la inyección de dependencias con **Hilt**. Teniendo en cuenta que se creará una **instancia única** de cada uno de ellos.
3. Modifica el '*repositorio*' para usar los métodos definidos.
4. Realiza la primera carga de datos de la base de datos de Room a partir de los datos **CocheDaoMock** . **Solo si la BD está vacía.**

Para ello, realiza la carga de datos en la BD al iniciar la aplicación. Utilizando el método **onCreate** de la clase **VehiculosApplication** .

Puedes inyectar de forma sencilla los datos de prueba y el Dao o el Repositorio de forma simple utilizando Hilt de la siguiente manera:

```

@HiltAndroidApp
class VehiculosApplication : Application() {
    @Inject
    lateinit var daoMock: CocheDaoMock
    @Inject
    lateinit var daoEntity: CocheDao
    ...
}

```



## Nota

Recuerda que los métodos de Room deben ser asíncronos y que debes tener en cuenta que la carga de datos se realiza solo si la BD está vacía.

5. Por último, comprueba con el App Inspection que la carga de datos se ha realizado correctamente.

App Inspection

Pixel 3a API 33 > com.vehiculos

Database Inspector | Network Inspector | Background Task Inspector

Databases

coches

Live updates

	id	fabricante	modelo	año	precio	porcentaje_descuento	descripcion	foto
coches.db	1	Volkswagen	Passat	2016	43210.0	0	Volkswagen Passat 2.0 TDI 15l	foto_1
coches	2	Ford	Mustang	2017	55000.0	10	Ford Mustang 5.0 Ti-VCT V8	foto_2
coches	3	Audi	A4	2018	46000.0	0	Audi A4 2.0 TDI 110kW (150CV)	foto_3
coches	4	BMW	X5	2019	86800.0	20	BMW X5 xDrive30d 5p	foto_4
coches	5	Mercedes	Clase C	2020	50000.0	0	Mercedes-Benz Clase C C 220	foto_5
coches	6	Volkswagen	Golf	2021	29000.0	0	Volkswagen Golf 1.5 TSI 96kW	foto_6
coches	7	Ford	Focus	2022	30000.0	5	Ford Focus 1.0 EcoBoost 92kW	foto_7
coches	8	Audi	A3	2023	32000.0	0	Audi A3 Sportback 30 TFSI 81	foto_8
coches	9	BMW	Serie 3	2020	45200.0	0	BMW Serie 3 320d 140kW (190	foto_9
coches	10	Mercedes	Clase A	2021	38200.0	0	Mercedes-Benz Clase A A 200	foto_10
coches	11	Audi	e-tron	2022	87310.0	10	Audi e-tron 55 quattro 265kW	foto_11
coches	12	BMW	Serie 5	2023	60000.0	20	BMW Serie 5 520d 140kW (190	foto_12
coches	13	Mercedes	Clase G	2020	148225.0	0	Mercedes-Benz Clase G G 350	foto_13
coches	14	Volkswagen	Tiguan	2021	39000.0	0	Volkswagen Tiguan 2.0 TDI 110	foto_14
coches	15	Ford	Kuga	2022	38250.0	0	Ford Kuga 1.5 EcoBoost 110kW	foto_15
coches	16	Audi	Q3	2023	43270.0	0	Audi Q3 35 TFSI 110kW (150CV)	foto_16
coches	17	Ford	Mondeo	2021	32000.0	10	Ford Mondeo 2.0 TDCI 110kW	foto_17