Adrián Pérez

FILTRAR POR POTENCIA

PLAN DE PRUEBAS

PRUEBAS DE ACEPTACIÓN

Serán llevadas a cabo por el Product Owner durante el Sprint Review.

Se identifican los siguientes escenarios:

Prueba 00: éxito

1. El usuario selecciona filtrar por potencia.
2. El sistema muestra al usuario las posibles potencias por las que se puede filtrar.
3. El usuario selecciona la potencia con la que desea hacer el filtrado.
4. El usuario selecciona aceptar filtro.
5. El sistema muestra al usuario todos los puntos de carga con esa potencia.

Prueba 01: No existe ningún punto de carga con los filtros aplicados.

1. El usuario selecciona filtrar por potencia.
2. El sistema muestra al usuario las posibles potencias por las que se puede filtrar.
3. El usuario selecciona la potencia con la que desea hacer el filtrado.
4. El usuario selecciona aceptar filtro.
5. Se verifica que el sistema muestre un mensaje alertando de que no existe ningún punto de carga con los filtros aplicados.

Prueba 02: Selección de cancelar durante la aplicación de filtros sin ningún filtro escogido.

1. El usuario selecciona filtrar por potencia.
2. El sistema muestra al usuario las posibles potencias por las que se puede filtrar.
3. El usuario selecciona la potencia con la que desea hacer el filtrado.
4. El usuario selecciona cancelar.
5. Se verifica que el sistema vuelve a la ventana principal.

Prueba 03: Selección de cancelar durante la aplicación de filtros con algún filtro escogido.

1. El usuario selecciona filtrar por potencia.
2. El sistema muestra al usuario las posibles potencias por las que se puede filtrar.
3. El usuario selecciona cancelar.
4. Se verifica que el sistema vuelve a la ventana principal.

Prueba 04: Pulsar fuera de los márgenes de la ventana emergente de aplicación de filtros de potencia con algún filtro escogido.

1. El usuario selecciona filtrar por potencia.
2. El sistema muestra al usuario las posibles potencias por las que se puede filtrar.
3. El usuario selecciona la potencia con la que desea hacer el filtrado.
4. El usuario pulsa fuera de los márgenes de la ventana emergente.
5. Se verifica que el sistema vuelve a la ventana principal.

Prueba 05: Pulsar fuera de los márgenes de la ventana emergente de aplicación de filtros de potencia sin ningún filtro escogido.

1. El usuario selecciona filtrar por potencia.

2. El sistema muestra al usuario las posibles potencias por las que se puede filtrar.
3. El usuario pulsa fuera de los márgenes de la ventana emergente.
4. Se verifica que el sistema vuelve a la ventana principal.

Prueba 06: Los puntos de carga que no proporcionan información sobre su potencia no aparecen tras aplicar el filtrado.

1. El usuario selecciona filtrar por potencia.
2. El sistema muestra al usuario las posibles potencias por las que se puede filtrar.
3. El usuario selecciona la potencia con la que desea hacer el filtrado.
4. El usuario selecciona aceptar filtro.
5. Se verifica que el sistema no muestra los puntos de carga que no disponen de información sobre su potencia.

PRUEBAS INTERFAZ

Las pruebas serán automatizadas a través de Espresso. Para esta prueba se utilizarán los puntos de carga proporcionados por chargers_es_100.json

Se identifican los siguientes escenarios:

Filtrar por potencia

a. Filtrado válido seleccionando solo una potencia, sin aparecer los puntos de carga que no aporten información sobre su potencia

b. Filtrado válido seleccionando más de una potencia, sin aparecer los puntos de carga que no aporten información sobre su potencia

c. Filtrado válido, pero sin puntos de carga existentes para ese filtro

d. Filtrado cancelado sin filtros(el usuario cancela el filtrado seleccionando el botón de cancelar sin haber seleccionado algún filtro)

e. Filtrado cancelado con filtros(el usuario cancela el filtrado seleccionando el botón de cancelar habiendo seleccionado algún filtro)

f. Filtrado incorrecto tras pulsar fuera de la ventana emergente sin haber seleccionado algún filtro

g. Filtrado incorrecto tras pulsar fuera de la ventana emergente habiendo seleccionado algún filtro

Tabla 1. Casos de prueba de interfaz

Identificador	Entrada	Resultado
A1.a	Elige filtrar por potencia, selecciona 7.4KW, y pulsa aceptar	Lista de puntos de carga filtrados con esa potencia. Deben aparecer exactamente 7.

A1.b	Elige filtrar por potencia, selecciona 7.4KW y 43KM, y pulsa aceptar	Lista de puntos de carga filtrados con esas potencias. Deben aparecer exactamente 28. EL primero debe ser el de Repsol Ibil de Repsol la Raza(Andalucía) y el último el de la calle del puerto del ayuntamiento de Huelva.
A1.c	Elige filtrar por potencia, selecciona 2.3KW y pulsa aceptar	Lista vacía de puntos de carga y aparece un mensaje informativo.
A1.d	Elige filtrar por potencia y pulsa cancelar	Se vuelve a la ventana de tipos de filtrado disponibles
A1.e	Elige filtrar por potencia, selecciona 50KW y pulsa cancelar	Se vuelve a la ventana de tipos de filtrado disponibles
A1.f	Elige filtrar por potencia y pulsa fuera de la ventana emergente	Se vuelve a la pantalla inicial de la aplicación
A1.g	Elige filtrar por potencia, selecciona 50KW y pulsa fuera de la ventana emergente	Se vuelve a la pantalla inicial de la aplicación

Decido implementar el caso A1.b mencionado anteriormente como prueba de interfaz.

MÉTODOS SUSCEPTIBLES DE PRUEBAS

Clase **MainView.java**:

Método showLoadErrorDialog -> Crea un alertDialog que avisa de un error determinado

Método filtrosDialog -> Crea un alertDialog para elegir los filtros

Método filtrarPotenciasDialog -> Se encarga de mostrar la ventana emergente de filtrado y mandarle a MainPresenter las potencias seleccionadas para filtrar.

Método onOptionsItemSelected -> Se añade la pestaña de filtros

Clase **MainPresenter.java**:

Método onAceptarFiltroPotenciaClicked -> Se encarga de realizar el filtrado por potencia. Recibe como parámetro las potencias a filtrar(se las pasa el MainView.java desde filtrarPotenciasDialog)

Clase **Charger.java:**

Método ListarTiposConector

Método contienePotencia -> Devuelve true si alguno de los cargadores de ese punto tiene la potencia pasada como parámetro.

PRUEBAS UNITARIAS

Usaremos JUnit y Mockito(solo en el primer método) para el desarrollo de las pruebas unitarias.

Clase **MainPresenter.java** método onAceptarFiltroPotenciaClicked:

```
public void onAceptarFiltroPotenciaClicked(List<Double> potencias)
```

Se asume que las potencias son válidas porque se fuerza al usuario a que seleccione unas que ya están preestablecidas a través de la interfaz de la aplicación.

Utilizamos mock de MainView en el constructor de MainPresenter.

Existen seis casos:

A. Filtrado con varios cargadores y un solo tipo de potencia
B. Filtrado con varios cargadores y varias potencias
C. Filtrado pero no se encuentran cargadores para esas potencias
D. Filtrado con solo un cargador y varias potencias
E. Un cargador con más de un conector coincidente con las potencias pasadas por parámetro
F. Se filtra sin seleccionar ningún tipo de potencia
G. Se filtra pero no existe ningún cargador.

a=charger

c=conector

Entrada	Resultado
Param: 7.4KW Tenemos: a(c=7.4KW), a2(c2=43KW), a3(c3=7. 4KW)	[a,a3]
Param: 7.4KW y 43.0KW	[a, a2,a3]

Tenemos: a(c=7.4KW), a2(c2=7.4KW), a3(c3=43KW)	
Param: 7.4KW y 43.0KW Tenemos: a(c=50KW), a2(c2=50KW), a3(c3=50KW)	[]
Param: 7.4KW y 43.0KW Tenemos: a(c=7.4KW)	[a]
Param: 7.4KW y 43.0KW Tenemos: a(c=50KW, c2=7.40KW, c3=43KW), a2(c=50KW)	[a]
Param: Tenemos: a(c=50KW, c2=7.40KW, c3=43KW), a2(c=50KW)	[a, a2]
Param: 7.4KW Tenemos:	[]

Clase **Charger.java** método listarTiposConector:

Para la realización de este test solo se ha utilizado la herramienta JUnit.

```
public List<String> listarTiposConector(){
```

Existen tres casos:

A. Sin ningún conector
B. Con un conector
C. Con varios
D. Caso en el que dos conectores tengan el mismo title.

Entrada	**Resultado**
nada	Lista de conectores vacía.
Conector c1	Lista de conectores con tamaño 1.

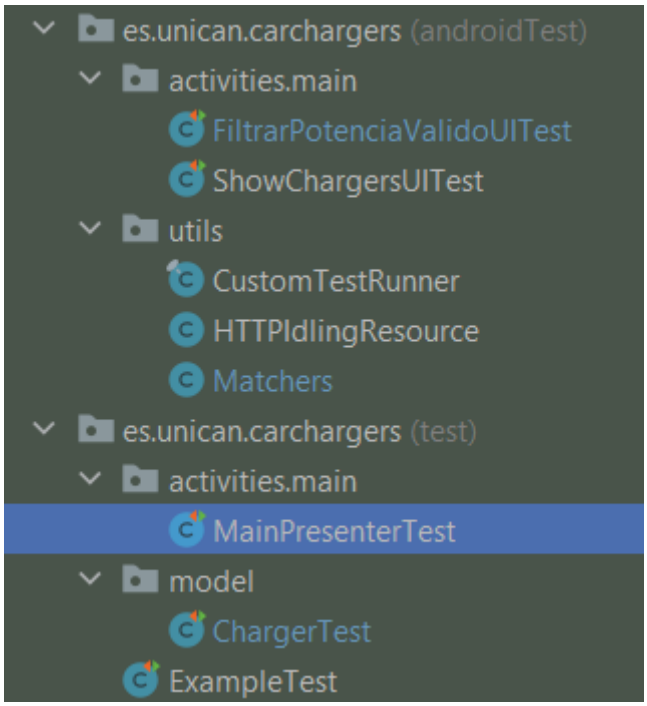
Conector c1, c2 y c3	Lista de conectores con tamaño 3.
Conector c1, c2 y c3 c1 y c2 tienen el mismo title.	Lista de conectores con tamaño 3. A pesar de tener el mismo título se añaden a la lista por separado

REPORTE FINAL

En primer lugar, respecto a las pruebas de interfaz veo necesario comentar el hecho de que tuve inicialmente ciertos problemas para su implementación en código principalmente a la hora de contar el número de cargadores que salían por pantalla filtrados, solucionándolo finalmente creando un matcher que se encargase de ello. Además, tuve que comprender la estructura del fichero chargers_es_100.json porque necesitaba encontrar los ID de los puntos de carga que aparecen tanto en primer lugar como en último tras realizar el filtrado para así en el test comprobar que se ha hecho dicho filtrado correctamente.

En cuanto a las pruebas unitarias, me resultó más compleja la del método “onAceptarFiltroPotenciaClicked” ya que requirió el uso de Mocks y un mayor número de casos de prueba en comparación al método “listarTiposConector”. Es importante destacar que se creó un mock para poder simular el comportamiento de la clase "MainView" dado que esta es una vista propia de Android.

Los tests se organizaron en carpetas de acuerdo a lo establecido en las normas, estando las pruebas unitarias dentro de es.unican.carchargers (test) y la de interfaz en es.unican.carchargers (androidTest)



Gracias a los tests de “onAceptarFiltroPotenciaClicked” pudimos detectar y solucionar un fallo en el código del propio método que sucedía cuando no existían cargadores que se ajustasen a la búsqueda de potencias dada.

Por último, destacar que tanto el plan de pruebas como las pruebas fueron implementadas y desarrolladas por Adrián Pérez.