**Plan de pruebas US 500954 Filtrar por marcas.**

Tabla 1. Casos de prueba de aceptación

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Entrada** | **Resultado** |
| A1 | [“REPSOL”] | Lista y toast con 45 gasolineras. |
| A2 | [“REPSOL”, “BALLENOIL”] | Lista y toast con 51 gasolineras. |
| A3 |  | Lista y toast con 164 gasolineras. |
| A4 | [“REPSOL”] | Lista y toast con 164 gasolineras. |
| A5 | [“Todas”] | Lista y toast con 164 gasolineras. |
| A6 | [“SVM S.L.”] | Mensaje de fallo y toast con 0 gasolineras. |

**PRUEBAS UNITARIAS.**

En esta historia de usuario no se modifica nada respecto al acceso y la persistencia de datos. Por lo tanto, las pruebas unitarias consistirán en pruebas en las clases de dominio, negocio y presentación.

**Pruebas unitarias de dominio.**

Deberían proberse los métodos de la clase Filter.

* + setGasBrands( gasBrands : List<String> ) : IFilter
  + toFilter(g : List<Gasolinera>) : List<Gasolinera>
  + brandsFilter( g : Gasolinera) : Boolean
  + clear() : Void
* Método brandsFilter( g : Gasolinera) : Boolean

|  |  |  |
| --- | --- | --- |
| Identificador | Entrada | Valor esperado |
| UD1.a | Filter(“REPSOL”) Gasolinera(“REPSOL”) | true |
| UD1.b | Filter(“SVM”) Gasolinera(“REPSOL”) | false |
| UD1.c | Filter() Gasolinera(“REPSOL”) | true |
| UD1.d | Filter(“REPSOL”) Gasolinera(null) | false |

* clear() : Void

|  |  |  |
| --- | --- | --- |
| Identificador | Entrada | Valor esperado |
| UD2.a | Filter(“REPSOL”) | Filter(null) |

**Pruebas unitarias de negocio:**

Deberían probarse los métodos de la clase MainPresenter para ello serán necesarias las clases de GasolinerasRepository, IFilter, Filter y un Mock de IMainConctract.View.

* Método getBrandsSelections()
* Método onFiltersPopUpBrandsSelected()
* Método onFiltersPopUpBrandsOneSelected(int index, boolean value)
* Método onFiltersPopUpAccepted()
* Método setFiltersPopUpValues()

**Método onFiltersPopUpBrandsOneSelected(int index, boolean value):**

|  |  |  |
| --- | --- | --- |
| **Identificador.** | **Entrada.** | **Valor esperado.** |
| **UGIC.1a** | index = 0  value = true  tempListSelection= [(“Todos”, false), (“Marca1”, true), (“Marca2”, false)] | Llama a: view.updateFiltersPopUpBrandsSelection(index, value) para desactivar todas las marcas que no sean “Todos”. Para activar todos, llamará a: view.updateFiltersPopUpBrandsSelection(index, value).  Se mostrarán todas las gasolineras de todas las marcas. |
| **UGIC.1b** | index = 0  value = false  tempListSelection= [(“Todos”, true), (“Marca1”, false), (“Marca2”, false)] | Llama a: view.updateFiltersPopUpBrandsSelection(index, value) porque siempre debe haber algo seleccionado.  Se mostrarán todas las gasolineras de todas las marcas. |
| **UGIC.1c** | index = 1  value = true  tempListSelection= [(“Todos”, true), (“Marca1”, false), (“Marca2”, false)] | Llama a: view.updateFiltersPopUpBrandsSelection(index, value) para desactivar la casilla de todos.  Llama a:  view.updateFiltersPopUpBrandsSelection(index, value) para mostrar los detalles de una gasolinera específica.  Se mostrarán todas las gasolineras de la marca seleccionada. |
| **UGIC.1d** | index = 2  value = true  tempListSelection= [(“Todos”, false), (“Marca1”, true), (“Marca2”, false)] | Llama a: view.updateFiltersPopUpBrandsSelection(0, true) para activar la casilla de todos.  Llama a:  view.updateFiltersPopUpBrandsSelection(i, false) para desmarcar todas las gasolineras marcadas.  Se mostrarán todas las gasolineras de todas las marcas. |
| **UGIC.1e** | index = 0  value = false  tempListSelection= [(“Todos”, true), (“Marca1”, true), (“Marca2”, false)] | Llama a: view.updateFiltersPopUpBrandsSelection(index, value) para desactivar la casilla de todos.  Se mostrarán únicamente las gasolineras de las marcas previamente marcadas. |

**Método onFiltersPopUpBrandsSelected():**

|  |  |  |
| --- | --- | --- |
| **Identificador.** | **Entrada.** | **Valor esperado.** |
| **UGIC.2a** | TempList= [“Marca1”] | Llama a getBrandsSelections(tempFilter) que debe generar correctamente la lista con “Marca 1”.  tempListSelection= [(“Todos”, false), (“Marca1”, true), (“Marca2”, false)]  Llama a view.showFiltersPopUpBrandSelector(tempList Selection) con los valores correctos. |
| **UGIC.2b** | tempList = [] | Llama a getBrandsSelections(tempFilter) que debe generar correctamente la lista con “Todos”.  tempListSelection= [(“Todos”, false), (“Marca1”, true), (“Marca2”, false)]  Llama a view.showFiltersPopUpBrandSelector(tempListSelection). |
| **UGIC.2c** | tempList = [“Marca1”, “Marca 2”] | Llama a getBrandsSelections(tempFilter) que debe generar correctamente la lista con “Todos”.  tempListSelection= [(“Todos”, true), (“Marca1”, false), (“Marca2”, false)]  Llama a view.showFiltersPopUpBrandSelector(tempListSelection). |

**PRUEBAS DE INTEGRACIÓN.**

Deberán probarse los métodos de la clase MainPresenter, para ello usaremos la clase GasolinerasRepository, Filter y el mockup de I MainConctract.View (Se usará la opción de cargar datos mediante json en vez del servicio).

* Método onFiltersClicked()
* Método onFiltersPopUpBrandsSelected()
* Método onFiltersPopUpBrandsOneSelected(int index, boolean value)
* Método onFiltersPopUpBrandsAccepted()
* Método onFiltersPopUpAcceptClicked()
* Método onFiltersPopUpClearFiltersClicked()
* Método load ()
* Método getBrandsSelections (IFilter f)

**Método onFiltersPopUpBrandsAccepted():**

|  |  |  |
| --- | --- | --- |
| **Identificador.** | **Entrada.** | **Valor esperado.** |
| **IT.1a** | tempListSelection= [(“Todos”, true), (“Marca1”, false), (“Marca2”, false)]  tempList=[“Marca1”] | Se deberán mostrar todas las gasolineras. Además, se deberá verificar que tempList= [“Todos”] y que se realiza una llamada a los métodos tempFilter.setBrands(Arrays.*asList*(BrandsEnum.*values*())) y a view.updateFiltersPopupTextViews(null, getStringOfSelections(tempListSelection)). |
| **IT.1b** | tempListSelection= [(“Todos”, false), (“Marca1”, true), (“Marca2”, true)]  tempList=[“Todos”] | Se deberá comprobar que se muestran las gasolineras de las marcas seleccionadas. Además, se verificará que tempList poseer las marcas seleccionadas. |
| **IT.1c** | tempListSelection= [(“Todos”, false), (“Marca1”, false), (“Marca2”, false)]  tempList=[“Marca1”] | Se deberá comprobar que se muestran todas las gasolineras. |
| **IT.1e** | tempListSelection= [(“Todos”, true), (“Marca1”, false), (“Marca2”, false)]  tempList=[] | Se deberá verificar que cuando la lista esta vacía y se añaden todas las marcas, se muestran todas las gasolineras. |

**PRUEBAS DE UI.**

En este caso se aplica la técnica basada en casos de uso para la definición de las pruebas a realiza. Los casos de prueba definidos serán los mismo que los de las pruebas de aceptación, pero automatizados a través de Junit y Expresso.

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Entrada** | **Resultado** |
| UIT.1 | [“REPSOL”] | Lista de gasolineras y toast con 45 gasolineras. |
| UIT.2 | [“REPSOL”,  “BALLENOIL”] | Lista de gasolineras y toast con 164 gasolineras. |
| UIT.3 | [“Todas”] | Lista con todas las gasolineras y toast con 164 gasolineras. |
| UIT.4 |  | Lista con todas las gasolineras y toast con 164 gasolineras. |