PLAN DE PRUEBAS PARA LA HISTORIO DE USUARIO: FILTRAR POR PALABRAS CLAVE

Arronti Cultura

1. Pruebas de Aceptación

Prueba 00: Éxito al mostrar
1. El usuario escribe ciertas palabras clave.
2. El sistema muestra la lista de eventos de las palabras clave escritas en el paso 1.
3. Se verifica que los eventos que se han mostrado corresponden a las palabras clave y que se muestra un Toast al usuario confirmándole que se ha realizado la búsqueda.

Prueba 01: Se busca sin introducir ninguna palabra clave
1. El usuario no escribe palabras clave, pero le da a ENTER.
2. Se verifica que el sistema no aplica ningún tipo de filtro sobre la lista de eventos.

Prueba 02: Imposibilidad de obtención de eventos de ciertas palabras clave
1. El usuario escribe ciertas palabras clave.
2. El sistema no puede mostrar la lista de eventos filtrada porque no hay eventos disponibles relacionados con esas palabras clave.
3. Se verifica que el sistema muestra un Toast en la que se le notifica al usuario de que no hay eventos disponibles relacionados con esas palabras y se le recomienda filtrar por otras palabras clave.

Prueba 03: Se buscan palabras con mayúsculas
1. El usuario escribe palabras con mayúsculas.
2. Se verifica que el sistema busca sin tener en cuenta si hay mayúsculas o no.

Prueba 04: Se buscan palabras con tildes
1. El usuario escribe palabras con tildes.
2. Se verifica que el sistema busca sin tener en cuenta si hay tildes o no.

Prueba 05: Se buscan palabras con la letra Ñ
1. El usuario escribe palabras con la letra Ñ.
2. Se verifica que el sistema busca teniendo en cuenta las Ñ.

2. Pruebas Unitarias

Los casos de prueba para el método *onKeywordsFilter()* de la clase *EventsPresenter* se encuentran codificados en la clase *EventsPresenterTest* y son los siguientes:

Identificador	Entrada	Valor Esperado
UGIC.1a	“Palacio de Festivales”	Guarda en *cachedEvents* 5 coincidencias
UGIC.1b	Null	Vuelve a dejar la lista inicial con 345 eventos
UGIC.1c	“PalabraRara”	Lista vacía en *cachedEvents*

3. Pruebas de Integración

Los casos de prueba para el método *loadData()* de la clase *EventsPresenter* se encuentran codificados en la clase *EventsPresenterITest* y son los siguientes:

Identificador	Entrada	Valor Esperado
IGIC.1a	Repositorio No Disponible	cachedEvents=null
IGIC.1b	Repositorio Disponible	cachedEvents.size() = 345
IGIC.1c	Repositorio No Disponible tras ya tener la lista de eventos y recargar.	cachedEvents.size() = 345

4. Pruebas de Interfaz

Las pruebas de interfaz serán automatizadas a través de *Espresso* y se encuentran codificadas en la clase *EventsActivityUITest*

Identificador	Entrada	Resultado
IVF.1a	“Palacio de Festivales” (Se va escribiendo poco a poco por la búsqueda automática y comprobando que va buscando poco a poco)	Lista con cinco coincidencias
IVF.1b	“Palacio de Festivales” sin internet	El dispositivo no tiene conectividad.
IVF.1c	“PalabraRara” (Se va escribiendo poco a poco por la búsqueda automática y comprobando que va buscando poco a poco)	No existen eventos disponibles relacionados.
IVF.1d	Limpiar los filtros con el botón ArrontiCultura	Lista con todos los eventos (345)

5. Ejecución de las pruebas y fallos encontrados

5.1 Pruebas Unitarias

No se encuentran problemas.

5.2 Pruebas de Integración

No se encuentran problemas.

5.3 Pruebas de Interfaz

No se encuentran problemas en la interfaz. Si se han encontrado problemas en la codificación de las pruebas a la hora de comprobar los Toast y el tamaño de la lista. Para comprobar el Toast se ha utilizado un DecorView y para comprobar el tamaño de la lista se ha creado un método que devolver un matcher al pasarle un tamaño de lista para comprobar si coincide en un check(matcher()).

Plan realizado y ejecutado por Mario Fernández González