Manuel Fernández

Plan de pruebas: Añadir comentario parada

Nuestro proyecto se dividirá en tres capas: vista, presenter y DAO, en este documento se especificará las diferentes pruebas que se realizarán para comprobar el correcto funcionamiento de las capas conjunta e independientemente. El tipo de pruebas a realizar sobre el código son unitarias, de integración y de aceptación.

Se comenzará realizando las pruebas unitarias, éstas se agruparán dependiendo de la capa en la que se encuentren los métodos que se vayan a comprobar. En la capa DAO se encontrarán las clases RemoteFetch, ParseJson, Línea, Parada y Database. Se obviará realizar test sobre los métodos “get” y “set” de estas clases, los cuales trabajan con atributos privados de la clase, suponiendo así que en estos métodos no puede haber error alguno y sobre métodos cuya funcionalidad ya se ha probado anteriormente. Por lo tanto, las pruebas de esta historia en esta capa se realizarán sobre la clase Database.

En la capa presenter se deberán probar métodos de la clase recientemente añadida DetallesParadaPresenter, algunos de estos métodos únicamente sirven como conexión entre las otras dos capas del proyecto, por lo que se podrá comprobar su correcto funcionamiento durante la realización de las pruebas de aceptación.

El caso de la capa vista es diferente, ya que los métodos existentes en este nivel son propios de Android o triviales, así que no se verificará su correcto comportamiento mediante pruebas unitarias. Aun conociendo estos datos, el correcto funcionamiento de la capa vista podrá ser comprobado durante las pruebas de aceptación. Por lo tanto, si se tiene en cuenta estas pautas, deberemos realizar los siguientes casos de prueba:

CASOS DE PRUEBA

Database: getCommentParada()

a. Lectura satisfactoria (parada obtenida).
b. Lectura satisfactoria, parada sin comentarios (parada obtenida)
c. Lectura no satisfactoria, parada no existente (parada == null)
d. Lectura no satisfactoria, id no correspondiente a la parada (parada ==null)

La comprobación del addComment() se probará llamando al anterior método implementado, getCommentParada() .

Database: addComment()

a. Agregación satisfactoria a parada sin comentario (comprobar parada con get)
b. Actualización satisfactoria a parada con comentario (comprobar parada con get)
c. Agregación satisfactoria de comentario vacío a parada sin comentario (comprobar parada con get)
d. Agregación no satisfactoria id repetido (muestra mensaje informativo)
e. Agregación satisfactoria de comentario con carácter conflictivo (comprobar parada con get).

PRUEBAS UNITARIAS

Para implementar estas pruebas se utilizarán los casos de prueba detallados utilizando únicamente el método cuya funcionalidad quiere ser comprobada. Durante la realización de estos test será necesario desarrollar diferentes Json locales que permitan conseguir las diferentes situaciones necesarias, si se sigue el mismo orden en el que se ha detallado, se debería implementar unos test sobre estas características:

En la clase Database encontramos tanto el método getCommentParada()

Caso de prueba	**Nombre parada**	**Id parada**	**Resultado**
U12. a	La Pereda	463	Parada (La Pereda, Cerca de un buen restaurante, 435031.38, 4814426.15, 463)
U12. b	Camarreal Peñacastillo	499	Parada (Camarreal Peñacastillo , , 429986.36, 4811045.72, 499)
U12.c	Unican	5	null
U12.d	La Pereda	1	null

Como el método addComment()

Caso de prueba	**Id**	**Nombre**	**Comentario**	**Llamada a get**
U13. a	463	La Pereda	Cerca de un buen restaurante	Parada (La Pereda, Cerca de un buen restaurante, 435031.38, 4814426.15, 463)
U13. b	463	La Pereda	Cerca de un buen restaurante que cerró.	Parada (La Pereda, Cerca de un buen restaurante que cerró, 435031.38, 4814426.15, 463)
U13. c	499	Camarreal Peñacastillo	-	Parada (Camarreal Peñacastillo ,,429986.36,4811045.72, 499)
U13.d	463	Camarreal Peñacastillo	Cerca de un buen restaurante	Mensaje informativo previo
U13.e	499	Camarreal Peñacastillo	Junto al parque de la araña	Parada (Camarreal Peñacastillo , Junto al parque de la araña ,429986.36,4811045.72, 499)

Una vez comprobado el correcto funcionamiento de los métodos de la capa DAO, se debería realizar el mismo procedimiento con los posibles métodos conflictivos de las otras dos capas del proyecto. Tanto en la capa presenter como en la capa vista, se considerará que la comprobación

de su correcto comportamiento se puede realizar mediante los siguientes tipos de test al contener estas capas métodos cuya funcionalidad ya ha sido probada o ser estos triviales.

PRUEBAS DE INTENGRACIÓN

Para la realización de estas pruebas se deberán comprobar la correcta funcionalidad entre capas, y no de forma independiente como se ha realizado anteriormente. En esta historia, se ha añadido la clase DetallesParadaPresenter a la capa presenter. Los métodos existentes a esta nueva clase se consideran triviales, o su única función es la conexión básica entre las otras dos capas del proyecto. Por ello, su funcionalidad se considerará como óptima si se pasan las pruebas de aceptación pertinentes.

TEST DE ACEPTACIÓN

Se realizarán los test de aceptación en función a lo pedido por el product owner, y sus peticiones fueron claras. En el momento en el que la aplicación muestre alguna lista de paradas, debe existir la posibilidad de seleccionar una y añadir a esta un comentario personalizado que perdure en el tiempo. A estas indicaciones se le añadirán una serie de pruebas alternativas para aumentar su cobertura. Para realizar esta comprobación se realizarán las pruebas sobre un Nexus 5 API 25, se proporciona este detalle ya que los test dependen de la posición en la que la aplicación muestra los datos, y esta varía dependiendo del tamaño de la pantalla en la que se ejecute.