Tercera Práctica

Sergio Quijano Rey Fernando Valdés Navarro Carlos Corts Valdivia
Ignacio Carvajal Herrera

28.04.2022

Notas generales

Para especificar el diseño de los tests, usaremos la plantilla que hemos usado en el examen de teoría asíncrono. Plantilla que ha sido dada por los profesores de la asignatura.

Pruebas unitarias

1. `CarConfigurationRepository`
   - **Condiciones**: El repositorio funciona como se espera. Se realizan las siguientes operaciones con éxito: añadir configuraciones configuraciones, buscar configuraciones existentes, eliminar configuraciones, los identificadores dan el comportamiento esperado (no se añaden configuraciones que ya existen, no tiene efecto borrar dos veces la misma configuración)
   - **Datos requeridos**: Partimos de unos datos iniciales para realizar las comprobaciones. Se crean en `create_basic_repo()`
   - **Casos de prueba**:
     1. Comprobar almacenamiento
     2. Añadir configuraciones
     3. Buscar configuraciones
     4. Eliminar configuraciones
     5. Intentar eliminar una configuración dos veces
2. `ActiveConfigurationRepository`
   - **Condiciones**: el repositorio con la configuración actual funciona como se espera. Podemos cambiar la configuración activa en ese momento, el modelo, color, tapicería y extra. El algoritmo que calcula el precio funciona correctamente
   - **Datos requeridos**: el repositorio parte de una configuración activa por defecto. Para los modelos, colores, tapicerías y extras usamos unas listas prefijadas en el fichero `tests/ActiveConfiguration_test.dart`, para servir como opciones a usar
   - **Casos de prueba**:
     1. Activar una configuración en concreto
     2. Hacer `set` del modelo
     3. Hacer `set` del color
     4. Hacer `set` de la tapicería
     5. Hacer `set` del extra
     6. El cómputo del precio es correcto para un caso concreto
3. `Option`
   - **Condiciones**: la clase `Option`, que constituye una parte fundamental de nuestro modelo, funciona como se espera
   - **Datos requeridos**: por la simpleza de la clase, al principio de cada caso de test creamos

una opción sobre la que vamos a realizar las comprobaciones
- **Casos de prueba**:
  1. Los datos de la opción son correctos
  2. El cálculo del precio asociado a las opciones es correcto

Pruebas de widgets

1. Pruebas sobre la página principal
   - **Condiciones**: la página principal funciona correctamente.
   - **Datos requeridos**: reinicializamos el `DataController`, porque es un `singleton` y queremos que las pruebas corran sobre un estado *inicial* de la aplicación
   - **Casos de prueba**:
     1. Eliminar una configuración almacenada y comprobar que se ha borrado
     2. Al pulsar en configurar, se debe cambiar la vista a la configuración
2. Pruebas sobre la vista de compra
   - **Condiciones**: la página de compra funciona correctamente.
   - **Datos requeridos**: en este caso no hace falta reinicializamos el `DataController`. Usamos los datos que vienen por defecto
   - **Casos de prueba**:
     1. Creamos una nueva configuración
        - Navegamos hasta la vista de compra, desde la vista principal
        - Guardamos la configuración, con los datos que se asignan por defecto
        - Volvemos a la vista principal y comprobamos que esté la nueva configuración con los datos por defecto

Resultado de los tests

En la siguiente figura se puede observar cómo los tests de la aplicación pasan sin problemas:

```
sergio at asus-laptop in ~/GitRepos/DesarrolloSoftware at ᚠ[master]:
$ sudo flutter test test/*
[sudo] password for sergio:
   Woah! You appear to be trying to run flutter as root.
   We strongly recommend running the flutter tool without superuser privileges.
  /
□□
Running "flutter pub get" in DesarrolloSoftware ...                 3.9s
00:13 +16: All tests passed!
sergio at asus-laptop in ~/GitRepos/DesarrolloSoftware at ᚠ[master]:
$
```

Figure 1: Resultado de lanzar todos los tests, tanto unitarios como de *widgets*

Tests de integración

- Hemos creado un test de integración para comprobar que el proceso de creación y guardado de una configuración funciona correctamente
- Sin embargo, para correr estos tests hemos tenido problemas con el móvil (estos *tests* corren en un dispositivo físico, y nuestros móviles bloquean la instalación aunque tengamos las opciones para desarrolladores habilitadas y configuradas)

- En `integration_test` se puede consultar este test de integración, que es muy parecido al *test* de *widget* realizado sobre la vista de compra

Uso de git

- Igual que en la práctica anterior, hemos usado *github* como sistema de versión de controles
- Dicho repositorio se puede encontrar en este enlace
- Hemos usado características como creación de *issues* para asignación de tareas, uso de ramas y *pull requests* para desarrollo en paralelo