Práctica 3

Iván Gijón
Alejandro Ramos
Víctor Mojica
Gonzalo Alganza

4 de mayo de 2024

UNIVERSIDAD
DE GRANADA

Índice

1. Introducción

Durante esta práctica hemos realizado las pruebas sobre nuestra aplicación desarrollada en la anterior práctica.
Para lograr un buen diseño y análisis de las pruebas hemos optado por realizar previamente un Análisis de Requisitos (funcionales y no funcionales). Gracias a este análisis y al uso de las pruebas unitarias que nos ofrece Flutter, hemos encontrado cosas que no funcionaban bien y que entorpecían el funcionamiento de nuestra aplicación.
Estas pruebas únicamente del modelo,dejando las pruebas sobre la interfaz para una futura práctica.

2. Análisis de Requisitos

En primer lugar vamos a especificar los requisitos funcionales de nuestra aplicación.
Estos requisitos muestran que debe poder realizar nuestra aplicación

2.1. Requisitos Funcionales

- La aplicación permitirá al usuario crear una o varias personalizaciones de una Casa.
- El usuario puede crear 3 tipos de casa diferentes (Chalet,Casa de Campo o Apartamento)
- Se podrá añadir diferente número de Dormitorios a cada Casa.
- Cada casa tendrá siempre una Sala de Estar.
- Se podrá personalizar la cocina de cada casa, añadiendo un Lavavajillas, una Isla, o ambas opciones.
- Se podrá personalizar el baño de cada casa, añadiendo un Bidet, un Jacuzzi, o ambas opciones.
- Existirá un apartado para que el usuario pueda ver todas las casas que ha personalizado, a modo de historial.
- En todo momento se podrá cancelar una personalización de una Casa.

- Se debe poder navegar hacia atrás durante el proceso de creación de una Casa.

Estos son los requisitos no funcionales, estos requisitos especificarán como debe ser el rendimiento, la seguridad, la usabilidad de nuestra aplicación

2.2. Requisitos No Funcionales

- La aplicación funcionará fluidamente evitando largas esperas al usuario.
- La aplicación debe ser fácil de usar y comprender para los usuarios.
- La aplicación debe ser compatible con una variedad de dispositivos Android e iOS
- El código de la aplicación debe estar bien estructurado, usando patrones y un buen diseño.

3. Planificación de pruebas

En nuestra aplicación usamos el patrón builder para crear 3 tipos de casas distintas y 2 decoradores distintos. Dado que vamos a realizar 2 grupos para las pruebas, pensamos que lo mejor sería separarlo por los tipos de decoradores que tenemos, probando cada decorador y aspectos relacionados en cada uno de los grupos de test.
Estas pruebas las realiza el programador de la aplicación tras el proceso de desarrollo, es decir, una vez se haya programado la aplicación,dichas pruebas terminaran cuando se verifique que se cumplen los requisitos funcionales y que no haya errores en la aplicación.

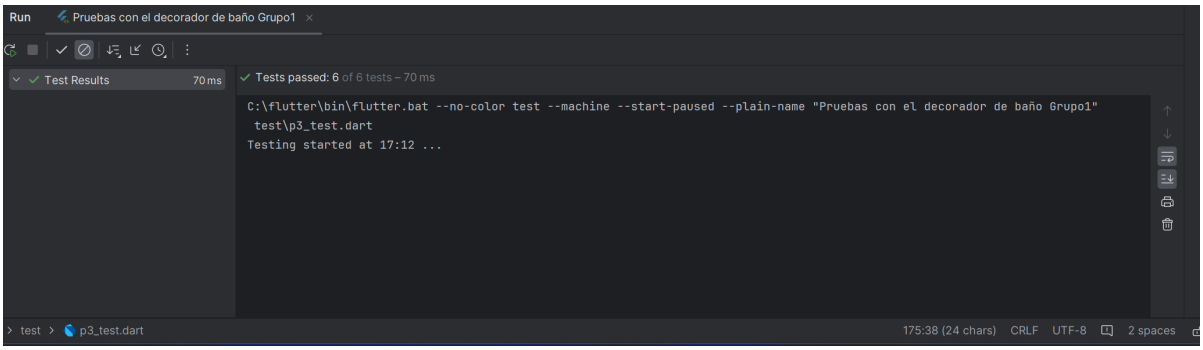
3.1. Grupo 1: Basado en el Decorador Baño

En este grupo de 6 pruebas se va a probar diferentes configuraciones del Decorador del Baño, con diferentes Builders y parámetros.
El objetivo principal de este grupo es evaluar el funcionamiento del Decorador de baño en la creación de distintos tipos de casa.
Las pruebas que se van a hacer en este grupo son las siguientes.

- Prueba de decorador de baño con un decorador de bidet, para comprobar si funciona el decorador de bidet.
- Prueba de creación de una casa de campo con un baño con un bidet, para comprobar si funciona el decorador de bidet en la creación de una casa de campo. Además de comprobar la correcta creación de una Casa de Campo.
- Prueba de creación de un apartamento con un baño con bidet y con Jacuzzi. Para probar el funcionamiento de los dos decoradores del baño y de la creación de un apartamento mediante el patrón builder.
- Creación de un Apartamento con un baño con 2 bidet y un jacuzzi.Para probar que se pueden añadir varios decoradores de bidet a un mismo apartamento.
- Comprobación de que una casa no se puede crear sin Baño, para comprobar el manejo de excepciones implementado y verificar que una casa no se puede crear si no le añades un baño.

- Comprobación de que todos los atributos están instanciados, método toString de Casa, para comprobar que todas las partes de la casa se han asignado correctamente y que se visualiza correctamente en el método toString de la casa.

La totalidad de las pruebas realizadas en este grupo se pueden ver con más detalle en la sección de Análisis de Pruebas.
Funcionamiento del los test del grupo 1:



3.2. Grupo 2: Basado en el Decorador de la Cocina y Builder

El objetivo principal de este grupo de 6 pruebas, ha sido comprobar el funcionamiento del Decorador de la Cocina, algunos Builders de casas y la clase globals.
Algunos casos de prueba que hemos realizado han sido los siguientes:

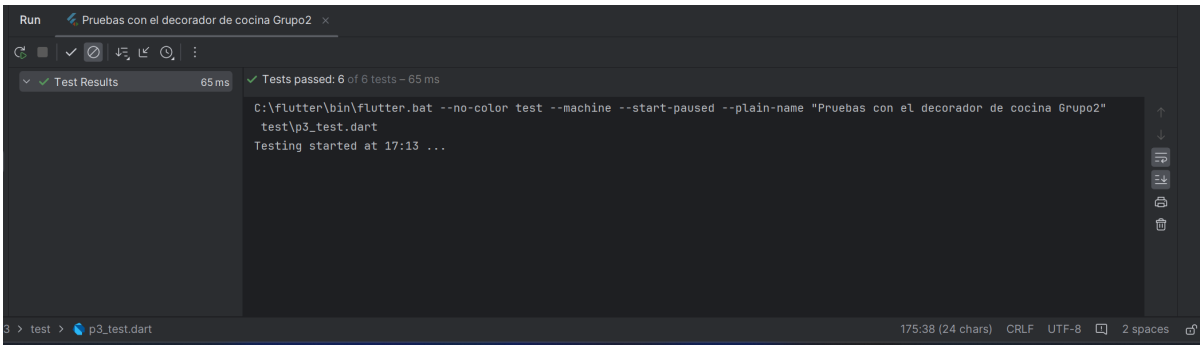
- Prueba de creación de un Chalet con decorador de isla y lavavajillas.Para comprobar el correcto funcionamiento de los decoradores de la Cocina y de la correcta creación de un Chalet.

- Prueba de decorador de cocina con dos islas.Para comprobar que se pueden añadir varios decoradores de Isla a una misma cocina.

- Prueba Singleton ListaCasas.Para comprobar que cuando se crea una nueva personalización de Casa se guarda en el array global de nuestra aplicación.

- Prueba de una cocina con una isla y un apartamento.Para comprobar la correcta creación de un apartamento y del decorador de Cocina con isla.
- Prueba de decorar una cocina de una casa de campo con tres lavavajillas. Para comprobar que se puede decorar una Cocina con varios decoradores de Lavavajillas.
- Prueba de añadirle a una cocina múltiples islas y lavavajillas.Para comprobar que se pueden añadir varios decoradores de Isla y lavavajillas a la vez.

La totalidad de las pruebas realizadas en este grupo se pueden ver con más detalle en la sección de Análisis de Pruebas. Funcionamiento test grupo2:



4. Análisis de pruebas

Elemento a probar	**Condiciones**	**Datos requeridos**
Clase decorador de baño bidet	Un baño con un decorador de bidet debería de devolver un baño con un único bidet	Baño Estándar
Clase Builder Casa Campo	El builder de casa de campo debería de devolver una casa de campo	Baño y Cocina
Clase Builder Chalet y Decorador de Cocina (isla y lavavajillas)	La cocina del chalet debe tener un decorador de isla y lavavajillas. La casa debe ser un Chalet	Casa con baño de Chalet, sala de Estar de Chalet y cocina con isla y lavavajillas.
Decorador de cocina (dos islas)	La cocina debe tener dos islas	Cocina
Clase Singleton ListaCasas	Al crear una casa Chalet y una casa de tipo Casa De Campo, la lista globals debe tener longitud 2, el primer elemento debe ser un Chalet y el segundo una Casa de Campo	Casa de Campo y Chalet
Clase Builder Apartamento y decorador cocina (isla)	La cocina debe tener una isla y la casa debe ser un apartamento	Cocina y Baño
Builder Casa Campo con decorador cocina (tres lavavajillas)	Devolver una casa de campo que tenga una cocina con tres lavavajillas	Cocina y un Baño
Decorador cocina (tres islas y dos lavavajillas)	Debe devolver una cocina con tres islas y dos lavavajillas	Cocina Estándar
Clase Casa método toString	Debe devolver una lista con todos los atributos de la casa	Casa (Chalet,Casa de Campo o un Apartamento)

Elemento a probar	**Condiciones**	**Datos requeridos**
Excepciones en los 3 Builders	Debe devolver una excepción si falta una parte de la casa cuando mientras crea la casa	Un builder (ChaletBuilder,Casa de Campo Builder, Apartamento Builder)
Clase Apartamento Builder con decorador de baño (2 bidets y 1 jacuzzi)	Devolverá una casa de tipo apartamento con un baño que contiene 2 bidets y 1 jacuzzi	Un Baño Estándar y una Cocina

Cuadro 1: Tabla de análisis de tests

5. Diseño de pruebas

Sobre el orden de prioridad de los métodos, consideramos que el test donde se muestra toda la información de la casa una vez creada es el de mayor prioridad ya que es sobre el objeto final, después, que no haya atributos nulos en la casa, con la comprobación de crear una casa sin baño. Hemos dado la misma prioridad a los test donde se prueban tanto los builder como los decoradores en un mismo test, ya que se están haciendo ambas comprobaciones, luego al test que comprueba el buen uso de la clase globals y por último, los tests que únicamente prueban los decoradores.

Casos de prueba (ordenados por prioridad)	**Entornos de prueba**	**Datos de prueba**	**Relación con las condiciones de prueba (trazabilidad)**
Comprobación de todos los atributos de la casa mediante el método toString de Casa	Tenemos un apartamento creado con todos sus atributos y se comprueba que dichos atributos son correctos mediante el método toString	Builder Apartamento con todos los atributos inicializados.	Comprobamos que el builder de apartamento funciona correctamente y que contiene todos los atributos que le hemos metido mediante el método toString de la casa.También se comprueba que el método toString funciona correctamente.
Comprobación de que una Casa no se puede crear sin Baño	Tenemos el builder de Casa de Campo vamos a comprobar que si no se agrega un Baño da una excepción	Una Casa de Campo sin Baño	Comprobamos que el builder de apartamento no funciona correctamente cuando no se agrega la cocina.

Casos de prueba (ordenados por prioridad)	**Entornos de prueba**	**Datos de prueba**	**Relación con las condiciones de prueba (trazabilidad)**
Creación de una casa de campo con un baño con un bidet	Mediante un builder de casa de campo crearemos una casa de campo usando un baño con un bidet creado anteriormente	Un baño con un decorador de bidet	Comprobamos que el builder de casa de campo crea una casa con un baño con un decorador de bidet correctamente.
Creación de un Apartamento y creación de una cocina con una isla	Creamos un apartamento y después le añadimos una cocina estandar que decoramos con una isla.	Un Apartamento y una cocina con un decorador de isla	Comprobamos que el builder de Apartamento funciona y comprobamos que el decorador de cocina funciona correctamente
Creación de una Casa de campo con una cocina que tenga tres lavavajillas	Creamos una casa de campo y después le añadimos una cocina estándar que decoramos con tres lavavajillas.	Una Casa de campo y una cocina con tres decoradores de lavavajillas	Comprobamos que el builder de casa de campo y que el método toString de la casa funcionen correctamente.También se comprueba que el decorador haga su función.

Casos de prueba (ordenados por prioridad)	**Entornos de prueba**	**Datos de prueba**	**Relación con las condiciones de prueba (trazabilidad)**
Creación de un Apartamento con un baño con bidet y jacuzzi	Creamos un apartamento con un baño con decorador de bidet y jacuzzi mediante un builder de apartamento	Un baño con un decorador de bidet	Comprobamos que el builder de apartamento funciona correctamente, además de comprobar que se le puede agregar varias veces un decorador a un baño.
Singleton ListaCasas	A partir de una casa de tipo Chalet y una Casa de Campo, las añadimos a la lista global casasCreadas	Referencia a la lista global de casas creadas con un Chalet y una Casa de Campo	Se deben verificar las condiciones de que se creen una casa Chalet y una Casa de Campo, y que la lista global de casas creadas tenga una longitud de 2. Además, que los tipos de las casas en la lista sean Chalet y Casa de Campo.
Creación de un Apartamento con un baño con 2 bidets y jacuzzi	Creamos un apartamento y añadiríamos a dicho apartamento un baño con dos bidets y un jacuzzi.	Un baño con dos decoradores de bidet y un decorador de jacuzzi y un Apartamento	Comprobamos que el builder de apartamento funciona correctamente, además de comprobar que se le puede agregar varios bidets al baño con el decorador.

Casos de prueba (ordenados por prioridad)	**Entornos de prueba**	**Datos de prueba**	**Relación con las condiciones de prueba (trazabilidad)**
Creación de un Chalet con decorador de cocina	A partir del builder del Chalet añadimos a la cocina un decorador de isla y otro de lavavajillas	Una Cocina con isla y lavavajillas	Se debe verificar que la cocina tenga un decorador de isla y un lavavajillas, y que la casa tenga un baño Chalet y una sala de estar Chalet, además de una lista de dormitorios vacía.
Creación de un baño con un decorador de bidet	A partir de un baño simple se le agrega un decorador para que tenga bidet	Un baño simple sin nada	Comprobamos que el decorador de bidet funciona correctamente y funciona como debería, agregando atributos extra a un baño.
Decorador de cocina con dos islas	A partir de una cocina estándar, añadimos dos decoradores de isla	Una Cocina con dos islas	Se debe verificar la condición de que la cocina tenga dos decoradores de isla.
Creación de una cocina con múltiples islas y dos lavavajillas	A partir de una casa creamos una cocina estandar que decoramos con tres islas y dos lavavajillas	Una cocina con varios decoradores, tres de isla y dos de lavavajillas	Comprobamos que los decoradores de cocina funcionan y que podemos entrelazarlos como queramos.

Cuadro 2: Tabla de diseño de pruebas

6. Informe de Bugs encontrados y solucionados gracias a los tests

Gracias a el desarrollo de tests hemos encontrado algunos bugs que no habíamos visto anteriormente. Estos son:

- El builder de Apartamento le ponía el tipo Apartamento a la cocina en el método de el baño, el cual se lo debería de haber puesto a el baño. Gracias a el test donde comprobábamos que el builder de Apartamento funcionara correctamente hemos conseguido solucionar el problema.

- Cuando se programó y probó el test para comprobar que si no se añade una cocina saca una excepción, nos dimos cuenta de que no devolvía ninguna excepción y de que la aplicación fallaba. De manera se interrumpía la experiencia para el usuario. Tras realizar una depuración del código y ver las posibles soluciones hemos decidido poner como nullable la variable Cocina del Builder y comprobar cada vez que se crea la casa que la Cocina no sea null, en caso de ser null lanzamos una excepción.
  De esta manera se puede puede controlar de manera eficiente y correcta la creación de la casa en casa de que falte una cocina.

  De la misma forma se ha cambiado para el baño, de manera que la aplicación lanza una excepción cuando no se añade un baño en la construcción de la casa.

  Aquí un ejemplo de como se ha resuelto: Se cambia el tipo de las variables del Builder para que puedan ser null añadiendo ?.

```
abstract class CasaBuilder{

  late Casa casa;
  Banio? banio;
  Cocina? cocina;
  late List<Dormitorio>dormitorios;
```

En la construcción de la casa cuando se añade la cocina se comprueba que no sea null.Y se lanza la excepción cuando si lo es.

```
void setCocina() {
  if(cocina==null){
    throw Exception('La cocina no está inicializada');
  }else{
    cocina?.tipo = "Casa de Campo";
    casa.cocina = cocina!;
  }
}
```

De la misma forma se hace para el baño, y estos cambios se realizan en los 3 builders de nuestra aplicación (ChaletBuilder, ApartamentoBuilder y CasaDeCampoBuilder).