Software Plus

Monica Alvarez
Juan Camilo Chafloque
Sebastián Gutiérrez
Laura Jimenez
Santiago Palacios
Sergio Posada

Smart Closet

Reporte del plan de pruebas

Ingeniería de Software

Pontificia Universidad Javeriana
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas
2020-1

1. Introducción

Se realizaron pruebas para cada caso de uso que tiene la aplicación conforme se iban creando y desarrollando dichos casos de uso.

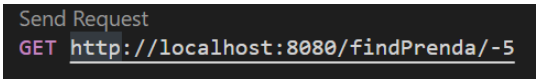
- Pruebas unitarias (Componentes pequeños, Ej. Agregar/Eliminar favorito en una prenda)
- Pruebas de integración (Relación entre varios componentes y entidades, Ej. Crear un atuendo).

Se verificó que los datos quedarán almacenados correctamente y de la forma esperada en la base de datos de H2. Además, se tenía como regla grupal entre los desarrolladores que cada vez que se encontraban errores, no se seguía desarrollando nuevos casos de uso hasta tener el caso de uso actual, funcionando correctamente.

Para el proceso de pruebas se utilizó un servicio REST que ofrece Spring Boot llamado MockMVC, que se utiliza para realizar pruebas unitarias y pruebas de integración. Este servicio esta basado en los protocolos HTTP (Get, Put, Post, Delete) y funciona mediante *requests* que hace Spring a la base de datos, simulando las peticiones que haría un usuario desde la aplicación. A continuación, se mostrarán dos ejemplos; uno para las pruebas unitarias y otro para las pruebas de integración.

2. Ejemplo de prueba unitaria de Smart Closet

Caso de ejemplo: Se tiene una prenda con un id determinado. Se verifica inicialmente los atributos de dicha prenda para ver en qué estado esta su campo de *favorito,* y luego cambiar el estado.



Como primer paso, se realizaba una petición GET al servidor, para obtener una prenda cualquiera dado un id. Para el ejemplo anterior se utilizó la prenda con el id 5, que inicialmente se encontraba marcada como favorita.

Como segundo paso, se necesitaba saber si el caso de uso y los métodos implementados en verdad estaban funcionando de la manera correcta, por lo que se utilizaba el protocolo HTTP con su debida url que se encontraba asociada al método a probar, que en este caso era el método de modificar favorito.

De la imagen anterior se puede ver como se implementa el caso de prueba y el llamado simulado a la base de datos con el request PUT, para hacer el llamado a la función asociada. Se puede ver que el servidor retorna la prenda que se modificó, que en este caso es la prenda con el id 5, y se puede ver que el campo de favorito se cambió correctamente.

Run | Run Selected | Auto complete | Clear | SQL statement:

```
select * from prenda where id_prenda = -5
```

```
select * from prenda where id_prenda = -5;
```

ID_PRENDA	ABRIGO	COLOR	DESCRIPCION	DISPONIBLE	FAVORITO	FORI
-5	2	Amarillo	Camisa amarilla manga corta	TRUE	FALSE	3

(1 row, 8 ms)

Cabe aclarar que paralelamente con el plan de pruebas también se hacia uso de la base de datos H2 para verificar que en verdad los datos se estaban guardando de manera correcta y la persistencia era cumplía con la propiedad de Durabilidad.

Por último, también se tenía un proceso transversal con la parte del front-end y la aplicación como tal. Cada vez que se realizaban pruebas, además de verificar la persistencia y el manejo de los datos en el servidor y en la base de datos, también se tenia un control de pruebas en la parte de diseño de la aplicación, ya que algunos de los casos de uso, tenían un efecto directo con los cambios que realizaba el sistema en la aplicación y que se le iban a mostrar al usuario final.

Por ejemplo, siguiendo con el ejemplo anterior, se verificaba que, al hacer un cambio en el estado de favorito de alguna prenda, se verificaba que, en el menú de prendas en la aplicación, el botón de favoritos cambiara conjuntamente con el cambio en la

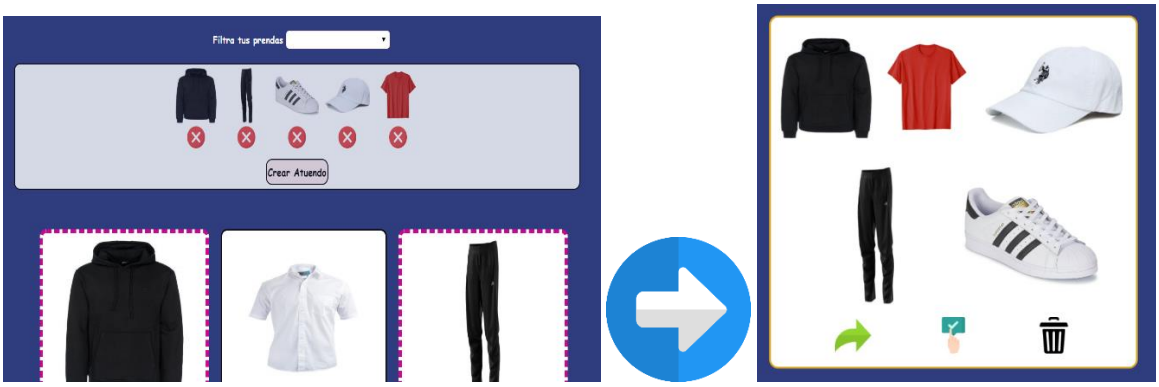
base de datos, y que además la prenda quedará visualizada en el menú de favoritos de la aplicación o que fuera eliminada del menú, según fuera el caso.

Este proceso se realizó para todas las pruebas unitarias de todos los componentes que se tenían en la aplicación, como lo fueron: Habilitar/Inhabilitar prendas, eliminar usuarios, eliminar prendas, eliminar atuendos, ver listas de prendas según su sección, ver lista de favoritos, editar una prenda o usuario, etc.

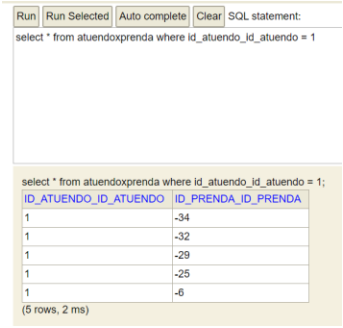
3. Ejemplo de prueba de integración de Smart Closet

Para el ejemplo de prueba de integración se mostrará el funcionamiento y todo el proceso de pruebas para el caso de uso de crear atuendo, ya que este caso de uso estaba relacionado con otros dos, los cuales eran el componente de historial de atuendo y el componente de calendario.

Lo primero que se hacia a la hora de crear un atuendo era verificar que las prendas mostradas en las diferentes visualizaciones de la aplicación eran en verdad las prendas que el usuario escogió para la creación de su nuevo atuendo.



Así como en las pruebas unitarias, para las pruebas de integración se tenia un mayor control y seguimiento de la base de datos, verificando que cualquier creación o cualquier modificación de algún componente se estuviera guardando de la manera correcta.



Para mostrar la parte de la integración, se utilizó el botón de “poner” que tenía cada atuendo de la aplicación. Este botón tenia como funcionamiento principal, simular la prenda que el usuario se iba a poner para el día, por lo que, al presionar dicho botón, se iban a alterar los componentes de historial y calendario.

4. Resultados y Conclusiones

Como conclusiones principales del plan de pruebas que realizó Software Plus en el desarrollo de Smart Closet fueron las siguientes:

- Determinar cuáles son los errores
- Saber cómo arreglarlos
- Tenerlos presente en el futuro, ya que en el transcurso del desarrollo se pueden volver a presentar.
- Aprender de los errores y evitar que se repitan
- Garantizar una mejor calidad en el producto final

Como resultados se mostrarán a continuación los casos de uso con más errores y fallos y que necesitaron muchas pruebas:

- **Generar atuendo**: Este fue el caso de uso en el que mas se necesitaron pruebas, ya que, en las primeras versiones del algoritmo de recomendación, los resultados mostrados al usuario no eran buenos, ya que inicialmente no se mostraban todas las prendas o algunas veces se generaban atuendos sin alguna de las secciones de prendas obligatorias. En otros casos, las prendas generadas no cumplían con los estándares de colores que se habían consultado al inicio del proyecto con el experto, por lo que se tuvieron que realizar muchas pruebas en el algoritmo para terminar con una generación de atuendos de calidad.

- **Componente del clima y agregar prenda**: Para estos dos casos de uso, se necesitaron realizar varias pruebas en sus primeras versiones, debido principalmente a que estos dos casos de uso, eran los que estaban relacionados con las API que iba a utilizar Smart Closet. El clima con Open Weather Map y la creación de prendas con Google Image Search. La razón principal del porque se necesitaron realizar mas pruebas de lo usual para estos dos componentes se debe a la curva de aprendizaje que se requería para manejar las APIs correctamente, en especial la de Google Image Search.

- **Visualización de atuendos**: Para este caso de uso se necesitaron realizar diversas pruebas, en especial para la parte grafica y de visualización. Debido a que un atuendo era un conjunto de prendas, y cada prenda tenia asociada una imagen, al principio se le dificulto al equipo de desarrolladores como mostrar dicha colección de atuendos al usuario desde la aplicación, ya que

se no se tenía un número fijo de prendas por atuendo (Algunos podían tener tres prendas, otros cuatro, otros cinco y hasta seis prendas), por lo que el desarrollo del front-end para la visualización de los diferentes atuendos que tenia el usuario, fue un proceso demorado que necesitó de muchas pruebas.