SHOES COMPANY

Ficha: 2499719

Autores

Ángel Galindo

Daniel Rodríguez

Nicolas Escobar

Ricardo Arias

Santiago García

Sergio Rojas

CENTRO DE LA INDUSTRIA, LA EMPRESA Y LOS SERVICIOS TECNÓLOGO DE ANÁLISIS Y DESARROLLO DE SOFTWARE – HUILA 2022

SHOES COMPANY

Ficha: 2499719

Ángel Galindo

Daniel Rodríguez

Nicolas Escobar

Ricardo Arias

Santiago García

Sergio Rojas

DOCENTE: Carlos Julio Cadena Sarasty

CENTRO DE LA INDUSTRIA, LA EMPRESA Y LOS SERVICIOS TECNÓLOGO DE ANÁLISIS Y DESARROLLO DE SOFTWARE– HUILA 2022

INTRODUCCIÓN

El presente proyecto presentando por la empresa SC, propone una solución para la problemática que presentan a los pequeños emprendedores y locales de la ciudad de Neiva, brindando un aplicativo web, tanto en computadores como en sistemas Android. Este documento constara del sistema de testeo, resultados y conclusiones halladas al realizar diversos tanteos al aplicativo, a la web y a la base de datos, buscando encontrar, posibles errores, soluciones y dejar sistema funcionando.

PRUEBAS DE SOFTWARE

Existen diversos tipos de pruebas que se pueden realizar a un software, para poder definir cuáles son las más adecuadas toca saber en qué consisten cada una de estas y cuales llegan a ser más útiles, según las necesidades de cada proyecto. Algunos ejemplos de pruebas son los siguientes.

**1. Pruebas unitarias**

Las pruebas unitarias son de muy bajo nivel y se realizan cerca de la fuente de la aplicación. Consisten en probar métodos y funciones individuales de las clases, componentes o módulos que usa tu software. En general, las pruebas unitarias son bastante baratas de automatizar y se pueden ejecutar rápidamente mediante un servidor de integración continua.

**2. Pruebas de integración**

Las pruebas de integración verifican que los distintos módulos o servicios utilizados por tu aplicación funcionan bien en conjunto. Por ejemplo, se puede probar la interacción con la base de datos o asegurarse de que los microservicios funcionan bien en conjunto y según lo esperado. Estos tipos de pruebas son más costosos de ejecutar, ya que requieren que varias partes de la aplicación estén en marcha.

**3. Pruebas funcionales**

Las pruebas funcionales se centran en los requisitos empresariales de una aplicación. Solo verifican el resultado de una acción y no comprueban los estados intermedios del sistema al realizar dicha acción.

A veces, se confunden las pruebas de integración con las funcionales, ya que ambas requieren que varios componentes interactúen entre sí. La diferencia es que una prueba de integración puede simplemente verificar que puedes hacer consultas en la base de datos, mientras que una prueba funcional esperaría obtener un valor específico desde la base de datos, según dicten los requisitos del producto.

**4. Pruebas de extremo a extremo**

Las pruebas integrales replican el comportamiento de un usuario con el software en un entorno de aplicación completo. Además, verifican que diversos flujos de usuario funcionen según lo previsto, y pueden ser tan sencillos como cargar una página web o iniciar sesión, o mucho más complejos, como la verificación de notificaciones de correo electrónico, pagos en línea, etc.

Las pruebas integrales son muy útiles, pero son costosas de llevar a cabo y pueden resultar difíciles de mantener cuando están automatizadas. Se recomienda tener algunas pruebas integrales clave y depender más de pruebas de menor nivel (unitarias y de integración) para poder detectar rápidamente nuevos cambios.

**5. Pruebas de aceptación**

Las pruebas de aceptación son pruebas formales que verifican si un sistema satisface los requisitos empresariales. Requieren que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios. Sin embargo, también pueden ir más allá y medir el rendimiento del sistema y rechazar cambios si no se han cumplido determinados objetivos.

**6. Pruebas de rendimiento**

Las pruebas de rendimiento evalúan el rendimiento de un sistema con una carga de trabajo determinada. Ayudan a medir la fiabilidad, la velocidad, la escalabilidad y la capacidad de respuesta de una aplicación. Por ejemplo, una prueba de rendimiento puede analizar los tiempos de respuesta al ejecutar un gran número de solicitudes, o cómo se comporta el sistema con una cantidad significativa de datos. Puede determinar si una aplicación cumple con los requisitos de rendimiento, localizar cuellos de botella, medir la estabilidad durante los picos de tráfico y mucho más.

**7. Pruebas de humo**

Las pruebas de humo son pruebas básicas que sirven para comprobar el funcionamiento básico de la aplicación. Están concebidas para ejecutarse rápidamente, y su objetivo es ofrecerte la seguridad de que las principales funciones de tu sistema funcionan según lo previsto.

Las pruebas de humo pueden resultar útiles justo después de realizar una compilación nueva para decidir si se pueden ejecutar o no pruebas más caras, o inmediatamente después de una implementación para asegurarse de que la aplicación funciona correctamente en el entorno que se acaba de implementar.

OPCIONES ESCOGIDAS

Como opción principal para el testeo del aplicativo como grupo y empresa se optó por escoger, la primera opción, es decir, pruebas unitarias, esto con el fin de probar, verificar y testear, cada uno de los módulos implementados, pues, cada uno de estos se fue realizando por etapas de desarrollo, es decir, hasta no completar un módulo, no fueron realizado los siguientes.

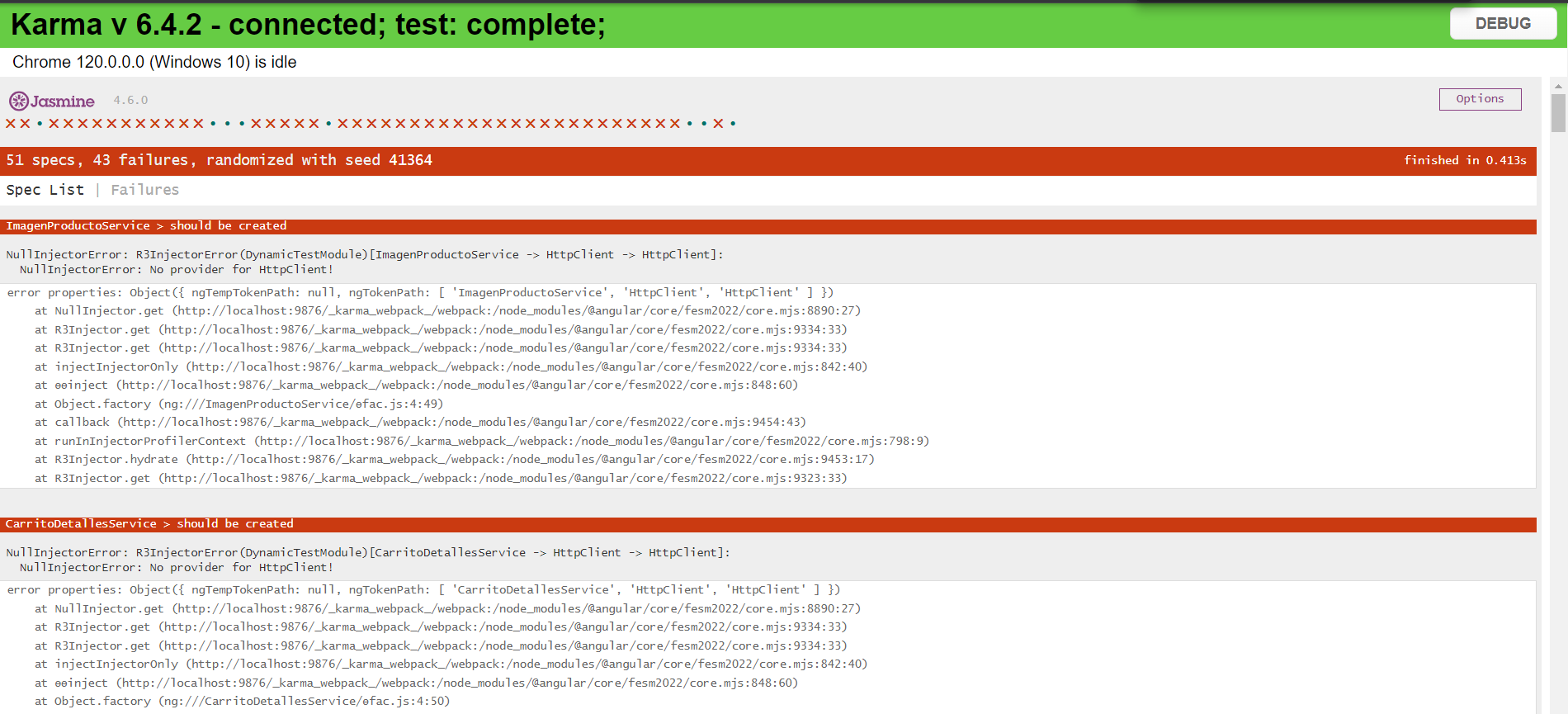
Una vez los módulos estuvieron listos, se empezó a aplicar pruebas integrales y pruebas funcionales, con el fin de verificar y revisar que los módulos trabajen correctamente unos con otros, además de confirmar, que la información y su funcionalidad funciones correctamente.

Estas pruebas se pudieron realizar gracias a la ayuda de las siguientes herramientas: Angular y Django; enfocadas en testear frontend y backend.

REALIZACIÓN DE PRUEBAS

Utilizamos “ng test” en angular para realizar las pruebas, y teníamos un error en todos los archivos, porque en el archivo de *spec* utilizado para realizarse las pruebas, faltaba colocar:

La importación del HttpClientModule.



CONCLUSIONES

Angular permite realizar pruebas unitarias con el archivo *spec* de cada componente, realizamos la corrección de la importación dicha, y todo salió correcto.