Universidad de Sevilla

Ingeniería Informática de Software

D05 - Advanced Topics

R04 - Testing Report

FECHA	VERSIÓN
31/05/2022	V1.3

REPOSITORIO: https://github.com/marinaramirofde/Acme-Toolkits.git

GRUPO DE PRÁCTICAS	E6.07
AUTORES	**ROLES**
Marina Ramiro Fernández marramfer12@alum.us.es	Manager, Developer, Tester and Operator
Ángel Lorenzo Casas anglorcas@alum.us.es	Developer, Tester

TABLA DE CONTENIDO

1.- Resumen ejecutivo

En este documento se documentará las distintas formas de testear y la manera en la que está organizado el repositorio a la hora del contenido de src/test/resource y src/test/java.

2.- Tabla de revisiones

Versión	Fecha	Descripción
V1.0	24/05/2022	Creación del documento.
V1.1	25/05/2022	Resumen del contenido
V1.2	28/05/2022	Ampliación del contenido
V1.3	31/05/2022	Revisión y finalización

3.- Introducción

-El apartado 4.1 - Testing en nuestro proyecto

Habla sobre la forma en la que el testing ha sido realizado en nuestro proyecto además del tipo de testing que hemos implementado específicamente.

-El apartado 4.2- Tipos de testing

Habla sobre los diferentes Test de Software que se realizan a lo largo del desarrollo y sus características, entre los diferentes tipos de Test que se abordan en este apartado se encuentran el Unit Testing, el Integration Testing, el System Testing también conocido como End To End testing, el Acceptance Testing y el Regression Testing cada uno de los cuales se diferencia de los otros por el rango de lo que testea o el objetivo de la realización de dicho Testing.

Además este apartado provee una ayuda visual para entender y diferenciar los distintos Testing.

-El apartado 4.3 - Conceptos de Testing

Donde se abordan varios conceptos sobre el Testing entre los cuales se encuentran el Testing formal e informal, además de ejemplos de algunas de las distintas pruebas que se pueden realizar en dichos Testing, también se comentan los diferentes tipos de pruebas que se han empleado a lo largo de la carrera y sus peculiaridades.

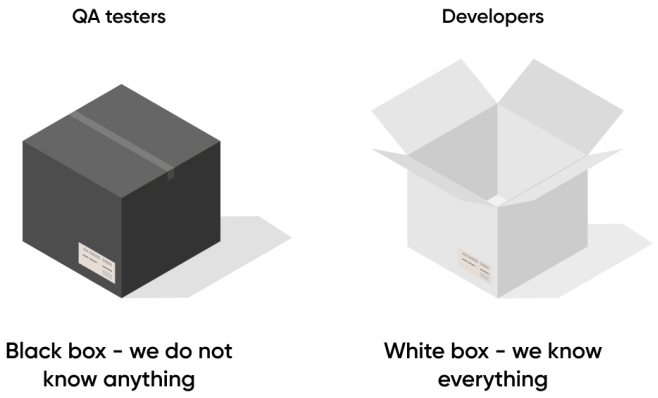
4.- Testing de un WIS

4.1 Testing en nuestro proyecto

Las pruebas que realizamos en nuestro proyecto son las denominadas E2E (extremo a extremo), las cuales prueban el software simulando la interacción que realizaría un usuario sobre la interfaz web. Se hacen pruebas positivas y negativas. En las positivas verificamos que el sistema funciona correctamente y en las negativas comprobamos que salten los errores tal y como esperábamos.

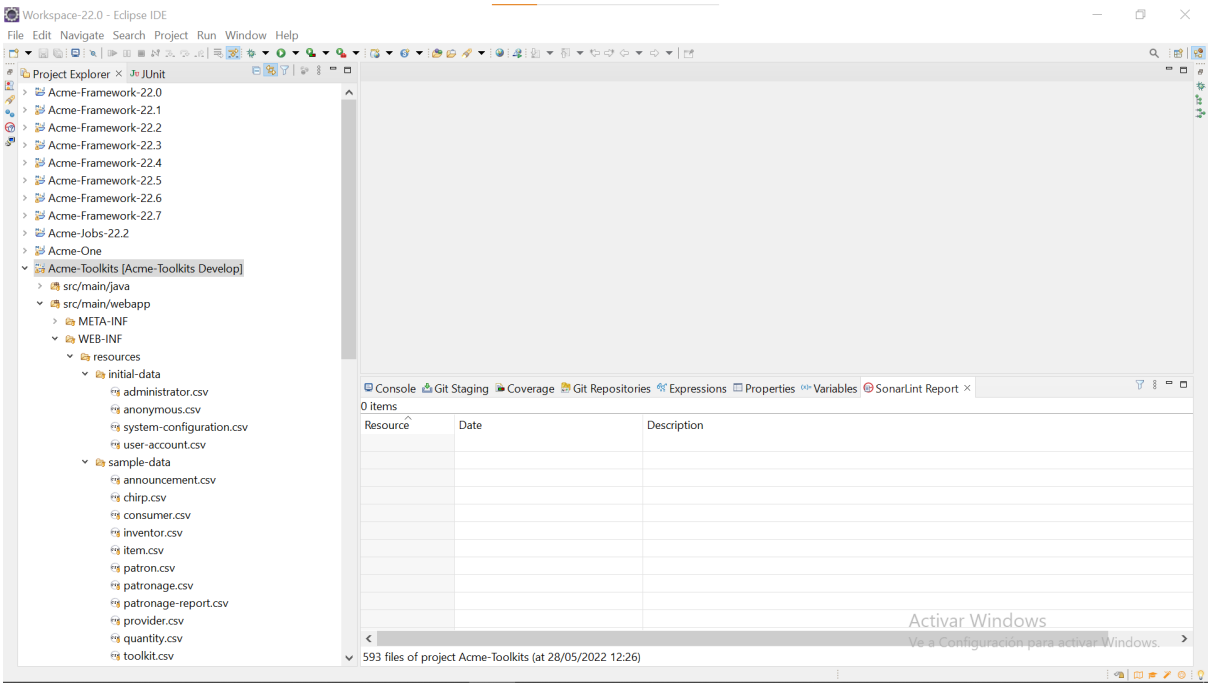
Las pruebas E2E pueden ser de muchos tipos: estáticas o dinámicas, funcionales o no funcionales, de caja blanca o negra, unitarias o integradas. Sin embargo, para las pruebas estáticas utilizamos SonarLint, que las hace automáticamente.

Black Box White Box testing:



Y para corroborar una imagen de lo que muestra el SonarLint al analizar nuestro proyecto y realiza las pruebas estáticas:

Imagen de sonarLint:



Tenemos varias carpetas implicadas en el testeo. La primera es “src/test/java” donde se encuentran cada una de las pruebas que realizaremos por cada funcionalidad del proyecto y están organizadas de la forma src/test/java/acme.testing/rol/entidad/metodo donde dentro de cada uno de los paquetes se pueden encontrar tests de distintos procedimientos del rol/entidad como Publish, Create, ListMine o List, Delete y Update de los cuales solo el ListMine o List posee casos solo positivos mientras que el Update, Delete, Create y Publish testean también casos negativos en los que la aplicación no tendría que funcionar y debe devolver error , en el caso de Create tambien se testea el hacking donde se debe comprobar que se devuelve un panic en los otros casos con tests positivos y negativos solo se teoriza debido a la incapacidad de poder realizarlos con las herramientas a mano en esos casos.

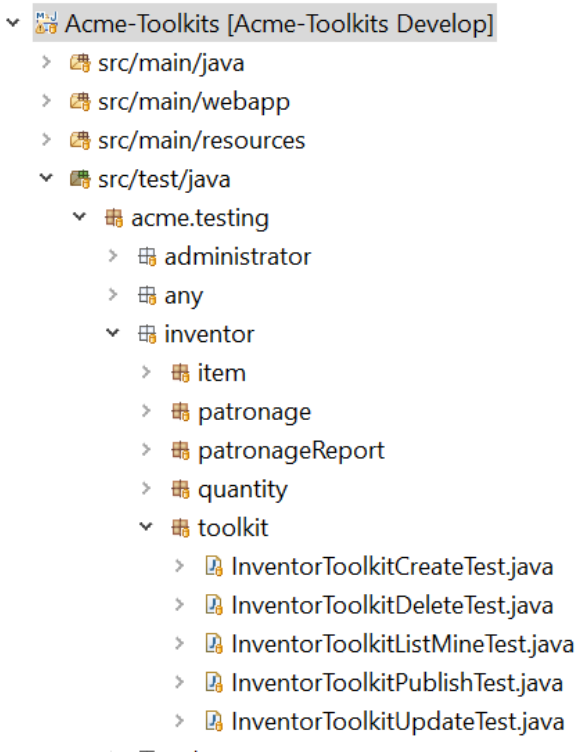


Imagen que muestra la organización en src/test/java en nuestro proyecto.

Por otro lado, la carpeta "src/test/resources" nos proporciona los datos con los que se trabajan en los test. Se especifica cierta información que completará los distintos formularios, entre otras cosas, y dependerá de si la prueba es positiva o negativa. Para las positivas se introducen datos que no constan de errores y en las negativas se trata cada uno de los distintos errores que pueden ocurrir. Aquí entran en juego también las reglas de negocio y otras restricciones que vienen implícitas.

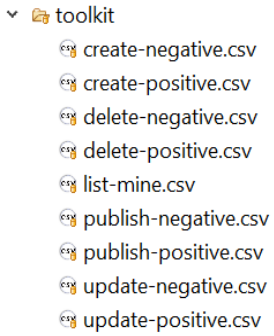


Imagen que muestra organización de los .csv en los src/test/resource

Además de estar en nuestro proyecto, los recursos .csv con los que se realizan los tests situados como ya sabemos en el src/test/resources. Están organizados de la forma src/test/resources/rol/entidad/nombre.csv el nombre.csv viene determinado por el método en el que se usa para testear (ej: create, update, delete). Si se usa para testear casos

positivos (en los cuales la aplicación debería funcionar correctamente) o casos negativos (en los cuales la aplicación debería dar error) (ej: x-positive, x-negative) de forma que el nombre.csv tiene el formato metodo-respuesta Test. Con ello podemos, como ya comentamos previamente, hacer el método un publish, update etc.. y la respuesta Test positive o negative. En el caso de que solo haya tests con respuestas positivas la respuesta Test no se incluye en el nombre como es el caso en los listMine o list.

4.2-Tipos de Testing.

Hay diferentes tipos de testing cuando queremos realizar tests, entre ellos se encuentran el:

-**Unit Testing**: Donde se testean unidades del software de forma aislada para observar su comportamiento en una situación en la que no interactúa con otras unidades, o lo hace lo mínimo posible.

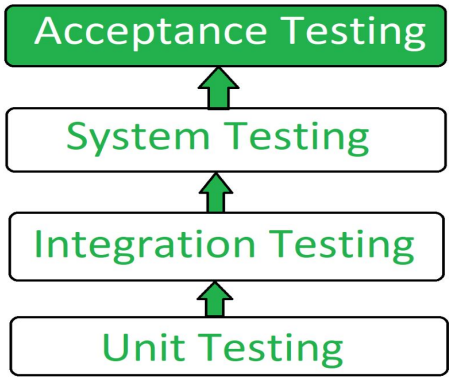
-**Integration Testing**: Donde se testean diferentes unidades de forma simultánea y las interacciones entre ellas en un entorno integrado, y que no está compuesto por todas las diferentes unidades o componentes.

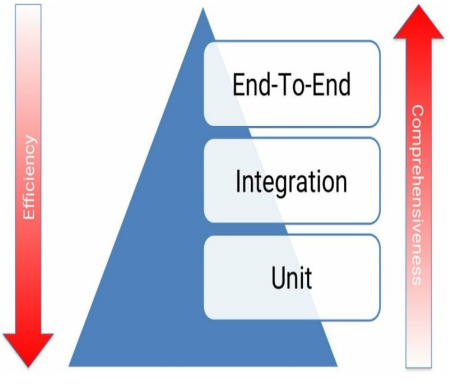
-**System Testing o END TO END TESTING**: Donde se testea todo el programa y sus diferentes componentes y unidades de forma simultánea, con el objetivo de observar que el software funciona correctamente o en el caso de que no funcione comprender donde están los fallos.

-**Acceptance Testing**: Donde se realiza una revisión con el objetivo de verificar que el sistema de software producido logra las especificaciones y realiza su cometido.

-**Regression Testing**: Donde se busca comprobar que una aplicación funciona correctamente tras cualquier cambio en el código, este tipo de testing se encarga de la estabilidad del código.

-Los distintos tipos de testing tienen una jerarquía como se puede apreciar en las siguientes imágenes:





4.3 - Conceptos de testing .

Los conceptos de testing de un sistema de información web que conocemos con anterioridad son:

- **Testing informal**: tests realizados por el desarrollador en los cuales comprueba manualmente el funcionamiento de la aplicación. En ellos se suelen probar el contenido de los formularios y el comportamiento de la aplicación tras el intento de acciones maliciosas que afecten su comportamiento.
- **Testing formal:** tests realizados por el desarrollador en un entorno controlado de pruebas en los cuales se comprueba si el comportamiento de la aplicación es el esperado tras un conjunto de acciones que forman el caso de prueba.

    - **Dentro del testing informal, conocemos distintas acciones a probar:**

- Comprobar los valores frontera en los formularios: no es necesario probar la inmensa mayoría de valores posibles, sino sólo aquellos que tienen mayor probabilidad de producir un error (valores nulos o vacíos, valores que excedan la capacidad del campo a testear, etc).

- Comprobar errores relacionados con la sesión del navegador: probar la accesibilidad de la aplicación desde el modo incógnito.

- Comprobar la protección frente a scripts: por ejemplo, scripts SQL que acceden a la base de datos o scripts que modifican la apariencia de la web.

    - **Dentro del testing formal, conocemos distintos tipos de pruebas:**

- Pruebas unitarias.
- Pruebas E2E.

Resumidamente, definimos que existen diferentes tipos de pruebas que se pueden realizar para testear una aplicación web. Algunos de estos tests que hemos aprendido a implementar durante el transcurso de la carrera son:

- **Pruebas de navegación**. Se implementan con herramientas como Gecko Driver o Selenium y consisten en simular navegaciones reales en un navegador web.

- **Pruebas de carga**. Se implementan con herramientas como Locust con el objetivo final de comprobar cómo se enfrenta la aplicación a grandes cargas de trabajo, es decir, numerosas peticiones HTTP de diferentes usuarios.

5.- Conclusión

Al realizar el testing de forma correcta podemos saber cuando algún componente/entidad o acción de nuestra aplicación no funciona como es debido y por tanto evitar los mayores bugs posibles a la hora de publicar la aplicación permitiéndonos tener una aplicación más libre de bugs y errores.

En este trabajo en concreto pasó que cuando el framework cambio a la versión 22.6 empezaron a surgir errores en los tests lo que nos alertó de que al haberse realizado algún cambio en el framework algunas acciones y entidades ya no funcionaban como deberían y por tanto debíamos arreglarlas.

6.- Bibliografía

Intencionalmente en blanca