DP II - 2022-2023

TESTING - INDIVIDUAL - REPORT

Repositorio: ***https://github.com/marolmmar1/C2.X02.git***

Tutor: RAFAEL CORCHUELO GIL

Alumno: MARCOS OLMEDO MARíN

Grupo - C2.X02

1 Índice

2 Resumen Ejecutivo

A continuación la finalidad del presente documento es la de recopilar y documentar el análisis de todos los tests realizados para el "Student #2" los cuáles se han realizado durante el Sprint final (Sprint 4) proporcionando resultados de que las *features* realizadas cumplen con la expectativas y no presentan ningún bugs que implique a nuestro cliente no estar satisfecho con nuestro trabajo.

Además en el siguiente documento se ha analizado el rendimiento de su ejecución de la aplicación comparándolo con diferentes dispositivos (siendo PC1 y PC2).

3 Tabla de Versionado

Versión	Fecha	Descripción
1.0	09/07/2023	Primera versión del documento

4 Introducción

En la siguiente sección se procederá a explicar cómo se han realizado las pruebas funcionales del requisito solicitado así como realizar la comparativa entre el mejor y peor dispositivo con los reportes aportados: ***tester-request-performance.csv*** y ***tester-test-performance.csv***

5 Contenido

5.1 About Testing

Pruebas Funcionales > List Tutorial> Test Positivo

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/list-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int tutorialRecordIndex, final String code, final String title, final String estimatedTime) {

    super.signIn("assistant4", "assistant4");

    super.clickOnMenu("Assistant", "Tutorial List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.checkColumnHasValue(tutorialRecordIndex, 0, code);
    super.checkColumnHasValue(tutorialRecordIndex, 1, title);
    super.checkColumnHasValue(tutorialRecordIndex, 2, estimatedTime);

    super.signOut();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión como ***assistant4***. Una vez iniciada sesión, accederemos al menú de asistente y haremos click en “Tutorial List”, con lo que se nos mostrará la lista de forma ascendente de los tutoriales de este asistente.

Tras ordenar el listado, se irán comprobando los valores presentes en la aplicación con los esperados del csv.

Si todo se realiza de forma correcta cerrará sesión y el test habrá concluido con éxito.

Este tipo de prueba nos permite asegurar que los *tutoriales* asociados a ese usuario son correctos y que no aparecen otros pertenecientes a otros usuarios.

Pruebas Funcionales > List Tutorial >Test Hacking

```
.

@Test
public void test300Hacking() {

    super.checkLinkExists("Sign in");
    super.request("/assistant/tutorial/list-all");
    super.checkPanicExists();

    super.signIn("administrator", "administrator");
    super.request("/assistant/tutorial/list-all");
    super.checkPanicExists();
    super.signOut();

    super.signIn("lecturer1", "lecturer1");
    super.request("/assistant/tutorial/list-all");
    super.checkPanicExists();
    super.signOut();

    super.signIn("student1", "student1");
    super.request("/assistant/tutorial/list-all");
    super.checkPanicExists();
    super.signOut();

    super.signIn("company1", "company1");
    super.request("/assistant/tutorial/list-all");
    super.checkPanicExists();
    super.signOut();

    super.signIn("auditor1", "auditor1");
    super.request("/assistant/tutorial/list-all");
    super.checkPanicExists();
    super.signOut();

}
```

En este test, lo que hacemos es ir probando que ninguno de los demás roles, exceptuando el nuestro de *assistant*, puedan acceder a los listado de los *tutorials* y en ese caso, de intentar acceder, se devuelve un *error 500* indicando que no estamos autorizados.

Pruebas Funcionales > Show Tutorial> Show Positive

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/show-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int tutorialIndex, final String code, final String title, final String abstracts, final String goals, final String course, final String

    super.signIn("assistant4", "assistant4");

    super.clickOnMenu("Assistant", "Tutorial List");
    super.sortListing(0, "asc");
    super.clickOnListingRecord(tutorialIndex);
    super.checkFormExists();

    super.checkInputBoxHasValue("code", code);
    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    super.checkInputBoxHasValue("goals", goals);
    super.checkInputBoxHasValue("course", course);
    super.checkInputBoxHasValue("estimatedTime", estimatedTime);

    super.signOut();
}
```

Tras iniciar sesión con *assistant4*, accedemos al listado de tutoriales. Iteramos uno por uno comprobando que los datos existentes son coherentes con los datos esperados.

.

PRUEBAS FUNCIONALES > TUTORIAL> SHOW TUTORIAL> TEST HACKING

```
@Test
public void test300Hacking() {

    Collection<Tutorial> tutorials;
    String param;

    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant3");
    for (final Tutorial tutorial : tutorials)
        if (tutorial.isDraftMode()) {
            param = String.format("id=%d", tutorial.getId());

            super.checkLinkExists("Sign in");
            super.request("/assistant/tutorial/show", param);
            super.checkPanicExists();

            super.signIn("administrator", "administrator");
            super.request("/assistant/tutorial/show", param);
            super.checkPanicExists();
            super.signOut();

            super.signIn("assistant5", "assistant5");
            super.request("/assistant/tutorial/show", param);
            super.checkPanicExists();
            super.signOut();

            super.signIn("lecturer1", "lecturer1");
            super.request("/assistant/tutorial/show", param);
            super.checkPanicExists();
            super.signOut();

            super.signIn("student1", "student1");
            super.request("/assistant/tutorial/show", param);
            super.checkPanicExists();
            super.signOut();

            super.signIn("company1", "company1");
            super.request("/assistant/tutorial/show", param);
            super.checkPanicExists();
            super.signOut();

            super.signIn("auditor1", "auditor1");
            super.request("/assistant/tutorial/show", param);
            super.checkPanicExists();
            super.signOut();
        }
```

Tras listar el conjunto de tutoriales del asisitente 3, intentamos acceder aquellos tutoriales que siguen estando en modo borrador con otros perfiles.

Esto se hace de esta manera para reutilizar la función en tests como update o publish, para asegurarnos que la prohibición surge de la autorización del perfil que está intentando acceder a la url y no porque el tutorial no sea editable.

PRUEBAS FUNCIONALES > ASSISTANT> CREATE TUTORIAL> TEST POSITIVE

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/create-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int tutorialIndex, final String title, final String abstracts, final String goals, final String course, final String estimatedTime) {

    super.signIn("assistant6", "assistant6");

    super.clickOnMenu("Assistant", "Tutorial List");
    super.checkListingExists();

    super.clickOnButton("Create");
    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    super.fillInputBoxIn("goals", goals);
    super.fillInputBoxIn("course", course);
    super.clickOnSubmit("Create");

    super.clickOnMenu("Assistant", "Tutorial List");
    super.checkListingExists();
    super.sortListing(0, "asc");
    super.checkColumnHasValue(tutorialIndex, 1, title);
    super.checkColumnHasValue(tutorialIndex, 2, estimatedTime);

    super.clickOnListingRecord(tutorialIndex);
    super.checkFormExists();
    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    super.checkInputBoxHasValue("goals", goals);
    super.checkInputBoxHasValue("course", course);
    super.checkInputBoxHasValue("estimatedTime", estimatedTime);

    super.clickOnButton("Tutorial Sessions");
    super.checkListingExists();
    super.checkListingEmpty();

    super.signOut();
}
```

Este test se hace desde el perfil de assistant6. Tras acceder al formulario de creación y enviarlo, comprobamos que en la vista previa del listado de todos los tutoriales los datos coinciden. Después, hacemos click en esa misma entrada para analizarla y comprobar que todos los datos introducidos son coherntes con los que ha guardado el sistema.

También comprobamos que el nuevo tutorial posee el listado de sesiones correspondiente.

El ejecutar los test de esta manera nos permite detectar que los valores metidos en el formulario cumple con las reglas de negocio impuestas y no existen errores, y por lo tanto nos permite crear el formulario correctamente.

Además se comprueba que los datos metidos en cada campo, al listar y mostrar en el show, son los que hemos metido.

Pruebas Funcionales > Assistant> Create tutorial> Test Negative

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/create-negative.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test200Negative(final int tutorialIndex, final String title, final String abstracts, final String goals, final String course, final String estimatedTime) {

    super.signIn("assistant6", "assistant6");

    super.clickOnMenu("Assistant", "Tutorial List");
    super.clickOnButton("Create");
    super.checkFormExists();

    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    super.fillInputBoxIn("goals", goals);
    super.fillInputBoxIn("course", course);
    super.clickOnSubmit("Create");
    super.checkErrorsExist();

    super.signOut();
}
```

Este test es más corto que su contraparte positiva, quedándonos en el paso del envío de formulario. Esto se debe a que lo que estamos probando es que no sea posible introducir datos erróneos en el sistema.

Al hacer esto comprobamos que al introducir valores que no cumplen con las reglas de negocio, salta un error y así podemos comprobar que no hay error en la implementación de los mismos.

Pruebas Funcionales > Assistant> Update Tutorial> Test Positive

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/update-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int tutorialIndex, final String title, final String abstracts, final String goals, final String course, final String estimatedTime) {

    super.signIn("assistant18", "assistant18");

    super.clickOnMenu("Assistant", "Tutorial List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(tutorialIndex);
    super.checkFormExists();
    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    super.fillInputBoxIn("goals", goals);
    super.fillInputBoxIn("course", course);
    super.clickOnSubmit("Update");

    super.checkListingExists();
    super.sortListing(0, "asc");
    super.checkColumnHasValue(tutorialIndex, 1, title);
    super.checkColumnHasValue(tutorialIndex, 2, estimatedTime);

    super.clickOnListingRecord(tutorialIndex);
    super.checkFormExists();
    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    super.checkInputBoxHasValue("goals", goals);
    super.checkInputBoxHasValue("course", course);
    super.checkInputBoxHasValue("estimatedTime", estimatedTime);

    super.signOut();
}
```

Similar al test de show, este test accede a una cuenta de asistente e inspecciona diferentes tutoriales del mismo. A diferencia del Show, se sobreesciben los datos del formulario y se envía, pulsando en el botón update.

Una vez actualizamos, comprobamos que los cambios se han reflejado en la vista previa del listado, y también inspeccionamos en detalle el tutorial, comprobando que todos los cambios se han guardado con éxito.

El ejecutar los test de esta manera nos permite detectar que los valores metidos en el formulario cumple con las reglas de negocio impuestas y no saltan errores y permite actualiazar el formulario correctamente.

Pruebas Funcionales > Assistant> Update Tutorial> Test Negative

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/update-negative.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test200Negative(final int tutorialIndex, final String title, final String abstracts, final String goals, final String course, final String estimatedTi

    super.signIn("assistant18", "assistant18");

    super.clickOnMenu("Assistant", "Tutorial List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(tutorialIndex);
    super.checkFormExists();
    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    super.fillInputBoxIn("goals", goals);
    super.fillInputBoxIn("course", course);
    super.clickOnSubmit("Update");

    super.checkErrorsExist();

    super.signOut();
}
```

De forma similar al test negativo de create, seguimos los pasos iniciales de la contraparte positiva, pero asegurándonos de que no nos deja implementar en el sistema datos erróneos.

Por lo tanto, sólo comprobamos que existen errores una vez sobrescribimos los datos en el formulario y pulsamos el botón “Update”.

El ejecutar los test de esta manera nos permite detectar que los valores metidos en el formulario no cumple con las reglas de negocio impuestas, saltan excepciones y no permite actualizar el formulario correctamente.

Pruebas Funcionales > Assistant> Tutorial Session

Todos los test de las sesiones de un tutorial son de la misma naturaleza que los de los tutoriales, a excepción de publish, dado que sólo tutorial posee draftMode.

Para más información, leer el reporte de análisis del entregable 3

5.2 Performance Requests Reports

A continuación se han generado los reportes de los **performance request** y **performance testing** con el mejor dispositivo y el peor dispositivo del equipo para tener un estudio aproximado de cómo funciona el software en el mejor y en el peor de los casos; ambos obtuvieron muy buenos tiempos. Es por ello que en este documento hemos reflejado los gráficos de uno de ellos para mostrar estos resultados.

➔ Gráficos

Para la realización de este apartado se ha agrupado en una hoja de excel los tiempos recogidos en los reports performance requests y se han agrupado por su simple-path para calcular el promedio de tiempo invertido en estos mismos.

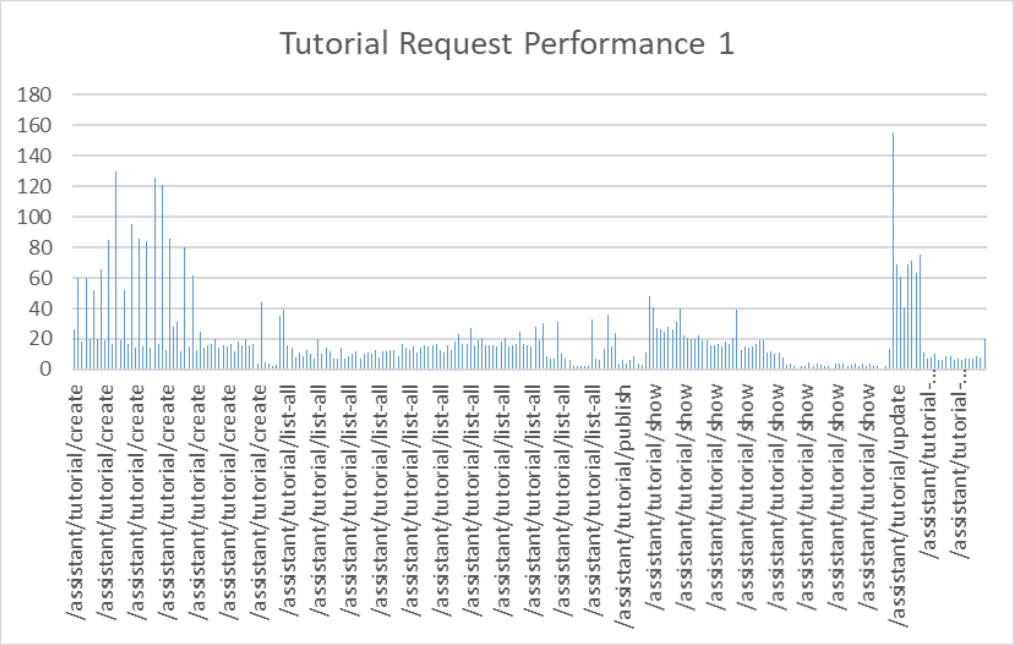

Gráfico del promedio del tiempo de las solicitudes - PC 1

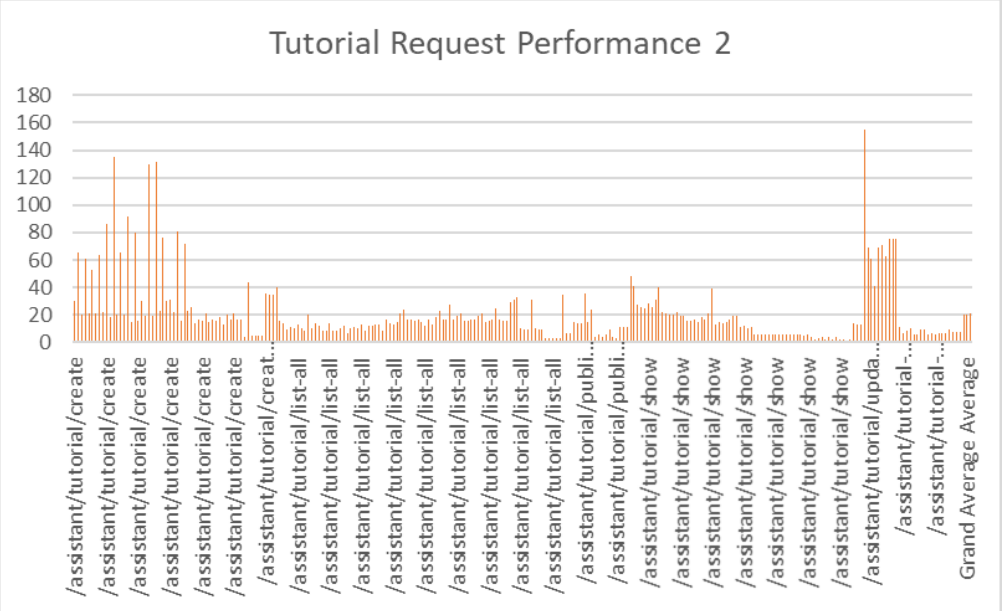

*Gráfico del promedio del tiempo de las solicitudes - **PC 2***

➔ Análisis de datos

Para la realización de este apartado se ha llevado a cabo un análisis estadístico de los tiempos obtenidos en los reportes, haciendo una comparativa de los resultados obtenidos entre dos dispositivos del grupo, para posteriormente analizar los tiempos en cada dispositivo.

TUTORIAL ESTADÍSTICAS	
Mean	20,03822197
Standard Error	1,490736721
Median	14
Mode	2
Standard Deviation	23,09439398
Sample Variance	533,3510335
Kurtosis	10,24060535
Skewness	2,941049205
Range	154
Minimum	1
Maximum	155
Sum	4809,173274
Count	240
Confidence Level(95,0%)	2,936661011

Interval (ms) =	***17,10156096***	***22,97488298***
Interval (s) =	***0,017101561***	***0,022974883***

TUTORIAL ESTADÍSTICAS 2	
Mean	21,00367134
Standard Error	1,43574229
Median	15
Mode	6
Standard Deviation	22,88196679
Sample Variance	523,5844042
Kurtosis	10,7446477
Skewness	2,977412508
Range	154
Minimum	1
Maximum	155
Sum	5334,932521
Count	254
Confidence Level(95,0%)	2,827529048

Interval (ms):	***18,17614229***	***23,83120039***
Interval (s):	***0,018176142***	***0,0238312***

5.3 Performance Tests Reports

A continuación se han generado los reportes de los **performance request** y **performance testing** con el mejor dispositivo y el peor dispositivo del equipo para tener un estudio aproximado de cómo funciona el software en el mejor y en el peor de los casos; ambos obtuvieron muy buenos tiempos. Es por ello que en este documento hemos reflejado los gráficos de uno de ellos para mostrar estos resultados.

➔ Gráficos

Para la realización de este apartado se ha agrupado en una hoja de excel los tiempos recogidos en los reports performance tests y se han agrupado por su test-class y por su test-method después, pudiendo así calcular el promedio de tiempo invertido en cada método de cada clase.

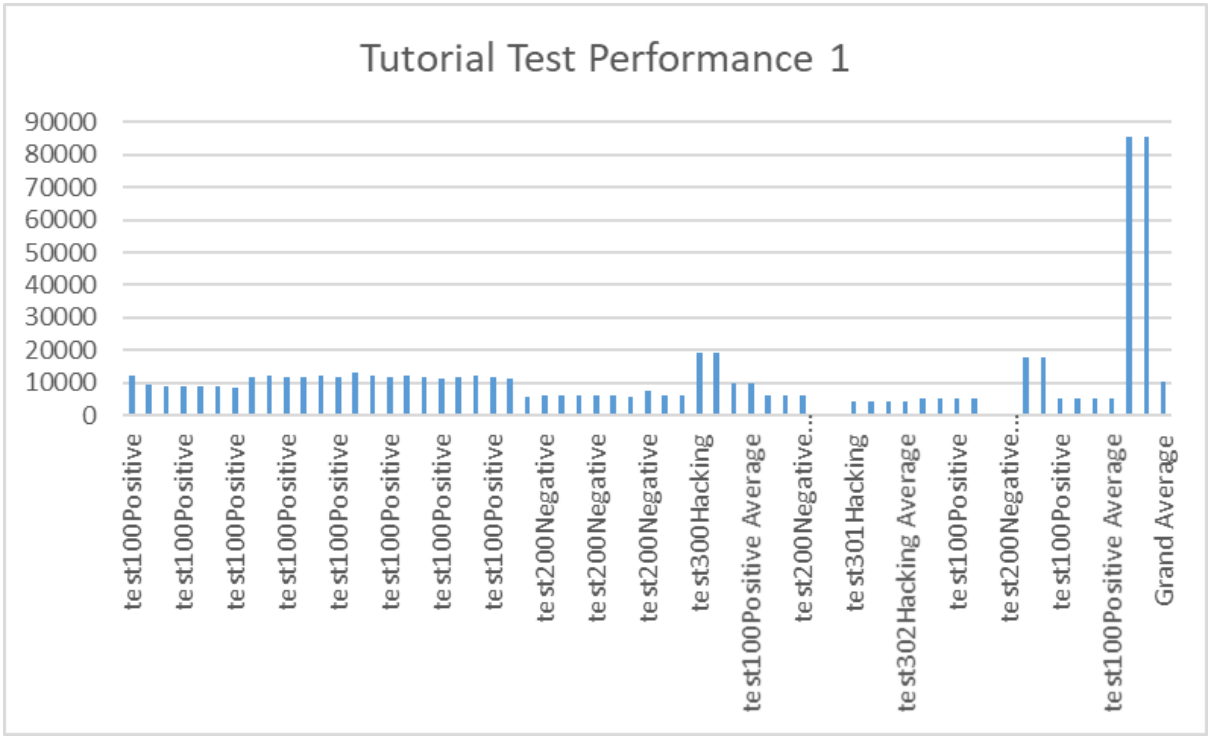

Gráfico del tiempo promedio de ejecución de los tests- PC1

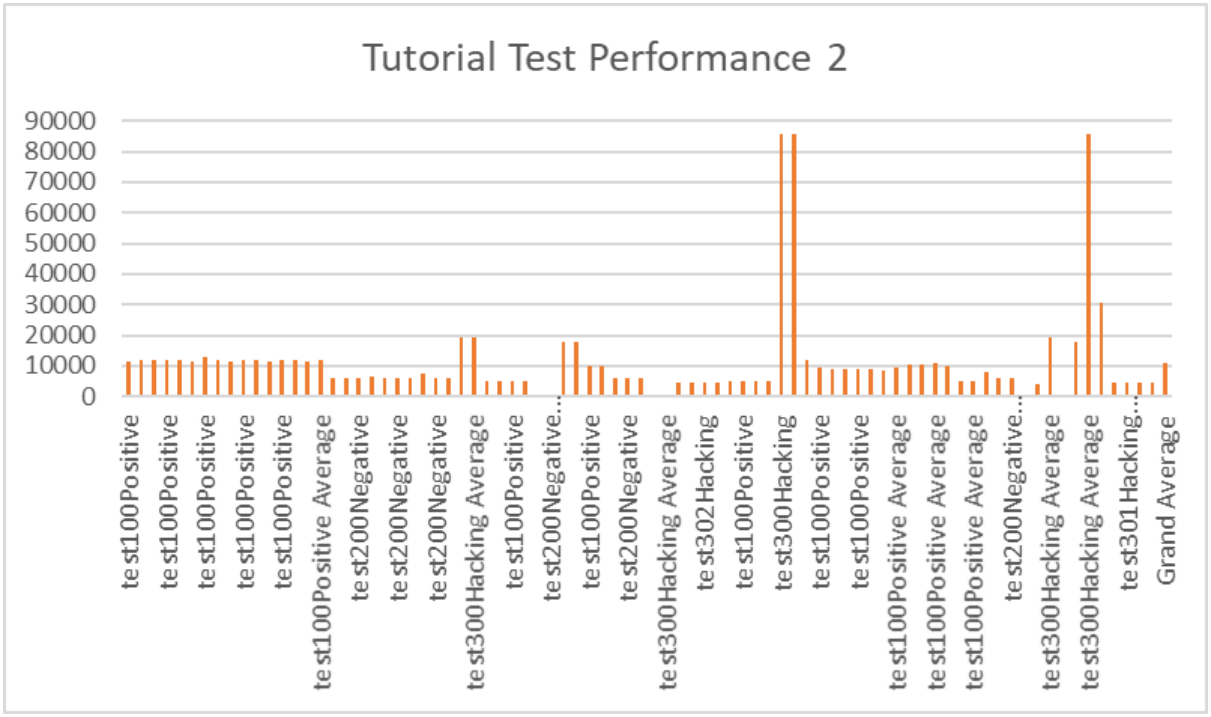

Gráfico del tiempo promedio de ejecución de los tests PC2

6 Lessons Learnt About Testing

A continuación se enumeran las lecciones aprendidas durante este proyecto en grupo acerca del "testeo" llevado a cabo durante este proyecto. Podemos destacar las siguientes lecciones aprendidas como ventajas del testing:

- Permite una organización de una forma más óptima los tests.
- Mejora la gestión del alcance de los test, aportando una mejor optimización en cuanto al tiempo.
- Podemos ver que sin duda el testeo influye muy notable y positivamente respecto a la calidad de código, puesto que podemos detectar errores en una etapa más temprana de desarrollo y de forma más rápida.
- Observando también cómo favorece al trabajo generalizado, pues permite trabajar de manera más ágil.
- Sobre todo cara a los costes de mantenimiento del proyecto, esta es una buena manera de sobrellevarlos.

7 Conclusiones

Intencionalmente en blanco.

8 Bibliografía

Intencionalmente en blanco.