DP II - 2022-2023

TEST REPORT

https://github.com/pedlopruz/Acme-L3-D04

Miembros:

- Carolina Carrasco Díaz
- Pedro López Ruz
- Ángel Lorenzo Casas
- Manuel Navarro Sicre
- Manuel Ortiz Blanco

Tutor: RAFAEL CORCHUELO GIL

GRUPO - C1.04.12

1 Índice

2 Resumen Ejecutivo 2
3 Tabla de Versionado 2
4 Introducción 2
5 Contenido 3
6 Conclusión 13
7 Bibliografía 13

2 Resumen Ejecutivo

A continuación la finalidad del presente documento será la de tener documentado por escrito como se ha realizado la tarea de testing, explicando las conclusiones obtenidas y analizando el rendimiento al aplicar los mismos.

3 Tabla de Versionado

Versión	Fecha	Descripción
1.0	20/03/2023	Primera versión del documento
1.1	21/05/2023	Rellenar el documento

4 Introducción

La intención de este documento es la de recopilar y documentar el análisis de todos los tests realizados para el estudiante 3, los cuáles se han realizado en el Sprint 4 proporcionando resultados de que las features realizadas cumplen con la expectativas y no tienen bugs que provocan que nuestros clientes no están contento con nuestro trabajo. Además analizaremos cómo ha sido el rendimiento de su ejecución de la aplicación en varios Pcs.

5 CONTENIDO

Pruebas Funcionales

- **Listar Tutorial**

  - **Test Positivo**

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

    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 con un assistant, esto hará que nos redirija al menú de assitant, se nos mostrará la lista de forma ascendente de tutorials asociados a ese assistant (comprobando que se muestra) y se comprobará que los valores de las columnas coincide con los que le hemos metido como parámetro. Si todo se realiza de forma correcta cerrará sesión y el test estará correcto.

El ejecutar los test de esta manera nos permite detectar que los tutorials asociados a ese assistant son correctos y que no aparecen otros los cuales pertenecen a otros assistant.

```
@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 assistant, no pueden acceder a los listado de los tutorials y que devuelve un error 500 indicando que no estamos autorizados.

Show Tutorial

Test Positive

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

    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();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión con un assitant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de tutorials de forma ascendente pulsamos en cada tutorials asociados a ese assistant (comprobando que se muestra) y se comprobará que los valores de cada campo coincide con los que le hemos metido como parámetro. Si todo se realiza de forma correcta cerrará sesión y el test estará correcto.

El ejecutar los test de esta manera nos permite detectar que los tutorials asociados a ese assistant son correctos y que no aparecen otros, los cuales pertenecen a otros assistant y que los valores que se van guardando en cada campo son correctos.

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();
```

En este test, lo que hacemos es ir probando que ninguno de los demás roles, exceptuando el assistant asociado a esos tutorials , no pueden acceder a los formularios de cada tutorial y que devuelve un error 500 indicando que no estamos autorizados que es lo que estamos buscando.

Create Tutorial

- **Test Positive**

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/create-positive.csv", encoding = "
public void test100Positive(final int tutorialIndex, final String code, final Stri

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

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

    super.clickOnButton("Create");
    super.fillInputBoxIn("code", code);
    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, 0, code);
    super.checkColumnHasValue(tutorialIndex, 1, title);
    super.checkColumnHasValue(tutorialIndex, 2, estimatedTime);


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

     super.signOut();
}
```

En este test, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de tutorials de forma ascendente (comprobando que se muestra), tras esto al pulsar el botón de create saldrá el formulario de creación, se añadirá valores positivos en cada campo y tras esto se pulsará el botón de create.

Una vez que se ha creado correctamente comprobamos que existe la lista de tutorial asociado a ese assistant, comprobamos que la lista muestra correctamente los datos del tutorial que hemos creado

y que al pulsar sobre él y redirigir al formulario de show se comprueba que los datos de cada campo sean los que hemos añadidos.

Por último se comprueba que hay un botón denominado Tutorial Session y que al pulsar sobre él da lugar a una lista de tutorial session y que esta está vacía.

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 crear el formulario correctamente.

Además se comprueba que los datos metidos en cada campo luego al listar y mostrar en el show son los que hemos metido y por último que se genera un botón que deriva a los tutorial sessions de ese tutorial.

Test Negative

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/create-negative.csv", encoding = "utf-8", numLinesTo
public void test200Negative(final int tutorialIndex, final String code, final String title, final St

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

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

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

    super.signOut();
}
```

En este test, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, pulsamos el botón de create y comprobamos que sale el formulario de creación, se añadirá valores negativos en uno de los campo por cada iteración para así comprobar tras pulsar el botón create que en ese campo muestra un mensaje de error.

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.

Test Negative

```
@Test
public void test300Hacking() {

    super.checkLinkExists("Sign in");
    super.request("/assistant/tutorial/create");
    super.checkPanicExists();

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

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

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

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

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

En este test, lo que hacemos es ir probando que ninguno de los demás roles, exceptuando assistant, no pueden crear tutorial y que devuelve un error 500 indicando que no estamos autorizados que es lo que estamos buscando.

Update Tutorial

- ***Test Positive***

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/update-positive.csv", encoding = "utf-8", r
public void test100Positive(final int tutorialIndex, final String code, final String title,

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

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

    super.clickOnListingRecord(tutorialIndex);
    super.checkFormExists();
    super.fillInputBoxIn("code", code);
    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, 0, code);
    super.checkColumnHasValue(tutorialIndex, 1, title);
    super.checkColumnHasValue(tutorialIndex, 2, estimatedTime);

    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();
}
```

En este test, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de tutorials de forma ascendente (comprobando que se muestra), pulsamos en cada tutorial asociados a ese assistant tras esto aparecerá el formulario con todos los datos(se comprueba que existe), se actualizará con valores positivos en cada campo y tras esto se pulsará el botón de update.

Una vez que se ha creado correctamente comprobamos que existe la lista de tutorial asociado a ese assistant, comprobamos que la lista muestra correctamente los datos del tutorial que hemos actualizado y que al pulsar sobre él y redirigir al formulario de show se comprueba que los datos de cada campo sean los que hemos añadidos.

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 crear el formulario correctamente.

Test Negative

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/update-negative.csv", encoding = "utf-8", numLinesTo
public void test200Negative(final int tutorialIndex, final String code, final String title, final St

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

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

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

    super.checkErrorsExist();

    super.signOut();
}
```

En este test, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de tutorials de forma ascendente (comprobando que se muestra), pulsamos en cada tutorial asociados a ese assistant tras esto aparecerá el formulario con todos los datos(se comprueba que existe), se actualizará con valores negativos los cuales no cumplen las reglas de negocio en cada campo y tras esto se pulsará el botón de 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 crear el formulario correctamente.

Test Hacking

```
@Test
public void test300Hacking() {

    Collection<Tutorial> tutorials;
    String param;

    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant1");
    for (final Tutorial tutorial : tutorials) {
        param = String.format("id=%d", tutorial.getId());

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

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

        super.signIn("lecturer2", "lecturer2");
        super.request("/assistant/tutorial/update", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("student2", "student2");
        super.request("/assistant/tutorial/update", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("assistant2", "assistant2");
        super.request("/employer/job/update", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("company2", "company2");
        super.request("/assistant/tutorial/update", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("auditor2", "auditor2");
        super.request("/assistant/tutorial/update", param);
        super.checkPanicExists();
        super.signOut();
    }
}
```

En este test, lo que hacemos es ir probando que ninguno de los demás roles, exceptuando el assistant que es el nuestro, no pueden actualizar un tutorial y que devuelve un error 500 indicando que no estamos autorizados, que es lo que estamos buscando.

Publish Tutorial

<u>Test Positive</u>

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/publish-positive.csv", encoding = "utf-8", numLinesT
public void test100Positive(final int tutorialIndex, final String code) {

    super.signIn("assistant10", "assistant10");

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

    super.clickOnListingRecord(tutorialIndex);
    super.checkFormExists();
    super.clickOnSubmit("Publish");
    super.checkNotErrorsExist();

    super.signOut();
}
```

En este test, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de tutorials de forma ascendente (comprobando que se muestra), pulsamos en cada tutorial asociados a ese assistant tras esto aparecerá el formulario con todos los datos(se comprueba que existe), tras esto se pulsará el botón de publish y observaremos que no produce errores.

El ejecutar los test de esta manera nos permite detectar que se publica tutorial los cuales tienen un tutorial session asociado.

Test Negative

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial/publish-negative.csv", encoding = "utf-8", numLinesT
public void test200Negative(final int tutorialIndex, final String code) {

    super.signIn("assistant5", "assistant5");

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

    super.checkColumnHasValue(tutorialIndex, 0, code);
    super.clickOnListingRecord(tutorialIndex);
    super.checkFormExists();
    super.clickOnSubmit("Publish");
    super.checkAlertExists(false);

    super.signOut();
}
```

En este test, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de tutorials de forma ascendente (comprobando que se muestra), pulsamos en cada tutorial asociados a ese assistant tras esto aparecerá el formulario con todos los datos(se comprueba que existe), tras esto se pulsará el botón de publish y observaremos que produce error.

El ejecutar los test de esta manera nos permite detectar que no se publica tutorials los cuales tienen un tutorial session asociado.

Test Hacking

```
    @Test
    public void test300Hacking() {

        Collection<Tutorial> tutorials;
        String params;

        tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant10");
        for (final Tutorial tutorial : tutorials)
            if (tutorial.isDraftMode()) {
                params = String.format("id=%d", tutorial.getId());

                super.checkLinkExists("Sign in");
                super.request("assistant/tutorial/publish", params);
                super.checkPanicExists();

                super.signIn("administrator", "administrator");
                super.request("assistant/tutorial/publish", params);
                super.checkPanicExists();
                super.signOut();

                super.signIn("lecturer1", "lecturer1");
                super.request("assistant/tutorial/publish", params);
                super.checkPanicExists();
                super.signOut();

                super.signIn("student1", "student1");
                super.request("assistant/tutorial/publish", params);
                super.checkPanicExists();
                super.signOut();


              super.signIn("student1", "student1");
              super.request("assistant/tutorial/publish", params);
              super.checkPanicExists();
              super.signOut();


              super.signIn("company1", "company1");
              super.request("assistant/tutorial/publish", params);
              super.checkPanicExists();
              super.signOut();

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

En este test, lo que hacemos es ir probando que ninguno de los demás roles, exceptuando el assistant que es el nuestro, no pueden publicar un tutorial y que devuelve un error 500 indicando que no estamos autorizados, que es lo que estamos buscando.

Test Hacking

```
@Test
public void test301Hacking() {

    Collection<Tutorial> tutorials;
    String params;

    super.signIn("assistant1", "assistant1");
    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant1");
    for (final Tutorial tutorial : tutorials)
        if (!tutorial.isDraftMode()) {
            params = String.format("id=%d", tutorial.getId());
            super.request("/assistant/tutorial/publish", params);
        }
    super.signOut();
}
```

En este test, lo que hacemos es comprobar que no pueden publicar un tutorial que ya ha sido publicado previamente y que devuelve un error 500 indicando que no estamos autorizados, que es lo que estamos buscando.

Test Hacking

```
@Test
public void test302Hacking() {

    Collection<Tutorial> tutorials;
    String params;

    super.signIn("assistant1", "assistant1");
    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant2");
    for (final Tutorial tutorial : tutorials) {
        params = String.format("id=%d", tutorial.getId());
        super.request("/assistant/tutorial/publish", params);

    }
    super.signOut();
}
```

En este test, lo que hacemos es comprobar que no pueden publicar un tutorial que pertenece a otro assistant y que devuelve un error 500 indicando que no estamos autorizados, que es lo que estamos buscando.

Listar Tutorial Session

- ***Test Positivo***

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial-session/list-positive.csv", encoding = "utf-8", numL
public void test100Positive(final int tutorialRecordIndex, final String code, final int tutorialSess

    super.signIn("assistant1", "assistant1");

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

    super.checkColumnHasValue(tutorialRecordIndex, 0, code);
    super.clickOnListingRecord(tutorialRecordIndex);
    super.checkInputBoxHasValue("code", code);
    super.clickOnButton("Tutorial Sessions");

    super.checkListingExists();
    super.checkColumnHasValue(tutorialSessionRecordIndex, 0, title);
    super.clickOnListingRecord(tutorialSessionRecordIndex);

    super.signOut();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de forma ascendente de tutorials asociados a ese assistant (comprobando que se muestra) y se comprobará que el valor de las columna “code” coincide con los que le hemos metido como parámetro para cada tutorial. Tras esto, se accede al formulario de cada tutorial y se comprueba que el valor de “code” sigue coincidiendo con el que le pasamos como parámetro. Comprobado esto se pulsará el botón Tutorial Session para mostrar la lista de tutorial sessions asociados al tutorial(se comprueba que aparece) y por último se comprueba que el atributo “título” coincide con el pasado como parámetro. Si todo se realiza de forma correcta cerrará sesión y el test estará correcto.

El ejecutar los test de esta manera nos permite detectar que los tutorials asociados a ese assistant son correctos y que no aparecen otros los cuales pertenecen a otros assistant, además que los tutorial sessions asociados a ese tutorial aparezca correctamente y no los de otro tutorial y sobre todo que se muestran las listas con los datos correctos.

Test Hacking

```
@Test
public void test300Hacking() {
    Collection<Tutorial> tutorials;
    String param;

    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant1");
    for (final Tutorial tutorial : tutorials)
        if (tutorial.isDraftMode()) {
            param = String.format("id=%d", tutorial.getId());

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

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

            super.signIn("assistant2", "assistant2");
            super.request("/assistant/tutorial-session/list", param);
            super.checkPanicExists();
            super.signOut();

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


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

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

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

En este test, lo que hacemos es ir probando que ninguno de los demás roles ni los demás assistant, no pueden acceder a los listado de los tutorial sessions y que devuelve un error 500 indicando que no estamos autorizados.

Show Tutorial Session

- ***Test Positivo***

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial-session/show-positive.csv", encoding = "utf-8", numL
public void test100Positive(final int tutorialIndex, final String code, final int tutorialSessionRec

    super.signIn("assistant2", "assistant2");

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

    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    super.checkInputBoxHasValue("nature", nature);
    super.checkInputBoxHasValue("inicialPeriod", inicialPeriod);
    super.checkInputBoxHasValue("finalPeriod", finalPeriod);
    super.checkInputBoxHasValue("link", link);

    super.signOut();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de forma ascendente de tutorials asociados a ese assistant (comprobando que se muestra).

Tras esto, se accede al formulario de cada tutorial y se pulsará el botón Tutorial Session para mostrar la lista de tutorial sessions asociados al tutorial(se comprueba que aparece) y por último se pulsará sobre cada tutorial sessions y se comprueba que todos los atributos coinciden con los pasados como parámetro. Si todo se realiza de forma correcta cerrará sesión y el test estará correcto.

El ejecutar los test de esta manera nos permite detectar que los tutorials asociados a ese assistant son correctos y que no aparecen otros, los cuales, pertenecen a otros assistant, además que los tutorial sessions asociados a ese tutorial aparezca correctamente y no los de otro tutorial y sobre todo que se muestran las listas y los formularios asociados con los datos correctos según las reglas de negocio impuestas.

Test Hacking

```
@Test
public void test300Hacking() {

    Collection<TutorialSession> tutorialSessions;
    String param;

    super.signIn("assistant1", "assistant1");
    tutorialSessions = this.repository.findManyTutorialSessionByAssistantUsername("assistant1");
    for (final TutorialSession tutorialSession : tutorialSessions)
        if (tutorialSession.getTutorial().isDraftMode()) {
            param = String.format("id=%d", tutorialSession.getTutorial().getId());

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

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

            super.signIn("assistant2", "assistant2");
            super.request("/assistant/tutorial-session/show", param);
            super.checkPanicExists();
            super.signOut();

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



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

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

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

En este test, lo que hacemos es ir probando que ninguno de los demás roles y los demás assistant, no pueden acceder a los datos de los tutorial sessions y que devuelve un error 500 indicando que no estamos autorizados.

Create Tutorial Session

- ***Test Positivo***

```
@CsvFileSource(resources = "/assistant/tutorial-session/create-positive.csv", encoding = "utf-8", nu
public void test100Positive(final int tutorialRecordIndex, final int tutorialSessionRecordIndex, fin

    super.signIn("assistant3", "assistant3");

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

    super.clickOnListingRecord(tutorialRecordIndex);
    super.clickOnButton("Tutorial Sessions");

    super.clickOnButton("Create");
    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    super.fillInputBoxIn("nature", nature);
    super.fillInputBoxIn("inicialPeriod", inicialPeriod);
    super.fillInputBoxIn("finalPeriod", finalPeriod);
    super.fillInputBoxIn("link", link);
    super.clickOnSubmit("Create");

    super.checkListingExists();
    super.sortListing(0, "asc");
    super.checkColumnHasValue(tutorialSessionRecordIndex, 0, title);

    super.clickOnListingRecord(tutorialSessionRecordIndex);
    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    super.checkInputBoxHasValue("nature", nature);
    super.checkInputBoxHasValue("inicialPeriod", inicialPeriod);
    super.checkInputBoxHasValue("finalPeriod", finalPeriod);
    super.checkInputBoxHasValue("link", link);
    super.signOut();
```

En este test en primer lugar, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de forma ascendente de tutorials asociados a ese assistant (comprobando que se muestra).

Tras esto, se accede al formulario de cada tutorial y se pulsará el botón Tutorial Session para mostrar la lista de tutorial sessions asociados al tutorial, después se pulsará el botón de Create y nos saldrá el formulario de creación de un nuevo tutorial session, se añadirá los valores correctos, con la reglas de negocio, en cada campo y se pulsará el botón de Create.

Tras esto, se comprueba que el listado de tutorial session se muestra correctamente con los valores asociados y finalmente se va accediendo a cada tutorial session comprobando que los datos en cada atributo sean los metidos a la hora de haberlos creado.

El ejecutar los test de esta manera nos permite detectar que los tutorials asociados a ese assistant son correctos y que no aparecen otros, los cuales, pertenecen a otros assistant, además que los tutorial sessions asociados a ese tutorial aparezcan correctamente y no los de otro tutorial, y sobre todo que se crean los tutorial session con valores que cumplen las reglas de negocio y que al crearse se muestran las listas y los formularios asociados con los datos correctos.

Test Negativo

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial-session/create-negative.csv", encoding = "utf-8", nu
public void test200Negative(final int tutorialRecordIndex, final int tutorialSessionRecordIndex, fi

    super.signIn("assistant3", "assistant3");

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

    super.clickOnListingRecord(tutorialRecordIndex);
    super.clickOnButton("Tutorial Sessions");

    super.clickOnButton("Create");
    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    super.fillInputBoxIn("nature", nature);
    super.fillInputBoxIn("inicialPeriod", inicialPeriod);
    super.fillInputBoxIn("finalPeriod", finalPeriod);
    super.fillInputBoxIn("link", link);
    super.clickOnSubmit("Create");
    super.checkErrorsExist();

    super.signOut();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de forma ascendente de tutorials asociados a ese assistant (comprobando que se muestra).

Tras esto, se accede al formulario de cada tutorial y se pulsará el botón Tutorial Session para mostrar la lista de tutorial sessions asociados al tutorial, después se pulsará el botón de Create y nos saldrá el formulario de creación de un nuevo tutorial session, se añadirá los valores erróneos de acuerdo con las reglas de negocio en cada campo de forma escalonada para que no salte varios errores a la vez y podamos distinguir que ha fallado, finalmente se pulsará el botón de Create y saltará una error.

El ejecutar los test de esta manera nos permite detectar que los tutorials asociados a ese assistant son correctos y que no aparecen otros, los cuales, pertenecen a otros assistant, además que los tutorial sessions asociados a ese tutorial aparezcan correctamente y no los de otro tutorial, y sobre todo que no se crean los tutorial session con valores que no cumplen las reglas de negocio, saltando excepciones en cada uno de ellos.

Test Hacking

```
@Test
public void test300Hacking() {

    final Collection<Tutorial> tutorials;
    String param;

    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant1");
    for (final Tutorial tutorial : tutorials) {
        param = String.format("tutorialId=%d", tutorial.getId());

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

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

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

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


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

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

En este test, lo que hacemos es ir probando que ninguno de los demás roles, no pueden crear tutorial sessions y que devuelve un error 500 indicando que no estamos autorizados.

Test Hacking

```
@Test
public void test301Hacking() {

    final Collection<Tutorial> tutorials;
    String param;

    super.checkLinkExists("Sign in");
    super.signIn("assistant1", "assistant1");
    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant1");
    for (final Tutorial tutorial : tutorials)
        if (!tutorial.isDraftMode()) {
            param = String.format("masterId=%d", tutorial.getId());
            super.request("/assistant/tutorial-session/create", param);
            super.checkPanicExists();
        }
}
```

En este test, lo que hacemos es comprobar que no pueden crear un tutorial session asociado un tutorial que ya ha sido publicado previamente y que devuelve un error 500 indicando que no estamos autorizados, que es lo que estamos buscando.

Test Hacking

```
@Test
public void test302Hacking() {

    final Collection<Tutorial> tutorials;
    String param;

    super.checkLinkExists("Sign in");
    super.signIn("assistant11", "assistant11");
    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant14");
    for (final Tutorial tutorial : tutorials) {
        param = String.format("masterId=%d", tutorial.getId());
        super.request("/assistant/tutorial-session/create", param);
        super.checkPanicExists();
    }
}
```

En este test, lo que hacemos es comprobar que no pueden crear un tutorial session asociado un tutorial que no pertenece al assistant asociado y que devuelve un error 500 indicando que no estamos autorizados, que es lo que estamos buscando.

Update Tutorial Session

- ***Test Positivo***

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial-session/update-positive.csv", encoding = "utf-8", nu
public void test100Positive(final int tutorialRecordIndex, final int tutorialSessionRecordIndex, fin

    super.signIn("assistant3", "assistant3");

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

    super.clickOnListingRecord(tutorialRecordIndex);
    super.clickOnButton("Tutorial Sessions");

    super.clickOnListingRecord(tutorialSessionRecordIndex);
    super.checkFormExists();

    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    super.fillInputBoxIn("nature", nature);
    super.fillInputBoxIn("inicialPeriod", inicialPeriod);
    super.fillInputBoxIn("finalPeriod", finalPeriod);
    super.fillInputBoxIn("link", link);
    super.clickOnSubmit("Update");

    super.checkListingExists();
    super.sortListing(0, "asc");
    super.checkColumnHasValue(tutorialSessionRecordIndex, 0, title);


    super.clickOnListingRecord(tutorialSessionRecordIndex);
    super.checkFormExists();
    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    super.checkInputBoxHasValue("nature", nature);
    super.checkInputBoxHasValue("inicialPeriod", inicialPeriod);
    super.checkInputBoxHasValue("finalPeriod", finalPeriod);
    super.checkInputBoxHasValue("link", link);

    super.signOut();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de forma ascendente de tutorials asociados a ese assistant (comprobando que se muestra).

Tras esto, se accede al formulario de cada tutorial y se pulsará el botón Tutorial Session para mostrar la lista de tutorial sessions asociados al tutorial(se comprueba que aparece), después se pulsará sobre cada tutorial session, nos aparece su formulario con los datos asociados a cada campo, se añadirá los valores correctos, con la reglas de negocio, en cada campo y se pulsará el botón de Update.

Tras esto, se comprueba que el listado de tutorial session se muestra correctamente con los valores asociados y finalmente se va accediendo a cada tutorial session comprobando que los datos en cada atributo sean los metidos a la hora de haberlos actualizarlo.

El ejecutar los test de esta manera nos permite detectar que los tutorials asociados a ese assistant son correctos y que no aparecen otros, los cuales, pertenecen a otros assistant, además que los tutorial sessions asociados a ese tutorial aparezcan correctamente y no los de otro tutorial, y sobre todo que se actualizan los tutorial session con valores que cumplen las reglas de negocio y que al crearse se muestran las listas y los formularios asociados con los datos correctos.

Test Negative

```
@ParameterizedTest
@CsvFileSource(resources = "/assistant/tutorial-session/update-negative.csv", encoding = "utf-8", n
public void test200Negative(final int tutorialRecordIndex, final int tutorialSessionRecordIndex, fi

    super.signIn("assistant3", "assistant3");

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

    super.clickOnListingRecord(tutorialRecordIndex);
    super.clickOnButton("Tutorial Sessions");

    super.clickOnListingRecord(tutorialSessionRecordIndex);
    super.checkFormExists();

    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    super.fillInputBoxIn("nature", nature);
    super.fillInputBoxIn("inicialPeriod", inicialPeriod);
    super.fillInputBoxIn("finalPeriod", finalPeriod);
    super.fillInputBoxIn("link", link);

    super.clickOnSubmit("Update");

    super.checkErrorsExist();

    super.signOut();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión con un assistant, esto hará que nos redirija al menú de assistant, se nos mostrará la lista de forma ascendente de tutorials asociados a ese assistant (comprobando que se muestra).

Tras esto, se accede al formulario de cada tutorial y se pulsará el botón Tutorial Session para mostrar la lista de tutorial sessions asociados al tutorial, después se pulsará sobre cada tutorial session, nos aparece su formulario con los datos asociados a cada campo, se añadirá valores incorrectos de acuerdo con las reglas de negocio, en cada campo y se pulsará el botón de Update.

El ejecutar los test de esta manera nos permite detectar que los tutorials asociados a ese assistant son correctos y que no aparecen otros, los cuales, pertenecen a otros assistant, además que los tutorial sessions asociados a ese tutorial aparezcan correctamente y no los de otro tutorial, y sobre todo que no se actualizan los tutorial session con valores que no cumplen las reglas de negocio, saltando excepciones en cada uno de ellos.

Test Hacking

```
@Test
public void test300Hacking() {

    final Collection<Tutorial> tutorials;
    String param;

    tutorials = this.repository.findManyTutorialsByAssistantUsername("assistant1");
    for (final Tutorial tutorial : tutorials) {
        param = String.format("id=%d", tutorial.getId());

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

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

        super.signIn("assistant2", "assistant2");
        super.request("/assistant/tutorial-session/update", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("auditor1", "auditor1");
        super.request("/assistant/tutorial-session/update", param);
        super.checkPanicExists();
        super.signOut();


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

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

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

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

En este test, lo que hacemos es ir probando que ninguno de los demás roles, no pueden actualizar tutorial sessions y que devuelve un error 500 indicando que no estamos autorizados.

6 Test Request Performance

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.

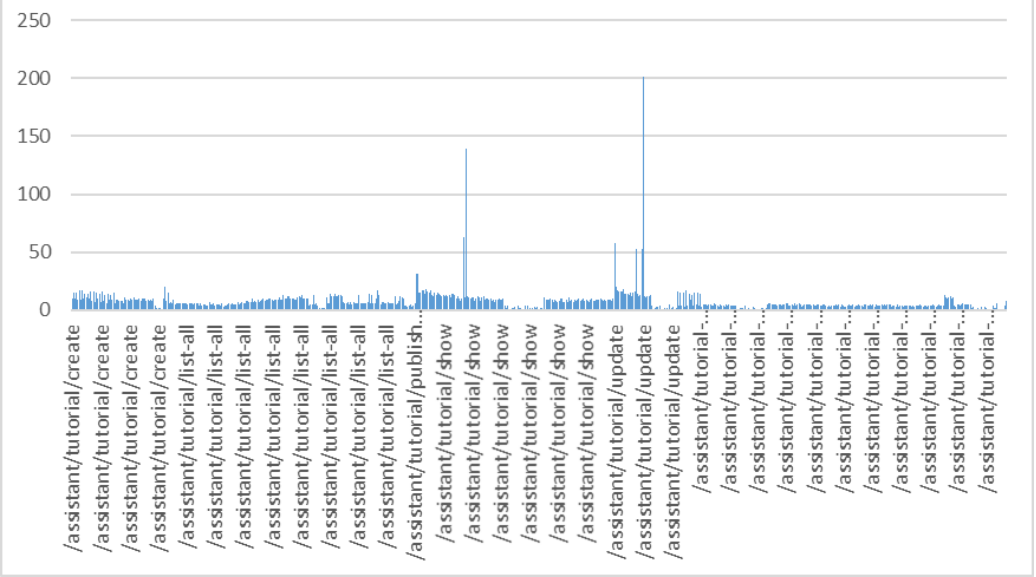

Gráfico del promedio del tiempo de las solicitudes - PC 1

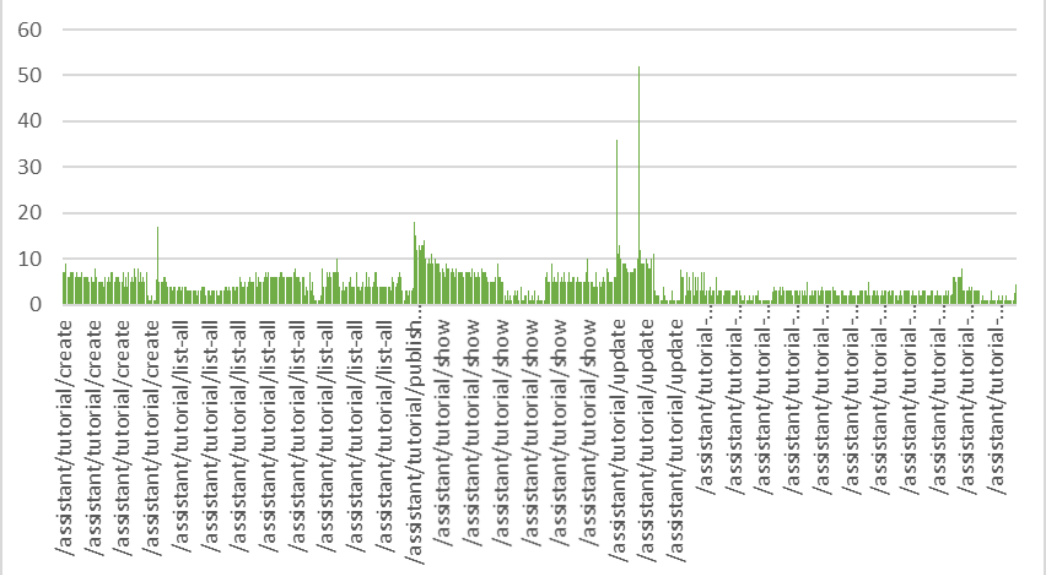

Gráfico del promedio del tiempo de las solicitudes - PC 2

➔ 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.

Result PC 1		
Mean	7,812308947	
Standard Error	0,422525688	
Median	6	
Mode	4	
Standard Deviation	10,81368631	
Sample Variance	116,9358116	
Kurtosis	190,4225108	
Skewness	12,09530096	
Range	200	
Minimum	1	
Maximum	201	
Sum	5117,06236	
Count	655	
Confidence Level(95,0%)	0,829670561	
Interval (ms):	6,982638386	8,64198
Interval (s):	0,006982638	0,008642

Result PC 2		
Mean	4,508527132	
Standard Error	0,134901015	
Median	4	
Mode	3	
Standard Deviation	3,42606086	
Sample Variance	11,73789301	
Kurtosis	67,73468534	
Skewness	5,790963124	
Range	51	
Minimum	1	
Maximum	52	
Sum	2908	
	645	
Confidence Level(95,0	0,264898978	
Interval (ms):	4,243628154	4,773426
Interval (s):	0,004243628	0,004773

Como observamos tanto en las gráficas como en los datos obtenidos el PC2 ha obtenido un rendimiento mejor que el PC1, ya que como se observa en el intervalo de confianza el de PC1 es el doble que el de PC2 aunque la diferencia entre ambos no llega a ser algo alarmante.

7 Test Performance Report

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.

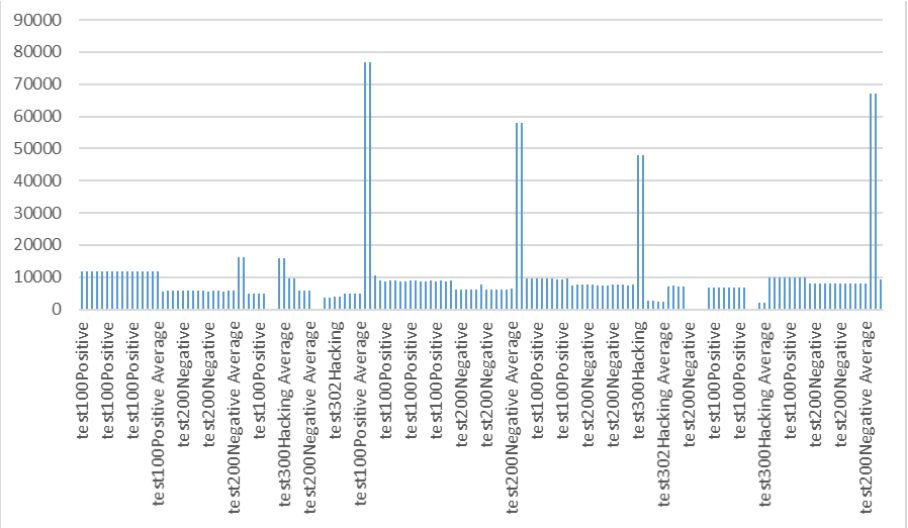

Gráfico del tiempo promedio de ejecución de los tests (de 0 a 450 segundos) - PC1

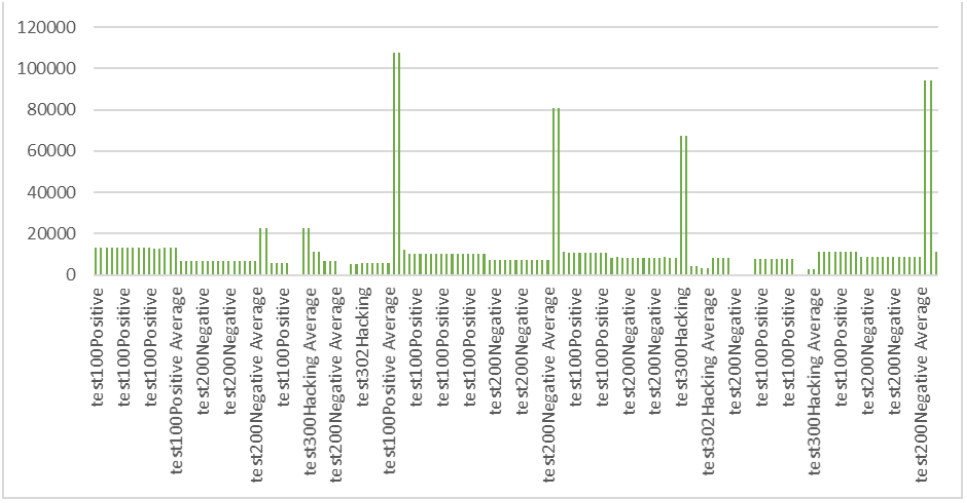

Gráfico del tiempo promedio de ejecución de los tests (de 0 a 450 segundos) - PC1

8 Z-Test

- ***Z-TEST DEL PC 1:***

z-Test: Two Sample for Means		
	BEFORE- PC1	*AFTER PC 1*
Mean	7,817054264	7,817054264
Known Variance	118,5441186	118,5441186
Observations	645	645
Hypothesized Mean Diffe	0	
z	0	
P(Z<=z) one-tail	0,5	
z Critical one-tail	1,644853627	
P(Z<=z) two-tail	1	
z Critical two-tail	1,959963985	

- ***Z-TEST DEL PC 2:***

z-Test: Two Sample for Means		
	BEFORE - pc 2	*AFTER - pc 2*
Mean	4,508527132	4,508527132
Known Vai	11,73789301	11,73789301
Observatic	645	645
Hypothesi:	0	
z	0	
P(Z<=z) on	0,5	
z Critical o	1,644853627	
P(Z<=z) tw	1	
z Critical tv	1,959963985	

Aquí observamos que los tiempos se mantienen igual antes que después porque significa que el rendimiento se va a mantener en el tiempo aunque hagamos más iteraciones.

9 Conclusiones

10 Bibliografía

Intencionalmente en blanco.