DP II - 2022-2023

TEST REPORT

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

Miembro:

- Manuel Ortiz Blanco

Tutor: RAFAEL CORCHUELO GIL

GRUPO - C1.04.12

1 Índice

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	26/05/2023	Primera y última versión del documento

4 Introducción

La intención de este documento es la de recopilar y documentar el análisis de todos los tests realizados para el estudiante 4, 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 Practicum

Test Positivo

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

    super.signIn("company2", "company2");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.checkColumnHasValue(practicumRecordIndex, 0, code);
    super.checkColumnHasValue(practicumRecordIndex, 1, title);

    super.signOut();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión con una company, esto hará que nos redirija al menú de company, se nos mostrará la lista de forma ascendente de practicums asociados a ese company (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 practicums asociados a ese company son correctos y que no aparecen otros los cuales pertenecen a otra company.

Test Hacking

```
@Test
public void test300Hacking() {

    super.checkLinkExists("Sign in");
    super.request("/company/practicum/list-all");
    super.checkPanicExists();

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

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

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

    super.signIn("assistant1", "assistant1");
    super.request("/company/practicum/list-all");
    super.checkPanicExists();
    super.signOut();

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

}
```

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

Show Practicum

Test Positive

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

    super.signIn("company2", "company2");

    super.clickOnMenu("Company", "Practicum List");
    super.sortListing(0, "asc");
    super.clickOnListingRecord(practicumIndex);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de practicums de forma ascendente pulsamos en cada practicum asociados a esa company(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 practicums asociados a esa company son correctos y que no aparecen otros, los cuales pertenecen a otros company y que los valores que se van guardando en cada campo son correctos.

Test Hacking

```
@Test
public void test300Hacking() {
    Collection<Practicum> practicums;
    String param;
    practicums = this.repository.findManyPracticumsByCompanyUsername("company1");
    for (final Practicum p : practicums)
        if (p.isDraftMode()) {
            param = String.format("id=%d", p.getId());

            super.checkLinkExists("Sign in");
            super.request("/company/practicum/show", param);
            super.checkPanicExists();
            super.signIn("administrator", "administrator");
            super.request("/company/practicum/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("assistant5", "assistant5");
            super.request("/company/practicum/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("lecturer1", "lecturer1");
            super.request("/company/practicum/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("student1", "student1");
            super.request("/company/practicum/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("company1", "company1");
            super.request("/company/practicum/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("auditor1", "auditor1");
            super.request("/company/practicum/show", param);
            super.checkPanicExists();
            super.signOut();
        }
```

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

Create Practicum

Test Positive

```
@CsvFileSource(resources = "/company/practicum/create-positive.csv", encoding = "utf-8", numLinesToSkip =
public void test100Positive(final int practicumIndex, final String code, final String title,
    final String abstracts, final String goals, final String course, final String estimatedTime) {

    super.signIn("company15", "company15");

    super.clickOnMenu("Company", "Practicum 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("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");
    super.checkColumnHasValue(practicumIndex, 0, code);
    super.checkColumnHasValue(practicumIndex, 1, title);

    super.clickOnListingRecord(practicumIndex);
    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("Practicum Session");
    super.checkListingExists();
    super.checkListingEmpty();

    super.signOut();
}
```

En este test, lo que hacemos es iniciar sesión con un company, esto hará que nos redirija al menú de company, se nos mostrará la lista de practicums 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 practicums asociado a ese company, comprobamos que la lista muestra correctamente los datos del practicum 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 Practicum Session y que al pulsar sobre él da lugar a una lista de practicums sessions 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 practicum session de ese practicum.

Test Negative

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

    super.signIn("company2", "company2");

    super.clickOnMenu("Company", "Practicum 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 company, esto hará que nos redirija al menú de company, 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 Hacking

```
@Test
public void test300Hacking() {

    super.checkLinkExists("Sign in");
    super.request("/company/practicum/create");
    super.checkPanicExists();

    super.signIn("administrator", "administrator");
    super.request("/company/practicum/create");
    super.checkPanicExists();
    super.signOut();

    super.signIn("student1", "student1");
    super.request("/company/practicum/create");
    super.checkPanicExists();
    super.signOut();

    super.signIn("assistant1", "assistant1");
    super.request("/company/practicum/create");
    super.checkPanicExists();
    super.signOut();

    super.signIn("lecturer1", "lecturer1");
    super.request("/company/practicum/create");
    super.checkPanicExists();
    super.signOut();

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

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

Update Practicum

Test Positive

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

    super.signIn("company7", "company7");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(practicumIndex);
    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(practicumIndex, 0, code);
    super.checkColumnHasValue(practicumIndex, 1, title);

    super.clickOnListingRecord(practicumIndex);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de practicums de forma ascendente (comprobando que se muestra), pulsamos en cada practicum asociados a esa company 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 practicums asociado a ese company, comprobamos que la lista muestra correctamente los datos del practicum 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 = "/company/practicum/update-negative.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test200Negative(final int practicumIndex, final String code, final String title,
    final String abstracts, final String goals, final String course, final String estimatedTime) {

    super.signIn("company7", "company7");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(practicumIndex);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de practicums de forma ascendente (comprobando que se muestra), pulsamos en cada practicum asociados a ese company 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<Practicum> practicums;
    String param;
    practicums = this.repository.findManyPracticumsByCompanyUsername("compnay4");
    for (final Practicum p : practicums) {
        param = String.format("id=%d", p.getId());
        super.checkLinkExists("Sign in");
        super.request("/company/practicum/update", param);
        super.checkPanicExists();
        super.signIn("administrator", "administrator");
        super.request("/company/practicum/update", param);
        super.checkPanicExists();
        super.signOut();
        super.signIn("lecturer2", "lecturer2");
        super.request("/company/practicum/update", param);
        super.checkPanicExists();
        super.signOut();
        super.signIn("student2", "student2");
        super.request("/company/practicum/update", param);
        super.checkPanicExists();
        super.signOut();
        super.signIn("assistant2", "assistant2");
        super.request("/company/practicum/update", param);
        super.checkPanicExists();
        super.signOut();
        super.signIn("company1", "company1");
        super.request("/company/practicum/update", param);
        super.checkPanicExists();
        super.signOut();

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

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

Publish Practicum

Test Positive

```
@ParameterizedTest
@CsvFileSource(resources = "/company/practicum/publish-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int practicumIndex, final String code) {

    super.signIn("company2", "company2");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "desc");
    super.checkColumnHasValue(practicumIndex, 0, code);

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

    super.signOut();
}
```

En este test, lo que hacemos es iniciar sesión con un company, esto hará que nos redirija al menú de company, se nos mostrará la lista de practicums de forma

ascendente (comprobando que se muestra), pulsamos en cada practicum asociados a ese company 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 practicum los cuales tienen un practicum session asociado.

Test Negative

```
@ParameterizedTest
@CsvFileSource(resources = "/company/practicum/publish-negative.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test200Negative(final int practicumIndex, final String code) {

    super.signIn("company7", "company7");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "desc");

    super.checkColumnHasValue(practicumIndex, 0, code);

    super.clickOnListingRecord(practicumIndex);
    super.checkFormExists();
    super.clickOnSubmit("Publish");
    super.checkAlertExists(false);

    super.signOut();
}
```

En este test, lo que hacemos es iniciar sesión con un company, esto hará que nos redirija al menú de company, se nos mostrará la lista de practicums de forma ascendente (comprobando que se muestra), pulsamos en cada practicum asociados a ese company 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 publican los practicums los cuales no tienen un practicum session asociado.

Test Hacking

```
@Test
public void test300Hacking() {
    Collection<Practicum> practicums;
    String params;
    practicums = this.repository.findManyPracticumsByCompanyUsername("company1");
    for (final Practicum practicum : practicums)
        if (practicum.isDraftMode()) {
            params = String.format("id=%d", practicum.getId());

            super.checkLinkExists("Sign in");
            super.request("company/practicum/publish", params);
            super.checkPanicExists();

            super.signIn("administrator", "administrator");
            super.request("company/practicum/publish", params);
            super.checkPanicExists();
            super.signOut();

            super.signIn("lecturer1", "lecturer1");
            super.request("company/practicum/publish", params);
            super.checkPanicExists();
            super.signOut();

            super.signIn("student1", "student1");
            super.request("company/practicum/publish", params);
            super.checkPanicExists();
            super.signOut();

            super.signIn("assistant1", "assistant1");
            super.request("company/practicum/publish", params);
            super.checkPanicExists();
            super.signOut();

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

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

Test Hacking

```
@Test
public void test301Hacking() {

    Collection<Practicum> practicums;
    String params;

    super.signIn("company1", "company1");
    practicums = this.repository.findManyPracticumsByCompanyUsername("company1");
    for (final Practicum p : practicums)
        if (!p.isDraftMode()) {
            params = String.format("id=%d", p.getId());
            super.request("/company/practicum/publish", params);
        }
    super.signOut();
}
```

En este test, lo que hacemos es comprobar que no pueden publicar un practicum 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<Practicum> practicums;
    String params;

    super.signIn("assistant1", "assistant1");
    practicums = this.repository.findManyPracticumsByCompanyUsername("company1");
    for (final Practicum p : practicums) {
        params = String.format("id=%d", p.getId());
        super.request("/company/practicum/publish", params);

    }
    super.signOut();
}
```

En este test, lo que hacemos es comprobar que no pueden publicar un practicum con un rol distinto y que devuelve un error 500 indicando que no estamos autorizados, que es lo que estamos buscando.

Listar Practicum Session

Test Positivo

```
@ParameterizedTest
@CsvFileSource(resources = "/company/practicum-session/list-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int practicumRecordIndex, final String code, final int practicumSessionRecordIndex, final String title) {

    super.signIn("company2", "company2");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.checkColumnHasValue(practicumRecordIndex, 0, code);
    super.clickOnListingRecord(practicumRecordIndex);
    super.checkInputBoxHasValue("code", code);
    super.clickOnButton("Practicum Session");

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

    super.signOut();
}
```

En este test en primer lugar, lo que hacemos es iniciar sesión con un company, esto hará que nos redirija al menú de company, se nos mostrará la lista de forma ascendente de practicums asociados a ese company(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 practicum. Tras esto, se accede al formulario de

cada practicum y se comprueba que el valor de “code” sigue coincidiendo con el que le pasamos como parámetro. Comprobado esto se pulsará el botón Practicum Session para mostrar la lista de practicum session asociados al practicum(se comprueba que aparece) y por último se comprueba que el atributo “title” 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 prcacticums asociados a ese company son correctos y que no aparecen otros los cuales pertenecen a otros company, además que los practicum session asociados a ese practicum aparezca correctamente y no los de otro practicum y sobre todo que se muestran las listas con los datos correctos.

Test Hacking

```
@Test
public void test300Hacking() {
    Collection<Practicum> practicums;
    String param;
    practicums = this.repository.findManyPracticumsByCompanyUsername("company2");
    for (final Practicum p : practicums)
        if (p.isDraftMode()) {
            param = String.format("id=%d", p.getId());
            super.checkLinkExists("Sign in");
            super.request("/company/practicum-session/list", param);
            super.checkPanicExists();
            super.signIn("administrator", "administrator");
            super.request("/company/practicum-session/list", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("assistant1", "assistant1");
            super.request("/company/practicum-session/list", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("lecturer1", "lecturer1");
            super.request("/company/practicum-session/list", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("student1", "student1");
            super.request("/company/practicum-session/list", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("company1", "company1");
            super.request("/company/practicum-session/list", param);
            super.checkPanicExists();
            super.signOut();

            super.signIn("auditor1", "auditor1");
            super.request("/company/practicum-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 company, puedan acceder a los listado de los practicum session y que devuelve un error 500 indicando que no estamos autorizados.

Show Practicum Session

Test Positivo

```
@ParameterizedTest
@CsvFileSource(resources = "/company/practicum-session/show-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int practicumIndex, final String code, final int practicumSessionRecordIndex,
    final String title, final String abstracts, final String inicialPeriod, final String finalPeriod, final String link) {

    super.signIn("company2", "company2");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");
    super.clickOnListingRecord(practicumIndex);
    super.clickOnButton("Practicum Session");
    super.checkListingExists();
    super.clickOnListingRecord(practicumSessionRecordIndex);
    super.checkFormExists();

    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de forma ascendente de practicums asociados a ese company(comprobando que se muestra).

Tras esto, se accede al formulario de cada practicum y se pulsará el botón Practicum Session para mostrar la lista de practicum session asociados al practicum(se comprueba que aparece) y por último se pulsará sobre cada practicum session 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 practicums asociados a ese company son correctos y que no aparecen otros, los cuales, pertenecen a otros company, además que los practicum session asociados a ese practicum aparezca correctamente y no los de otro practicum 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<PracticumSession> practicumSessions;
    String param;
    practicumSessions = this.repository.findManyPracticumSessionByCompanyUsername("company2");
    for (final PracticumSession ps : practicumSessions)
        if (ps.getPracticum().isDraftMode()) {
            param = String.format("id=%d", ps.getPracticum().getId());
            super.checkLinkExists("Sign in");
            super.request("/company/practicum-session/show", param);
            super.checkPanicExists();
            super.signIn("administrator", "administrator");
            super.request("/company/practicum-session/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("assistant1", "assistant1");
            super.request("/company/practicum-session/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("lecturer1", "lecturer1");
            super.request("/company/practicum-session/show", param);
            super.checkPanicExists();
            super.signOut();

            super.signIn("student1", "student1");
            super.request("/company/practicum-session/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("company1", "company1");
            super.request("/company/practicum-session/show", param);
            super.checkPanicExists();
            super.signOut();
            super.signIn("auditor1", "auditor1");
            super.request("/company/practicum-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 company, no pueden acceder a los datos de los practicum session y que devuelve un error 500 indicando que no estamos autorizados.

Create Practicum Session

Test Positivo

```
@ParameterizedTest
@CsvFileSource(resources = "/company/practicum-session/create-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int practicumRecordIndex, final int practicumSessionRecordIndex, final String title,
    final String abstracts, final String inicialPeriod, final String finalPeriod, final String link) {

    super.signIn("company7", "company7");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(practicumRecordIndex);
    super.clickOnButton("Practicum Session");

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

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

    super.clickOnListingRecord(practicumSessionRecordIndex);
    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de forma ascendente de practicums asociados a ese company(comprobando que se muestra).

Tras esto, se accede al formulario de cada practicum y se pulsará el botón Practicum Session para mostrar la lista de practicum session asociados al practicum, después se pulsará el botón de Create Practicum Session y nos saldrá el formulario de creación de un nuevo practicum 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 practicum session se muestra correctamente con los valores asociados y finalmente se va accediendo a cada practicum 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 practicums asociados a ese company son correctos y que no aparecen otros, los cuales, pertenecen a otros company, además que los practicums session asociados a ese practicum aparezcan correctamente y no los de otro practicum, y sobre todo que se crean los practicum 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 = "/company/practicum-session/create-negative.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test200Negative(final int practicumRecordIndex, final int practicumSessionRecordIndex, final String title,
    final String abstracts, final String inicialPeriod, final String finalPeriod, final String link) {

    super.signIn("company7", "company7");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(practicumRecordIndex);
    super.clickOnButton("Practicum Session");

    super.clickOnButton("Create Practicum Session");
    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de forma ascendente de practicums asociados a ese company(comprobando que se muestra).

Tras esto, se accede al formulario de cada practicum y se pulsará el botón Practicums Session para mostrar la lista de practicum session asociados al practicum, después se pulsará el botón de Create Practicum Session y nos saldrá el formulario de creación, 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 practicums asociados a ese company son correctos y que no aparecen otros, los cuales, pertenecen a otros company, además que los practicum session asociados a ese practicum aparezcan correctamente y no los de otro practicum, y sobre todo que no se crean los practicum 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<Practicum> practicums;
    String param;
    practicums = this.repository.findManyPracticumsByCompanyUsername("company1");
    for (final Practicum p : practicums) {
        param = String.format("practicumId=%d", p.getId());

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

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

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

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

        super.signIn("assistant1", "assistant1");
        super.request("/company/practicum-session/create", param);
        super.checkPanicExists();
        super.signOut();

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

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

Test Hacking

```
@Test
public void test301Hacking() {

    final Collection<Practicum> practicums;
    String param;

    super.checkLinkExists("Sign in");
    super.signIn("company1", "company1");
    practicums = this.repository.findManyPracticumsByCompanyUsername("company1");
    for (final Practicum p : practicums)
        if (!p.isDraftMode()) {
            param = String.format("practicum=%d", p.getId());
            super.request("/company/practicum-session/create", param);
            super.checkPanicExists();
        }
}
```

En este test, lo que hacemos es comprobar que no pueden crear un practicum session asociado a un practicum 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<Practicum> practicums;
    String param;

    super.checkLinkExists("Sign in");
    super.signIn("company3", "company3");
    practicums = this.repository.findManyPracticumsByCompanyUsername("company1");
    for (final Practicum p : practicums) {
        param = String.format("practicumId=%d", p.getId());
        super.request("/company/practicum-session/create", param);
        super.checkPanicExists();
    }
}
```

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

Create Practicum Session Exception

Test Positivo

```
@ParameterizedTest
@CsvFileSource(resources = "/company/practicum-session/create-exception-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int practicumRecordIndex, final int practicumSessionRecordIndex, final String title,
    final String abstracts, final String inicialPeriod, final String finalPeriod, final String link) {

    super.signIn("company4", "company4");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(practicumRecordIndex);
    super.clickOnButton("Practicum Session");

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

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

    super.clickOnListingRecord(practicumSessionRecordIndex);
    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de forma ascendente de practicums asociados a ese company(comprobando que se muestra).

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

Tras esto, se comprueba que el listado de practicum session se muestra correctamente con los valores asociados y se marca como exceptional y finalmente se va accediendo a cada practicum 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 practicums asociados a ese company son correctos y que no aparecen otros, los cuales, pertenecen a otros company, además que los practicum session asociados a ese practicum aparezcan correctamente y no los de otro practicum, y sobre todo que se crean los practicum 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 Hacking

```
@Test
public void test300Hacking() {
    final Collection<Practicum> practicums;
    String param;

    practicums = this.repository.findManyPracticumsByCompanyUsername("company2");
    for (final Practicum p : practicums) {
        param = String.format("companyId=%d", p.getId());

        super.checkLinkExists("Sign in");
        super.request("/company/practicum-session/create-exceptional", param);
        super.checkPanicExists();

        super.signIn("administrator", "administrator");
        super.request("/company/practicum-session/create-exceptional", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("lecturer1", "lecturer1");
        super.request("/company/practicum-session/create-exceptional", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("student1", "student1");
        super.request("/company/practicum-session/create-exceptional", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("auditor1", "auditor1");
        super.request("/company/practicum-session/create-exceptional", param);
        super.checkPanicExists();
        super.signOut();

        super.signIn("assistant1", "assistant1");
        super.request("/company/practicum-session/create-exceptional", param);
        super.checkPanicExists();
        super.signOut();
    }
```

En este test, lo que hacemos es ir probando que ninguno de los demás roles excepto company, no pueden crear practicum session exceptional y que devuelve un error 500 indicando que no estamos autorizados.

Test Hacking

```
@Test
public void test301Hacking() {

    final Collection<Practicum> practicums;
    String param;

    super.checkLinkExists("Sign in");
    super.signIn("company3", "company3");
    practicums = this.repository.findManyPracticumsByCompanyUsername("company3");
    for (final Practicum p : practicums)
        if (!p.isDraftMode()) {
            param = String.format("practicumId=%d", p.getId());
            super.request("/company/practicum-session/create-exceptional", param);
            super.checkPanicExists();
        }
}
```

En este test, lo que hacemos es comprobar que no pueden crear un practicum exceptional asociado a un audit que no haya sido publicado 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<Practicum> practicums;
    String param;

    super.checkLinkExists("Sign in");
    super.signIn("company3", "company3");
    practicums = this.repository.findManyPracticumsByCompanyUsername("company2");
    for (final Practicum p : practicums) {
        param = String.format("practicumId=%d", p.getId());
        super.request("/company/practicum-session/create-exceptional", param);
        super.checkPanicExists();
    }
}
```

En este test, lo que hacemos es comprobar que no pueden crear un practicum session exceptional asociado a un practicum que no pertenece al company asociado

y que devuelve un error 500 indicando que no estamos autorizados, que es lo que estamos buscando.

Update Practicum Session

Test Positivo

```
@ParameterizedTest
@CsvFileSource(resources = "/company/practicum-session/update-positive.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test100Positive(final int practicumRecordIndex, final int practicumSessionRecordIndex,
    final String title, final String abstracts, final String inicialPeriod, final String finalPeriod, final String link) {

    super.signIn("company2", "company2");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(practicumRecordIndex);
    super.clickOnButton("Practicum Session");

    super.clickOnListingRecord(practicumSessionRecordIndex);
    super.checkFormExists();

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

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

    super.clickOnListingRecord(practicumSessionRecordIndex);
    super.checkFormExists();
    super.checkInputBoxHasValue("title", title);
    super.checkInputBoxHasValue("abstracts", abstracts);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de forma ascendente de practicums asociados a ese company(comprobando que se muestra).

Tras esto, se accede al formulario de cada practicum y se pulsará el botón Practicum Session para mostrar la lista de practicum session asociados al practicum(se comprueba que aparece), después se pulsará sobre cada practicum 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 practicum session se muestra correctamente con los valores asociados y finalmente se va accediendo a cada practicum session comprobando que los datos en cada atributo sean los metidos a la hora de haberlos actualizado.

El ejecutar los test de esta manera nos permite detectar que los practicums asociados a ese company son correctos y que no aparecen otros, los cuales, pertenecen a otros company, además que los practicums session asociados a ese practicum aparezcan correctamente y no los de otro practicum, y sobre todo que se actualizan los

practicum session con valores que cumplen las reglas de negocio y que al actualizarse se muestran las listas y los formularios asociados con los datos correctos.

Test Negative

```
@ParameterizedTest
@CsvFileSource(resources = "/company/practicum-session/update-negative.csv", encoding = "utf-8", numLinesToSkip = 1)
public void test200Negative(final int practicumRecordIndex, final int practicumSessionRecordIndex, final String title,
    final String abstracts, final String inicialPeriod, final String finalPeriod, final String link) {

    super.signIn("company2", "company2");

    super.clickOnMenu("Company", "Practicum List");
    super.checkListingExists();
    super.sortListing(0, "asc");

    super.clickOnListingRecord(practicumRecordIndex);
    super.clickOnButton("Practicum Session");

    super.clickOnListingRecord(practicumSessionRecordIndex);
    super.checkFormExists();

    super.fillInputBoxIn("title", title);
    super.fillInputBoxIn("abstracts", abstracts);
    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 company, esto hará que nos redirija al menú de company, se nos mostrará la lista de forma ascendente de practicum asociados a ese company(comprobando que se muestra).

Tras esto, se accede al formulario de cada practicum y se pulsará el botón Practicum Session para mostrar la lista de practicum session asociados al practicum, después se pulsará sobre cada practicum 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 practicum asociados a ese company son correctos y que no aparecen otros, los cuales, pertenecen a otros companys, además que los practicum session asociados a ese practicum aparezcan correctamente y no los de otro practicum, y sobre todo que no se actualizan los practicum 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<Practicum> practicums;
    String param;
    practicums = this.repository.findManyPracticumsByCompanyUsername("company2");
    for (final Practicum p : practicums) {
        param = String.format("id=%d", p.getId());
        super.checkLinkExists("Sign in");
        super.request("/company/practicum-session/update", param);
        super.checkPanicExists();
        super.signIn("administrator", "administrator");
        super.request("/company/practicum-session/update", param);
        super.checkPanicExists();
        super.signOut();
        super.signIn("assistant1", "assistant1");
        super.request("/company/practicum-session/update", param);
        super.checkPanicExists();
        super.signOut();
        super.signIn("auditor1", "auditor1");
        super.request("/company/practicum-session/update", param);
        super.checkPanicExists();
        super.signOut();
        super.signIn("lecturer1", "lecturer1");
        super.request("/company/practicum-session/update", param);
        super.checkPanicExists();
        super.signOut();

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

        super.signIn("company1", "company1");
        super.request("/company/practicum-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 practicum session 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.

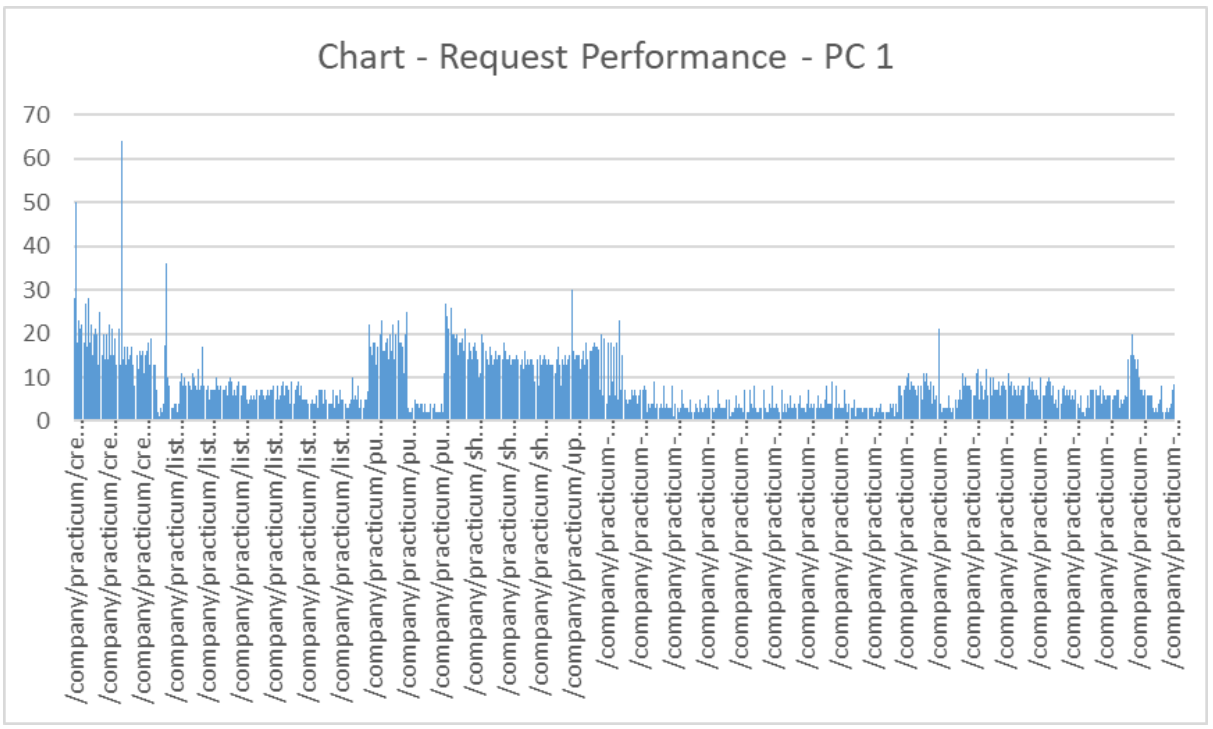

Gráfico del promedio del tiempo de las solicitudes - PC 1

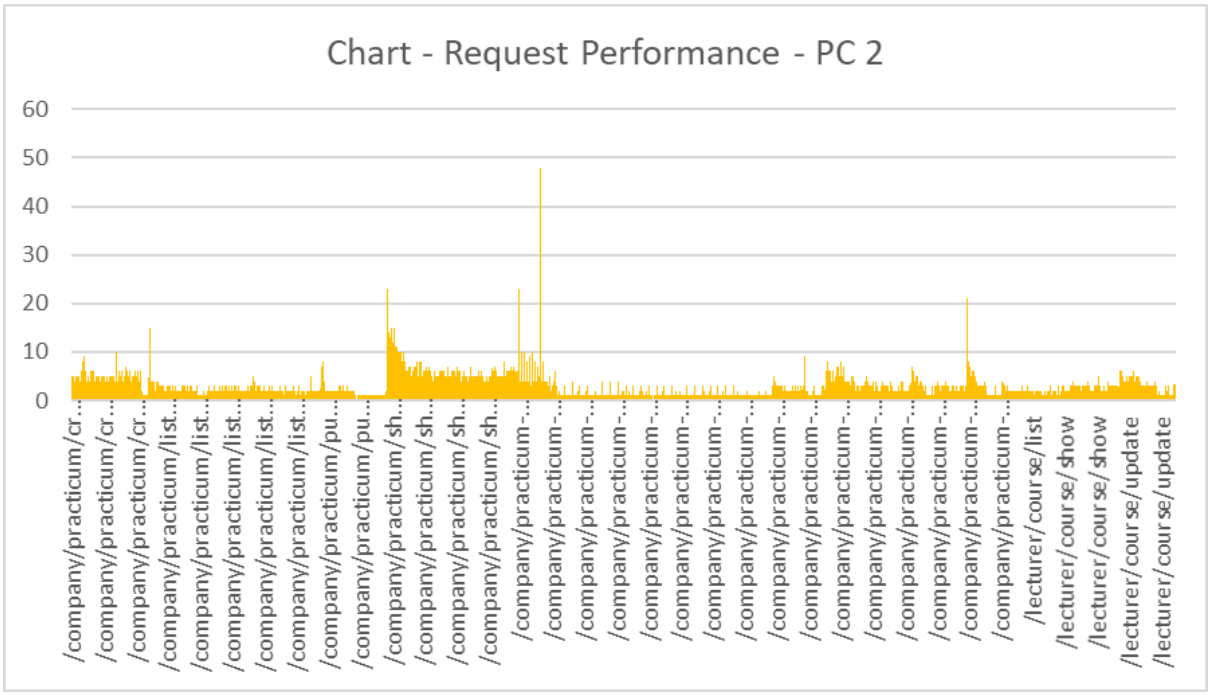

Gráfico del promedio del tiempo de las solicitudes - PC 2

➔ 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.

RESULTS BY PC 1	
Mean	8,47632312
Standard Error	0,236864559
Median	7
Mode	3
Standard Deviation	6,346909492
Sample Variance	40,2832601
Kurtosis	10,47628517
Skewness	2,143570515
Range	63
Minimum	1
Maximum	64
Sum	6086
Count	718
Confidence Level(95,0%)	0,465030998

Interval (ms):	8,011292122	8,941354
Interval (s):	0,008011292	0,008941

RESULTS BY PC 2	
Mean	3,245283019
Standard Error	0,099164081
Median	3
Mode	1
Standard Deviation	2,887701629
Sample Variance	8,338820699
Kurtosis	74,79172307
Skewness	6,086305901
Range	48
Minimum	0
Maximum	48
Sum	2752
Count	848
Confidence Level(95,0%	0,194636156

Interval (ms):	3,050646863	3,439919
Interval (s):	0,003050647	0,00344

➔ Z test

Para la realización de este apartado se ha llevado a cabo un ***análisis Z test*** con los tiempos obtenidos en los reportes de antes y después.

z-Test: Two Sample for Means		
	BEFORE	*AFTER*
Mean	8,47632312	8,491832174
Known Variance	40,2832601	39,98681438
Observations	718	729
Hypothesized Mean Differenc	0	
z	-0,046559584	
P(Z<=z) one-tail	0,481432122	
z Critical one-tail	1,644853627	
P(Z<=z) two-tail	0,962864244	
z Critical two-tail	1,959963985	

PC 1: α = 1 - Confidence level = 1- 0,46503 = 0,53497
PC 2 : α = 1-Confidence level = 1- 0,19463 = 0,80537

Como observamos el P-Value (P(Z<=z) two tail) que es 0,962864 se encuentra entre α y 1.00 , es decir entre (α ,1.00] esto quiere decir que los cambios no han supuesto ninguna mejora significativa, es decir, los tiempos de muestreo son diferentes, ***pero globalmente son los mismos.***

Por lo que podemos concluir con que no podemos asegurar cual de los dos PC (PC 1 o PC 2) es más rápido.

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.

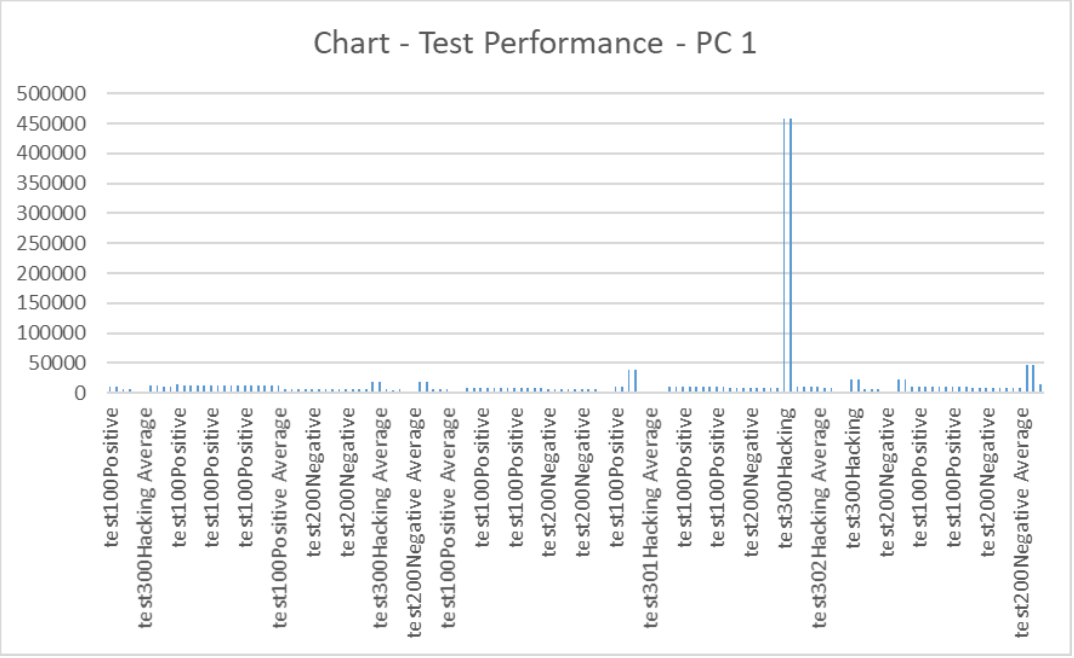

Gráfico del tiempo promedio de ejecución de los tests (de 0 a 450 segundos) - PC1

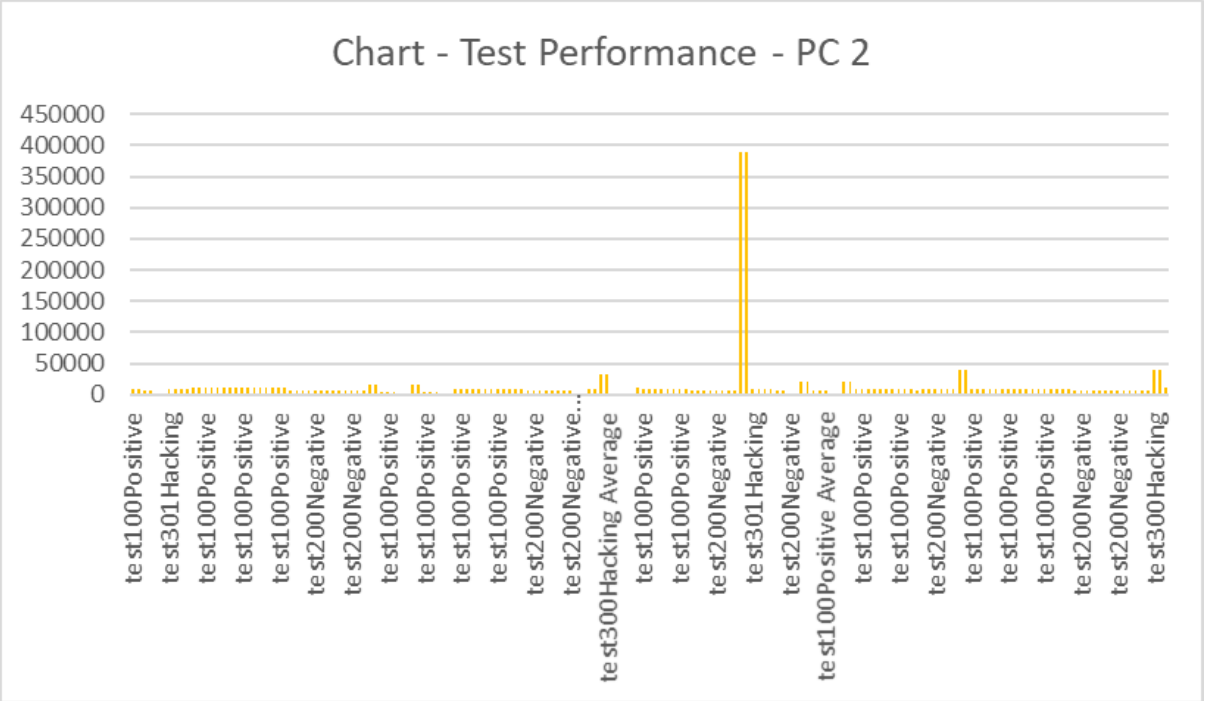

Gráfico del tiempo promedio de ejecución de los tests (de 0 a 400 segundos) - PC2

8 Conclusiones

Intencionalmente en blanco.

9 Bibliografía

Intencionalmente en blanco.