

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

DP2-Informe de Planificación y Progreso


Student 4



Grado en Ingeniería Informática – Ingeniería del Software

Diseño y Pruebas II

Curso 2023 – 2024

<u>Group:</u>	C2.X02
<u>Repository:</u>	https://github.com/SoniaRM/Acme-SF-D04-24.5.0
<u>Student #4</u>	<div data-bbox="402 1662 552 1854"></div> <div data-bbox="392 1861 810 1966"><p>UVUS: josmirmar2 Name: José Manuel Miret Martín Email: josmirmar2@alum.us.es</p></div>
<u>Date:</u>	Sevilla Julio 2, 2024

Índice de contenido

1. Versiones	3
2. Listado de tareas	3
3. Capturas de Pantalla	3
4. Presupuesto	3
5. Lista de registros	4
6. Conflictos surgidos	4
7. Comparación de costes estimados / reales	4
8. Tareas específicas para la convocatoria 2	5
9. Presupuesto actualizado	6
10. Tareas específicas para la convocatoria 3	7
11. Presupuesto actualizado	9

1. Versiones

Versión	Fecha	Autor
1.0	16/05/2024	José Manuel Miret Martín
1.1	25/05/2024	José Manuel Miret Martín
1.2	02/07/2024	José Manuel Miret Martín
1.3	07/10/2024	José Manuel Miret Martín

2. Listado de tareas

Task 9. Produce a test suite for Requirements #6 and #7. Para ello llevaré a cabo el rol de tester
Task 10. Produce a testing report. Para hacerlo llevaré a cabo el rol de tester.
Task 23. Produce an analysis report. Para ello llevaré a cabo el rol de project manager.
Task 24. Produce a planning and progress report. Para ello llevaré a cabo el rol de project manager

3. Capturas de Pantalla

4. Presupuesto

Tarea	Tiempo estimado (en horas)	Tiempo dedicado(en horas)	Coste estimado	Coste real	Rol
Task 9	11	21,65	220	433	Tester
Task 10	3,2	5,32	64	106,4	Tester

Task 23	0,5	0,5	10	10	Project Manager
Task 24	0,5	0,6	10	12	Project Manager

Coste total real : 561,4€

Coste total estimado : 304€

5. Lista de registros

Para esta entrega no realicé ningún registro.

6. Conflictos surgidos

Con respecto a los conflictos, para esta entrega podríamos destacar el tema de tener que actualizar el framework. Fue una cuestión que me planteó varios conflictos que tuve que dedicarle bastante tiempo a resolver. A su vez, otra cuestión que me quitó bastante tiempo fue el resolver conflictos que habían de la anterior entrega. no eran graves conflictos, pero eran problemas que se tenían que resolver para esta entrega.

Con respecto a las tareas de este sprint, los tres documentos a realizar (las tareas 10, 23 y 24) no me habían generado problema grave, excepto la tarea 10. Debido a mi falta de conocimiento de ese documento y además de la cantidad de fallos a la hora de grabar los datos necesarios para elaborar el grafo de mis registros con los promedios de cada función testeada junto con los cálculos de intervalo de confianza de mis tests. Por otro lado, el testing me ha llevado más tiempo del esperado debido a mi falta de conocimiento sobre cómo debía elaborarse, junto con los pequeños errores y la refactorización del código de la entrega anterior. Además, gran parte de ese tiempo se invirtió en rehacer los tests varias veces, ya que no abarcaban todo el código desarrollado.

7. Comparación de costes estimados / reales

En esta ocasión, el coste estimado se desvió bastante de lo esperado. Esto es debido a la tarea 9, que me ha llevado mucho más tiempo de lo esperado debido a lo comentado en el anterior apartado. Esto ha hecho que el coste esperado sea mucho más elevado que el coste estimado. Por otro lado, la tarea 10 me llevó más tiempo de lo esperado, siendo el coste estimado más bajo que el real de esta tarea en específico. Las otras 2 tareas han tenido mejores estimaciones, por lo que el coste al final es muy similar.

8. Tareas específicas para la convocatoria 2

Para esta segunda convocatoria se ha comentado varios cambios a realizar en el código para corregir lo elaborado en la anterior convocatoria, entregable 4. Estas tareas específicas son:

Reelaborar los tests para que se prueben todos los casos de prueba posibles, detectando posibles errores:

Testing requirements

9) Produce a test suite for Requirements #6 and #7.

[] Realmente los tests no son correctos porque hay datos inválidos que permiten persistir las entidades y al revés. Es decir, si se indican fechas de un patrocinio o una invoice erróneas, como las que he puesto de ejemplo en los requisitos del D03, los tests deberían de devolver fallos y no los devuelven. De la misma forma, cuando se introducen datos correctos, por ejemplo una fecha en el futuro en el due date de la invoice, el test debería de dar un error de validación y no lo muestra. Los tests se deberían de haber usado para detectar este tipo de errores.

Para esta tarea se han elaborado de nuevo todos los tests para las entidades sponsorship e invoices, tanto los de tipo hacking como los de tipo safe, para que prueben todos los casos posibles y detectando posibles errores no deseados:

```
if (!super.getBuffer().getErrors().hasErrors("startDate")) {
    Date minimumDeadline;

    if (!super.getBuffer().getErrors().hasErrors("moment")) {
        super.state(MomentHelper.isAfterOrEqual(object.getStartDate(), object.getMoment(), "startDate", "sponsor.sponsorship.form.error.invalid-moment"));

        minimumDeadline = object.getEndDate() == null ? null : MomentHelper.deltaFromMoment(object.getEndDate(), 1, ChronoUnit.MONTHS);
        super.state(object.getEndDate() == null || MomentHelper.isBefore(object.getStartDate(), minimumDeadline), "startDate", "sponsor.sponsorship.form.error.invalid-moment");
    } else {
        super.state(false, "startDate", "sponsor.sponsorship.form.error.invalid-moment");
    }
}

if (!super.getBuffer().getErrors().hasErrors("endDate")) {
    Date maximumDeadline;

    if (!super.getBuffer().getErrors().hasErrors("moment")) {
        super.state(MomentHelper.isAfterOrEqual(object.getEndDate(), object.getMoment(), "endDate", "sponsor.sponsorship.form.error.invalid-moment"));

        maximumDeadline = object.getStartDate() == null ? null : MomentHelper.deltaFromMoment(object.getStartDate(), 1, ChronoUnit.MONTHS);
        super.state(object.getStartDate() == null || MomentHelper.isAfter(object.getEndDate(), maximumDeadline), "endDate", "sponsor.sponsorship.form.error.invalid-moment");
    } else {
        super.state(false, "endDate", "sponsor.sponsorship.form.error.invalid-moment");
    }
}

if (!super.getBuffer().getErrors().hasErrors("project")) {
    Boolean isDraftMode = this.repository.projectIsDraftMode(object.getProject().getId());
    super.state(!isDraftMode, "project", "sponsor.sponsorship.form.error.not-published-project");
}

101 }
102
103 if (!super.getBuffer().getErrors().hasErrors("project")) {
104     Boolean isDraftMode = this.repository.projectIsDraftMode(object.getProject().getId());
105     super.state(!isDraftMode, "project", "sponsor.sponsorship.form.error.not-published-project");
106 }
107
108 }
```


Coste total real : 304€

Coste total estimado : 683€

Esta vez el coste se ha alejado mucho más de lo previsto, debido a que en las tareas 9, 10 y 23 ha aumentado el tiempo. Estas tres tareas se han alejado del coste estimado, sobre todo la de la tarea 9, siendo su coste real más del doble con respecto al coste estimado para esa tarea.

10. Tareas específicas para la convocatoria 3

Para esta tercera convocatoria se ha comentado varios cambios a realizar en el código para corregir lo elaborado en la anterior convocatoria, entregable 4. Estas tareas específicas son:

Reelaborar los tests para que se prueben todos los casos de prueba posibles, detectando posibles errores:

9) Produce a test suite for Requirements #6 and #7.

[] Faltan pruebas para detectar los errores que he comentado anteriormente. Sobre todo, uno fácil de detectar podría ser el de las fechas. La fecha del sistema debería de ser aceptada en los sponsorships.

Para esta tarea se han elaborado de nuevo todos los tests para las entidades sponsorship e invoices, tanto los de tipo hacking como los de tipo safe, para que prueben todos los casos posibles y detectando posibles errores no deseados:

```
62
63 @Override
64 public void validate(final Invoice object) {
65     assert object != null;
66     if (!super.getBuffer().getErrors().hasErrors("code"))
67         super.state(this.repository.existsOtherByCodeAndId(object.getCode(), object.getId()), "code", "sponsor.invoice.form.error.duplicated");
68
69     if (!super.getBuffer().getErrors().hasErrors("dueDate")) {
70         Date minimumDeadline;
71         minimumDeadline = MomentHelper.deltaFromMoment(object.getRegistrationTime(), 30, ChronoUnit.DAYS);
72         super.state(MomentHelper.isAfter(object.getDueDate(), minimumDeadline), "dueDate", "sponsor.invoice.form.error.to-close-from-registration");
73     }
74
75     if (!super.getBuffer().getErrors().hasErrors("quantity")) {
76         double before;
77         double res;
78         Collection<Invoice> invoices;
79
80         if (object.getTax() != null) {
81             invoices = this.repository.findManyInvoicesBySponsorshipId(object.getSponsorship().getId());
82             before = this.repository.findOneInvoiceById(object.getId()).totalAmount().getAmount();
83             res = object.totalAmount().getAmount() - before;
84             for (Invoice i : invoices)
85                 res += i.totalAmount().getAmount();
86         } else
87             res = -10000000000.00;
88
89         super.state(object.getQuantity().getAmount() > 0, "quantity", "sponsor.invoice.form.error.negative-salary");
90         super.state(Arrays.asList(this.repository.findAcceptedCurrencies().split(",")).contains(object.getQuantity().getCurrency()), "quantity", "sponsor.invoice.form.error.invalid-currency");
91         super.state(res <= object.getSponsorship().getAmount().getAmount(), "quantity", "sponsor.invoice.form.error.bad-total-amount");
92     }
93 }
94
95 }
```

```

@Override
public void authorise() {
    boolean status;
    int masterId;
    Sponsor sponsor;
    Sponsorship sponsorship;

    masterId = super.getRequest().getData("masterId", int.class);
    sponsorship = this.repository.findOneSponsorshipById(masterId);
    sponsor = sponsorship == null ? null : sponsorship.getSponsor();
    status = sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsor);

    super.getResponse().setAuthorised(status);
}

@Override
public void load() {
    Invoice object;
    Sponsor sponsor;
    int masterId;
    Sponsorship sponsorship;

    sponsor = this.repository.findOneSponsorById(super.getRequest().getPrincipal().getActiveRoleId());

    masterId = super.getRequest().getData("masterId", int.class);
    sponsorship = this.repository.findOneSponsorshipById(masterId);

    object = new Invoice();
    object.setDraftMode(true);
    object.setSponsorship(sponsorship);
    object.setSponsor(sponsor);
    object.setRegistrationTime(MomentHelper.getCurrentMoment());

    super.getBuffer().addData(object);
}

@Override
public void authorise() {
    super.getResponse().setAuthorised(true);
}

@Override
public void load() {
    Sponsorship object;
    Sponsor sponsor;

    sponsor = this.repository.findOneSponsorById(super.getRequest().getPrincipal().getActiveRoleId());

    object = new Sponsorship();
    object.setDraftMode(true);
    object.setSponsor(sponsor);
    object.setMoment(MomentHelper.getCurrentMoment());

    super.getBuffer().addData(object);
}

@Override
public void bind(final Sponsorship object) {
    assert object != null;

    int projectId;
    Project project;

    projectId = super.getRequest().getData("project", int.class);
    project = this.repository.findOneProjectById(projectId);

    super.bind(object, "code", "moment", "startDate", "endDate", "email", "link", "type", "amount");
    object.setProject(project);
}

```

Elaborar correctamente el Informe de Testing Student 4:

10) Produce a testing report.

[X] La conclusión no se hace explícita. No se indica si hay evidencia estadística para determinar que el rendimiento de un ordenador sea mejor que otro. Se indica que las medias son comparables y, si son comparables, ¿cuál es la conclusión?

	Ordenador 1	Ordenador 2
Media	11,2247446	12,9046383
Varianza (con	112,534893	149,733192
Observaciones	469	469
Diferencia hip	0	
z	-2,24644353	
P(Z<=z) una cc	0,01233781	
Valor crítico d	1,64485363	
Valor crítico d	0,02467561	
Valor crítico d	1,95996398	

Análisis: Estos son los resultados de la prueba Z para determinar si las dos medias son comparables. Los intervalos de ambos casos cumplen las expectativas, ya que en ningún caso comienzan o terminan después de 1 segundo.

A su vez, podemos afirmar que las medias son comparables, dado que el valor P (P value) es menor que 0.05 en el valor crítico de Z.

Para llegar a estos resultados, se han ejecutado todas las pruebas diseñadas para las entidades "invoices" y "sponsorships".

Para esta tarea se ha re-elaborado la conclusión del informe de testing, haciéndolo mucho más explícito y determinando cuál ordenador es mejor en rendimiento.

11. Presupuesto actualizado

Tarea	Tiempo estimado (en horas)	Tiempo dedicado(en horas)	Coste estimado	Coste real	Rol
Task 9	11	27,13	220	522,6	Tester
Task 10	3,2	8,62	64	172,4	Tester
Task 23	0,5	0,8	10	16	Project Manager
Task 24	0,5	0,8	10	16	Project Manager

Coste total real : 304€

Coste total estimado : 727€

Esta vez el coste se ha alejado mucho más de lo previsto, debido a que en las tareas 9 y 24 ha aumentado el tiempo. Estas dos tareas se han alejado del coste estimado, sobre todo la de la tarea 9, siendo su coste real más del doble con respecto al coste estimado para esa tarea.