DP2 2022
Progress Report

Acme Recipes

Enlace al repositorio

Miembros:

- Ballestero López, Jesús (jesballop@alum.us.es)
- Botello Romero, Francisco (frabotrom@alum.us.es)
- León Valderrama, Juan José (jualeoval@alum.us.es)
- Moreno Pérez, Juan Carlos (juamorper2@alum.us.es)
- Serrano Mena, Antonio Roberto (antsermen@alum.us.es)
- Suárez García, Antonio José (antsuagar@alum.us.es)

Tutor: Manuel Jesús Jiménez Navarro

GRUPO S02

Versión 5.0

02/09/2022

Índice

Resumen Ejecutivo

El trabajo realizado por el grupo ha sido más que satisfactorio. Todos los miembros han cumplido con sus tareas sin ningún tipo de problema.

Este quinto sprint comprende cuatro tareas de documentación, una de implementación, y por último una de testeo. Esta última consistió en ejecutar la suite de tests unitarios y una prueba manual del sistema exhaustiva por parte de todos los testers.

El sistema funciona de manera impecable y está listo para recibir cualquier petición de cambio y/o mantenimiento en el futuro, gracias a la buena cobertura de tests, así como a la alta cohesión y el bajo acoplamiento que se consiguen al utilizar el framework adecuado.

Historial de versiones

Fecha	Versión	Descripción	Sprint
05/07/2022	0.1	- Creación del documento	1
06/07/2022	0.2	- Introducción - Resumen ejecutivo - Contenido - Conclusión	1
10/07/2022	1.0	- Actualización de contenidos	1
16/07/2022	2.0	- Progreso en el sprint 2 añadido	2
18/07/2022	2.1	- Cambios en la política de versionado - Adición de apartados correspondientes al tercer entregable	3
31/07/2022	2.2	- Progreso del sprint 3 añadido	3
01/08/2022	2.3	- Actualización de contenidos	3
08/08/2022	3.0	- Actualización de contenidos	3
25/08/2022	4.0	- Progreso del sprint 4 añadido	4
02/09/2022	5.0	- Progreso del sprint 5 añadido - Finalización del documento	5

Introducción

Este informe recoge el modo en el que el grupo ha desarrollado los entregables, dando a conocer la cronología de todo el proceso.

En la sección *Progreso,* se describirán la asignación y elaboración de tareas para cada miembro del grupo. Después, se explicará el nivel de esfuerzo empleado a la hora de realizar dichas tareas. Por último, se detalla qué tal fue la comunicación grupal.

Las conclusiones obtenidas tras haber realizado este informe de planificación se encuentran en la sección *Conclusiones*. Dicha sección contiene algunos aspectos claves sobre la planificación de las tareas y el presupuesto realizado.

La sección *Bibliografía* recoge la bibliografía relevante que ha sido utilizada para la elaboración de este informe. Concretamente, esta sección se encuentra vacía en este documento, puesto que no se ha empleado ninguna bibliografía para elaborarlo.

Finalmente, a modo de resumen, este documento cuenta con la siguiente estructura: portada, índice, resumen ejecutivo, tabla de control de versiones, introducción, progreso, conclusiones y bibliografía.

Progreso

D01 - Introduction

En primer lugar, a cada miembro del grupo se le asignó la tarea de colocar en el menú de Anonymous su link favorito, junto a su información personal.

Para ello, todos debían instalar el entorno de desarrollo y sus herramientas, junto a la creación del proyecto en GitHub con una rama diferente por tarea, para evitar conflictos con la rama principal (master). Dicha tarea, junto con la instalación y creación de todo se completó con éxito por parte de todos, y no hubo mayor problema.

Después, se debían realizar una serie de informes (incluyendo este), los cuales repartimos de forma equitativa, colocando las tareas oportunas en el kanban de GitHub.

Por último, ha de constatar que el equipo ha trabajado de forma satisfactoria y conjunta, comunicándose en todo momento por WhatsApp y Discord (esta última ha sido utilizada para realizar reuniones y trabajo en equipo, ya que permite establecer una comunicación en tiempo real por voz) para resolver las dudas que fueron surgiendo.

D02 - The data model

A partir de los conocimientos adquiridos y las herramientas utilizadas durante el primer entregable, el segundo entregable ha sido mucho más rápido de realizar.

En primer lugar se realizó un documento con las pautas a seguir a la hora de trabajar en una tarea, abarcando desde su creación hasta su corrección, incluyendo la política de commits.

Tras ello se asignaron las tareas de forma equitativa y teniendo en cuenta las relaciones entre las clases a implementar.

La primera tarea en ser realizada fue el diagrama UML de dominio de clases, el cual era esencial para el resto de tareas.

Una vez diseñado el diagrama y tras ser revisado por el grupo, se empezó a implementar requisitos, lo cual se hizo en menos de una semana.

Finalmente, el project manager lanzó la release *D02 - The data model.*

Gracias a la experiencia previa en la asignatura y el documento con las pautas de trabajo no surgieron dudas fuera del proceso de diseño del diagrama de dominio de clases, las cuales fueron pocas y se resolvieron rápidamente.

D03 - Displaying data

Las tareas fueron completadas y testeadas muy cerca de la fecha acordada, tras un breve periodo vacacional por parte del equipo de desarrollo entre la finalización del segundo sprint y el comienzo del tercero.

El desarrollo comenzó con las features de menor acoplamiento y la implementación de tests automatizados.

Para realizar cálculos de los precios de cada recipe teniendo en cuenta el intercambio de divisa, se implementó la feature *MoneyExchange* que hace uso de la api de ExchangeRate.

Tan solo tres features produjeron dudas que no fueron planteadas al inicio del sprint durante la implementación: Mostrar recipes según ingredients y utensils, tests de peeps con fecha automatizada y el cambio de unidades.

En cuanto a recipes by thing:

AcmeFramework implementa una funcionalidad para buscar cadenas contenidas en un listado, pero lo que queremos mostrar no es parte de recipes. Para solventarlo, creamos un campo “payload” en el modelo que contiene todos los ingredientes y utensilios asociados a dicha receta. Dicho campo no será visible en el listado pero si podrá accederse desde el input que el framework implementa por defecto en las listas.

```
@Override
public void unbind(final Request<Recipe> request, final Recipe entity, final Model model) {
    assert request != null;
    assert entity != null;
    assert model != null;

    String payload;

    request.unbind(entity, model, "code", "heading","totalPrice");

    payload = String.format(
        "%s; %s; %s",
        entity.getChef(),
        this.repository.findAllThingNamesOfRecipe(entity),
        this.repository.findAllThingCodesOfRecipe(entity));
    model.setAttribute("payload", payload);

}
```

En cuanto a peeps:

Una forma muy simple puede ser distribuirlos homogéneamente en los últimos 30 días preservando su orden en el fichero de datos de prueba.

Necesitamos que los tests de listado de peeps funcionen indefinidamente, pero como sólo se muestran los de los últimos 30 días esto no ocurrirá si implementamos una clase de tests similar a las demás features.

La solución que le hemos dado a este problema es introducir varios peeps con una fecha especial, en concreto enero de 1900, que es modificada antes de comenzar los tests. Dicha modificación (que podemos ver en las siguientes figuras) consiste en restar a la fecha actual un número de días correspondiente al día del mes en la fecha de 1900. De esta manera los cuatro peeps que usamos como datos de test datarán entre uno y cuatro días antes del día que sean ejecutados.

```
@Override
@BeforeAll
public void beforeAll() {
    super.beforeAll();
    FactoryHelper.autowire(this);
    this.patchPeeps();
}

protected void patchPeeps() {
    Collection<Peep> peeps;
    Date moment;
    final Date deadline = new Date("1901/01/01 00:00");

    peeps = this.repository.findAPeepsToPatch(deadline);
    for (final Peep peep : peeps) {
        moment = this.adjustMoment(peep.getInstantiationMoment());
        peep.setInstantiationMoment(moment);
        this.repository.save(peep);
    }
}

private Date adjustMoment(final Date instantiationMoment) {
    Calendar res;
    res = Calendar.getInstance();
    final int deltaDays = instantiationMoment.getDate();
    res.add(Calendar.DATE, -deltaDays);
    return res.getTime();
}
```

```
String moment;

moment = new SimpleDateFormat("yyyy/MM/dd HH:mm").format(this.adjustMoment(new Date(instantiationMoment)));

super.clickOnMenu("Any","Peeps Recent list");
super.checkListingExists();
super.sortListing(0, "desc");

super.checkColumnHasValue(recordIndex, 0, moment);
super.checkColumnHasValue(recordIndex, 1, heading);
super.checkColumnHasValue(recordIndex, 2, writer);
super.checkColumnHasValue(recordIndex, 3, text);
super.checkColumnHasValue(recordIndex, 4, email);
```

Con respecto a las unidades:

El cliente ha sido informado de las consecuencias que conlleva no implementar un sistema de gestión de unidades y ha pedido una estimación de los sobrecostes asociados a implementar las funcionalidades asociadas a dicho sistema.

El sobrecoste queda recogido en el anexo del planning report, y en caso de ser aceptado comenzaremos de inmediato con la implementación.

D04 - Editing data

Este cuarto sprint ha sido el más costoso en cuanto a horas dedicadas, pero ha sido el más sencillo sin duda alguna. Todos los miembros finalizaron sus tareas sin plantear dudas sobre las propias funcionalidades o la arquitectura de la aplicación, pues al finalizar el tercer sprint ya teníamos las ideas bien claras.

No se han tenido que tomar decisiones fuera de lo planeado (salvo una excepción) gracias a la experiencia adquirida durante el desarrollo de una aplicación muy similar pocos meses atrás.

Surgieron excepciones en los tests al mergear varios pull requests sin ejecutar la suite completa antes de cada merge, pero se arreglaron sin mayor problema.

Spam detector y system configuration:

Para que la funcionalidad de detección de spam pudiese leer los datos almacenados en la configuración del sistema, fue necesario cambiar el patrón con el que se almacenan dichos datos. Fue un cambio muy sencillo entre “,” y “:” para separar los términos y sus pesos.

D05 - Final delivery

Para el último sprint no quedaban muchas tareas por realizar, se trata de cuatro documentos, un cambio de dependencias y una buena revisión de la aplicación por parte de los testers.

Lo primero fue actualizar el planning report con las tareas del sprint.

Tras esto se crearon los dos documentos en los que se exponen los conocimientos adquiridos durante la asignatura con respecto a los sistemas de información web.

Finalmente se realizó la tarea de cambio de dependencias y se dejó constancia de todo ello en este progress report.

Acme Framework como componente jar independiente:

De manera similar a la independización del componente de detección de spam, el framework fue exportado como fichero .jar e importado de nuevo en la carpeta “library” dentro del proyecto.
No surgieron problemas durante la realización de la tarea.

Conclusiones

Consideramos esencial la realización de este documento, pues hace de testimonio sobre el desempeño grupal durante el periodo de trabajo. A partir de la información que se recoge podemos aplicar las medidas necesarias en el siguiente sprint para remediar carencias y mejorar el rendimiento.

Gracias al informe realizado durante los primeros sprints, pudimos mejorar el modo de trabajo empleado en el tercero y así alcanzar una mayor fluidez.

Bibliografía

No procede.