UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por nombre alumno
en Universidad de Burgos — 15 de marzo de 2017
Tutor: nombre tutor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

A.2. Planificación temporal

Sprint 0 (1/2/2017 - 8/2/2017)

En este primer «Sprint» determinamos que posibles herramientas de traducción pueden ser útiles para este proyecto sin profundizar demasiado. Se hace también la primera toma de contacto con Thoth para ver su funcionamiento básico.

Por otro lado, analizamos las posibilidades del proyecto, determinando los caminos en los que puede derivar dicho proyecto. Es decir, en estos primeros pasos no sabemos como funcionan estas herramientas, es por ello que el resultado sea más simple del esperado o por el contrario resulten complicaciones que lleven un tiempo mayor al esperado.

Además de esto, hacemos una introducción a las herramientas de documentación y gestión. Por ello aclaramos que:

- Para la gestión de versiones haremos uso de GitHub, asociando un gestor de tareas llamado Zenhub que funciona como un «plugin» en el buscador.
- Realizo un primer contacto con LaTeX

En esta primeras semanas no me aclaro mucho con el funcionamiento del gestor de versiones, y es por ello que no hago un buen uso de los «commits» ni de los «issues» que proporciona Github.

Sprint 1 (8/2/2017 - 15/2/2017)

Ya en la segunda semana realizo una evaluación más exhaustiva de las herramientas de traducción, analizando los pros y los contras de ellas. Por lo tanto tomamos la decisión de centrarnos en GWT como la principal y con la que vamos a llevar a cabo el proyecto.

Definimos como tareas semanales:

- Evaluar los pros y contras de las diferentes herramientas de traducción de código.
- Documentar esa evaluación con LaTeX, familiarizándome con la manera de documentar.
- Realizar las primeras pruebas con GWT, de una forma simple.

En esta semana hago alguna prueba simple con GWT, gracias a los ejemplos que proporciona la página oficial a modo de tutorial. También realizo alguna prueba simple con JSweet para ver su funcionamiento real y si es, de verdad, útil para poder llevar a cabo el proyecto. En el último ejemplo que hago me ocurre un problema con GWT que no termino de solucionar y que me obliga a posponer la prueba de ese ejemplo para el siguiente sprint.

También profundizo algo más en la documentación y en el uso de las herramientas para documentar. Hasta este punto no he asociado las tareas o «issues» con los «milestones» y por lo tanto no queda registrado el tiempo del sprint.

Sprint 2 (15/2/2017 - 22/2/2017)

En este tercer sprint, lo primero que hago es solucionar el anterior problema que tuve con GWT. Consistía en configurar bien el entorno de Eclipse y el «plugin» para poder hacer la ejecución de GWT con «Super Dev Mode», alternativa implantada por los desarrolladores para evitar la necesidad de instalar una extension de GWT en el buscador en el cual se lanza la aplicación.

También en esta semana descubrimos una nueva herramienta relacionada con el tema, que se llama Vaadin y que nos puede servir, por lo menos para hacer una comparativa más completa de las herramientas de traducción.

La parte más importante de esta semana es la prueba de traducción del «core» de Thoth, que aunque no sale como esperamos, ha resultado útil para conocer con mayor profundidad tanto la aplicación como la herramienta.

Por lo tanto como tareas para este sprint:

- Solucionar el error surgido con GWT.
- Realizar pruebas de traducción con el «core» de Thoth.
- Incluir la nueva herramienta en la comparativa.

Muy a mi pesar, en el «milestone» de esta semana, aunque he pasado cada tarea al estado de realizada o «done» no las he cerrado hasta darnos cuenta al final del sprint, es por ello que el «burndown» queda de esta manera.

Sprint 2 (22/2/2017 - 1/3/2017)

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

B.2. Objetivos generales

B.3. Catalogo de requisitos

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

Manual para GWT:

Para poder trabajar con GWT debo descargar el SDK de GWT proporcionado en la página web oficial. Los requisitos previos para crear un aplicación web con Google Web Toolkit son básicamente dos: tener instalada la SDK de Java en su versión 1.6 o cualquiera superior a esta y tener instalado también Apache Ant o en su defecto Apache Maven.

Es fácil saber si cumplo ambos requisitos. Una vez descargado el SDK de GWT desde la página oficial [1], accedo a la carpeta desde la consola de comandos, y ahí intento ejecutar el comando «webAppCreator». En caso de que la consola me devuelva un error en el que indica que Java no ha podido reconocerlo como un comando interno, quiere decir que no cumplo esos requisitos previos.

Posteriormente puedo proceder de varias formas y con diferentes plataformas. En mi caso he elegido la plataforma Eclipse sobre la que trabajaré en su versión 4.4 que es también denominada con el nombre de Luna.

GWT se instala en Eclipse como un «plugin» y para ello debo ir, dentro de Eclipse a «añadir un nuevo software» donde encontraremos una ventana

[1] http://www.oracle.com/technetwork/java/javase/downloads/index.html

donde escribir la dirección del software a instalar. Para mi versión (llamada Luna) la URL es «https://developers.google.com/eclipse/docs/install-eclipse-4.4» que una vez añadida mostrará los paquetes a instalar. Estos deben ser Google Plugin for Eclipse y GWT Designed for GPE, y una vez seleccionados se procederá a su instalación normal. Después de reiniciar procedo a referenciar el SDK de GWT yendo a «preferencias», «Google» y dentro de «Web Toolkit» añadir el SDK que descargué anteriormente.

Para añadir un nuevo proyecto a Eclipse tengo que importarlo desde Maven, puedo hacerlo añadiendo un proyecto de Maven existente. Una vez seleccionado el proyecto, hay que configurar su ejecución. Debemos ir a «Run Configuration» y seleccionar el constructor de Maven. Una vez ahí, puedo añadirle un nombre específico y seleccionando el directorio base el proyecto a ejecutar y por último el «Goal», o meta debe ser «gwt:run», ya que en caso contrario no se producirá una ejecución correcta.

Antes de la ejecución se debe instalar el constructor de Maven desde el debuguer, sino puede dar algunos errores. Una vez finalizada la instalación, podremos ejecutar un proyecto con GWT apareciendo así el modo desarrollador y ahí se podrá lanzar la aplicación en el buscador por defecto.

El «plugin» de GWT es sencillo de utilizar y solo con hacer click en «New Web Application Project» aparecerá un «wizard» en el cual hay que introducir el nombre del proyecto y el paquete principal. Se recomienda desmarcar «Use Google App Engine» ya que además de que no nos interesa, puede que nos dé algún error. Hecho esto se creará un proyecto automáticamente en forma de ejemplo, con un paquete cliente, otro servidor y uno compartido.

(05/03/2017) Los proyectos en GWT se componen de una parte cliente y otra servidor. He creado un proyecto con sólo la parte del cliente, la cual es la más importante. En el paquete «src/client» he incluido la clase principal que llamará a las otras. En realidad no se si esto es correcto al cien por cien. Solución del error del «prefuse» es descargando de la página oficial [2] el proyecto, incluyo el paquete prefuse y añado, desde «Java Build Path» la librería «lucene» porque es necesaria para algunas clases del paquete. Inicialmente muestra unos errores del tipo «did you forget to inherit a required module?» referente a que dentro del proyecto, la clase con extensión .gwt.xml no incluye algunas de las rutas de los recursos que utilizo. No estiendo muy bien el error. (continuar con la explicación)

Pruebas realizadas en la semana 5:

Una vez nos dimos cuenta de que el fallo de tratar de hacer la aplicación en la parte del servidor era que GWT no reconocía algunas de las librerías

[2] http://prefuse.org/

claves, tanto en la parte visual como en el núcleo de la aplicación, decidimos buscar otros caminos alternativos.

El funcionamiento de GWT consiste en traducir a «JavaScript» la parte del cliente y la compartida. En consecuencia decidimos hacer pruebas en las que las partes mas fundamentales del núcleo se encontrasen en el lado del servidor. De esta forma cuando el cliente necesitase hacer algún uso de métodos con librerías no reconocidas por GWT, simplemente llamase al servidor ya que este podría soportar dichos métodos.

En los primeros intentos nos dimos cuenta de que estas llamadas no se podían hacer de una forma simple, ya que la comunicación entre cliente y servidor no funcionaba y no obteníamos los resultados que esperábamos. Aún así seguimos haciendo pruebas para asegurarnos, metiendo dentro del paquete «compartido» las partes del núcleo mas cercanas a lo que nosotros consideramos la vista. El problema seguía siendo esa comunicación. Interpretaba como del lado del cliente lo que nosotros queríamos que formara parte del servidor, dando errores debido a que GWT no trabaja con esas librerías.

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía