UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por nombre alumno
en Universidad de Burgos — 10 de enero de 2019
Tutor: nombre tutor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En el siguiente apartado se presentará la planificación del proyecto y la metodología adoptada junto con las técnicas y herramientas utilizadas.

La metodología que se ha utilizado para el desarrollo del proyecto ha sido *Scrum*, una metodología de desarrollo ágil.

Para saber qué recursos se necesitan, hay que analizar todas las partes que forman el proyecto. El analisis podemos desglosarlo en:

- Planificación temporal: Tiempo que tendremos que invertir en cada una de las fases del desarrollo.
- Estudio de viabilidad: Aquí tendremos en cuenta las licencias necesarias, beneficios, costes y normativas de leyes a cumplir.

A.2. Planificación temporal

La metodología comentada anteriormente, *Scrum*, facilita nuestra organización pues está diseñada para poder realizar entregas parciales y periódicas en pequeños y cortos espacios de tiempo, donde unos prontos resultados y una flexibilidad considerable son fundamentales. Para concretar todo lo comentado, necesitábamos una herramienta que nos permitiera desarrollar el producto acorde a esta metodología y así es como se empezó a utilizar *GitHub* como plataforma donde alojar el proyecto.

*T*anto Scrum como la herramienta *GitHub*, contienen numerosos términos, pero a continuación explicaremos aquellos que se han utilizado en el proyecto.

- *Milestone:* También llamado *Sprint*. En nuestro caso, tiempo comprendido entre una o dos semanas. A cada *Milestone* le corresponde un número determinado de *issues*. Al inicio de cada *Sprint* hemos tenido como costumbre realizar una pequeña reunión de nos más de una hora para comentar dificultades tenidas hasta ese momento y qué cosas nuevas íbamos a planificar.

- *Issue:* Así llamamos a cada una de las pequeñas tareas a realizar en este periodo de tiempo. No hay número máximo de *issues* por *milestone*. En nuestro caso, ha oscilado entre siete o quince, en función de la duración del sprint y del tiempo del que disponíamos.

- *ZenHub:* A través de *ZenHub*[1], una plataforma que se integra en *GitHub* mediante la instalación de una extensión para el navegador, podemos gestionar nuestro tiempo de una manera más eficiente ya que permite asignar tiempos a cada una de las *issues*. De esta manera, te permitía hacer cálculos para evitar crear más issues de las que ibas a ser capaz de realizar y poder organizar mejor los siguientes *sprints*.

- *Board:* Otro de los elementos más significativos de *ZenHub* es el tablero. Cuenta con numerosas *pipelines* personalizables entre las que mover *issues* en función de su estado de realización.

A continuación se mostrará el avance a lo largo de las iteracciones del proyecto:

Sprint 0: Inicio del proyecto:

Reunión inicial en la que se comenta en rasgos generales la estructura y finalidad del desarrollo de este proyecto.

Issues:

- Elegir el *framework* que se va a utilizar. Ver qué versión de *CakePHP* es la más estable hoy en día.

[1]Se ha valorado la utilización de alguna alternativa como puede ser *Trello* pero nos hemos decantado por *ZenHub* al integrarse perfectamente en *GitHub*.

- Diseñar la base de datos *MySQL* a través de un modelo Entidad-Relación.
- Buscar un servidor gratuito donde poder alojar la aplicación.
- Crear la estructura de directorios del proyecto.

Este *sprint* me ha servido para tomar un primer contacto con el proyecto, preparar el entorno de desarrollo en el que a través de *XAMPP* se ha tenido que simular en local un servidor *Apache* con *PHP* y *MySQL*. Además, se ha realizado una primera búsqueda de lo que sería el servidor donde lo vamos a alojar, me ha servido para refrescar conceptos de base de datos y realizar el modelo entidad relación.

En cuanto al framework a utilizar, me decanté por *CakePHP 3.5* pues aunque en el trabajo utilizamos una versión bastante más antigua[2] no deja de ser el mismo *framework* y satisfacía las necesidades presentadas.

Sprint 1: Base de datos y administración:

Issues:

- Subir proyecto base al servidor.
- Contactar con el cliente: Carga inicial de datos: Necesitábamos conocer qué estructura de ficheros venía utilizando hasta ahora para poder nosotros diseñar la base de datos en función de sus necesidades.
- Crear tablas en Base de Datos MySQL.
- Documentación: Cargar la plantilla Latex.

Una vez que teníamos claro qué *framework* se iba a utilizar, que habíamos creado correctamente la estructura del proyecto, el paso fue subirlo a un servidor. La opción que elegimos en este primer momento, fue la más económica, eligiendo un hosting gratuito en https://domitienda.com/hosting-ilimitado/ con las siguientes características:

- Hosting 1 Dominio.
- 250 MB de espacio en disco.

[2] CakePHP 2.x

- 1 buzón de correo.
- Certificado Let's Encrypt.
- Protección permanente de aplicaciones
- Soporte 24 horas.

El cliente envió una hoja de cálculo con los datos que manejaba hasta ahora por lo que se decidió dejar para el siguiente *sprint* la carga de tablas en base de datos mientras se desgranaba el documento.

Para la realización de la documentación, elegimos *LaTeX* acompañado del editor *TeXstudio* por lo que importamos la plantilla facilitada desde la universidad.

Sprint 2: Base de datos y carga inicial:

Issues:

- Crear diagrama relacional y crear estructura de tablas en base de datos.
- Carga inicial de datos en la aplicación.

Los objetivos de este *sprint* son claros. Una vez desgranados los datos de la hoja de cálculo facilitada por el cliente, modelar un diagrama relacional y crear la estructura de tablas en base de datos. Con esto cumplido, el siguiente paso era alimentar nuestra base de datos con datos reales. Esta carga inicial la hemos hecho leyendo los datos directamente de la hoja de cálculo facilitada por el cliente.

Dado que existían pequeñas disconcordancias entre los propios datos y no teniendo la seguridad de que ser los datos y estructura definitiva, nos limitamos a hacer un primer boceto de lo que sería nuestro diagrama relacional y se empezó a preparar la estructura MVC (Modelo, Vista y Controlador) para las entidades que teníamos fijas. Desde el *framework* es necesario establecer una comunicación entre el cliente y la base de datos. Esto es lo que se trató de hacer en este *sprint*.

Sprint 3: Carga de datos con una primera administración:

Issues:

- Renombrar tablas de base de datos.
- Cargar en base de datos los datos facilitados en un excel. II
- Empezar la administración de los datos.

Una vez que teníamos la estructura montada en el proyecto, seguimos dando forma a nuestro diagrama hasta su versión definitiva. Con la base de datos ya bien montada, utilizando la librería *PhpSpreadsheet* empezamos a cargar los datos.

En cuanto a la administración de los datos, no fue viable por lo que lo dejamos para el siguiente *sprint*.

Sprint 4: Terminar la carga de datos y empezar la administración de las entidades:

Issues:

- Terminar de cargar la información en base de datos.
- Administración de la tabla *countries*.
- Administración de la tabla *regions*.
- Administración de la tabla *fuels*.
- Administración de la tabla *technologies*.

En este *sprint* ha habido problemas a la hora de cerrar la carga inicial de todos los datos pues nos dimos cuenta de que había tablas cuyos datos no existían y es que el cliente estaba pendiente de facilitarlos.

Sin embargo, con la mayoría de tablas con sus datos correspondientes cargados, se ha empezado a construir la administración para cada una de ellas.

El primer proceso es laborioso pues tienes que realizar los *mockups* de las pantallas que se quieren construir.

En este *sprint* se ha invertido más tiempo del planificado pues los tiempos de elaboración de *mockups*, maquetación de una cabecera y plantilla común a toda la aplicación no han sido los previstos. Esto nos vino bien para a partir de ahora, ser más cuidadosos con el presupuesto de horas pues las tareas eran menos previsibles y más complejas.

Sprint 5: Terminar la administración:

Issues:

- Administración de la tabla *arcs*.
- Administración de la tabla *typelines*.
- Administración entre *arcs* y *typelines* (*arcs_typelines*).
- Administración entre *regions* y *technologies* (*regions_technologies*).
- Administración entre *regions* y *arcs* (*regions_arcs*).
- Administración de la tabla *rangedemands*.

Mientras que para las tablas en las que la administración era a través de un formulario no existía mayor problema, la tabla *rangedemands* se abastece de un fichero con formato hoja de cálculo que el usuario sube a la aplicación. Esto ha resultado bastante problemático por el gran número de registros a insertar en la tabla dándonos problemas con la memoria disponible y el tiempo de ejecución empleado.

Otro problema detectado ha sido el tamaño máximo de fichero que podemos subir a la aplicación. Por defecto, en la configuración de *PHP* que trae *Apache*, este valor viene limitado. Nosotros hemos tenido que entrar al fichero `php.ini` y modificar el valor de una propiedad dejándola como: `upload_max_filesize=100M`

Al margen del problema con tamaño del archivo que ha quedado corregido y tras muchas horas de pruebas, se han intentado solucionar los inconvenientes pero en este *sprint* no ha sido posible, por lo que arrastramos el desarrollo de esta funcionalidad al siguiente.

Sprint 6: Logotipo de Weblectric y documentación del proyecto:

Issues:

- Logotipo para la aplicación: *Weblectric*.
- Documentación: 2 Objetivos del proyecto.
- Documentación: 3 Conceptos teóricos.
- Documentación: 4 Técnicas y herramientas.
- Documentación: 5 Aspectos relevantes del desarrollo del proyecto.
- Documentación: 6 Trabajos relacionados.

Aprovechando el parón de navidades, se ha decidido aprovechar para diseñar un logotipo para la aplicación y formalizar por escrito en la documentación del proyecto todas aquellas cosas que estaban en el aire.

Escribir el apartado .ºbjetivos del proyecto"(2) y "Técnicas y herramientas"(4) no nos ha llevado demasiado tiempo porque era material ya trabajado y comentado previamente, en cambio, si que ha requerido de una labor más a fondo los apartados Çonceptos teóricos"(3), .ªspectos relevantes del desarrollo del proyecto"(5) y "Trabajos relacionados"(6) pues ha requerido labor de investigación sobre algún algoritmo de optimización para el apartado número 3 y de competencias profesionales ya existentes en el mercado para el apartado número 6.

...

...

...

A.3. Estudio de viabilidad

En esta sección se hablará del presupuesto económico equivalente al desarrollo del proyecto y de su viabilidad legal.

Viabilidad económica

Se va a proceder a hacer un estudio de viabilidad económica del proyecto justificando todos los gastos que se hubieran tenido si este desarrollo formara parte del mundo laboral en lugar del plan de estudios del grado.

A través de *ZenHub* podemos asignar el tiempo que estimamos en la planificación del *sprint* para el desarrollo de una *issue*. En nuestro caso, salen

un total de: XXX horas. Suponiendo que la hora de trabajo del desarrollador está pagada a 8 €, tenemos un total de:

$$8€/\text{h} \times 000\text{h} = 280€$$

Además se tendrá en cuenta el trabajo llevado a cabo por los tutores del proyecto. Sumando al valor anterior 1 hora de reunión por *sprint* y 3 horas extras para revisar documentación, plataforma y correos electronicos con consultas procedentes del desarrollador y cobrando a 10 €/h, hacen un total de:

$$7\textit{sprints} \times 10€/\text{h} \times 4\text{h} = 0000€$$

De momento sumamos un total de XXXX €que si le aplicamos el XX % de seguridad social, se queda en un total de:

$$\text{XXXX} + (\text{XXXX} \times 0{,}00) = \text{YYYY}$$

Por último, destacar el coste material. En la parte software no tenemos gastos pues las aplicaciones y herramientas son todas gratuitas. En la parte hardware, podríamos valorar:

- Ordenador portátil: 600 €.
- Monitor: 150 €.
- Teclado y ratón: 25 €.

Sumando el todo nos da un total de 775 €y según la agencia tributaria [3], los equipos para procesos de información tienen una vida útil de 8 años (70.080 horas) y como nuestro proyecto ha tenido una duración de XXX horas, deja un total de:

$$\frac{775€}{12 \text{ Meses} \times 8 \text{ Años}} \times 3{,}5 \text{ Meses} = 28,26€$$

Por último, sumar el gasto producido por el servidor donde hemos tenido alojada la web. Aunque gran parte de la duración del proyecto lo hemos

[3] Tablas de amortización en 2018: https://bit.ly/2ND4vCw

Tabla A.1: Costes totales del proyecto

Costes	Importe
Desarrolladores	XXXX €
Tutores del proyecto	280 €
Seguridad Social	XXXX €
Material hardware	28,26 €
Servidor	6 €
Totales	YYYY €

tenido en un *hosting* gratuito, en la recta final y por causa de las limitaciones que nos ofrecía el plan, en el último mes de proyecto, los gastos han sido:

$$\frac{18€}{3 \text{ Meses}} = 6€$$

Con todos los costes mencionados, el desglose e importe final se recoge en la tabla A.1:

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

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