UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Visor de espectros
Documentación Técnica

Presentado por Iván Iglesias Cuesta
en Universidad de Burgos — 2 de junio de 2018
Tutor: Dr. José Francisco Díez Pastor
Cotutor: Dr. César Ignacio García Osorio

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Para que un proyecto se desarrolle con normalidad y con el menor número de imprevistos posibles es esencial que cuente con una fase de planificación. Aquí se estima el tiempo, trabajo y dinero que puede llegar a usarse durante la realización del proyecto. Para ello, se debe analizar en detalle cada parte del proyecto. De cara al futuro, el análisis del proyecto puede servir para predecir como de bien puede desarrollarse una continuación del mismo.

La planificación del proyecto consta de dos partes:

- **Planificación temporal**: en esta parte se analiza y planifica el tiempo que se va a dedicar a cada parte del proyecto, fecha de inicio y final aproximado, teniendo en cuenta el trabajo necesario para cada parte.
- **Estudio de viabilidad**: en esta parte se analiza como de viable es la realización del proyecto, se divide a su vez en dos apartados:
  - **Económica**: en esta parte se estiman los costes y los beneficios que puede suponer el proyecto.
  - **Legal**: en esta parte se analizan los conceptos legales del proyecto, como podrían ser las licencias del proyecto o la política de protección de datos.

A.2. Planificación temporal

La planificación temporal se organiza mediante *sprints*. Cada *sprint* dura una o dos semanas. Al terminar cada *sprint* se realiza una reunión con los

tutores para dar por terminado

Sprint 0

En este *sprint* marcó el inicio del proyecto. La lista de tareas está disponible en Sprint 0[1]. En reuniones previas se habló con los tutores en que iba a consistir en proyecto, pero no estaba claro con que tecnologías desarrollarlo. Se decidió hacer una evaluación de las tecnologías posibles y crear unos prototipos básicos. Todas las tareas se completaron a tiempo.

Sprint 1

En este *sprint* se habló sobre el despliegue de la aplicación. Los tutores comentaron que en proyectos webs anteriores el despliegue se solía dejar para las etapas finales del proyecto, haciendo que todos los problemas asociados surjan en esas finales, retrasando el despliegue y, a veces, no llegar a desplegar la aplicación.

Se conocía la plataforma Heroku para ello y se probó a usarla, adicionalmente se buscaron otras alternativas. También se usó el prototipo escogido para crear el proyecto definitivo y se trabajó en la memoria. La última parte fue estudiar las guías de estilo de Python, aplicarlas en los prototipos y documentar su código.

La lista de tareas está disponible en Sprint 1[2]. Todas las tareas se completaron a tiempo.

Sprint 2

Dado que en la aplicación se necesita que los usuarios suban contenido, se necesita control de usuarios, en este *sprint* se investigaron formas de ofrecerlo. También debido a la necesidad de disponer almacenamiento persistente se tuvo que mirar otras formas de despliegue y buscar como He-

[1] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/1?closed=1

[2] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/2?closed=1

roku lo ofrece, que de serie no lo hace. Al final se decidió cambiar a Nanobox.

La lista de tareas está disponible en Sprint 2[3]. Todas las tareas se completaron a tiempo.

Sprint 3

En este *sprint* se podría decir que comienza el desarrollo del proyecto, basado en el prototipo. Como tal se mejoró el aspecto visual de la aplicación, se cambio la estructura para tener partes diferenciadas y mantenibles, añadir control de usuarios y mejorar la subida de ficheros.

También se planteó añadir subir y visualizar un *dataset* completo, escribir el manual de despliegue y los casos de uso.

La lista de tareas está disponible en Sprint 3[4]. El manual de despliegue y la subida de *datasets* no se pudieron completar. El gráfico *burndown* del *sprint* se ve en la figura A.1.

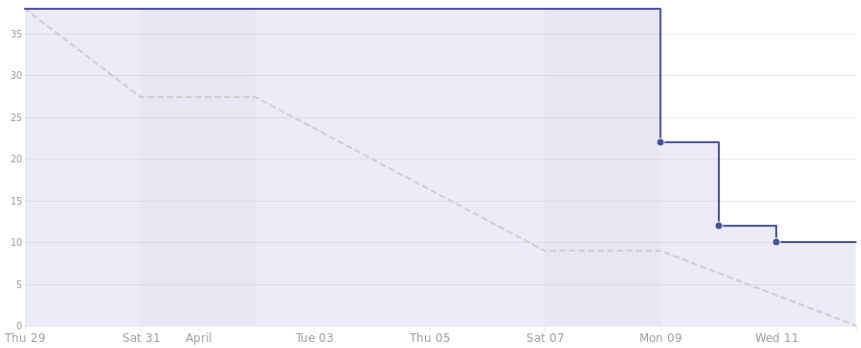

Figura A.1: Burndown del *sprint* 3

[3] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/3?closed=1

[4] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/4?closed=1

Sprint 4

En este *sprint* se completan las tareas que no habían dado tiempo del *sprint* anterior y se planteó usar una base de datos en lugar de almacenamiento para guardar los datos, por lo que se realizó una comparación entre formas de almacenar los datos. También se añadió la opción de borrar un *dataset* ya almacenado.

Al igual que en el *sprint* anterior se estructuró la aplicación como conjunto, en este *sprint* se estructura la parte de visualización. También se arreglaron dos *bugs* que se introdujeron en el *sprint* anterior en la aplicación desplegada.

La lista de tareas está disponible en Sprint 4[5]. Todas las tareas se completaron a tiempo. El gráfico *burndown* del *sprint* se ve en la figura A.2.

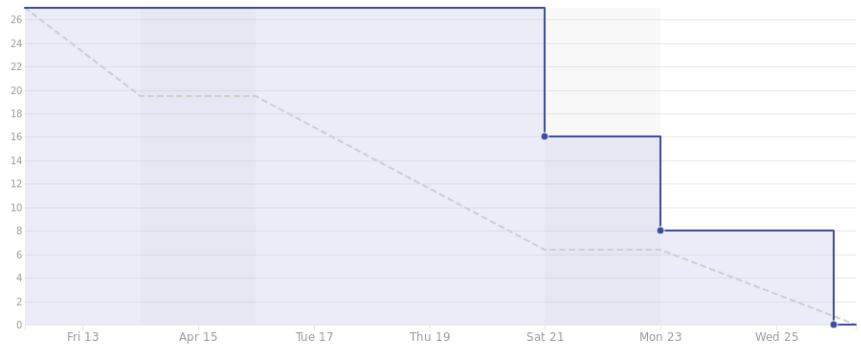

Figura A.2: Burndown del *sprint* 4

Sprint 5

De la comparación del *sprint* anterior se decidió usar MongoDB para el almacenamiento, por lo cual la mayoría de los esfuerzos se centraron en adaptar la aplicación para usar MongoDB en todos sus aspectos: guardar, borrar y coger los datos. También se solucionó un *bug* en la parte de visualización.

[5] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/5?closed=1

La lista de tareas está disponible en Sprint 5[6]. Todas las tareas se completaron a tiempo. El gráfico *burndown* del *sprint* se ve en la figura A.3.

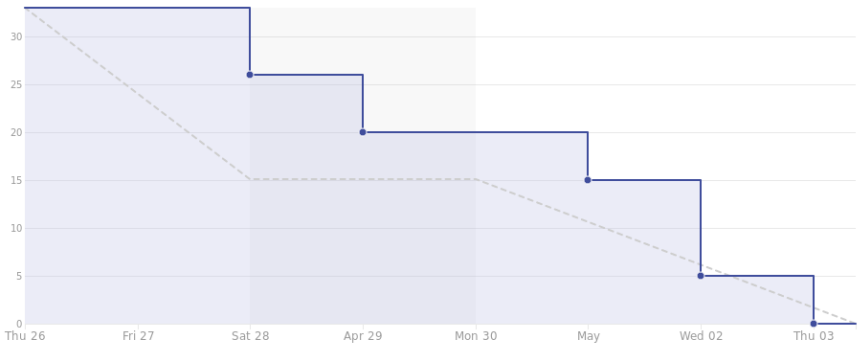

Figura A.3: Burndown del *sprint* 5

Sprint 6

Para este *sprint* se planteó cambiar la forma en la que se guardan los *datasets*, de forma que cuando el sistema de aprendizaje automático esté implementado sea más sencillo pasar los datos para el entrenamiento. También se añadió soporte de comentarios en el *dataset*. Debido a un cambio en como la geóloga organizaba sus datos, se tuvo de adaptar la subida de *datasets* a este cambio. Por último se empezó a añadir controles en la parte de visualización para el procesado de los espectros.

La lista de tareas está disponible en Sprint 6[7]. La tarea de los controles no se completó a tiempo. El gráfico *burndown* del *sprint* se ve en la figura A.4.

[6] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/6?closed=1

[7] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/7?closed=1

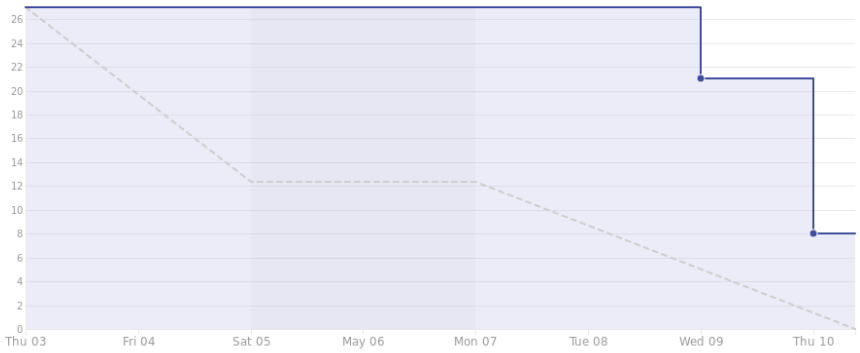

Figura A.4: Burndown del *sprint* 6

Sprint 7

Este *sprint* se centró en completar la tarea del *sprint* anterior, cambiar la parte de visualización para que se muestre en un tabla los ejemplos subidos y sus metadatos, documentar el código y avanzar en la memoria y anexos.

Durante el *sprint*, se vio que la tarea sobre procesamiento de datos era demasiado extensa, dividiéndola en dos partes, la primera que sería añadir los controles para el procesamiento en la interfaz, y una segunda para añadir el código de procesamiento de datos, para realizar en el siguiente *sprint*.

La lista de tareas está disponible en Sprint 7[8]. No dio tiempo a escribir la introducción de la memoria. El gráfico *burndown* del *sprint* se ve en la figura A.5.

[8] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/8?closed=1

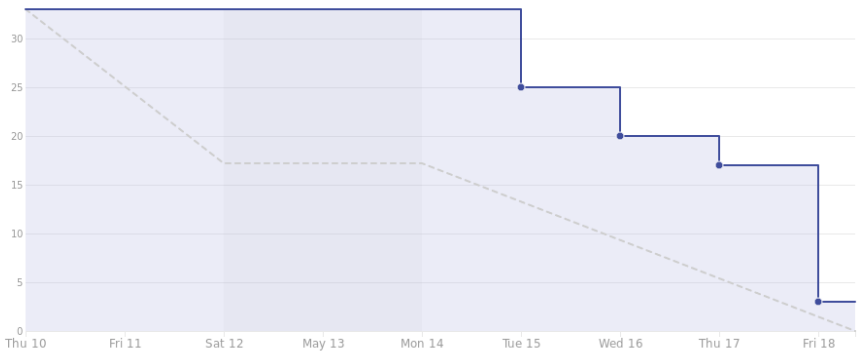

Figura A.5: Burndown del *sprint* 7

Sprint 8

Este *sprint* se centró en completar la tarea del *sprint* anterior, arreglar un *bug* del despliegue, añadir al proyecto el código de procesamiento de datos y enlazarlo a los controles, añadir instrucciones de esta parte y ampliar la planificación temporal con links al repositorio y capturas de los gráficos *burndown*.

Se recuerda que el código de procesamiento es gran parte de los resultados del proyecto previo en las colaboraciones.

La lista de tareas está disponible en Sprint 8[9]. Todas las tareas se completaron a tiempo. El gráfico *burndown* del *sprint* se ve en la figura A.6.

[9] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/9?closed=1

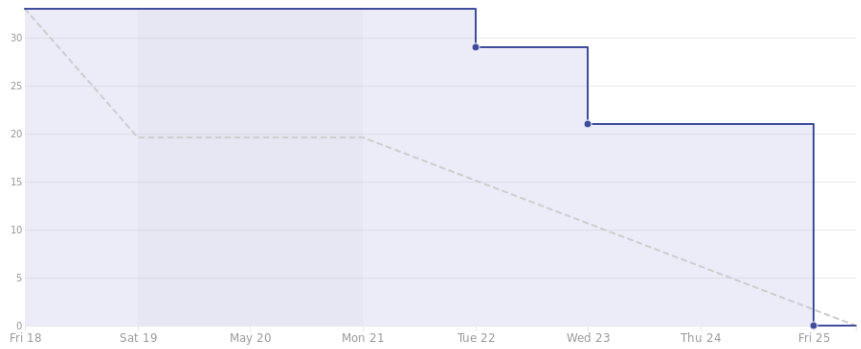

Figura A.6: Burndown del *sprint* 8

Sprint 9

Este *sprint* se centró en añadir la funcionalidad de los espectros individuales: subida, visualización, procesado y borrado. También se trabajó en los objetivos de la memoria.

La lista de tareas está disponible en Sprint 9[10]. Todas las tareas se completaron a tiempo. El gráfico *burndown* del *sprint* se ve en la figura A.7.

Figura A.7: Burndown del *sprint* 9

[10] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/10?closed=1

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

Requisitos funcionales

- RF-1 Control de usuarios: la aplicación debe permitir controlar usuarios.
  - RF-1.1 Integración con Google: la aplicación debe poder hacer uso de cuentas de Google para el control de usuarios.
  - RF-1.2 Inicio de sesión: el usuario debe poder iniciar sesión con una cuenta de Google.
  - RF-1.3 Cierre de sesión: el usuario debe poder cerrar sesión cuando haya terminado.
- RF-2 Gestión de Datasets: la aplicación debe poder almacenar y gestionar datasets de los usuarios.
  - RF-2.1 Creación: el usuario debe poder subir y almacenar un dataset.
  - RF-2.2 Edición: el usuario debe poder modificar un dataset creado anteriormente.
  - RF-2.3 Borrado: el usuario debe poder borrar un dataset creado anteriormente.

    - RF-2.4 Visualización: el usuario debe poder visualizar un dataset creado anteriormente.
- RF-3 Procesamiento: el usuario debe poder procesar los datos de un dataset creado anteriormente.
    - RF-3.1: Procesamiento de dataset: el usuario debe poder realizar operaciones de procesamiento de datos sobre un dataset creado.
    - RF-3.2: Exportación: la aplicación debe poder exportar los datos después del procesado.
- RF-4 Gestión de modelos: la aplicación debe poder trabajar con modelos de aprendizaje automático.
    - RF-4.1 Creación/entrenamiento: el usuario debe poder crear y entrenar un modelo basado en un dataset creado.
    - RF-4.2 Predicción: el usuario debe poder predecir un espectro basado en un modelo creado anteriormente.
    - RF-4.3 Interpretación: la aplicación debe poder dar una interpretación del modelo creado y de la predicción.

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 de despliegue

El despliegue de la aplicación se hace mediante la herramienta Nanobox, cuyo propósito principal es el de facilitar la tarea de despliegue. Para ello combina servicios de virtualización con servidores virtuales privados o VPS, ocupándose de la configuración de la máquina virtual que el proveedor de VPS nos proporcione.

El primer paso es registrarse en alguno de los proveedores disponibles[1]. Para este proyecto se ha elegido Digital Ocean debido a que durante el proyecto se contaba con el "Student Developer Pack" de GitHub, el cual contiene un crédito gratuito de 50$ para esa plataforma.

El siguiente paso es crear una cuenta en Nanobox[2]. Una vez hecho, hay

[1] https://docs.nanobox.io/providers/hosting-accounts/
[2] https://nanobox.io/

que enlazar esta cuenta con la creada anteriormente, desde las opciones de la cuenta en la pestaña "Hosting Accounts", una vez ahí seguir las instrucciones que se muestran en la página.

Después de tener las cuentas preparadas hay que descargar la herramienta Nanobox para el sistema adecuado e instalarla, seguir las instrucciones de la documentación oficial[3] y las del instalador.

Desde la página principal se pulsa en "Lauch New App" para crear la nueva aplicación, seguir las instrucciones en la página. Al terminar se muestran instrucciones para desplegar la aplicación.

Como se puede comprobar, Nanobox cumple el propósito de facilitar el despliegue. Para cualquier otra duda sobre la herramienta toda la documentación oficial está disponible en https://docs.nanobox.io/.

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

[3] https://docs.nanobox.io/install/

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía