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 — 17 de mayo 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. Los *sprints* empiezan en la reunión y acaban en la

siguiente.

Sprints 0-2

Estos *sprints* se centran en investigar las tecnologías que se van a usar en el proyecto, toma de contacto con las herramientas, construcción de prototipos en las tecnologías y desplegar la aplicación básica.

Se comentó con los tutores 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. Por eso se ha desplegado la aplicación desde las etapas iniciales del proyecto.

Sprint 3

Este *sprint* se centra en mejorar visualmente la aplicación y encabezar en desarrollo del proyecto hacia algo definitivo. Se mejora la estructura lógica del proyecto, la visualización de un espectro individual, se añade la subida y visualización de *datasets* y soporte de usuarios mediante cuenta de Google.

Hubo dos tareas que no se llegaron a completar, la subida y visualización y el manual de despliegue definitivo.

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. También se añadió la opción de borrar un *dataset* ya almacenado.

En este *sprint* se han completado todas las tareas.

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.

En este *sprint* se han completado todas las tareas.

Sprint 6

Para este *sprint* se planteó cambiar la forma en la que se guardan los *datasets* para que más adelante sea más sencillo de pasar los datos al sistema de aprendizaje automático (muchos dataframes a un dataframe). También se añadió soporte de comentarios en el *dataset* y se añadió soporte para un fichero excel en la subida que contenga los metadatos del *dataset*. Por último se empezó a añadir controles en la parte de visualización para el procesado de los espectros.

Esta última tarea no se pudo completar.

Sprint 7

Este *sprint* se centró en completar la tarea del *sprint* anterior, mostrar en la parte de visualización una tabla con los ejemplos del *dataset* y sus metadatos, documentar el código y avanzar en la memoria y anexos.

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