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 — 27 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 necesario para 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* se 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 etapas finales, retrasando el despliegue y, a veces, impidiendo desplegar la aplicación.

Se conocía la plataforma Heroku así que fue la primera opción que se probó., 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.

[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

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 e investigar como Heroku lo ofrece, dado que por defecto no lo hace. Al final se decidió cambiar a Digital Ocean con 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.

[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

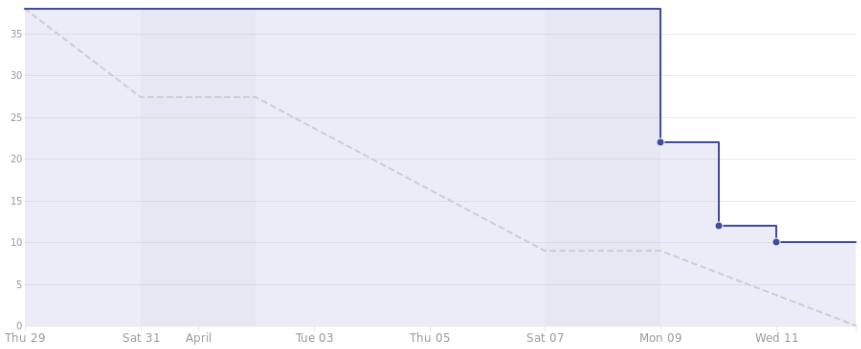

Figura A.1: Burndown del *sprint* 3

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.

Similar a lo ocurrido en el *sprint* anterior, se cambió la estructura de la aplicación, en este *sprint* se estructuró la parte de visualización. También se arreglaron dos errores 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.

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

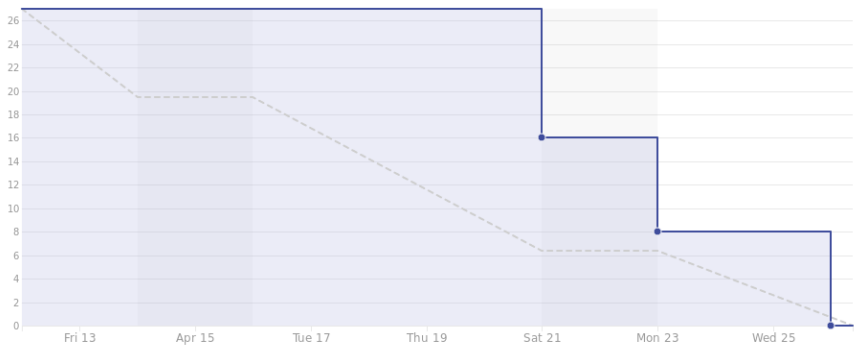

Figura A.2: Burndown del *sprint* 4

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 error en la parte de visualización.

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.

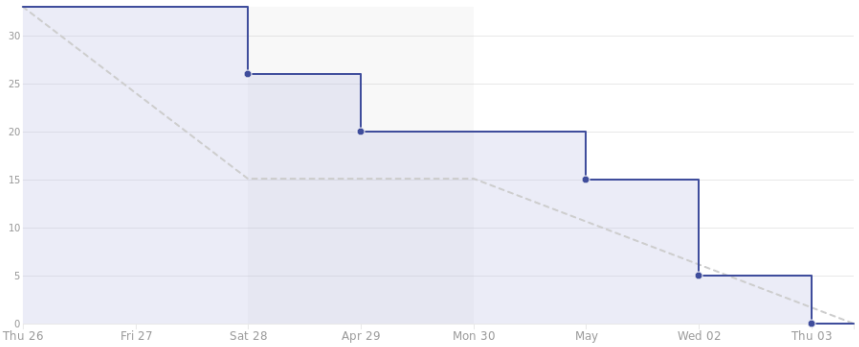

Figura A.3: Burndown del *sprint* 5

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

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 que 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.

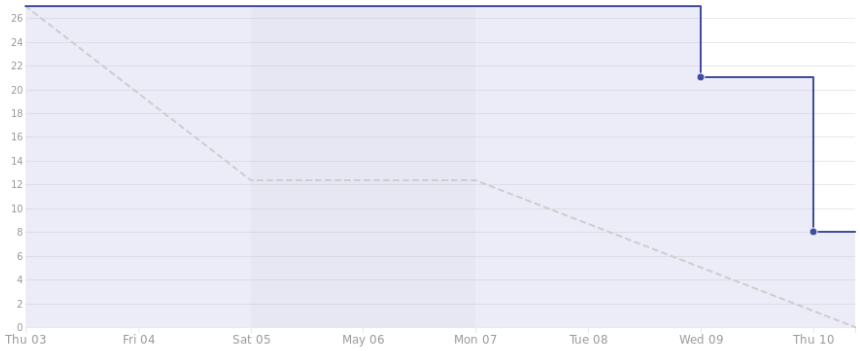

Figura A.4: Burndown del *sprint* 6

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 para 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

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

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.

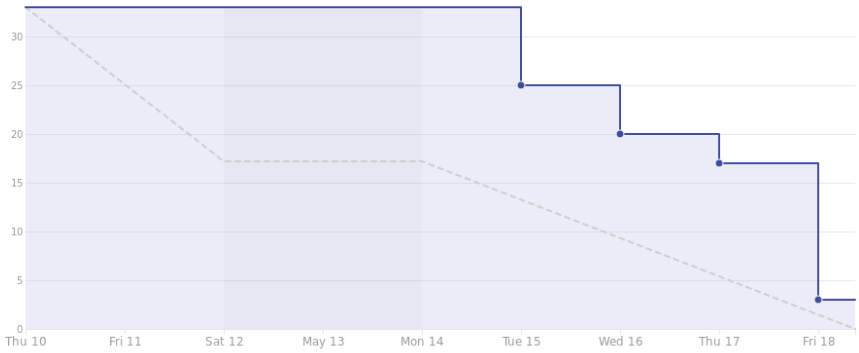

Figura A.5: Burndown del *sprint* 7

Sprint 8

Este *sprint* se centró en completar la tarea del *sprint* anterior, arreglar un error 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.

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

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

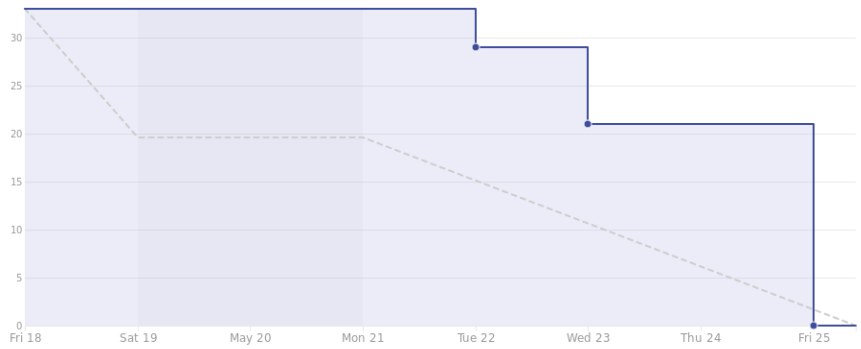

Figura A.6: Burndown del *sprint* 8

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

Sprint 10

Este *sprint* se centró en añadir la funcionalidad de minería de datos respecto a creación de clasificadores y varias mejoras visuales de la interfaz.

Este *sprint* duró dos semanas por desarrollarse a la vez que la época de exámenes. Las tareas relacionadas con la mejora de la web no hubo problemas en completarlas, pero las relacionadas a la creación de clasificadores llevaron más tiempo del esperado y no se pudieron completar a tiempo.

La lista de tareas está disponible en Sprint 10[11]. El gráfico *burndown* del *sprint* se ve en la figura A.8.

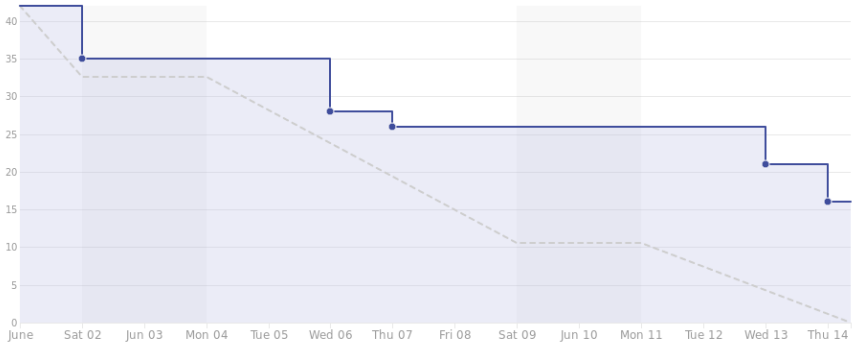

Figura A.8: Burndown del *sprint* 10

Sprint 11

En este *sprint* se completaron las tareas pendientes del *sprint* anterior, además se realizó también la tarea de codificar la predicción de nuevos espectros. También se avanzó bastante en la memoria, aunque no se llegaron a terminar completamente todas las secciones previstas.

Debido a la cercanía de la entrega este *sprint* tuvo más carga de trabajo que los anteriores. Se puede considerar el último *sprint* de desarrollo porque

[11] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/11?closed=1

se terminaron de implementar los objetivos del proyecto.

La lista de tareas está disponible en Sprint 11[12]. El gráfico *burndown* del *sprint* se ve en la figura A.9. Debido a que gran parte de las tareas eran de documentación, se avanzaba en ellas en paralelo y se cerraron al final del *sprint*, de ahí la forma del gráfico.

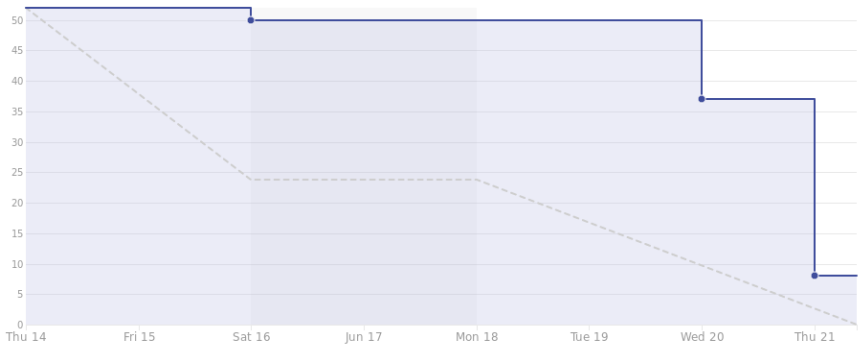

Figura A.9: Burndown del *sprint* 11

A.3. Estudio de viabilidad

Viabilidad económica

Costes de personal

El proyecto se ha llevado a cabo por un desarrollador contratado a tiempo parcial durante 4 meses. Se considera un salario neto de 1000 € mensuales (ver tabla A.1).

La cotización a la seguridad social se ha calculado como horas comunes, según el régimen general de 2018 (28,30 %) [7].

Costes de hardware

En esta sección se enumeran los costes del hardware usado durante el desarrollo.

[12] https://github.com/IvanBeke/TFG-Visor-de-espectros/milestone/12?closed=1

Concepto	Coste
Salario neto	1000 €
Retención IRPF (19 %)	360,53 €
Seguridad social (28,30 %)	537,00 €
Salario bruto (mensual)	1897,53 €
Total 4 meses	7590,12 €

Tabla A.1: Costes de personal

Para el desarrollo se ha usado un ordenador portátil valorado en 800 €, con amortización en 4 años (ver tabla A.2).

$$\frac{800\,€}{4\ años * 12\ meses} = 16{,}67\,€$$

Concepto	Coste	Amortización
Ordenador portátil	800 €	16,67 €
Total 4 meses	66,67 €	

Tabla A.2: Costes de hardware

Costes de software

El proyecto se ha desarrollado usando el sistema Ubuntu, por lo que en este aspecto no habría costes. La licencia de PyCharm Professional supone un gasto de 8,90 € mensuales [6]. La licencia de GitKraken Pro supone un pago único anual de 49$, que aproximadamente corresponde a 42 € [2]. Los costes se resumen en la tabla A.3.

Concepto	Coste
PyCharm Professional (mensual)	8,90 €
GitKraken Pro	42 €
Total 4 meses	77.60 €

Tabla A.3: Costes de software

Costes del servidor

Como servidor se ha elegido un *Droplet* estándar con 2GB de memoria, 1 CPU virtual, 50GB de disco duro y 2TB de transferencia cuyo coste es de 10$ mensuales, aproximadamente 8,60€ mensuales [1]. Los costes se resumen en la tabla A.4.

Concepto	**Coste**
Digital Ocean (mensual)	8,60€
Total 4 meses	34,40€

Tabla A.4: Costes del servidor

Costes totales

En la tabla A.5 se agrupan todos los costes calculados del proyecto, dando el total de 7768,99€.

Concepto	**Coste**
Personal	7590,12€
Hardware	66,67€
Software	77,60€
Servidor	34,60€
Total	7768,99€

Tabla A.5: Costes totales del proyecto

Beneficios

Para obtener beneficios del proyecto se podría plantear añadir a la aplicación las siguientes alternativas:

- Limitaciones de almacenamiento: la cantidad de ficheros que puede subir un usuario estaría limitada por un plan de pago.
- *Freemium* [8]: este modelo funciona ofreciendo unas funcionalidades básicas a todos los usuarios, pero bloqueando algunas a usuarios que no hayan pagado, por ejemplo, la creación de clasificadores.

- Publicidad: podría incluirse en la aplicación publicidad relacionado con el tema.

Para obtener el máximo beneficio estas opciones podrían combinarse.

Viabilidad legal

A lo hora de añadir una licencia al proyecto hay que tener en cuenta a que licencias están sometidas las dependencias. Con la ayuda de la herramienta Requires.io[13] , se han listado las dependencias, versión usada y licencia (ver tabla A.6).

De las dependencias usadas, todas son licencias bastante permisivas que permiten el uso con libertad, por lo que no tenemos que preocuparnos de incompatibilidades con la licencia que se escoja.

Con ayuda de las recomendaciones GNU [4], se ha escodigo la licencia GPL-3.0 [3]. Esta licencia permite la modificación, uso y distribución del software, siempre que esto se haga bajo la misma licencia, se indiquen los cambios y se mencione al autor original.

[13] https://requires.io/

Dependencia	Versión	Licencia
dash	0.21.1	MIT
dash-core-components	0.22.1	MIT
dash-html-components	0.10.1	MIT
dash-table-experiments	0.6.0	MIT
dash_renderer	0.11.3	MIT
Flask	1.0.2	BSD
Flask-Bootstrap	3.3.7.1	BSD
Flask-Dance	0.14.0	MIT
Flask-PyMongo	0.5.1	BSD
Flask-WTF	0.14.2	BSD
gunicorn	19.8.1	MIT
ipython	6.3.1	BSD
numpy	1.14.3	BSD
numpydoc	0.8.0	BSD
pandas	0.23.0	BSD
plotly	2.5.1	MIT
pymongo	3.6.1	Apache License 2.0
pyOpenSSL	17.5.0	Apache License 2.0
PyWavelets	0.5.2	MIT
scikit-learn	0.19.1	BSD 3-Clause
scipy	1.1.0	BSD
Werkzeug	0.14.1	BSD
WTForms	2.2.1	BSD
xlrd	1.1.0	BSD

Tabla A.6: Dependencias del proyecto

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apéndice se describen los objetivos generales de la aplicación y se detallan sus requisitos, tanto funcionales como no funcionales.

B.2. Objetivos generales

- Ofrecer control de usuarios.
- Permitir a los usuarios subir *datasets* y espectros.
- Que los *datasets* y espectros se puedan visualizar.
- Que sobre la visualización se pueda aplicar operaciones de procesamiento.
- Poder entrenar modelos de aprendizaje automático con los *datasets* subidos.
- Poder usar los modelos mencionados anteriormente para predecir nuevos ejemplos subidos.
- Que la aplicación final sea útil para la investigadora, que hace las veces de cliente en este proyecto.

B.3. Catálogo 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 Datasets**: la aplicación debe poder almacenar y gestionar conjuntos de espectros.
  - **RF-2.1 Subida**: el usuario debe poder subir un *dataset*.
  - **RF-2.2 Eliminado**: el usuario debe poder eliminar un *dataset* almacenado.
  - **RF-2.3 Visualización**: el usuario debe poder visualizar un *dataset* almacenado.
- **RF-3 Espectros**: la aplicación debe poder almacenar y gestionar espectros.
  - **RF-3.1 Subida**: el usuario debe poder subir un espectro.
  - **RF-3.2 Eliminado**: el usuario debe poder eliminar un espectro.
  - **RF-3.3 Visualización**: el usuario debe poder visualizar un espectro.
- **RF-4 Procesamiento**: el usuario debe poder aplicar operaciones de preprocesamiento.
  - **RF-4.1: Procesamiento de *dataset***: el usuario debe poder aplicar operaciones de preprocesamiento sobre un *dataset* visualizado.
  - **RF-4.2: Procesamiento de espectro**: el usuario debe poder aplicar operaciones de preprocesamiento sobre un espectro visualizado.

- **RF-5 Minería de datos**: el usuario debe poder usar técnicas de minería de datos.
  - **RF-5.1 Creación**: el usuario debe poder crear modelos personalizados.
  - **RF-5.2 Evaluación**: la aplicación debe ofrecer métricas del modelo entrenado.
  - **RF-5.3 Predicción**: el usuario debe poder usar los modelos que ha creado para predecir nuevos espectros.

Requisitos no funcionales

- **RNF-1 Usabilidad**: la aplicación debe ser intuitiva y fácil de usar.
- **RNF-2 Escalabilidad**: el rendimiento debe poder aumentar al incrementar los recursos.
- **RNF-3 Manteninibilidad**: debe ser sencillo añadir funcionalidad nueva a la aplicación.
- **RNF-4 Compatibilidad**: la aplicación debe poder funcionar en los principales navegadores.
- **RNF-5 Responsividad**: la aplicación debe adaptarse al tamaño de la pantalla.
- **RNF-6 Facilidad de despliegue**: la aplicación debe poder desplegarse en un servidor de forma sencilla.

B.4. Especificación de requisitos

En esta sección se desarrollan los casos de uso de la aplicación. A continuación, se expone el diagrama de casos de uso que los resume.

CU-1	**Iniciar sesión**
Versión	1.0
Autor	Iván Iglesias Cuesta
Requisitos asociados	RF-1, RF-1.1, RF-1.2
Descripción	El usuario inicia sesión en la aplicación.

CU-1	**Iniciar sesión**
Precondición	Navegador web abierto y página de la aplicación cargada.
Acciones	1. Pulsar botón “Iniciar sesión con Google”. 2. Introducir o seleccionar cuenta de Google.
Postcondición	Redirección a la aplicación con sesión iniciada.
Excepciones	▪ Cuenta no existente. ▪ Combinación de nombre y contraseña incorrecta.
Importancia	Alta

Tabla B.1: Iniciar sesión.

CU-2	**Cerrar sesión**
Versión	1.0
Autor	Iván Iglesias Cuesta
Requisitos asociados	RF-1, RF-1.3
Descripción	El usuario cierra sesión en la aplicación.
Precondición	Sesión iniciada en la aplicación.
Acciones	1. Pulsar en el botón cuyo texto es el correo. 2. Pulsar "Cerrar sesión".
Postcondición	Redirección a la aplicación con sesión cerrada.
Excepciones	▪ La sesión no estaba iniciada.
Importancia	Alta

Tabla B.2: Cerrar sesión.

CU-3	**Subir dataset**
Versión	1.0
Autor	Iván Iglesias Cuesta
Requisitos asociados	RF-2, RF-2.1
Descripción	El usuario sube un dataset a la aplicación para su guardado.
Precondición	Sesión iniciada en la aplicación.
Acciones	1. Pulsar en el botón "Subir dataset". 2. Descargar y rellenar la plantilla. 3. Crear un fichero `.zip` con los datos y la plantilla. 4. Rellenar el formulario de subida. 5. Seleccionar el fichero creado. 6. Presionar el botón "Subir".
Postcondición	Redirección a la página de los archivos guardados.
Excepciones	▪ El formato del dataset no es correcto. ▪ Existe un dataset con el mismo nombre.
Importancia	Alta

Tabla B.3: Subir dataset.

CU-4	**Subir espectro**
Versión	1.0
Autor	Iván Iglesias Cuesta
Requisitos asociados	RF-3, RF-3.1
Descripción	El usuario sube un espectro a la aplicación para su guardado.
Precondición	Sesión iniciada en la aplicación.
Acciones	1. Pulsar en el botón “Subir espectro”. 2. Rellenar el formulario de subida. 3. Seleccionar el espectro. 4. Presionar botón “Subir”.
Postcondición	Redirección a la página de los archivos guardados.
Excepciones	▪ El formato del espectro no es correcto. ▪ Existe un espectro con el mismo nombre.
Importancia	Alta

Tabla B.4: Subir espectro.

CU-5	**Visualizar dataset**
Versión	1.0
Autor	Iván Iglesias Cuesta
Requisitos asociados	RF-2, RF-2.3, RF-4, RF-4.1
Descripción	El usuario visualiza un espectro en la aplicación.
Precondición	Sesión iniciada en la aplicación, dataset guardado en la aplicación y estar en la página de "Mis ficheros".
Acciones	1. Escoger un dataset que visualizar. 2. Pulsar el botón "Visualizar" en el dataset escogido.
Postcondición	Se muestra la página de visualización.
Excepciones	▪ No se ha podido cargar el dataset.
Importancia	Alta

Tabla B.5: Visualizar dataset.

CU-6	**Visualizar espectro**
Versión	1.0
Autor	Iván Iglesias Cuesta
Requisitos asociados	RF-3, RF-3.3, RF-4, RF-4.2
Descripción	El usuario visualiza un espectro en la aplicación.
Precondición	Sesión iniciada en la aplicación, espectro guardado en la aplicación y estar en la página de "Mis ficheros".
Acciones	1. Escoger un espectro que visualizar. 2. Pulsar el botón "Visualizar" en el espectro escogido.
Postcondición	Se muestra la página de visualización.
Excepciones	▪ No se ha podido cargar el espectro.
Importancia	Alta

Tabla B.6: Visualizar espectro.

CU-7	**Guardar clasificador**
Versión	1.0
Autor	Iván Iglesias Cuesta
Requisitos asociados	RF-5, RF-5.1, RF-5.2
Descripción	El usuario crear y entrena un modelo usando como datos un dataset guardado.
Precondición	Sesión iniciada en la aplicación, dataset guardado en la aplicación y estar en la página de "Mis ficheros".
Acciones	1. Escoger el dataset que se quiera usar como base. 2. Pulsar el botón "Crear modelo" en el dataset escogido. 3. Seleccionar en el desplegable el modelo a usar. 4. (Opcional) Rellenar el formulario con los parámetros del modelo. 5. Pulsar el botón "Crear y evaluar modelo". 6. Para guardar el clasificador: *a*) Completar el formulario. *b*) Pulsar el botón "Guardar". 7. Para no guardar el clasificador: *a*) Pulsar el botón "Descartar".
Postcondición	Se redirige a los archivos guardados.
Excepciones	▪ Los parámetros del modelo introducidos son erróneos. ▪ Ya existe un clasificador con el nombre introducido.
Importancia	Media

Tabla B.7: Guardar clasificador.

CU-8	**Predecir espectro**
Versión	1.0
Autor	Iván Iglesias Cuesta
Requisitos asociados	RF-5, RF-5.3
Descripción	El usuario usa un clasificador creado para predecir un espectro.
Precondición	Sesión iniciada en la aplicación, dataset guardado en la aplicación, algún clasificador creado y estar en la página de "Mis ficheros".
Acciones	1. Escoger el espectro que se quiera predecir. 2. Pulsar el botón "Predecir" en el espectro escogido. 3. Seleccionar en el desplegable el clasificador a usar. 4. Pulsar el botón "Predecir espectro". 5. Para guardar el clasificador:
Postcondición	Se muestra la predicción.
Excepciones	▪ No se puede cargar el clasificador.
Importancia	Media

Tabla B.8: Predecir espectro.

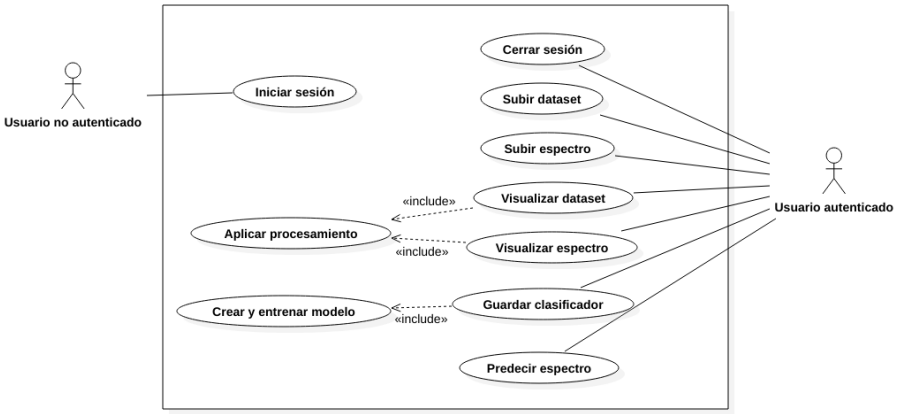

Figura B.1: Diagrama de casos de uso

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación del programador

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

Manual de despliegue

El despliegue de la aplicación se hace con ayuda de la herramienta Nanobox.

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 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.

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

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 se puede recurrir a su documentación oficial [5].

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

[1] Digital Ocean Pricing. https://www.digitalocean.com/pricing/.

[2] GitKraken Pricing. https://www.gitkraken.com/pricing.

[3] Gnu general public license v3.0. https://www.gnu.org/licenses/gpl-3.0.en.html.

[4] How to choose a license for your own work. https://www.gnu.org/licenses/license-recommendations.en.html.

[5] Nanobox documentation. https://docs.nanobox.io/.

[6] PyCharm: JetBrains toolbox Subscription. https://www.jetbrains.com/pycharm/buy/#edition=personal.

[7] Seguridad Social: Bases y tipos de cotización 2018.

[8] Wikipedia. Freemium — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Freemium&oldid=106331717, 2018. [Internet; descargado 27-junio-2018].