UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Impact Factor Oracle
Documentación Técnica

Presentado por Gadea Lucas Pérez
en Universidad de Burgos — 4 de junio de 2023
Tutores: Virginia Ahedo y Álvar Arnaiz

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La planificación temporal es esencial para el éxito de cualquier proyecto, especialmente en el desarrollo de *software*. En esta sección se detallará cómo se ha llevado a cabo el cronograma siguiendo una metodología ágil tipo Scrum, mediante el uso de *sprints*. Se describirán los pasos necesarios para planificar y gestionar de manera eficiente el tiempo y los recursos disponibles, asegurando así el cumplimiento de los objetivos del proyecto en el plazo establecido.

A.2. Metodología Scrum y Kanban

La metodología Scrum es un marco de trabajo ágil que se utiliza para la gestión de proyectos. Se basa en la colaboración entre el equipo de desarrollo y el cliente, así como en la entrega continua de productos funcionales en ciclos cortos de tiempo conocidos como *sprints* [15]. Scrum se centra en la flexibilidad y la adaptabilidad, permitiendo más fácilmente reajustes y los cambios de manera rápida y efectiva ante situaciones inesperadas.

En el contexto del presente trabajo, el uso de la metodología Scrum será beneficioso puesto que nos permitirá tener un enfoque más dinámico y flexible, lo que nos posibilitará adaptarnos a las necesidades del proyecto a medida que avanzamos. Además, Scrum nos proporcionará una mayor transparencia y comunicación, lo que nos capacitará para tomar decisiones informadas y asegurar que el proyecto se entregue a tiempo y con éxito.

Por otro lado, se ha complementado la metodología Scrum con el modelo Kanban[1] propio de la metodología Kanban [14]. Esta combinación nos permitirá visualizar y organizar de manera efectiva las tareas, los hitos y los recursos disponibles, facilitando así la gestión y el seguimiento del proyecto.

Antes de continuar con la planificación del proyecto, es importante familiarizarnos con algunos conceptos clave que son fundamentales para comprender la metodología Scrum. A continuación, se proporcionará una breve descripción de los roles, artefactos y la medición de las tareas en *story points*.

Artefactos en Scrum

En cuanto a los artefactos de Scrum, hay tres principales: el *product backlog*, el *sprint backlog* y el incremento.

1. El *product backlog* es una lista priorizada de todas las funcionalidades, características o mejoras que se desean implementar en el producto final.

2. El *sprint backlog* es una selección de elementos del *backlog* del producto que se abordarán en un *sprint* específico.

3. Y finalmente, el incremento es el resultado tangible y potencialmente entregable al final de cada *sprint*, que incorpora nuevas funcionalidades o mejoras al producto.

Roles en Scrum

En Scrum, existen tres roles principales: el Scrum Master, el Product Owner y el Equipo de Desarrollo.

1. El Scrum Master es responsable de garantizar que el equipo siga las prácticas y principios de Scrum, eliminando cualquier obstáculo que pueda afectar a la productividad.

2. El Product Owner es el encargado de definir y priorizar los elementos del *product backlog*, asegurándose de que se cumplan las necesidades del cliente.

[1]Este modelo se refiere comúnmente a un lienzo o plantilla visual utilizada para estructurar y desarrollar ideas de negocio.

3. Por último, el Equipo de Desarrollo es responsable de la implementación de las tareas y la entrega del producto.

Dado que en este caso el equipo está conformado solamente por los dos tutores y la alumna, es importante adaptar los roles y las prácticas de Scrum para que se ajusten a esta situación específica. Se pueden asignar los roles de Scrum Master y Product Owner a cada uno de los tutores y, el rol de Equipo de Desarrollo, correspondería a la alumna.

Medición de las tareas

En cuanto a la medición de las tareas en *story points*, Scrum utiliza esta técnica para estimar el esfuerzo relativo requerido para completar una tarea. Los *story points* son una medida abstracta que no se relaciona directamente con el tiempo, sino con la complejidad y el esfuerzo necesario para completar una tarea en comparación con otras. Esta medida ayuda al equipo a planificar su capacidad de trabajo en cada *sprint* y a realizar estimaciones más precisas en función de su experiencia previa.

A.3. Herramienta

ZenHub [9] es una herramienta de gestión de proyectos que se utiliza en muchos entornos empresariales para mejorar la eficiencia en el seguimiento y administración de tareas. Con ZenHub, es posible planificar y visualizar fácilmente las actividades de un proyecto, monitorizar el progreso de cada tarea y hacer un seguimiento de los plazos de entrega. También ofrece funciones útiles como integración con GitHub, herramientas de informes y análisis de datos, y seguimiento de errores y problemas.

Además, ZenHub utiliza la secuencia de Fibonacci para medir el peso de las tareas en *story points* (que es una escala comúnmente utilizada en el contexto de la metodología ágil). La secuencia de Fibonacci es una serie matemática en la que cada número es la suma de los dos anteriores: 1, 2, 3, 5, 8, 13, 21, etc. La razón del uso de la secuencia de Fibonacci para los *story points* es evitar la tendencia a dar estimaciones precisas y lineales. De esta forma, se busca que los equipos se enfoquen más en la relativa complejidad y esfuerzo de una tarea en lugar de estimaciones exactas de tiempo.

A.4. Planificación temporal

El presente proyecto comienza en septiembre y se extiende hasta junio. Durante este período de tiempo, es importante asegurarnos de que todas las tareas e hitos estén claramente definidos.

Para lograr esto, se han realizado reuniones periódicas para discutir el progreso del proyecto y asegurarnos de que estamos en el camino correcto. Además, hemos implementado *sprints* regulares (de entre una y dos semanas de duración) para asegurarnos de que estamos avanzando de manera constante y cumpliendo con nuestras metas a tiempo.

Con esta planificación temporal sólida, estamos seguros de que podremos completar el proyecto a tiempo y cumplir con los objetivos establecidos. Sin embargo, es importante ser flexibles y estar preparados para hacer ajustes según sea necesario a medida que avanzamos en el proyecto, puesto que se realiza al mismo tiempo que trascurre el curso académico.

En las siguientes tablas se recogen los distintos *sprints* junto con su duración, objetivos y tareas. Además, se adjuntará el gráfico *burndown*[2] de cada uno de los *sprints*.

Sprint 1

Sprt.	Duración	Objetivos	Tareas
1	19/09/2022 03/10/2022	Comenzar el desarrollo del proyecto	▪ Elegir un modelo de referencias bibliográficas. ▪ Definición de conceptos.

Tabla A.1: Sprint 1

[2]Gráfico *burndown*: se trata de una herramienta comúnmente utilizada en la gestión de proyectos ágiles para realizar un seguimiento de un *sprint*. Este gráfico muestra la cantidad de trabajo que queda por hacer en relación con el tiempo disponible para hacerlo. A medida que se completa el trabajo, la línea del gráfico debería disminuir hasta alcanzar cero en el momento en que finaliza el *sprint*.

Figura A.1: Sprint 1: Burndown chart

Sprint 2

Sprt.	Duración	Objetivos	Tareas
2	03/10/2022 17/10/2022	Tareas de investigación y documentación	▪ Búsqueda de antecedentes. ▪ Familiarizarse con el procesador de textos LaTeX.

Tabla A.2: Sprint 2

Figura A.2: Sprint 2: Burndown chart

Sprint 3

Sprt.	Duración	Objetivos	Tareas
3	17/10/2022 14/11/2022	Investigación y creación de un prototipo inicial	▪ Búsqueda de información sobre MIAR. ▪ Documentación sobre el JCR y el índice de impacto. ▪ Prototipo que permita la búsqueda de artículos en GS. ▪ Extracción de datos: ¿qué datos se pueden extraer? ▪ Reflexión sobre el diseño de la BBDD. ▪ Prueba del prototipo.

Tabla A.3: Sprint 3

Figura A.3: Sprint 3: Burndown chart

Sprint 4

Sprt.	Duración	Objetivos	Tareas
4	14/11/2022 28/11/2022	Base de datos y prototipo	▪ Creación de la base de datos en MariaDB. ▪ Prueba de la BBDD. ▪ Mejoras del prototipo.

Tabla A.4: Sprint 4

NOTA

Como se puede observar no se ha añadido un gráfico *burn-down* en este *sprint*. Esto se debe a que las tareas no fueron cerradas a tiempo. Sin embargo, es preciso aclarar que se trata de un error en la configuración de las tareas. Si bien todas las tareas fueron completadas a tiempo, fue preciso borrarlas en ZenHub y volverlas a crear más adelante cuando el *sprint* ya había terminado debido a que estaban inicialmente mal configuradas.

Sprint 5

Sprt.	Duración	Objetivos	Tareas
5	28/11/2022 12/12/2022	Base de datos y prototipo	▪ Se muda la BBDD a Postgres. ▪ Pruebas e investigación para extraer el DOI.

Tabla A.5: Sprint 5

Figura A.4: Sprint 5: Burndown chart

Sprint 6

Sprt.	Duración	Objetivos	Tareas
6	09/01/2023 23/01/2023	Documentación y obtención del DOI	▪ Obtención del DOI a través de Crossref. ▪ Avance de la memoria y los anexos. ▪ Mejora de la BBDD.

Tabla A.6: Sprint 6

Figura A.5: Sprint 6: Burndown chart

Sprint 7

Sprt.	Duración	Objetivos	Tareas
7	06/02/2023 20/02/2023	Preparación para la conexión remota a los servidores de la universidad.	▪ Conectarse a la VPN de la universidad. ▪ Crear un servicio virtualizado en Miniconda. ▪ Mejorar la BBDD.

Tabla A.7: Sprint 7

Figura A.6: Sprint 7: Burndown chart

Sprint 8

Sprt.	Duración	Objetivos	Tareas
8	20/02/2023 06/03/2023	Conexión a los servidores y mejora de los prototipos. Avances en la API y documentación.	▪ Mejorar el prototipo. ▪ Prototipo con varios hilos de ejecución (usar servidores de la universidad). ▪ Nuevo prototipo con Selenium. ▪ Sistema inteligente de requests. ▪ Revisión de PoP. ▪ API: Autentificación y back-end.

Tabla A.8: Sprint 8

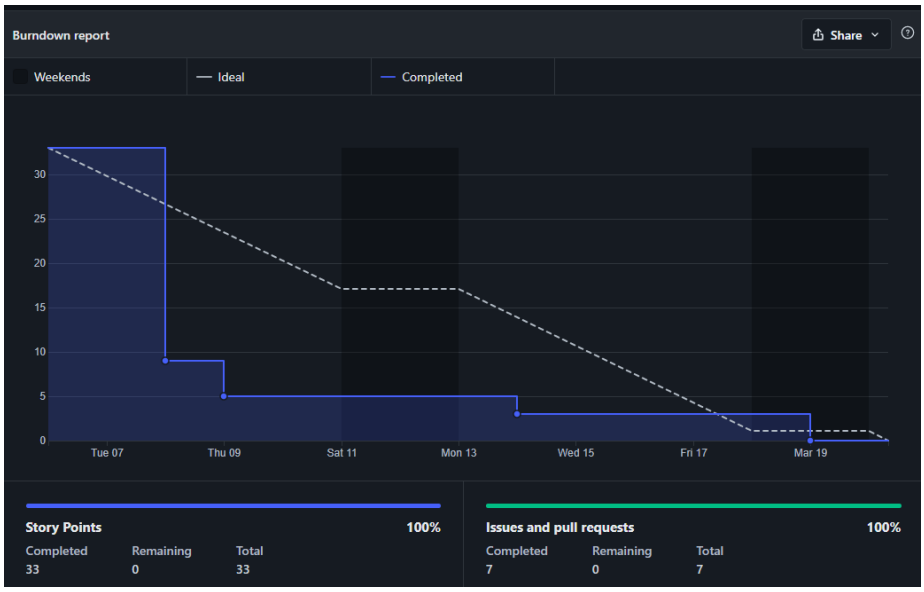

Figura A.7: Sprint 8: Burndown chart

Sprint 9

Sprt.	Duración	Objetivos	Tareas
9	06/03/2023 20/03/2023	Finalización del prototipo de extracción de Crossref.	▪ Plan de prueba para Crossref. ▪ Prototipo de Crossref. ▪ Comprobar límite de llamadas a Crossref. ▪ Crear entorno virtualizado. ▪ Fichero requirements. ▪ Licencia Crossref para Cited-by. ▪ Cálculo del IF.

Tabla A.9: Sprint 9

Figura A.8: Sprint 9: Burndown chart

Sprint 10

Sprt.	Duración	Objetivos	Tareas
10	03/04/2023 17/04/2023	Procesamiento de los datos, creación de los modelos de predicción y diseño de interfaces.	▪ Normalizar listados JCR. ▪ Comparar JCR obtenidos con los valores reales. ▪ Modelos de Crossvalidation. ▪ Diseño de las interfaces de la aplicación. ▪ Diagramas UML de la API.

Tabla A.10: Sprint 10

Figura A.9: Sprint 10: Burndown chart

Sprint 11

Sprt.	Duración	Objetivos	Tareas
9	24/04/2023 15/05/2023	Creación de un prototipo de aplicación web y exportación de los modelos de predicción	▪ Prototipo de API ▪ API con manejo de errores ▪ API con conexión a BBDD ▪ API con interfaz de prueba ▪ API con internacionalización ▪ Generar ficheros binarios para los modelos de predicción.

Tabla A.11: Sprint 11

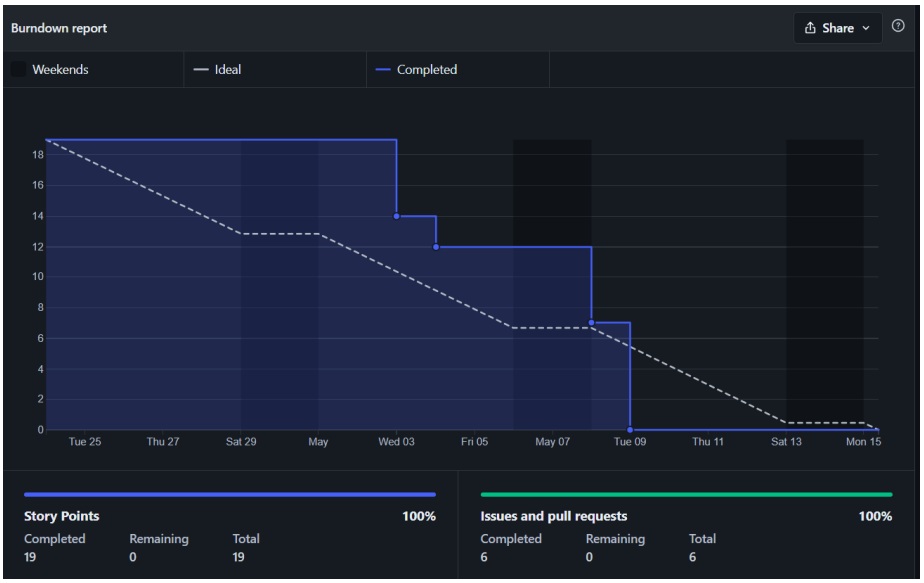

Figura A.10: Sprint 11: Burndown chart

Sprint 12

Sprt.	Duración	Objetivos	Tareas
9	15/05/2023 22/05/2023	Aplicación con todas las funcionalidades básicas en base al prototipo y corrección de la carga de información en la BBDD.	▪ Funcionalidades básicas de la API ▪ Cargar datos correctos

Tabla A.12: Sprint 12

Figura A.11: Sprint 12: Burndown chart

Sprint 13

Sprt.	Duración	Objetivos	Tareas
9	15/05/2023 22/05/2023	Lanzar la aplicación en Heroku y mejora de funcionalidades.	▪ Serie temporal ▪ Añadir el cuartil a las revistas ▪ Configurar API en Heroku ▪ Configurar BBDD en Heroku

Tabla A.13: Sprint 13

Figura A.12: Sprint 13: Burndown chart

Sprint 14

...

Overview

Finalmente, se incluirá un gráfico acumulativo[3] (ver figura A.13) que muestra la cantidad total de trabajo realizado de forma más resumida.

[3]Gráfico acumulativo: en este gráfico, el eje vertical representa la cantidad total de trabajo completado, mientras que el eje horizontal representa el tiempo transcurrido durante el proyecto. A medida que se completa el trabajo, la línea del gráfico se eleva, lo que indica el progreso general del proyecto

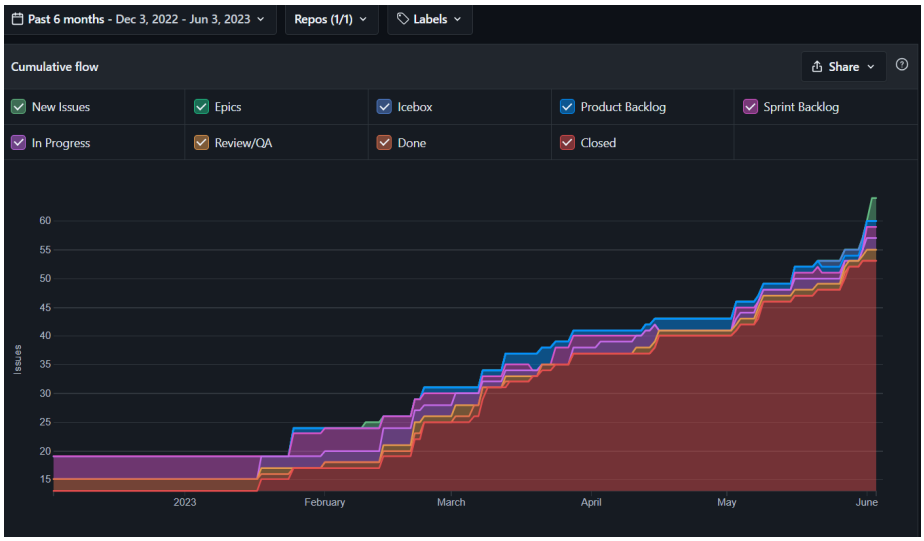

Figura A.13: Gráfico acumulativo de todo el proyecto

A.5. Estudio de viabilidad

El estudio de viabilidad es un componente esencial en el proceso de evaluación de cualquier proyecto o iniciativa. En el contexto de nuestra investigación, se ha desarrollado una aplicación *open source* destinada a beneficiar y servir a la comunidad científica.

Dado que la intención no es obtener beneficios económicos directos de este proyecto, se considerará un enfoque empresarial realista para asegurar la viabilidad y sostenibilidad a largo plazo de nuestro proyecto.

En las siguientes secciones, se analizarán en detalle los aspectos de viabilidad económica (sección A.5) y legal (sección A.5), con el objetivo de establecer un marco sólido que asegure la continuidad y éxito del proyecto.

Viabilidad económica

En primer lugar, la viabilidad económica se refiere a la evaluación de los recursos financieros necesarios para llevar a cabo el proyecto y asegurar su sostenibilidad a largo plazo. Aunque nuestra aplicación no tiene como objetivo obtener ganancias, es importante considerar los costos asociados al desarrollo, mantenimiento y mejora continua de la plataforma. Además, se deben analizar posibles fuentes de financiamiento, como subvenciones, donaciones o colaboraciones con instituciones interesadas en respaldar este tipo de iniciativas científicas abiertas.

Algunos aspectos relevantes para poder hacer estos cálculos son los siguientes:

- La empresa se situaría en España, lo que implica tener en cuenta las regulaciones y requisitos legales y fiscales del país.
- El proyecto tiene una duración estimada de 8 meses y una semana, desde octubre hasta la segunda semana de junio. Esto equivale aproximadamente a 34 semanas, considerando una duración promedio de 4 semanas por mes.
- El equipo encargado del proyecto está formado por una desarrolladora (la alumna), así como por el *Product Owner* y el *Scrum Master*, quienes actúan como tutores de la alumna en el desarrollo del proyecto.

Para todo esto, es fundamental considerar los costes y beneficios asociados al proyecto. Los **costes** se refieren a los desembolsos monetarios necesarios para llevar a cabo la iniciativa, incluyendo los gastos de inversión, los costes operativos y los costes de mantenimiento [12]. Por otro lado, los **beneficios** se refieren a las ganancias económicas que se esperan obtener como resultado de la implementación del proyecto. Estos beneficios pueden incluir ingresos generados por la venta de productos o servicios, ahorros en costes operativos, incremento en la productividad, entre otros [12].

Costes

Comenzaremos por calcular los costes totales de la supuesta empresa.

1. Costes de los empleados.

   Se procederá al cálculo del salario bruto de cada integrante del equipo de acuerdo con la normativa laboral vigente en España para un proyecto de desarrollo de *software*. El salario bruto se determina considerando factores como la categoría profesional y la experiencia de cada empleado.

   a) Desarrollador: en este caso, la alumna es considerada una programadora *junior*. En España, un programador *junior* (con menos de 3 años de experiencia laboral) puede esperar un salario medio de alrededor de 19.700 € brutos por año [10]. Además, según el artículo 34 del Estatuto de los Trabajadores, la duración fijada en convenio colectivo de la jornada laboral es un máximo de 40

horas de trabajo efectivo [1]. Ahora, para calcular el salario bruto mensual, se puede utilizar la siguiente fórmula:

$$\text{Salario Bruto Mensual} = \frac{\text{Salario Bruto Anual}}{12 \text{ meses}}$$

Sustituyendo el valor del salario bruto anual, obtenemos:

$$\text{Salario Bruto Mensual} = \frac{19{,}700\,€}{12\,\text{meses}} \approx 1{,}641{,}67\,€$$

b) Product Owner: el salario bruto promedio anual para un Product Owner en España es de 41.866 € [2]. Utilizando la misma fórmula para calcular el salario bruto mensual, obtenemos:

$$\text{Salario Bruto Mensual} = \frac{41{,}866\,€}{12\,\text{meses}} \approx 3{,}488,83\,€$$

c) Scrum Master: el salario bruto promedio anual para un Scrum Master en España es de 40.626 € [3]. Utilizando la misma fórmula para calcular el salario bruto mensual, obtenemos:

$$\text{Salario Bruto Mensual} = \frac{40{,}626\,€}{12\,\text{meses}} \approx 3{,}385,50\,€$$

Una vez obtenido el salario bruto, se deben aplicar los impuestos correspondientes de acuerdo con la legislación fiscal vigente. Los impuestos a tener en cuenta incluyen las contribuciones a la Seguridad Social [8], el impuesto sobre la renta[4] (IRPF) [7] y otros impuestos relacionados con la contratación laboral. La información detallada sobre los impuestos y las obligaciones legales se encuentra en los documentos oficiales y normativas correspondientes proporcionados por las autoridades fiscales y laborales citados anteriormente.

De forma resumida, Para el año 2023, se han establecido los siguientes porcentajes[5], los cuales se detallan en la Tabla A.14 y Tabla A.15.

Con base en los cálculos previos de los salarios brutos y los impuestos aplicados, se puede determinar el coste total de la empresa en cuanto a los empleados.

[4]En cuanto al IRPF, el porcentaje aplicado puede variar según el nivel de ingresos y la situación fiscal personal. Según la Agencia Tributaria de España, se pueden encontrar las tablas y los porcentajes correspondientes en su documentación oficial [4].

[5]El gobierno pone a disposición la plataforma ipyme [5], que proporciona información valiosa para facilitar el cálculo de los tramites empresariales.

Concepto	**Descripción**	**Porcentaje**
Cuota Contingencias Comunes	Gastos comunes en el ámbito laboral	23.60 %
Cuota Formación Profesional	Inversión en formación y capacitación	0.60 %
Desempleo	Prestaciones por desempleo	5.50 %
Accidentes de Trabajo	Prevención y compensación por accidentes laborales	1.50 %
FOGASA	Fondo de Garantía Salarial	0.20 %

Tabla A.14: Conceptos de Seguridad Social

Concepto	**Descripción**	**Porcentaje**
IRPF	Impuesto sobre la Renta de las Personas Físicas	Variable (retención en nómina)
IVA	Impuesto sobre el Valor Añadido	21.00 % (tipo general)

Tabla A.15: Otros conceptos

a) Desarrollador: Dado que el salario bruto mensual de la desarrolladora es de aproximadamente 1.641,67 €, se aplicarán los porcentajes correspondientes de los conceptos de la Seguridad Social según lo establecido en la Tabla A.14. El total de los conceptos de Seguridad Social se calcula sumando los porcentajes aplicados al salario bruto mensual:

$$\text{Total SS} = 0{,}236 + 0{,}006 + 0{,}055 + 0{,}015 + 0{,}002 \approx 0{,}314\,€$$

El IRPF es un impuesto variable y su porcentaje de retención en nómina depende del nivel de ingresos y la situación fiscal personal. Por lo tanto, no se tendrá en cuenta para el resultado de este cálculo.

Por último, el gasto de la empresa se calcula dividiendo el salario bruto mensual entre 1 menos el total de la Seguridad Social. Sustituyendo los valores correspondientes, obtenemos:

$$\text{Gasto mensual} = \frac{1641,67\,€}{1-0,314} \approx 2393,10\,€$$

b) Product Owner: Como el salario bruto anual para el Product Owner es de 3.488,83 €, aplicando los porcentajes de la Tabla A.14, el gasto total de la empresa para el Product Owner sería:

$$\text{Gasto mensual} = \frac{3488,83\,€}{1-0,314} \approx 5085,75\,€$$

c) Scrum Master: De manera similar, para el Scrum Master, partimos de un salario bruto mensual aproximado de 3.385,50 €. Aplicando los porcentajes de la Tabla A.14, el gasto total de la empresa para el Scrum Master sería:

$$\text{Gasto mensual} = \frac{3385,50\,€}{1-0,314} \approx 4935,13\,€$$

A modo de resumen, se agrupa el resultado de todos los cálculos en la Tabla A.16.

Empleado	**Salario Bruto Mensual**	**Gasto Mensual**
Desarrollador	1.641,67 €	2.393,10 €
Product Owner	3.488,83 €	5.085,75 €
Scrum Master	3.385,50 €	4.937,58 €
Total Proyecto	8.516,00 €	12.416,43 €

Tabla A.16: Resumen de costes de los empleados

2. Costes de *hardware*.

   Para calcular los costes de *hardware*, consideraremos únicamente el portátil de la alumna. Se trata de un MSI, modelo Prestige 15 A12UD-053XES con un procesador 12th Gen Intel(R) Core(TM) i7-1280P 2.00 GHz y una RAM de 32 GB. Este portatil se encuentra en el mercado a un precio de 1211,80 €.

   El coste de amortización del portátil de la alumna se basará en su vida útil estimada. Supongamos que el portátil tiene una vida útil de 4 años. Para calcular el coste anual de amortización, dividiremos el coste del portátil entre el número de años de vida útil:

$$\text{Coste anual de amortización} = \frac{1211,80\,€}{4\,\text{años}} \approx 302,95\,€/\text{año}$$

3. Costes de *software*.

   Comenzaremos por contemplar las licencias correspondientes a los sistemas operativos usados.

   - Windows 10 Pro: el valor aproximado de la licencia de Windows 10 Pro versión 22H2 es de 200 €. Al igual que en el caso del *hardware*, consideraremos una vida útil de 4 años y aplicaremos la amortización correspondiente.

     $$\text{Coste anual de amortización} = \frac{200\,€}{4\,\text{años}} \approx 50\,€/\text{año}$$

   - Linux Debian: el portatil tiene una segunda partición con un sistema Debian que, debido a su filosofía de «*free software*» [6], es gratuito y no supone costes adicionales.

   El resto del *software* utilizado está disponible de forma gratuita o puede ser usado gracias a las licencias de estudiantes que ofrece la universidad, por lo que no ha generado costes adicionales. Suponiendo que finaliza el plazo de estas licencias gratuitas, debemos considerar los siguientes costes:

   - Heroku: se trata de un servidor en la nube donde lanzar la aplicación desarrollada. Actualmente, se está utilizando una licencia gratuita de Heroku, gracias al *Student Pack* de GitHub proporcionado por la universidad. Sin embargo, una vez que la licencia gratuita expire, se deberá considerar el coste mínimo de los planes de Heroku. A saber, el coste mínimo para el Dyno básico (web Gunicorn API.paperrank.app:app) es de 7.00 $ (que equivale aproximadamente a 6.53 € mensuales). Además, será necesario tener en cuenta el coste adicional para disponer (al menos) de la versión mini de Heroku Postgres. Este coste es de 5.00 $ (4.66 € mensuales).
   - GitHub y extensión de ZenHub: La licencia de GitHub (versión empresarial) tiene un coste de 17.95 € mensuales. Además, la extensión de ZenHub tiene un coste adicional de 12 €.

Concepto	**Descripción**	**Coste Mensual**
Windows 10 Pro	Licencia de software	4.17 €
Heroku Dyno web	Servidor en la nube	6.53 €
Heroku Postgres	Servidor en la nube	4.66 €
GitHub Enterprise	Licencia de software	17.95 €
ZenHub	Extensión para GitHub	12.00 €
Total (mensual)		**45.31 €**

Tabla A.17: Costes de software mensuales

Se adjunta una tabla resumen con los costes de *software* estimados de forma mensual, considerando los elementos mencionados anteriormente (ver Tabla A.17).

Para obtener los gastos totales del proyecto, simplemente se multiplica el costo mensual por el número de meses:

$$\text{Gastos totales} = 45{,}31€ \times 8{,}14 \text{ meses} \approx 368{,}83€$$

Finalmente, aunque durante el desarrollo del proyecto se ha decidido no utilizar las licencias de las APIs de Google Scholar, Web of Science y Scopus, es importante tener en cuenta que en futuras investigaciones o aplicaciones más complejas, podría considerarse la necesidad de acceder a estas fuentes de información científica y académica. Estas APIs proporcionan un acceso especializado a una amplia gama de datos y recursos, lo cual podría requerir una inversión adicional en forma de licencias y sus correspondientes costos. Es recomendable evaluar detenidamente los requisitos del proyecto y el presupuesto disponible antes de tomar la decisión de adquirir estas licencias, en caso de que se considere necesario para la viabilidad y calidad de la aplicación.

Para visualizar los costes de forma clara, se ha elaborado una última tabla resumen con todos los costes finales (ver Tabla A.18).

Beneficios

Aunque la aplicación no tiene fines lucrativos y estará disponible de forma gratuita para la comunidad científica, es importante considerar una

Concepto	**Descripción**	**Coste Total**
Empleados	Costes totales de empleados	12.416,43 €
Hardware	Costes por el portatil	302,70 €
Software	Costes totales por licencias	402,75 €
Coste Total del Proyecto		13.121,88 €

Tabla A.18: Resumen de costes del proyecto

forma hipotética de generar ingresos para sostener el proyecto. Una opción sería incluir anuncios en la página web de la aplicación.

La inclusión de anuncios publicitarios podría proporcionar una fuente potencial de ingresos. Sin embargo, es fundamental analizar las implicaciones y opciones disponibles antes de tomar una decisión. En primer lugar, se debe tener en cuenta que la incorporación de anuncios puede afectar la experiencia del usuario y la usabilidad de la aplicación. Es importante encontrar un equilibrio entre la generación de ingresos y la satisfacción de los usuarios.

Una opción es utilizar redes de publicidad en línea, como Google AdSense, que permiten mostrar anuncios relevantes y personalizados en la página web. Estas redes se encargan de gestionar los anuncios y los ingresos generados por ellos, ofreciendo a los propietarios de sitios web una parte de los ingresos publicitarios.

Otra posibilidad es buscar acuerdos directos con empresas o instituciones que deseen promocionar sus productos o servicios en la aplicación. Esto podría implicar la negociación de contratos publicitarios personalizados y la integración de anuncios específicos en la página web.

Es importante tener en cuenta que la inclusión de anuncios publicitarios requerirá un trabajo adicional en términos de diseño, implementación y gestión de la publicidad. Se deberá evaluar la viabilidad económica de los ingresos generados por los anuncios en comparación con los costes asociados. Es fundamental también considerar las implicaciones éticas y de privacidad asociadas con la publicidad en línea. Se deben seguir las políticas y regulaciones vigentes para garantizar la transparencia, el respeto de la privacidad de los usuarios y la integridad del proyecto.

Viabilidad legal

Por otro lado, la viabilidad legal implica evaluar las cuestiones jurídicas y normativas que puedan afectar la operación y distribución de la aplicación *open source*. A pesar de que nuestro proyecto está destinado a ser utilizado por la comunidad científica en general, es necesario asegurarnos de cumplir con las regulaciones y requisitos legales relacionados con la privacidad, protección de datos y derechos de propiedad intelectual.

Web Scraping

La viabilidad legal de la aplicación es un aspecto crucial a considerar, especialmente cuando se utiliza una técnica polémica como *web scraping* para obtener información de fuentes externas. Es importante comprender los aspectos legales y los posibles problemas que pueden surgir en relación con los permisos y derechos de propiedad intelectual (especialmente en los sitios web que requieren registro de usuario).

En el caso específico de la aplicación, se realiza *web scraping* utilizando la API de Crossref, por lo que el primer paso es revisar y comprender los términos de servicio y las políticas de esta API. Estos documentos pueden establecer las condiciones específicas para el uso de la API, incluyendo cualquier restricción o requisito legal. Es recomendable revisar las políticas de Crossref regularmente para asegurarse de cumplir con los términos actualizados.

Según la página oficial, Crossref ofrece su API de REST públicamente [16], lo que permite a los usuarios acceder y utilizar los metadatos depositados por sus miembros. Crossref proporciona dichos metadatos sin restricciones de *copyright* y los usuarios pueden utilizarlos para cualquier propósito. En este sentido, nuestro proyecto no supone ningún problema legal (lo cual no sucedía con los sitios de Google Scholar, Scopus y Web of Science).

Sin embargo, es importante tener en cuenta que algunos *abstracts* contenidos en los metadatos pueden estar sujetos a derechos de autor por parte de los editores o autores originales. Por lo tanto, es esencial que al utilizar la API de Crossref se respeten los derechos de autor y se cumpla con los términos y condiciones establecidos por Crossref. Además, Crossref ha implementado un sistema de «piscinas» separadas para usuarios «educados» que incluyen información de contacto y respetan ciertas prácticas de acceso.

Licencias de software

Dado que en el presente proyecto se han utilizado una variedad de herramientas y paquetes con diferentes licencias (ver Tabla A.19), es fundamental tenerlas en cuenta al elegir una licencia para nuestro *software*. A continuación, se enumeran y explican las opciones de licencia más relevantes:

- **Licencia MIT**: esta licencia es ampliamente utilizada y permite el uso, copia, modificación y distribución del *software*, tanto en proyectos comerciales como no comerciales, con la condición de mantener el aviso de derechos de autor y la exención de responsabilidad. Es una opción flexible y abierta que fomenta la colaboración y la adopción del *software*.
- **Licencia BSD**: Las licencias BSD son flexibles y permiten la redistribución y el uso en proyectos comerciales y no comerciales. Estas licencias suelen requerir el mantenimiento del aviso de derechos de autor y la exención de responsabilidad. También pueden incluir cláusulas adicionales que limitan la utilización del nombre del titular de los derechos de autor en la promoción del *software* derivado.
- **Licencia Apache**: La Licencia Apache es una licencia de código abierto que permite el uso, modificación y distribución del *software* bajo ciertas condiciones. Estas condiciones incluyen el mantenimiento del aviso de derechos de autor, la exención de responsabilidad y la necesidad de proporcionar una copia de la licencia en cualquier distribución del *software*.
- **Licencia GNU**: La Licencia Pública General de GNU (GPL) es una licencia de *software* libre que garantiza a los usuarios la libertad de utilizar, estudiar, modificar y distribuir el *software*. La GPL también exige que cualquier *software* derivado se distribuya bajo los términos de la GPL, lo que garantiza que las mejoras y modificaciones sigan siendo de código abierto.

Después de considerar cuidadosamente las licencias de las herramientas utilizadas en nuestro proyecto y dada la naturaleza del mismo, se ha considerado que la más adecuada es la Licencia GNU. Esta licencia garantiza la libertad del *software* y el mantenimiento de su código abierto. Esta opción es ampliamente utilizada y respaldada por la comunidad del *software* libre y de código abierto.

Paquete	**Versión**	**Licencia**
Babel	2.12.1	MIT License
beautifulsoup4	4.12.2	MIT License
crossrefapi	1.5.0	CC Attribution 4.0 International License
cryptography	40.0.2	OSI Approved: Apache SW License, BSD License
Flask	2.3.2	BSD License
flask-babel	3.1.0	BSD 3 License
Flask-Cors	3.0.10	MIT License
Flask-Login	0.6.2	MIT License
Flask-Migrate	4.0.4	MIT License
Flask-WTF	1.1.1	BSD 3 License
frozenlist	1.3.3	Apache Software License (Apache 2)
gunicorn	20.1.0	MIT License
habanero	1.2.3	Copyright (C) 2021 Scott Chamberlain
httpcore	0.17.2	BSD License (BSD)
httpx	0.24.1	BSD License
idna	3.4	BSD License
imagesize	1.4.1	MIT License
ipykernel	6.23.1	BSD License (BSD 3-Clause License)
ipython	8.13.2	BSD License (BSD-3-Clause)
jedi	0.18.2	MIT License
Jinja2	3.1.2	BSD-3-Clause License
joblib	1.2.0	BSD License (BSD)
jupyter_client	8.2.0	BSD License (BSD 3-Clause License)
jupyter_core	5.3.0	BSD License (BSD 3-Clause License)
matplotlib	3.7.1	PSF
matplotlib-inline	0.1.6	BSD 3-Clause License
numpy	1.24.3	BSD License (BSD-3-Clause)
pandas	2.0.1	BSD License (BSD 3-Clause License)
pickleshare	0.7.5	MIT License
psycopg2	2.9.6	GNU Lesser General Public (LGPLv2)
PyJWT	2.7.0	MIT License
requests	2.31.0	Apache Software License (Apache 2.0)
scholarly	1.7.11	Unlicense
scikit-learn	1.2.2	BSD License (new BSD)
scipy	1.10.1	BSD License (Copyright (c) 2001-2002 Enthought,...)
selenium	4.9.1	Apache Software License (Apache 2.0)
SQLAlchemy	2.0.15	MIT License
xgboost	1.7.5	Apache Software License (Apache-2.0)
Bootstrap	5.2.3	MIT License
D3js	7.8.4	ISC
jQuery	3.6.4	MIT License
DataTables	1.13.4	MIT License
pytest	7.2.0	MIT License
Chartjs	4.0	MIT License

Tabla A.19: Información de los paquetes y licencias

Apéndice B

Especificación de Requisitos

B.1. Introducción

Para llevar a cabo el proyecto adecuadamente, se realizará un análisis exhaustivo de los requisitos funcionales y no funcionales de la aplicación. En la sección de Objetivos generales (B.2) se establecerán los resultados esperados y los criterios de éxito del proyecto. Seguidamente, se presentará el catálogo de requisitos (sección B.3), detallando las funcionalidades y características que la aplicación debe cumplir. Finalmente, se procederá a la especificación de requisitos (sección B.5), describiendo cada uno de los identificados en el catálogo y estableciendo su prioridad, complejidad y dependencias.

A través de este informe, se pretende sentar las bases y establecer las directrices necesarias para el desarrollo exitoso de la aplicación web que proporcionará a los usuarios la información clave sobre el impacto de las revistas científicas, con el objetivo de facilitar la toma de decisiones informadas en el ámbito de la publicación académica.

B.2. Objetivos generales

Los objetivos generales de este proyecto son los siguientes:

1. Realizar el proceso de extracción de datos relevantes para el cálculo del Índice de Impacto de las revistas científicas. Esta tarea constituye la parte más desafiante del proyecto, ya que implica llevar a cabo una investigación exhaustiva y realizar pruebas utilizando diversas técnicas

de *web scraping* y simulación con Selenium. Se requerirá desarrollar métodos eficientes para recopilar los datos necesarios que no estén fácilmente accesibles y estructurados.

2. Implementar modelos de predicción utilizando técnicas de *machine learning)*, en particular utilizando la biblioteca Scikit-learn. Estos modelos permitirán predecir el Índice de Impacto de las revistas científicas con base en las características y métricas disponibles. Se explorarán diferentes algoritmos y técnicas para obtener predicciones precisas y confiables.

3. Desarrollar una aplicación web utilizando el *framework* Flask. Esta aplicación será la interfaz principal para que los usuarios puedan acceder y visualizar los resultados del proyecto. La aplicación web proporcionará una experiencia de usuario intuitiva y amigable, permitiendo la búsqueda y filtrado de revistas, así como la visualización de los índices de impacto calculados y las predicciones generadas por los modelos.

El objetivo central del proyecto es proporcionar a los usuarios una herramienta integral que les permita evaluar y comparar el impacto de las revistas científicas de manera eficiente y precisa. Para lograr esto, se abordarán tres aspectos clave: la extracción de datos, los modelos de predicción y el desarrollo de la aplicación web. Cada uno de estos objetivos se complementa para lograr un sistema funcional y útil para la comunidad académica y científica.

Es importante reiterar que el proceso de extracción de datos es considerado como la parte más desafiante del proyecto debido a la necesidad de investigar y probar diferentes técnicas de *web scraping*. El éxito en esta etapa es fundamental para garantizar la disponibilidad de los datos necesarios para el cálculo del Índice de Impacto y para el entrenamiento de los modelos de predicción.

B.3. Catálogo de Historias de Usuario

En este caso, dado que se ha elegido seguir una metodología ágil, se considera un catálogo de historias de usuario (usadas en ambos marcos, Scrum y Kanban). Para poder redactar una historia de usuario, hay tres elementos clave [11]:

- Perfil: El rol del usuario final.
- Necesidad: El objetivo que tiene la función de *software* para el usuario final.
- Propósito: El objetivo de la experiencia del usuario final con la función de *software*.

Habiendo identificado estos elementos, la historia de usuario se redactará siguiendo este formato: **«Como [perfil], quiero [necesidad], para lograr [propósito]»**.

Se ha elaborado el catálogo de historias de usuario con el objetivo de cubrir todos los objetivos mencionados en el apartado anterior sin entrar en detalles (los requisitos más específicos con respecto a la aplicación web están definidos en la sección B.4). Para visualizar mejor el catálogo, se han clasificado las historias de usuario en tres categorías: extracción de datos (Tabla B.1), modelos de predicción (Tabla B.2) y aplicación web (Tabla B.3).

Historia de Usuario	**Descripción**
HU1	Como desarrollador, quiero obtener información relevante para el cálculo del Índice de Impacto (JCR) de las revistas científicas a partir de fuentes externas.
HU2	Como desarrollador, quiero aplicar técnicas de *web scraping* y simulación con Selenium para extraer los datos de manera eficiente y precisa.
HU3	Como desarrollador, quiero procesar y almacenar los datos extraídos en una base de datos para su posterior uso en el cálculo del Índice de Impacto y en los modelos de predicción.

Tabla B.1: HU Extracción de datos

Historia de Usuario	**Descripción**
HU4	Como desarrollador, quiero implementar modelos de predicción utilizando técnicas de *machine learning* (scikit-learn) para predecir el Índice de Impacto de las revistas científicas.
HU5	Como desarrollador, quiero evaluar y comparar diferentes algoritmos de *machine learning* para seleccionar los modelos más adecuados.
HU6	Como desarrollador, quiero entrenar los modelos utilizando los datos almacenados y ajustar sus parámetros para obtener predicciones precisas.

Tabla B.2: HU Modelos de predicción

Historia de Usuario	**Descripción**
HU7	Como usuario, quiero acceder a una aplicación web donde pueda buscar y filtrar revistas científicas para evaluar su Índice de Impacto.
HU8	Como usuario, quiero visualizar de forma clara y comprensible los índices de impacto calculados y las predicciones generadas por los modelos de predicción.
HU9	Como usuario, quiero interactuar con la aplicación web, seleccionar revistas específicas y obtener información detallada sobre cada una de ellas.

Tabla B.3: HU Aplicación web

Estas historias de usuario representan las necesidades y objetivos de los diferentes perfiles de usuarios en relación con la extracción de datos, los modelos de predicción y la aplicación web. Siguiendo una metodología ágil, estas historias de usuario servirán como guía para el desarrollo incremental y la entrega de funcionalidades de valor en cada iteración del proyecto.

B.4. Requisitos de la aplicación

En esta sección se enumerarán, en mayor detalle, los requisitos que se esperan de la aplicación web en concreto.

Requisitos funcionales

- RF1: Todo usuario debe poder registrarse e iniciar sesión.
- RF2: Validación de datos de usuario durante el registro:
  - RF2.1: Verificación del formato adecuado del correo electrónico.
  - RF2.2: Comprobación de los requisitos básicos de seguridad de la contraseña.
- RF3: Recuperación de contraseña en caso de olvido.
- RF4: Mostrar mensaje de error en caso de datos de inicio de sesión incorrectos.
- RF5: Opción para mostrar y ocultar la contraseña.
- RF6: Interfaz principal para elegir una revista y un año para predecir el JCR.
- RF7: Validación de campos obligatorios en la interfaz principal.
- RF8: Mensaje de confirmación antes de proceder al cálculo del JCR.
- RF9: Selección del modelo de predicción del JCR.
- RF10: Visualización del histórico del JCR y un gráfico representativo.
- RF11: Visualización de la predicción del JCR y un gráfico representativo.
- RF12: Navagación por *Breadcrums*.
- RF13: Interacción con el logo: redirección a la interfaz principal.
- RF14: Interfaz para mostrar la lista de revistas disponibles.
- RF15: Barra de búsqueda en la interfaz de revistas disponibles.
- RF16: Opción para cerrar sesión de usuario.

- RF17: Los gráficos deben mostrar información relevante con la que interactuar.
- RF18: Posibilidad de ocultar las líneas correspondientes a los modelos de predicción en el gráfico al pulsar sobre su leyenda.
- RF19: Los usuarios con cuenta deben poder modificar sus datos personales en su perfil de usuario.
- RF20: Cambio de idioma (español, inglés, francés e italiano).

Requisitos no funcionales:

- RNF1: Las interfaces deben contar con las barras superiores e inferiores comunes.
- RNF2: Barra superior con icono de perfil de usuario y símbolo de ayuda.
- RNF3: Interfaz de configuración de perfil de usuario con opciones de cierre de sesión, cambio de correo electrónico y nombre de usuario.
- RNF4: Enlace a la política de privacidad y términos de uso en el pie de página.
- RNF5: Aplicación *responsive* compatible con diferentes navegadores y sistemas operativos.
- RNF6: Interoperabilidad: compatibilidad con diferentes dispositivos y navegadores.
- RNF7: Disponibilidad: la aplicación debe estar disponible para su acceso y uso de manera constante.
- RNF8: Accesibilidad: la aplicación debe ser accesible para usuarios con discapacidades.
- RNF9: Soporte: la aplicación debe contar con un sistema de soporte para atender las consultas y problemas de los usuarios.
- RNF10: Mantenibilidad: la aplicación debe ser fácil de mantener, actualizar y corregir errores.
- RNF11: Seguridad: las contraseñas deben estar encriptadas y la aplicación debe seguir buenas prácticas de seguridad en general.

- RNF12: Privacidad: se deben proteger los datos personales de los usuarios y seguir las regulaciones de privacidad aplicables.
- RNF13: Escalabilidad: la aplicación debe poder manejar un aumento en la carga de usuarios y datos sin degradar su rendimiento.
- RNF14: Extensibilidad: la aplicación debe ser fácil de extender y agregar nuevas funcionalidades en el futuro.
- RNF15: Robustez: la aplicación debe ser capaz de manejar errores y excepciones de manera adecuada, sin colapsar o perder datos.
- RNF16: Internacionalización: la aplicación debe ser capaz de adaptarse a diferentes idiomas y culturas.

B.5. Especificación de requisitos

En esta sección se enumerarán los distintos casos de uso para la aplicación web desarrollada.

TODO

Diagrama de casos de usos

Finalmente, para poder visualizar todos estos casos, se incluye un diagrama de casos de uso.

TODO

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

C.5. Aplicación

Se pretende desarrollar una aplicación web sencilla que utilizará una arquitectura cliente-servidor de dos capas.

La funcionalidad principal de la aplicación será predecir el índice de impacto de una lista de revistas elegida por el usuario. Además, se proporcionarán gráficos e información adicional para ayudar a los usuarios a entender mejor los resultados. Por otro lado, se podrán distinguir dos perfiles de usuario: el usuario normal y el administrador. El administrador tendrá permisos para "reentrenar" la red que predice los índices y gestionar al resto de usuarios.

El *backend* se programará en Python y el *frontend* con HTML, CSS y JavaScript. Para ello, se utilizará Flask, que se trata de un *framework* de Python para el desarrollo de aplicaciones web.

Para la conexión con la base de datos, se hará uso de *psycopg2*, que es un módulo de Python que proporciona una interfaz para conectarse y interactuar con bases de datos PostgreSQL. Es una de las librerías más populares y ampliamente utilizadas para trabajar con PostgreSQL en Python. Además, es compatible con la mayoría de las versiones de Python y es muy fácil de utilizar.

Por otro lado, para la autenticación, se han evaluado varias opciones que ofrece Flask, a saber: Flask-Login, Flask-Security y Flask-JWT. Finalmente, se ha elegido Flask-Login debido a su sencillez y facilidad de uso. Esta opción permite al usuario iniciar sesión con un nombre de usuario y una contraseña y almacena la información de la sesión en las *cookies* del navegador. Este paquete incluye la protección de rutas con decoradores.

Interfaces

La aplicación estará conformada por seis interfaces principales, que se detallan a continuación:

- Interfaces iniciales con funciones de inicio de sesión y registro, verificando la validez de los datos y permitiendo recuperar contraseñas.
- Interfaz principal para seleccionar una revista y un año, con opción de calcular el JCR y elegir modelos de predicción.
- Interfaces con el histórico y la predicción del JCR, además de gráficos representativos.
- Interfaz con una lista de las revistas disponibles con barra de búsqueda.
- Área de usuario donde modificar las credenciales.

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

Esta sección está destinada a proporcionar información detallada sobre cómo utilizar el programa desarrollado en el proyecto. Aquí se describen las funciones y características clave de dicho software, así como los detalles sobre la configuración y la integración del programa con otros sistemas.

Base de datos

Para aquellos usuarios de sistemas operativos Windows, es importante tener en cuenta un detalle en el momento de obtener información de la base de datos en PostgresSQL. Cuando se recolectan los datos en formato CSV, es necesario cambiar la ruta de generación de estos archivos a la carpeta «Public» [1]. Esto se debe a que el usuario por defecto que genera PostgresSQL (usuario «postgres»), no tiene los permisos suficientes para leer e importar datos de los CSV en cuestión ya que, para ello, se utiliza el comando *copy* (que solo puede ser ejecutado por «Superusers»)[13]. Sin embargo, la carpeta

[1]Directorio que permite compartir archivos entre distintos usuarios en un mismo sistema Windows. Microsoft ha mantenido este directorio desde la versión de WindowsXP.

«Public» cuenta con los permisos necesarios para que el usuario que crea Postgres pueda acceder y leer los datos. Por lo tanto, es importante seguir este paso para garantizar que se pueda obtener correctamente la información recogida previamente.

D.4. Compilación, instalación y ejecución del proyecto

Despliegue en Heroku

A continuación se muestra la información en formato LaTeX:

> **NOTA**
>
> En el código de la aplicación web, al incluir `import` de clases propias, es preciso añadir al inicio de la ruta de paquetes el punto (por ejemplo, `import .miCarpeta.miClase`). Sin el punto inicial, Heroku no identificará los módulos por no estar en la raíz del repositorio.

Ficheros necesarios

Se deben ubicar los siguientes ficheros en la raíz del repositorio:

- `requirements.txt` → con todas las dependencias necesarias (`$ pip freeze > requirements.txt`)
- `Procfile` → indicando el módulo donde se encuentra `app.py` (fichero principal y su nombre).
- `runtime.txt` → indicar el lenguaje de programación y su versión (i.e.:`python-3.11.1`).

Creación de la aplicación

Para la creación de la aplicación en Heroku, se deben seguir los siguientes pasos previos:

1. Crear cuenta de Heroku (`https://www.heroku.com/what`), donde se solicitará elegir una forma de doble autentificación.

2. Solicitar el *Student Pack* (https://www.heroku.com/github-students) que ofrece GitHub Student para poder utilizar de forma gratuita (con limitaciones) los servicios de Heroku. Será necesario esperar algunos días antes de recibir una confirmación.

3. Descargar el *software* Heroku CLI (https://devcenter.heroku.com/articles/heroku-cli) adecuado para tu sistema operativo.

4. Crear aplicación. En este caso, se recomienda hacerlo desde un terminal en la raíz del repositorio de GitHub.

Los comandos para crear la aplicación son los siguientes:

1. Instalar el Dyno básico: `$ pip install gunicorn`

2. Ejecutar `$ heroku login`, que abrirá una pesataña en un navegador y solicitará las credenciales de la cuenta de Heroku creada.

3. Para crear la aplicación, bastará con ejecutar `$ heroku create nombre_app -region=eu`
   Como respuesta, deberá obtener la dirección de la aplicación y del repositorio remoto de Heroku con el siguiente formato:
   https://nombre_app.herokuapp.com/ | https://git.heroku.com/nombre_app.git

4. Finalmente, ejecutar `$ git push heroku main` → para subir el contenido de la rama `main` del repositorio GitHub al repositorio remoto de Heroku. Este comando se deberá ejecutar cada vez que hay cambios nuevos que se quieran desplegar.

5. Para desplegar la aplicación bastará con ejecutar `$ heroku open` →.

Finalmente, se puede comprobar que la aplicación se ha creado correctamente en el *dashboard* de Heroku (https://dashboard.heroku.com/apps/).

Añadir una base de datos

En este caso, como nuestra aplicación hace uso de una base de datos Postgres, se hará uso del complemento Heroku Postgres (https://devcenter.heroku.com/articles/getting-started-monitoring-postgres-database). Para ello será preciso seguir los siguientes pasos:

- Añadir (comando `$ addon` o directamente desde la web) un nuevo complemento: Heroku Postgres. Sin el *Student Pack* mencionado anteriormente, será necesario realizar un pago.
- Además será necesario configurar en tu código las credenciales de la base de datos (proporcionadas por Heroku) para poder establecer las conexiones. Es importante ejecutar de nuevo el comando `$ git push heroku main` tras realizar estos cambios.
- Volver a ejecutar `$ heroku open` y la aplicación ya estará en funcionamiento.

Mantenimiento y gestión de la aplicación

Algunos comandos útiles (https://devcenter.heroku.com/articles/heroku-cli-commands) para el mantenimiento y gestión de la aplicación son los siguientes:

- Destruir la aplicación: `$ heroku apps:destroy -a nombre_app` (pide confirmación).
- Reiniciar todos los *Dynos*: `$ heroku restart`.
- Registro:`$ heroku logs` (Añadir la opción `-tail` si se quiere que el log se mantenga abierto).

D.5. Pruebas del sistema

En el desarrollo de *software*, es esencial contar con una metodología adecuada para garantizar la calidad del producto final. En este sentido, el Test Driven Development (TDD) se ha convertido en una de las metodologías más populares en la actualidad. TDD consiste en escribir las pruebas antes de comenzar a desarrollar el código, lo que permite detectar errores de manera temprana y asegurar que el *software* cumpla con los requisitos establecidos. En este apartado, presentaremos los proyectos de pruebas que hemos llevado a cabo utilizando TDD como metodología principal.

Esta metodología se divide en tres fases principales: la fase roja, la fase verde y la fase morada.

- En la fase roja, se escriben las pruebas unitarias para el código que se desea implementar, las cuales deben fallar inicialmente (puesto que aún no existe el código a probar).

- En la fase verde, se escribe el código mínimo necesario para que las pruebas pasen satisfactoriamente.

- Finalmente, en la fase morada, se realiza la refactorización del código escrito anteriormente para mejorar su calidad y mantenerlo limpio y legible.

Este enfoque iterativo permite a los desarrolladores garantizar la calidad y funcionalidad del software en cada paso del proceso de desarrollo y ayuda a reducir los errores y *bugs* que puedan surgir en el proceso.

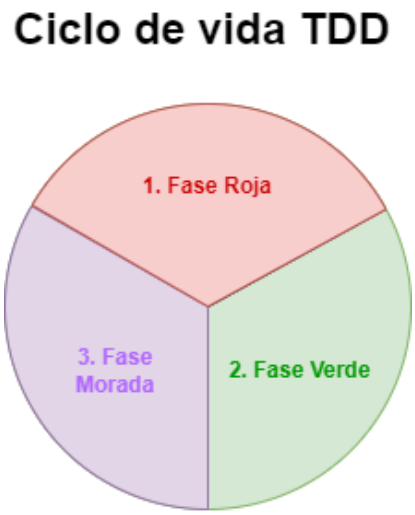

Figura D.1: Fases del la metodología TDD

Prototipo de Crossref

Este es el prototipo final para la extracción de los datos bibliográficos. A continuación, se documentará la información más relevante del proyecto de pruebas llevado a cabo.

Diseño y arquitectura

Puesto que el *software* a probar está compuesto de un solo *script* de Python, se ha generado un único módulo de pruebas llamado `crossref_test.py`.

Fase Roja

A lo largo de esta fase se definen las pruebas necesarias para crear un código que extraiga correctamente los datos de Crossref. Los casos de prueba diseñados son los siguientes:

ID	Nombre	Descipción	Salida esperada
C01	Revista existente	Método test_journal_exist. Se trata de extraer una revista que previamente se conoce que existe.	True
C02	Revista inexistente	Método test_journal_exist. Se trata de extraer una revista que previamente se conoce que no existe.	False
C03	Cargar datos de entrada	Método test_load_data. Se comprueba que se cargan bien los datos del CSV de entrada.	Carga correcta
C04	Formateado I	Método test_formatData. Se comprueba que se eliminan bien las etiquetas HTML y los espacios.	Cadena correcta
C05	Formateado II	Método test_formatData. Se comprueba que se eliminan bien las etiquetas «href».	Cadena correcta
C06	Formateado III	Método test_formatData. Se comprueba que una cadena correcta no cambia.	Cadena sin modificaciones
C07	Formateado IV	Método test_formatData. Se comprueba que no hay campos vacíos (rellenar con «n.m.»).	Cadena sin valores vacíos
C08	Formato de salida I	Método test_getArticles_returns_list_of_dicts. Se comprueba que se obtiene una lista de artículos.	Lista de diccionarios
C09	Formato de salida II	Método test_getArticles_returns_valid_keys. Se comprueba que las claves del diccionario son correctas.	Claves correctas
C10	Número de resultados	Método test_getArticles_correct_ article_count. Se comprueba que se obtiene el número correcto de artículos que aparecen en Crossref.	Número correcto

ID	Nombre	Descipción	Salida esperada
C11	Excepción de ISSN	Método test_getArticles_invalid_issn_raises_error. Se intenta buscar un ISSN incorrecto.	False
C12	Excepción de año I	Método test_getArticles_invalid_year. Se introducen los años de búsqueda en el orden incorrecto.	False
C13	Excepción de año II	Método test_getArticles_invalid_year. Se introducen años inválidos.	False
C14	Ejecución completa I	Método test_main. Pre-requisito: CSV de entrada de datos. Se comprueba que se genera una gráfica al finalizar el proceso. Post-requisito: Imagen con la gráfica generado.	True
C15	Ejecución completa II	Método test_main. Precondición: CSV de entrada de datos. Se comprueba que se generan los ficheros CSVs de resultados al finalizar el proceso. Post-requisito: Ficheros CSV generados.	True
C16	Ejecución completa III	Método test_main. Precondición: CSV de entrada de datos. Se comprueba que se genera el fichero de log al finalizar el proceso. Postcondición: Ficheros de log generados.	True
C17	Ejecución completa IV	Método test_main. Precondición: CSV de entrada de datos y fichero de log generado correctamente. Se comprueba que la información de log es correcta.	True

Tabla D.1: Casos de prueba para el prototipo de Crossref

Fase Verde

Tras la creación del código, finalmente se consigue una versión que pasa todos los test programados en la fase anterior, como se puede observar en la siguiente ilustración:

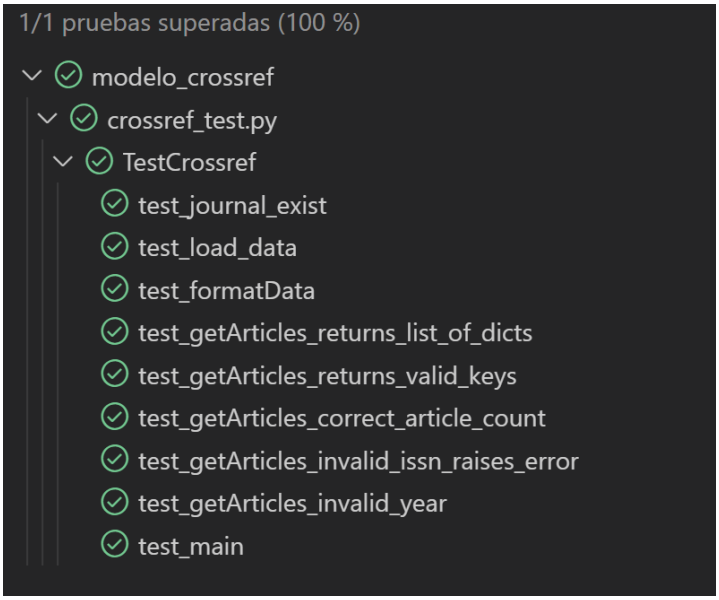

Figura D.2: Pruebas pasadas exitosamente

Fase Morada

En esta fase se han realizado acciones de refactorización. A saber:

- Documentación de código: Se han incluido las cabeceras y comentarios más importantes en el código para facilitar su comprensión y claridad. Asimismo, se han eliminado comentarios redundantes o innecesarios.
- Ordenación de *imports*: Se han ordenado los *imports* del código de forma que se identifique fácilmente la naturaleza de los paquetes y módulos empleados. De esta forma, se agrupan al inicio los módulos integrados de Python, posteriormente los paquetes de terceros y, finalmente, módulos o paquetes locales.
- Mensajes de salida I: En lugar de imprimir mensajes de salida en la consola, (ant. `print("[ERROR] Se ha producido un error")`), ahora se utilizan registros de *logging* para registrar los eventos de la función

(por ejemplo, `logging.error('Se ha producido un error')`). Los registros son una forma más eficiente y flexible de registrar información que imprimir en la consola.

- Mensajes de salida II: Utiliza f-strings en lugar de concatenar cadenas para mejorar el rendimiento y la legibilidad del código. Ejemplo: `logging.error('Se ha producido un error'+ codigo_error)` por `logging.error(f'Se ha producido un error {codigo_error}')`.
- Acceso a diccionarios I: En lugar de especificar los parámetros de búsqueda utilizando una cadena de consulta, se puede usar un diccionario y pasarlos como argumentos de palabras clave al método `requests.get()`. Esto puede hacer que el código sea más legible.
- Acceso a diccionarios II: En lugar de crear un diccionario vacío `author` y luego verificar si existe una lista de autores, se puede usar el método `get()` directamente en la lista de autores para obtener el primer elemento de la lista.
- Lista de comprensión: En lugar de crear un diccionario temporal publicacion para cada artículo, se puede usar una lista de comprensión para crear una lista de diccionarios de artículos. Esto puede hacer que el código sea más conciso.

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Entorno virtualizado

Se trata de un entorno aislado que te permite instalar y gestionar de forma independiente las bibliotecas y paquetes de Python que se necesitan para el proyecto, sin afectar a otros proyectos o al sistema operativo. Esto nos permite evitar problemas de dependencias y compatibilidad entre diferentes versiones de bibliotecas, además de facilitar la gestión y el mantenimiento del proyectos.

Miniconda, por su parte, es una distribución de Python que incluye un gestor de paquetes llamado Conda, que te permite instalar y gestionar de forma sencilla bibliotecas y paquetes de Python.

Así pues, se procede a crear un entorno virtualizado de Python utilizando Miniconda. El entorno en cuestión se encuentra empaquetado en la carpeta comprimida `environment.zip`. Para poder utilizar este entorno, bastará con realizar los siguientes pasos:

1. Instalar MiniConda en el dispositivo en el que se quiera ejecutar el prototipo.
2. Crear el entorno usando el siguiente comando en la consola de MiniConda: `conda env create -f environment.yml`. Es preciso estar en el mismo directorio que el fichero `environment.yml` o especificar la ruta completa.
3. A continuación, se debe activar el entorno con el siguiente comando: `conda activate crossref_entorno`.
4. Ejecutar el prototipo de Crossref.
5. Una vez finalizado, para desactivar el entorno, bastará con ejecutar el siguiente comando: `conda deactivate`.

Apéndice F

Glosario

API: Acrónimo de *Application Programming Interface* (Interfaz de Programación de Aplicaciones). Es un conjunto de reglas y protocolos que permite la interacción entre diferentes aplicaciones de *software*.

API REST: Una API basada en los principios del protocolo HTTP y diseñada para facilitar la comunicación y transferencia de datos entre sistemas. REST significa *Representational State Transfer* (Transferencia de Estado Representacional).

Babel: Una herramienta de traducción y localización de software que permite adaptar una aplicación para soportar diferentes idiomas y regiones.

Backend: La parte de un sistema o aplicación que se encarga del procesamiento y almacenamiento de datos, así como de la lógica de negocio. Normalmente, se refiere a la parte del sistema que no es visible para los usuarios finales.

BeautifulSoup4: Una biblioteca de Python que facilita el análisis y extracción de datos de documentos HTML y XML.

Bootstrap: Un *framework* de desarrollo web que proporciona estilos CSS y componentes JavaScript predefinidos, lo que facilita la creación de interfaces web responsivas y atractivas.

Chartjs: Una biblioteca JavaScript para la creación de gráficos interactivos y visualmente atractivos en páginas web.

Crossref: Una organización sin fines de lucro que proporciona servicios y datos relacionados con la identificación y el enlace de contenido académico.

crossrefapi: Un paquete de Python que proporciona una interfaz para interactuar con la API de Crossref y acceder a metadatos de publicaciones académicas.

Crossvalidation: Un método utilizado en aprendizaje automático (*machine learning*) para evaluar y validar el rendimiento de un modelo utilizando diferentes subconjuntos de datos.

Cryptography: Una biblioteca de Python que proporciona herramientas para la implementación de algoritmos criptográficos, como el cifrado y la generación de firmas digitales.

CSS: Acrónimo de *Cascading Style Sheets* (Hojas de Estilo en Cascada). Es un lenguaje utilizado para describir la presentación y el estilo de un documento HTML.

Cuartil: Un valor que divide un conjunto de datos ordenados en cuatro partes iguales, donde cada parte representa el 25 % de los datos.

D3js: Una biblioteca JavaScript para la creación de visualizaciones de datos interactivas y dinámicas en páginas web.

DataTables: Un complemento de JavaScript que proporciona funcionalidad avanzada de tablas interactivas y filtrado en páginas web.

DOI: Acrónimo de *Digital Object Identifier* (Identificador de Objeto Digital). Es un identificador único utilizado para identificar de manera persistente un objeto digital, como un artículo científico.

Flask: Un framework de desarrollo web minimalista y flexible para Python. Permite construir aplicaciones web rápidas y escalables.

flask-babel: Una extensión de Flask que simplifica la internacionalización y localización de aplicaciones web.

Flask-Cors: Una extensión de Flask que permite manejar la política de intercambio de recursos entre dominios (CORS) en aplicaciones web.

Flask-Login: Una extensión de Flask que proporciona funcionalidad de autenticación de usuarios y gestión de sesiones.

Flask-Migrate: Una extensión de Flask que facilita la gestión de migraciones de base de datos utilizando SQLAlchemy.

Framework Flask: Un conjunto de herramientas y librerías que facilitan el desarrollo de aplicaciones web utilizando Python y el framework Flask.

Frontend: La parte de un sistema o aplicación que interactúa directamente con los usuarios finales. Se refiere a la interfaz de usuario y la presentación visual.

frozenlist: Un paquete de Python que proporciona una lista inmutable (no modificable) que garantiza la integridad de los datos.

Google Scholar: Un motor de búsqueda especializado en literatura académica y científica. Proporciona acceso a artículos, tesis, resúmenes y otros recursos académicos.

Gráfico burndown: Un gráfico utilizado en la metodología ágil para representar el progreso de un proyecto a lo largo del tiempo, mostrando el trabajo pendiente y la tendencia de finalización.

gunicorn: Un servidor web HTTP para aplicaciones Python. Es comúnmente utilizado para implementar aplicaciones Flask en producción.

Heroku: Una plataforma en la nube que permite implementar, gestionar y escalar aplicaciones web.

HTML: Acrónimo de *Hypertext Markup Language* (Lenguaje de Marcado de Hipertexto). Es el lenguaje estándar utilizado para crear páginas web.

httpcore: Una biblioteca Python que proporciona una interfaz para realizar solicitudes HTTP de bajo nivel.

httpx: Una biblioteca Python que proporciona una interfaz de alto nivel para realizar solicitudes HTTP.

idna: Un módulo de Python que proporciona herramientas para la manipulación de nombres de dominio internacionalizados (IDN).

IF: Acrónimo de *Impact Factor* (Factor de Impacto). Es una métrica utilizada para evaluar la importancia relativa de una revista científica dentro de su campo.

Índice de Impacto: Una medida que indica la influencia relativa de una revista científica. Se basa en la frecuencia con la que los artículos de la revista son citados por otros investigadores.

ipykernel: Un kernel de IPython que permite ejecutar código Python en el entorno de Jupyter Notebook.

ipython: Un entorno interactivo de programación y exploración de datos que proporciona características adicionales sobre el intérprete de Python estándar.

JavaScript: Un lenguaje de programación interpretado utilizado principalmente para agregar interactividad y dinamismo a las páginas web.

JCR: Acrónimo de *Journal Citation Reports*. Es una base de datos que recopila información sobre las citas y el impacto de las revistas científicas.

jedi: Una biblioteca de Python que proporciona funcionalidades avanzadas de autocompletado y análisis estático para el código Python.

Jinja2: Un motor de plantillas de Python utilizado en el framework Flask para generar contenido dinámico en las aplicaciones web.

joblib: Una biblioteca de Python utilizada para la serialización y paralelización de tareas, especialmente en el contexto de machine learning.

jQuery: Una biblioteca de JavaScript rápida, pequeña y rica en características que simplifica la manipulación y el manejo de eventos en el código HTML.

jupyter_client: Un cliente de Jupyter Notebook que permite la comunicación con los kernels y el manejo de sesiones interactivas.

jupyter_core: Un conjunto de funcionalidades esenciales para el funcionamiento de los entornos Jupyter, incluyendo la gestión de notebooks y configuraciones.

Kanban: Una metodología ágil de gestión de proyectos que se centra en la visualización del flujo de trabajo y la optimización de la productividad.

LATEX: Un sistema de composición de documentos utilizado principalmente para la creación de documentos científicos y técnicos con alta calidad tipográfica.

Machine Learning: Un campo de estudio de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender y tomar decisiones basadas en datos.

MariaDB: Un sistema de gestión de bases de datos relacionales (RDBMS) derivado de MySQL. Ofrece compatibilidad con la mayoría de las características de MySQL y mejoras adicionales.

Matplotlib: Una biblioteca de Python ampliamente utilizada para la creación de gráficos estáticos, gráficos 2D y 3D, y visualizaciones de datos.

matplotlib-inline: Una extensión de Jupyter Notebook que permite la visualización de gráficos Matplotlib de forma integrada en el notebook.

Metadatos: Datos que describen características o propiedades de un objeto. En el contexto de los documentos académicos, los metadatos pueden

incluir información sobre los autores, título, resumen, fecha de publicación, etc.

MIAR: Acrónimo de *Matriz de Información para el Análisis de Revistas*. Es una base de datos en línea que proporciona información sobre las revistas académicas y científicas.

Miniconda: Una versión minimalista de Anaconda, un sistema de gestión de entornos y paquetes de Python, que proporciona solo los componentes esenciales.

NumPy: Una biblioteca de Python utilizada para realizar operaciones matemáticas y numéricas eficientes en matrices y arreglos multidimensionales.

Open Source: Un término que se refiere a programas o software cuyo código fuente es de acceso público y puede ser utilizado, modificado y distribuido por cualquier persona.

Pandas: Una biblioteca de Python utilizada para el análisis y manipulación de datos estructurados. Proporciona estructuras de datos y herramientas para el manejo eficiente de tablas y series temporales.

Paquete: Un conjunto de módulos y archivos relacionados que se agrupan y distribuyen juntos para facilitar su uso y reutilización en aplicaciones de software.

pickleshare: Un módulo de Python que proporciona una forma sencilla de almacenar y compartir objetos de Python mediante la serialización y deserialización utilizando el formato "pickle".

Postgres: Un sistema de gestión de bases de datos relacional de código abierto (RDBMS) conocido también como PostgreSQL.

Product Owner: Un rol en la metodología ágil Scrum que representa a los interesados y es responsable de gestionar el backlog del producto y priorizar las funcionalidades.

psycopg2: Un adaptador de base de datos de PostgreSQL para Python que permite interactuar con bases de datos PostgreSQL utilizando Python.

PyJWT: Una biblioteca de Python que proporciona herramientas para la codificación y decodificación de JSON Web Tokens (JWT).

pytest: Un marco de pruebas de Python que facilita la escritura y ejecución de pruebas unitarias, de integración y funcionales.

Requests: Una biblioteca de Python utilizada para realizar solicitudes HTTP de manera sencilla y eficiente.

scholarly: Un paquete de Python que proporciona una interfaz para interactuar con la API de Google Scholar y obtener información sobre publicaciones académicas.

scikit-learn: Una biblioteca de Python ampliamente utilizada para el aprendizaje automático (machine learning). Proporciona una amplia gama de algoritmos y herramientas para el análisis de datos y la construcción de modelos predictivos.

scipy: Una biblioteca de Python utilizada para el cálculo científico y el análisis de datos. Proporciona funcionalidades para el álgebra lineal, estadísticas, optimización y más.

Scopus: Una base de datos bibliográfica y de citas que contiene información sobre publicaciones académicas, patentes y conferencias.

Scrum: Un marco de trabajo ágil para la gestión y desarrollo de proyectos. Se centra en la colaboración, la adaptabilidad y la entrega incremental.

Scrum Master: Un rol en la metodología ágil Scrum responsable de facilitar el proceso y asegurar que el equipo siga las prácticas y principios de Scrum.

Selenium: Una suite de herramientas utilizada para la automatización de pruebas en aplicaciones web. Permite controlar navegadores web y simular interacciones de los usuarios.

Sprints: Iteraciones cortas y enfocadas en las que se realiza el trabajo en un proyecto ágil. Cada sprint tiene una duración fija y al final de cada uno se entrega un incremento potencialmente entregable del producto.

SQLAlchemy: Una biblioteca de Python que proporciona una capa de abstracción sobre los motores de bases de datos relacionales. Permite interactuar con bases de datos utilizando código Python en lugar de SQL directamente.

Story Points: Una medida utilizada en la metodología ágil para estimar el esfuerzo o la complejidad de una tarea. Ayuda a determinar la capacidad de trabajo de un equipo en un *sprint*.

Web of Science: Una base de datos bibliográfica y de citas ampliamente utilizada que proporciona información sobre publicaciones científicas, conferencias y patentes.

Web Scraping: La extracción de datos de páginas web de forma automatizada mediante el uso de programas o *scripts*.

xgboost: Una biblioteca de aprendizaje automático (*machine learning*) que se enfoca en el algoritmo de Gradient Boosting.

ZenHub: Una herramienta de gestión de proyectos basada en GitHub que proporciona características adicionales como tableros Kanban, seguimiento de problemas y más.

Bibliografía

[1]

[2]

[3]

[4]

[5]

[6]

[7] Agencia tributaria: Irpf.

[8] Seguridad social: Cotización.

[9] Zhenhub. [Internet; acceso 03-junio-2023].

[10] 2023.

[11] Team Asana. Historias de usuario: 3 ejemplos para generar valor para el usuario [2022] • asana, Jan 2022.

[12] Anthony E. Boardman, David H. Greenberg, Aidan R. Vining, and David L. Weimar. *Cost-benefit analysis: Concepts and practice*. Prentice Hall, 2006.

[13] Jorge Domínguez Chávez. *CLIENTE PSQL DE POSTGRESQL*. 04 2020.

[14] Julia Martins. ¿qué es la metodología kanban y cómo funciona? [2022] • asana, Oct 2022.

[15] Marta Palacio. *Scrum Master. Temario troncal 1.* 06 2021.

[16] Rosa-Clark. Rest api - crossref, Apr 2020.