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 — 3 de marzo 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 llevará 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. 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 y 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 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 la siguiente tabla se recogen los distintos *sprints* junto con su duración, objetivos y tareas.

Tabla A.1: Planificación de Sprints

Sprint	**Duración**	**Objetivos**	**Tareas**
1	19/09/2022 03/10/2022	Comenzar el desarrollo del proyecto	Elegir un modelo de referencias bibliográficas. Definición de conceptos.
2	03/10/2022 17/10/2022	Tareas de investigación y documentación	Búsqueda de antecedentes. Familiarizarse con la memoria en LaTeX.
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.
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.
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.
6	09/01/2023 23/01/2023	Documentación y obtención del DOI	Obtención del DOI a través de Crossref. Avence de la memoria y los anexos. Mejora de la BBDD

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

Una muestra de cómo podría ser una tabla de casos de uso:

B.2. Objetivos generales

B.3. Catálogo de requisitos

B.4. Especificación de requisitos

CU-1	**Ejemplo de caso de uso**
Versión	1.0
Autor	Alumno
Requisitos asociados	RF-xx, RF-xx
Descripción	La descripción del CU
Precondición	Precondiciones (podría haber más de una)
Acciones	1. Pasos del CU 2. Pasos del CU (añadir tantos como sean necesarios)
Postcondición	Postcondiciones (podría haber más de una)
Excepciones	Excepciones
Importancia	Alta o Media o Baja...

Tabla B.1: CU-1 Nombre del caso de uso.

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

C.5. API

Se pretende desarrollar una API REST sencilla que utilizará una arquitectura cliente-servidor de dos capas.

La funcionalidad principal de la API 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.

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")[1]. 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

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

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 ."[en]vironment.zip". Para poder hacer uso del entorno bastará con ejecutar el siguiente comando: `conda activate /ruta/al/environment.zip`

Bibliografía

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