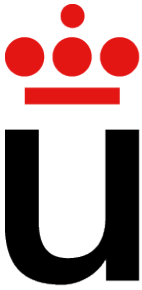

Universidad
Rey Juan Carlos

TITULACIÓN EN MAYÚSCULAS

Curso Académico 2022/2023

Trabajo Fin de Grado/Máster

TÍTULO DEL TRABAJO EN MAYÚSCULAS

Autor : Carmen González López

Tutor : Gregorio Robles Martínez

Trabajo Fin de Grado/Máster

Título del Trabajo con Letras Capitales para Sustantivos y Adjetivos

Autor : Carmen González López

Tutor : Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 202X, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 202X

Dedicado a

mi familia / mi abuelo / mi abuela

II

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se introduce el proyecto. Debería tener información general sobre el mismo, dando la información sobre el contexto en el que se ha desarrollado.

No te olvides de echarle un ojo a la página con los cinco errores de escritura más frecuentes[1].

Aconsejo a todo el mundo que mire y se inspire en memorias pasadas. Las memorias de los proyectos que he llevado yo están (casi) todas almacenadas en mi web del GSyC[2].

1.1. Estructura de la memoria

Describiremos la estructura de la memoria, exponiendo el contenido de cada uno de los capítulos, proporcionando así una guía organizada del trabajo de fin de grado para una mejor lectura y comprensión de este.

- **Capítulo 1: Introducción.**
- **Capítulo 2: Objetivos.** En esta sección se establece el objetivo general que se pretende alcanzar con este proyecto. Así como los objetivos específicos necesarios que van a guiar nuestro proyecto y la planificación de dichos objetivos(de todo el proyecto).
- **Capítulo 3: Estado del arte.** En esta sección se proporciona información detallada sobre el diseño, las características y los usos de cada una de las tecnologías usadas en el proyecto.

[1] `http://www.tallerdeescritores.com/errores-de-escritura-frecuentes`

[2] `https://gsyc.urjc.es/~grex/pfcs/`

- **Capítulo 4: Diseño e implementación.** En esta sección se detallan las fases que se han llevado a cabo para realizar el análisis de estudio. También se describe la arquitectura general del proyecto, los procedimientos utilizados para la recolección y el almacenamiento de los datos y la descripción del análisis de los datos.
- **Capítulo 5: Experimentos y validación.**
- **Capítulo 6: Resultados.**
- **Capítulo 7: Conclusiones.**

Capítulo 2

Objetivos

En un trabajo de fin de grado es muy importante describir el propósito final del proyecto, definir cuáles serán los aspectos a tratar, así como la planificación llevada a cabo. Todo esto ayudará a los lectores a hacer un seguimiento del proyecto. En este capítulo, se presentan los objetivos del trabajo de fin de grado y los procedimientos planteados para lograrlos.

2.1. Objetivo general

Mi trabajo fin de grado consiste en analizar repositorios de la plataforma de Github para observar las diferencias y similitudes que existen entre los desarolladores que usan ficheros UML frente al resto de desarrolladores.

2.2. Objetivos específicos

Para lograr el objetivo general de este proyecto se han establecido los siguientes objetivos específicos:

- Estudiar y probar el funcionamiento de Perceval
- Seleccionar una serie de repositorios de GitHub y almacenarlos en una estructura JSON.
- Observar los datos amacenados en los archivos JSON obtenidos e identificar los datos que son de interés para comparar en nuestro estudio.

- Desarrollar los programas de extracción y almacenamiento de los datos escogidos en MongoDB.

- Realizar el análisis de los resultados obtenidos para determinar las similitudes y diferencias más relevantes entre los desarrolladores que usan UML y los que no.

2.3. Planificación temporal

A finales de abril del año pasado empecé mi proyecto de fin de grado. El desarrollo de este trabajo comenzó en mayo de 2022, cuando el profesor Gregorio Robles Martínez me sugirió la idea del proyecto que he llevado a cabo.

Durante los meses de mayo y junio empecé a documentarme sobre Lindholmen dataset[1], que es una base de datos con los repositorios que utilizan diagramas UML en GitHub. De la misma manera, obtuve la información necesaria sobre las herramientas Grimoire Perceval[2] y GitHub API[3], las cuales se utilizan para analizar repositorios y extraer datos de estos. Después del estudio y análisis de estas herramientas, me decanté por Perceval, ya que con ella se obtienen más datos de Github.

De julio a septiembre, empecé a descargar y a almacenar en MongoDB algunos de los repositorios obtenidos en Lindholmen dataset usando Perceval para ver que datos se obtenía y cuales eran más relevantes para, posteriormente, realizar el análisis. También, inicié el desarrollo de varios programas con los que extaer los datos de interés que, previamente, había seleccionado.

Desde octubre hasta diciembre estuve desarrollando un porgrama para que guardase todos los datos de interés extraídos en MongoDB y otro programa que analizase dichos datos almacenados en MongoDB. A la vez, fui recopilando y almacenando los datos de una serie de repositorios de desarrolladores en los que hay archivos UML y repositorios en los que no hay archivos UML. Además, almacené los commits de dichos repositorios; ya que estos contienen información de interés para el desarrollo de nuestro análisis.

Durante los meses de enero y febrero del 2023, busqué cómo introducir los archivos JSON obtenidos, tanto de los repositorios, como de los commits en MongoDB de forma automática,

[1] `http://models-db.com/`

[2] `https://github.com/chaoss/grimoirelab-perceval`

[3] `https://docs.github.com/es/rest`

que hasta ese momento había hecho de manera manual. Esto supuso una mejora en el proceso de desarrollo del proyecto, ya que aceleró la introducción en MongoDB de los datos necesarios para el posterior análisis. Para ello, decidí crear un programa que ejecutase el proceso de mongoimport de cada uno de los archivos que había descargado.

(Entre marzo y abril, inicié el proceso de escritura de la memoria del proyecto...)

Capítulo 3

Estado del arte

En este apartado, se proporciona una comprensión completa y actualizada de las herramientas y librerías usadas en el proyecto. Esta exposición nos da una visión de la metodología empleada.

3.1. Tecnologías y herramientas

3.1.1. Github

Github es una plataforma web de desarrollo colaborativo de software[1] que permite la colaboración y el almacenamiento de proyectos de código abierto o cerrado. En esta plataforma, los usuarios pueden crear y unirse a proyectos para compartir ideas, discutir posibles soluciones y colaborar entre ellos para mejorar dichos proyecto. La función principal de Github [1] es que los usuarios puedan trabajar juntos en un proyecto desde cualquier lugar y en cualquier momento, ya que, registra el desarrollo de los proyectos de forma remota en la nube y no requiere de una infraestructura de hardware específica. Github ofrece una serie de bibliotecas para ayudar a los desarrolladores en su trabajo y herramientas para la gestión de versiones del software, la gestión de problemas, la revisión de código... Github integra la funcionalidad de Git que es un sistema de control de versiones distribuido. Se dice que es "distribuido" porque git tiene un historial completo de cambios, en el cual se puenden revisar los cambios realizados en el código a lo largo del tiempo. Esto permite ver las modificaciones de cada desarrollador e incluso deshacer-

[1] `https://github.com/`

las si es necesario. Además permite el trabajo de varios desarrolladores en un mismo proyecto. En resumen, GitHub es una plataforma que se ha convertido en una herramienta muy popular y esencial en la comunidad de desarrollo de software, ya que permite colaborar y trabajar en equipo para crear proyectos de alta calidad.

3.1.2. Mongodb

MongoDB[2] es un sistema de gestión de bases de datos no relacional (NoSQL) de código abierto. MongoDB ha sido diseñado para almacenar grandes volúmenes de datos; estos datos se almacenan y recuperan en formato BJSON [3] (Binary JSON), en lugar de en un formato de tabla relacional. El formato BJSON se inventó para solucionar algunos problemas que hacen que usar el formato de representación de objetos JSON no sea la mejor opción dentro de una base de datos. Con BJSON se optimiza la velocidad, el espacio y se mejora la eficiencia; solucionando así los problemas que tiene JSON de carecer de soporte para fechas y datos binarios y de no tener una longitud fija para los objetos y las propiedades JSON. En definitiva, MongoDB es un modelo de base de datos orientado a documentos. Cuando se almacena un documento a éste se le asigna un código que facilita el manejo de los datos que contiene.

MongoDB[4] tiene un modelo avanzado de consultas e idexación. Además, cuenta con una alta disponibilidad gracias a sus sistemas en la nube y por tanto es capaz de recuperarse en caso de fallo. Esta recuperación se consigue mediante el proceso de replicación automática. La consultas se ejecutan de la siguiente manera: MongoDB recibe por defecto al nodo (index) primario todas las lecturas y ejecuta todas las escrituras. El proceso de replicación permite tener siempre uns copia exacta del nodo primario, al replicar sus datos en los nodos secundarios. En el caso de que se produzca un fallo en el sistema, el nodo primario pasaría a ser uno de los secundarios contribuyendo a la recuperación del sistema. Otro proceso con el que cuenta es el fragmentación automática (auto-sharding) y el escalado horizontal que permiten la distribución de datos en múltiples servidores de forma que si uno falla, se sustituye rápidamente por otro servidor. MongoDB también ofrece controles de seguridad integrados para todos los datos.

Por último, MongoDB se puede ejecutar en una variedad de plataformas y sistemas ope-

[2] https://www.mongodb.com/

[3] https://www.mongodb.com/json-and-bson

[4] https://www.mongodb.com/es/nosql-explained

rativos, y es compatible con muchos lenguajes de programación. Por ello es utilizado en una amplia variedad de aplicaciones, incluyendo aplicaciones web, aplicaciones móviles, juegos y análisis de datos, entre otros.

3.1.3. JSON

JSON[5] (JavaScript Object Notation) es un formato ligero de intercambio de datos que se utiliza para transmitir y almacenar datos estructurados. Fue diseñado con el objetivo de que fuese un formato de texto portátil, mínimo y, originalmente, para ser utilizado con JavaScript. JSON es una colección no ordenada de pares clave-valor separados por comas y encerrados en llaves que se puede estructurar como un array o como un objeto [2]. En este proyecto usaremos la estructura JSON de objeto como podemos ver en la figura 3.1.

En el siguiente objeto JSON tenemos dos pares clave-valor entre { } y separados por una coma; en los cuales las claves tiene como valor las cadenas “string1” y “string2” {“key1”: “string1”, “key2” : “string2”}. Cada clave es una cadena de caracteres que se identifica con un valor correspondiente; estos valores pueden ser de cualquier tipo de datos: números, cadenas, booleanos, objetos y matrices. Una de las principales ventajas de JSON es que es fácil de leer y escribir para los humanos, lo que lo hace más fácil de entender y depurar en caso de problemas. Este tipo de estructuras son universales y por eso JSON es compatible con la mayoría de los lenguajes de programación y se ha convertido en un formato de datos muy popular. Por esta razón, al ser tan versátil se utiliza en aplicaciones web y móviles para transmitir datos entre el cliente y el servidor y como formato para el almacenamiento en bases de datos NoSQL, como MongoDB... [6].

[5] https://www.json.org/json-es.html

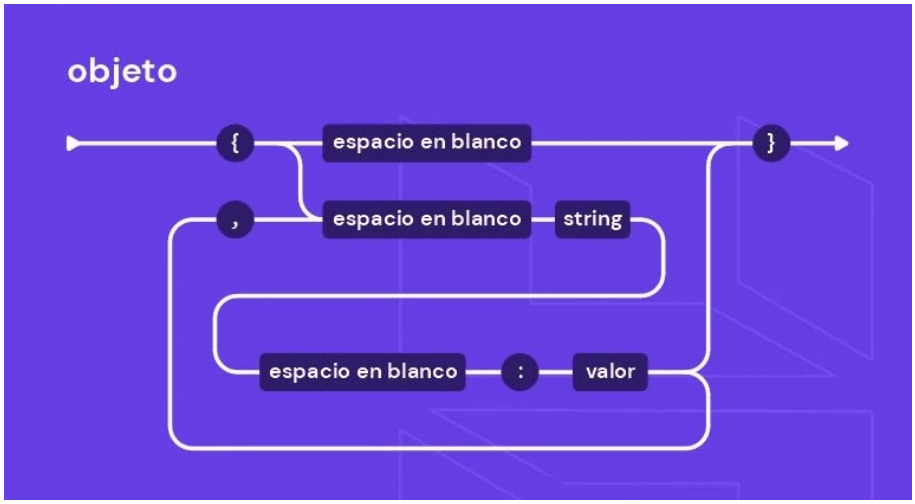

Figura 3.1: Estructura de un objeto JSON.

3.1.4. Python

Python[6] es un lenguaje de programación de alto nivel creado por Guido van Rossum [3] y lanzado por primera vez en 1991, cuyo nombre está inspirado en el grupo de cómicos "Monty Python".

Fue diseñado para facilitar la escritura de código y por tanto destaca por ser un lenguaje potente, sencillo, fácil de aprender y con una sintaxis clara y legible. Este, se considera un lenguaje multiparadigma, ya que, se apoya en diferentes estilos de programación como la programación orientada a objetos, la programación funcional y la programación imperativa. Python además cuenta con estructuras de datos para facilitar el desarrollo de software como listas, diccionarios, conjuntos y tuplas. También tiene una gran cantidad de bibliotecas y herramientas que facilitan el desarrollo de software; ya que, con ellas se obtienen soluciones en la tareas básicas de un programador.

Además, Python [7] cuenta con un intérprete propio y tiene compatibilidad con diferentes plataformas y sistemas operativos al ser un lenguaje de código abierto. Esta es una de las razones de que sea utilizado en una amplia variedad de aplicaciones, como el desarrollo web, el análisis de datos, la inteligencia artificial, el aprendizaje automático, la automatización de tareas y la creación de juegos, entre otros. Por todo esto, este lenguaje ha ganado mucha fuerza en la comunidad científica, educacional y de sofware libre.

[6] `https://www.python.org/`

3.1.5. Ubuntu

Ubuntu[7] es un sistema operativo de código abierto basado en Linux diseñado con una gran cantidad de software preinstalado para satisfacer las necesidades de la mayoría de los usuarios. Fue lanzado por primera vez en 2004 y desarrollado por una empresa llamada Canonnical [4], cuyo objetivo era generar un sistema fácil de instalar y utilizar, completo e innovador.

Actualmente Ubuntu [9] se ha convertido en uno de los sistemas operativos más populares en la comunidad de software libre y de código abierto. Eso se debe a que está disponible de forma gratuita para su descarga y uso de múltiples formas y, además, tiene una interfaz muy parecida a Windows lo que permite que sea intuitivo y fácil de utilizar por cualquier usuario.

Al igual que con Windows, con este sistema operativo también podemos navegar en la web, crear y modificar documentos; al ser compatible con formatos de archivos muy conocidos, y otras tareas que realizamos en nuestro día a día. Ubuntu se utiliza en una amplia variedad de sistemas, desde ordenadores hasta servidores y dispositivos móviles.

3.1.6. WSL

WSL[8](Windows Subsystem for Linux) es un sistema diseñado por Microsoft que permite a los desarrolladores ejecutar aplicaciones y herramientas de línea de comandos de Linux directamente en Windows (por la capa de compatibilidad), sin necesidad de utilizar una máquina virtual o instalar un sistema operativo Linux en un disco duro. Este subsistema de Windows para Linux fue lanzado en 2016 y no ha dejado de evolucionar desde entonces.

WSL [5] fue creado para los desarrolladores que necesitan utilizar herramientas y aplicaciones de línea de comandos específicas de Linux, pero prefieren trabajar en el entorno de Windows. También para aquellos que necesitan trabajar en los sistemas operativos Windows y Linux simultáneamente.

La versión WSL 1 utiliza una capa de compatibilidad para la transferencia de la ejecución de código entre Windows y Linux, mientras que WSL 2 tiene una versión actualizada de su arquitectura del software con respecto a su versión anterior. Además, utiliza una máquina virtual de Linux integrada que ofrece un rendimiento alto y muy parecido al de un sistema Linux real.

[7] `https://ubuntu.com/`

[8] `https://learn.microsoft.com/es-es/windows/wsl/`

3.1.7. LaTex

LaTeX[9] es un sistema de composición de (textos)(documentos), el cual, está formado por un conjunto de macros TeX. Fue desarrollado por Leslie Lamport en 1984, con el fin de facilitar el uso del lenguaje de composición tipográfica de textos(documentos) TeX. Tex, también es un sistema de composición tipográfica de textos que usa funciones avanzadas de automatización para generar documentos en los cuales se usan textos y fórmulas matemáticas con un determinado estándar. Fue desarrollado por Donald E. Knuth en 1978 como herramienta para la creación de textos que contengan fórmulas matematicas complejas.

LaTeX [8] usa las características del sistema TeX para generar documentos científicos, técnicos y académicos de alta calidad automaticamente, como artículos académicos, libros, tesis... Además, LaTeX ofrece con una variedad de paquetes, extensiones y estilos personalizados para adaptarse a las indicaciones específicas de cada usuario. Las extensiones con las que cuenta permiten insertar fácilmente imágenes, diagramas matemáticos, fórmulas matemáticas, textos de lenguajes de programación...

Estos sistemas de composición de textos están disponibles en gran cantidad de plataformas; además, cuentan con editores gratuitos que facilitan la creación de documentos sin la necesidad de tener conocimientos previos de programación. LaTeX utiliza una sintaxis de comandos que permiten al usuario controlar la estructura y el formato del documento, incluyendo la disposición de los elementos, el estilo de las fuentes, las ecuaciones matemáticas, y las referencias bibliográficas.

Una vez que se domina la sintaxis y las herramientas de LaTeX, se puede crear fácilmente documentos de alta calidad de manera eficiente. Por eso LaTeX es una herramienta muy utilizada y valorada en la comunidad científica, técnica y académica se utiliza ampliamente en universidades y centros de investigación de todo el mundo por su capacidad para producir documentos de alta calidad con un aspecto profesional, incluso si se requiere un formato complejo y detallado.

[9] `https://es.overleaf.com/`

3.2. Librerías

3.2.1. Perceval

Perceval[10] es uno de los componentes que contiene GrimoireLAb[11] que es uno de los proyectos fundadores de CHAOSS Software[12]. Este proyecto fue desarrollado para ayudar a investigadores en el análisis de repositorios de software y está formado por un conjunto de herramientas gratuitas, libres y de código abierto.

GrimoireLAb es el resultado de más de 10 años de desarrollo de Bitergia, grupo de investigación de software libre de la URJC y varios colaboradores. Cada herramienta se encuentra disponible en un repositorio distinto en Github y se puede instalar fácilmente como un módulo de paquetes de Python. Dichas herramientas se crearon para la recopilación automática e incremental de información de cualquier fuente de datos, el enriquecimiento de estos con información adicional y el almacenamiento y visualización de estos datos.

GrimoireLAb cuenta con unos componentes que se pueden usar juntos o por separado para las distintas tareas que se ejecutan en un análisis de datos. Perceval, como ya hemos mencionado, es uno de estos componentes que funciona como una biblioteca/módulo y proporciona una API de Python. Está diseñado para recuperar datos relacionados con el desarrollo de software de diferentes fuentes. Estas fuentes pueden ser de gestión de problemas o tareas, de gestión o revisión de código fuente, de foros o listas de correo, de wikis, entre otros. Posteriormente, se guarda la información obtenida en una base de datos en un formato estandarizado. Los datos que obtenemos con Perceval se almacenan como diccionarios de Python o como documentos JSON como el que mostramos a continuación:

```
{
  "backend_name": "GitHub",
  "backend_version": "0.2.2",
  "data": {
  ...
  },
  "origin": "https://github.com/grimoirelab/perceval",
```

[10] https://github.com/chaoss/grimoirelab-perceval
[11] https://chaoss.github.io/grimoirelab/
[12] https://chaoss.community/

```
    "perceval_version": "0.1.0",
    "timestamp": 1476139775.852378,
    "updated_on": 1451929343.0,
    "uuid": "c403532b196ed4020cc86d001feb091c009d3d26"
}
```


Esta herramienta es muy útil para desarrolladores, analistas de datos y cualquier persona interesada en la recuperación de información de diferentes fuentes de datos y su posterior análisis. Además, cualquier persona puede participar en este proyecto aportando soluciones para corregir errores o ideas para nuevos servicios o funciones. También, cuenta con una comunidad de desarrolladores que colaboran continuamente en el mantenimiento y la actualización de sus servicios.

3.2.2. Pymongo

Pymongo[13] es una librería de Python que contiene varias herramientas con las que podemos interactuar fácilmente desde un archivo de Python con MongoDB, siendo este una base de datos NoSQL. Con PyMongo, se pueden realizar una variedad de tareas en MongoDB, como crear y eliminar bases de datos y colecciones, insertar, actualizar y eliminar documentos, realizar consultas y filtrar datos...

3.2.3. Subprocess

El módulo "subprocess"[14] (subproceso), permite ejecutar un proceso que se encuentra dentro de un script de Python a la vez que se ejecuta dicho script. Esto es posible gracias a que este módulo interactúa con el sistema operativo haciendo posible ejecutar y administrar subprocesos directamente desde Python. Este módulo proporciona varias funciones y clases para interactuar con procesos externos con los cuales los desarrolladores pueden ejecutar comandos en la línea de comandos del sistema operativo y manipular la entrada y salida de estos procesos.

[13] https://pypi.org/project/pymongo/

[14] https://docs.python.org/es/3/library/subprocess.html

Capítulo 4

Diseño e implementación

A continuación, describiremos los métodos y herramienta utilizados para llevar a cabo el trabajo. Esto permite a los lectores entender cómo se ha desarrollado el proyecto para realizar el análisis de estudio. En este capítulo se detallan las fases del proyecto, las decisiones y justificaciones de las metodologías escogidas y el diseño e implementación para que otros puedan realizar el mismo análisis.

4.1. Arquitectura general

En la figura 4.1 mostraremos la arquitectura de este proyecto se divide en varias fases. En la primera fase se realiza la recopilación de datos de GitHub mediante la herramienta de Perceval. Con esta herramienta, nos conectamos a a la API de Github para extraer los datos con información de los repositorios y los commits ej documentos JSON. En la segunda fase procesaremos estos datos antes de almacenarlos en la base de datos MongoDB, de tal forma que nos quedamos con los datos que nos interesan para el análisis que vamos a realizar y descartamos el resto de datos. En la tercera fase realizaremos un análisis estadístico de los datos recopilados.

el archivo JSON generado por Perceval se envía al backend para su almacenamiento y análisis posterior. El backend puede ser una base de datos, un sistema de análisis de datos o cualquier otro sistema que el usuario haya configurado para manejar la información recopilada.

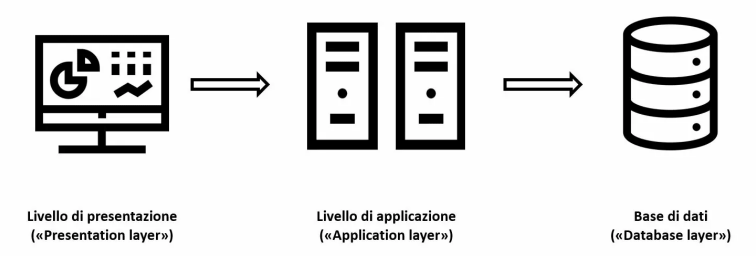

Figura 4.1: Descripción general de Perceval.

4.2. Procedimiento de instalación

Para poder realizar este proyecto es necesario tener intalados WSL (Windows Subsystem for Linux) en Windows y, en dicho Subsistema de Windows, tener Ubuntu como sistema operativo, además de la librería Perceval. A continuación, procederemos a explicar los pasos a seguir para realizar las instalaciones necesarias correctamente.

4.2.1. Instalación de WSL

En primer lugar, comprobaremos que nuestro ordenador cumple con los requisitos previos a la instalación, ya que para instalar WSL se necesita tener Windows 10 versión 2004 o posterior y nuestro hardware debe ser compatible con la virtualización de Hyper-V.

Para verificar la versión de Windows, se debe abrir la Configuración y después seleccionar “Sistema” y, por último hacer clic en “Acerca de”.

Para comprobar si nuestro procesador es compatible con Hyper-V debemos ir al menú de inicio de Windows y buscar “Información del sistema”.

Si cumplimos estos requisitos, lo primero que tenemos que hacer activar los permisos para que en Windows se pueda usar WSL. Para habilitar los permisos para usar WSL en Windows debemos seguir los siguientes pasos:

1. Ir al menú de inicio de Windows y buscar “Activar o desactivar las características de Windows”.

2. Marcar la casilla de verificación de “Subsistema de Windows para Linux” como podemos observar en la figura 4.2.

3. Hacer clic en “Aceptar” y esperar a que finalice el proceso.

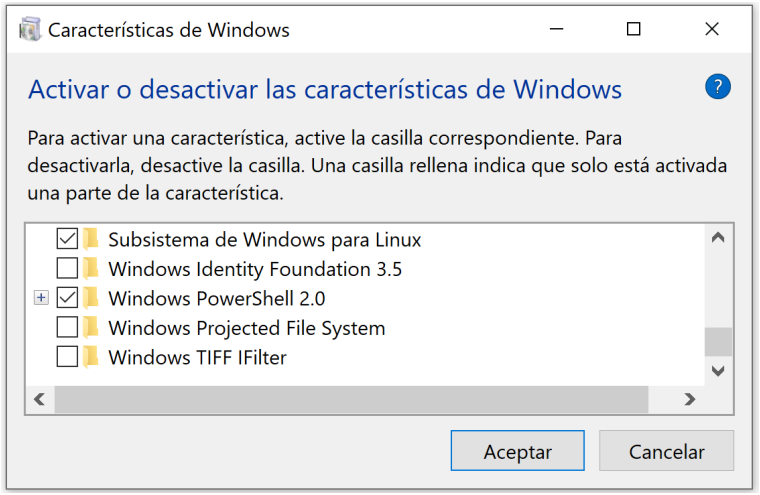

Figura 4.2: Características de Windows.

4. Por último, reiniciar el ordenador para completar la instalación.

Una vez que nos aseguremos de cumplir estos requisitos y tener habilitado WSL en el ordenador procedemos a instalar la distribución de Linux (Ubuntu) desde la Microsoft Store para usar en WSL.

4.2.2. Instalación de Ubuntu

A continuación, descargaremos la versión de GNU/Linux que queramos usar; en nuestro caso hemos descargado la versión Ubuntu 20.04.5 LTS.

Desde la propia tienda de aplicaciones de Windows buscar "Ubuntu", debemos seleccionar la versión deseada y hacer clic en "Obtener", como mostramos en la figura 4.3. Cuando se ha descargado podremos comprobar que Ubuntu se ejecuta con WSL.

4.2.3. Instalación de Perceval

Antes de instalar Perceval tenemos que conectarnos al Subsistema de Windows para Linux ya que instalaremos dicha librería en este entorno. A continuación, para acceder a WSL abriremos Visual Studio Code e instalaremos la extensión WSL que vemos en la figura **??**, que permite obtener toda la productividad de Windows mientras se trabaja con las herramientas y utilidades basadas en Linux. Con esta extensión se puede utilizar VS Code en WSL tal y como se haría desde Windows.

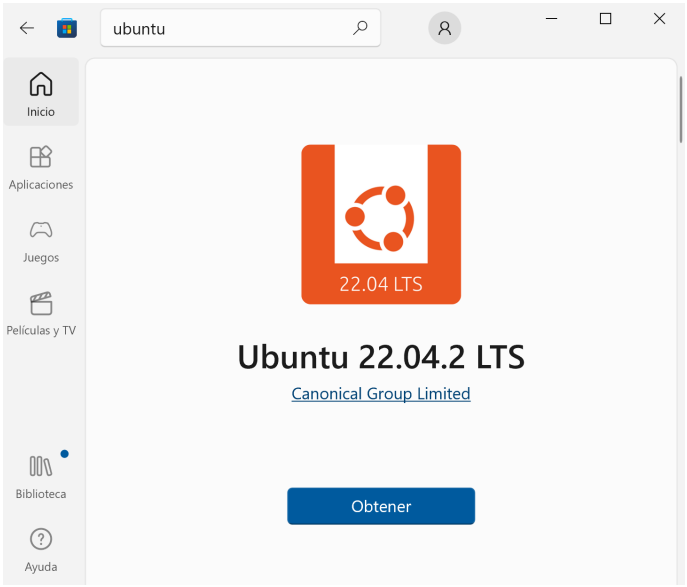

Figura 4.3: Características de Windows.

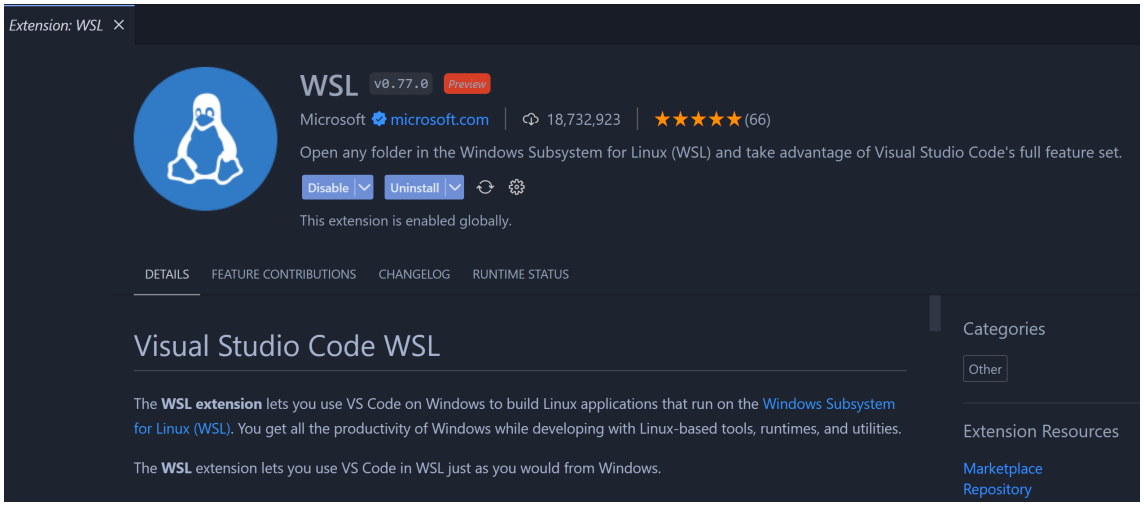

Figura 4.4: Estencion WSL de Visual Studio Code.

Una vez estemos conectados a WSL, pasamos a realizar los siguientes pasos para instalar Perceval en Ubuntu:

1. Instalar la última versión de Python y Pip, si aún no se tiene instalada en el sistema.

2. Instalar las dependencias requeridas para GrimoireLab Perceval utilizando el siguiente comando: *pip install perceval*

Para verificar que la instalación se ha completado con éxito usaremos el comando: *perceval –version* Si este comando devuelve la versión de GrimoireLab Perceval instalada, entonces la instalación se ha completado correctamente y se puede empezar a utilizar Perceval para recopilar datos de diversas fuentes de software.

4.3. Recopilación de datos

En esta parte del proyecto se lleva a cabo la recopilación de una lista de repositorios que utilizan archivos UML y otra en las que no se utilizan archivos UML. Para ello se recurre a Lindholmen dataset que proporciona a los investigadores listas de proyectos de código abierto que usan archivos UML y que se encuentran distribuidos en repositorios de GitHub.

A partir de este conjunto de datos se descarga el archivo “UMLFilesList” para obtener una lista de los repositorios que hacen uso de archivos UML. Este archivo proporciona los enlaces directos a los archivos UML de cada uno de los repositorios, lo que permite verificar que dichos archivos siguen formando parte de dichos repositorios. Con estos enlaces nos cercioramos de que los archivos continuen, cada uno de ellos, siendo parte de los diferentes proyectos, los cuales posteriormente procederemos a analizar.

Para obtener la lista de repositorios en los que no se usen archivos UML se procede a revisar los seguidores de los repositorios obtenidos anteriormente y se selecionan de forma aleatoria algunos de ellos para crear dicha lista.

Una vez que se obtienen ambas listas de repositorios, se procede a descargar los datos de los mismos, así como los de los commits, empleando la herramienta Grimoire Perceval. Esta herramienta soporta más de 30 fuentes de datos de los cuales se pueden obtener datos. En la figura 4.5 vemos un esquema del procedimiento que sigue Perceval. En éste, el cliente escoge

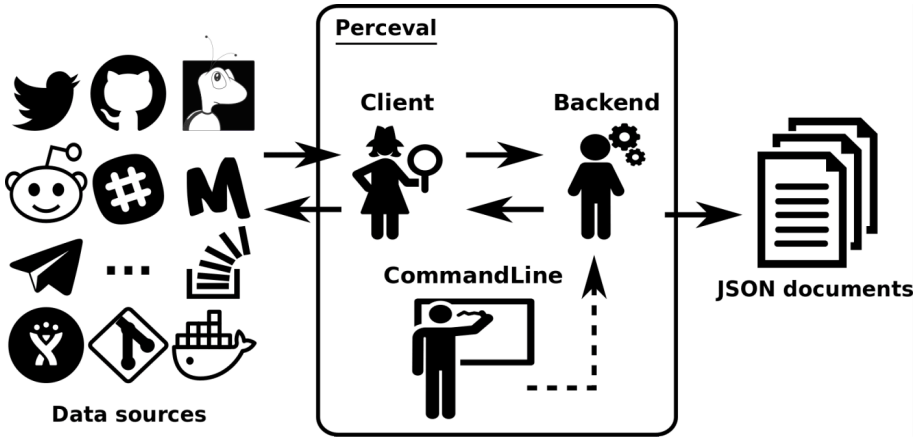

Figura 4.5: Esquema de arquitectura general.

la fuente de la cual desea obtener los datos a analizar y ejecuta Perceval desde la línea de comandos con los parámetros necesarios para recopilar la información de la fuente seleccionada. Posteriormente, Perceval procesa esta información y la almacena en documentos.

En este proyecto se eligió como fuente de datos Github. Partiendo de esta fuente de datos a Perceval se le pasaron los siguientes parámetros: el nombre del repositorio o dirección URL, el nombre y tipo de archivo con el que queremos que se guarden los documentos generados y, además, según el comando usado obtendremos los datos de los repositorios o de otras categorías como, en este caso, los commits, u otras como las issues, los pull request...

```
perceval github --category repository grimoirelab Perceval > repo.json
perceval git https://github.com/grimoirelab/perceval.git > commit.json
```

Gracias a esta herramienta, se pudo obtener información detallada de los repositorios y los commits, lo que resultó fundamental para el desarrollo de este proyecto de investigación. Estos datos se guardaron en archivos JSON, puesto que es una estructura de datos ampliamente utilizada y fácil de comprender. Para llevar a cabo esta tarea usamos una conexion WLS en la cual se instaló tanto Ubuntu como la librería de Perceval; ya que en Windows daba problemas. La intalación tanto de WLS como de Ubuntu y Perceval se han explicado en el apartado 4.2.

Después de finalizar el proceso de obtención de los datos de los repositorios se procedió a almacenarlos en la base de datos MongoDB. Para este fin, se buscó una forma de introducir los archivos JSON de forma automática. Para ello, se uso el módulo de python *subprocess* junto con *mongoimport*, que es una herramienta que se utiliza para importar datos de archivos en formato JSON, CSV o TSV con facilidad desde la línea de comandos. Con *mongoimport*, se pueden

importar grandes cantidades de datos de manera eficiente, además, se permite especificar la base de datos y la colección en la que se quieren almacenar.

Con estas dos herramientas se desarrolló un fichero que introduce todos los documentos JSON, obtenidos anteriormente, para crear la base de datos que vamos a estudiar. En este fichero se ejecuta *mongoimport* con cada uno de los archivos JSON obtenidos, tanto de los de los repositorios como de los commits.

4.4. Procesado de datos

Después de almacenar todos los archivos JSON de la lista de respositorios seleccionados, empezamos a observar los datos obtenidos y se eligieron los más adecuados para analizar los tipos de desarrolladores seleccionados. Posteriormente, se inició el desarrollo de varios ficheros con los que extaer los datos de mayor interés. Al mismo tiempo, se estudió en qué tipo de base de datos se deberían guardar los archivos JSON obtenidos con Perceval. Se optó por una base de datos que no fuese relacional (NoSQL), ya que los datos no estaban relacionados entre sí y este tipo de bases de datos son muy útiles para trabajar con grandes conjuntos de datos distribuidos. Por esta razón, se decidió almacenarlos en MongoDB que es una base de datos NoSQL de código abierto.

Para almacenar estos datos se desarrolló un fichero que se conecta a MongoDB y, a partir de aquí, se crea una nueva coleción y se introducen los datos que le pasamos. A la hora de importar los datos se decidió usar una estructura de diccionario como la que obsevamos a continuación.

```
{
"_id": { <ObjectId>},
"Desarrollador": String,
"Origin": String,
"Owners": String,
"Repositories": String,
"Language": String,
"License": Mixted,
"TypeDeveloper": String,
"Fork": Int32,
"Commit": Int32
}
```

A continuación, se explicaran los datos extraídos de los repositorios y los commits que se han guardado en esta estructura:

- Desarrollador: en este campo se indica si en dicho repositorio se usan archivos UML o no. Por tanto, los valores de este campo pueden ser "UML", si en el proyecto existen archivos de este tipo, o "NOUML", si en el proyecto no existen archivos UML.

- Origin: se refiere a la ubicación del repositorio. Esta información se usa cuando un usuario quiere clonar un repositorio de Github.

- Owners: en este campo se muestra el nombre del propietario del repositorio. Es decir, el usuario o la organización que creó el repositorio y que es responsable de los cambios que se producen en él. Con este campo se puede acceder a otros repositorios que sean propiedad del mismo usuario u organización. Además, también, se puede utilizar para identificar el perfil de GitHub del propietario del repositorio y ponerse en contacto con él.

- Repositories: nos muestra el nombre completo del repositorio, es decir, incluye tanto el nombre del usuario u organización, como el nombre del repositorio en sí, separados por una barra diagonal (/). Esta información nos sirve para identificar el repositorio al que pertenece la información extraída.

- Language: nos muestra el lenguaje de programación principal utilizado en el repositorio. Los valores pueden ser, por ejemplo, "Python", "Java", "JavaScript", "C++", "Ruby", entre otros.

- License: en este campo se muestra la licencia bajo la cual está disponible el código fuente del proyecto. Para que un repositorio sea verdaderamente de código abierto, tendrá que generarse una licencia. Con esta licencia cada propietario define el marco legal en el que se puede usar, distribuir o modificar el proyecto. Algunas licencias pueden ser más restrictivas que otras en cuanto a los derechos que otorgan a los usuarios y desarrolladores de software.

- TypeDeveloper: se indica si el repositorio en cuestión pertenece a un usuario individual o a una organización. Los valores pueden ser "User", si el propietario del repositorio es un usuario individual, o "Organization", si el propietario del repositorio es una organización

y no un usuario individual. Las organizaciones en GitHub están formadas por grupos de personas que trabajan juntas en un proyecto o en varios proyectos relacionados.

- Fork: se indica si ese repositorio es una copia de otro repositorio existente en GitHub. Los valores de este campo pueden ser "true", que significa que el repositorio es una bifurcación o "fork" de otro repositorio en GitHub o "false", que significa que el repositorio no es una copia de ningún otro repositorio de Github. Los usuarios de GitHub utilizan los "fork" para crear nuevas versiones de un proyecto existente para agregar nuevas funcionalidades, corregir errores, o personalizar un proyecto original. Cada vez que se crea un "fork", se crea una copia completa del repositorio original en el que se pueden realizar cambios sin afectar al repositorio original.

- Commit: se muestra el número de commits qua se han realizado en el repositorio. Un "commit" es una acción que se realiza para guardar los cambios realizados en los archivos de un repositorio. Todos estos "commits" se guardan en el historial del repositorio, por tanto, esto permite a los usuarios tener la posibilidad de revisar las modificaciones realizadas y revertir cambios en caso de ser necesario. Por lo tanto, el número de commits nos puede indicar el nivel de actividad de un proyecto, así como la frecuencia de actualizaciones, mejoras y mantenimiento del mismo.

4.5. Análisis de datos

En esta sección se describe el proceso de análisis de los datos obtenidos a partir de Perceval de los distintos repositorios de Github. Se comenzó por la revisión y limpieza de los datos, eliminando aquellos datos que no decidimos incluir en el análisis establecido previamente. Luego, se procedió a la exploración de los datos mediante el cálculo de estadísticos descriptivos para cada una de las variables de interés. Para ello se realizó un fichero en el que se hace el análisis de los datos extraídos usando Python, observando así las diferencias entre los desarrolladores que utilizan uml y el resto. De los 400 datos recogidos la mitad son desarrolladores que usan archivos UML y la otra mitad desarrolladores que no usan archivo UML.

En el fichero desarrollado se han analizado los siguientes puntos de cada uno de los repositorios:

4.5.1. Licencia del repositorio

Se realizará un proceso de análisis estadístico de los repositorios para saber si tienen o no licencia. Para cada repositorio, se ha accedido a el valor "Licencia" de los documentos JSON obtenidos con Perceval y se ha registrado dicha información correspondiente.

En este análisis lo primero será extraer los valores almacenados en la base de datos en el campo de licencia de cada uno de los documentos JSON que hemos generado para guardar los puntos a analizar de cada desarrollador. Posteriormente, usaremos la siguiente función () en la cual si no se encuentra ninguna información de la licencia en el valor de "Licencia" del objeto JSON, el repositorio no tiene licencia.

$$Licencia = \{\, 1\ sielrepositoriotieneunalicencia 0encualquierotrocaso \tag{4.1}$$

4.5.2. Propietario del repositorio

En este caso se procederá a estudiar el promedio de los tipos de propietarios de los desarrolladores UML y del resto de los desarrolladores. Los posibles tipos de propietarios que puede haber son "User" si es un usuario único o "Organization" si se trata de una organización compuesta por varios usuarios. Esta información se muestra también en los documentos JSON obtenidos de los repositorios.

$$Propietario = \{\, U\ sersielpropietariodelrepositorioesunusuario Organizationsielpropietariodelreposit \tag{4.2}$$

Después de haber definido los propietarios de cada uno de los repositorios, se procederá a realizar la media de todos ellos para conocer si los repositorios que contienen archivos UML suelen ser propiedad de organizaciones o de usuarios únicos. A continuación se muestra la fórmula usada para realizar las medias de los repositorios cuyos propietarios son usuario o desarrolladores:

$\bar{X} = \frac{1}{400} \sum_{i=1}^{400} x_i$

4.5.3. Existencia de fork del repositorio

$$Fork = \{ T\ ruesielrepositoriotieneunforkFalseencualquierotrocaso \tag{4.3}$$

Una vez que se han definido los repositorios que tiene fork y los que no se procederá a realizar la media de todos ellos para conocer si los desarrolladores UML suele realizar fork con respecto al resto de los desarrolladores. A continuación se muestra la fórmula usada para realizar la media de los repositorios que han realizado un fork:

$\bar{X} = \frac{1}{400} \sum_{i=1}^{400} x_i$

4.5.4. Lenguaje de programación principal del repositorio

$\bar{X} = \frac{1}{400} \sum_{i=1}^{400} x_i$

4.5.5. Cantidad total o frecuencia de los commits del repositorio

$\bar{X} = \frac{1}{400} \sum_{i=1}^{400} C_i$

Datos	Desarrolladores UML	Desarrolladores NO UML
Con licencia		
Usuario		
Organización		
Con fork		
JavaScript		
Python		
C		
C++		
Java		
PHP		
Ruby		
Pascal		
Html		
Swift		
Otros		
Num commits		

Capítulo 5

Experimentos y validación

Este capítulo se introdujo como requisito en 2019. Describe los experimentos y casos de test que tuviste que implementar para validar tus resultados. Incluye también los resultados de validación que permiten afirmar que tus resultados son correctos.

Capítulo 6

Resultados

En este capítulo se incluyen los resultados de tu trabajo fin de grado. Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

En este capítulo se presentan los resultados obtenidos tras la realización del estudio. Se han llevado a cabo distintos análisis estadísticos para evaluar la validez de las hipótesis planteadas en el capítulo anterior. Para ello, se ha utilizado una muestra de 200 participantes, divididos equitativamente entre hombres y mujeres. A continuación, se exponen los resultados obtenidos para cada una de las variables estudiadas y se interpretan los hallazgos más relevantes.

Capítulo 7

Conclusiones

7.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

Y si has llegado hasta aquí, siempre es bueno pasarle el corrector ortográfico, que las erratas quedan fatal en la memoria final. Para eso, en Linux tenemos aspell, que se ejecuta de la siguiente manera desde la línea de *shell*:

```
aspell --lang=es_ES -c memoria.tex
```

7.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a

2. b

7.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. Aquí viene uno.
2. Aquí viene otro.

7.4. Trabajos futuros

Ningún proyecto ni software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

Bibliografía

[1] J. Astigarraga and V. Cruz-Alonso. ¡ se puede entender cómo funcionan git y github! *Ecosistemas*, 31(1):2332–2332, 2022.

[2] T. Bray. The javascript object notation (json) data interchange format. Technical report, 2014.

[3] I. Challenger-Pérez, Y. Díaz-Ricardo, and R. A. Becerra-García. El lenguaje de programación python. *Ciencias Holguín*, 20(2):1–13, 2014.

[4] P. Martínez Juliá. Software libre, linux y ubuntu. *Eubacteria, nº 17 (2006)*, 2006.

[5] S. Medri. Forensic analysis of windows subsystem for linux on windows 11.

[6] J. A. Mora-Castillo. Serialización/deserialización de objetos y transmisión de datos con json: una revisión de la literatura. *Revista Tecnología en Marcha*, 29(1):118–125, 2016.

[7] J. R. M. Ríos, N. M. L. Mora, M. P. Z. Ordóñez, and E. L. L. Sojos. Evaluación de los frameworks en el desarrollo de aplicaciones web con python. *Revista latinoamericana de Ingenieria de Software*, 4(4):201–207, 2016.

[8] K. E. Rodríguez and J. A. Delgado. Edición de textos científicos en latex. 2014.

[9] I. H. P. Tavera et al. Software libre (ubuntu). *Vida Científica Boletín Científico de la Escuela Preparatoria*, (4):1, 2013.