Universidad
Rey Juan Carlos

ESCUELA DE INGENIERÍA DE FUENLABRADA

Ingeniería en Tecnologías de la Telecomunicación

TRABAJO FIN DE GRADO

REDISEÑO DE APLICACIÓN DE ANÁLISIS DE CÓDIGO PYTHON

Autor : José Matas Luque

Tutor : Dr. Gregorio Robles Martínez

Curso académico 2024/2025

Trabajo Fin de Grado

Rediseño de aplicación de análisis de código Python

Autor : José Matas Luque

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2024, siendo calificada por el siguiente tribunal:

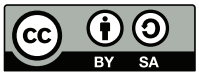
Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente **Calificación:**

Fuenlabrada, a de de 202X



II

Agradecimientos

Quisiera expresar mi más sincero agradecimiento a todas aquellas personas que, de una forma u otra, me han acompañado durante estos años, siendo parte de este largo camino cuyo fin llega con la realización de este proyecto.

En primer lugar, quiero agradecer profundamente a mi familia, cuyo amor y apoyo me han otorgado el ánimo y confianza necesarios en esta etapa.

A mis amigos y compañeros, quienes han vivido conmigo este viaje académico, gracias por los momentos compartidos y por todo lo que me han permitido aprender.

Por último, quisiera agradecer también a los profesores que han contribuido a mi formación durante este ciclo, permitiéndome llegar hasta aquí. En particular un especial agradecimiento a mi tutor, Gregorio Robles, por la paciencia y la ayuda dada durante el desarrollo de este proyecto.

Resumen

El objetivo principal de este proyecto es rediseñar y extender la funcionalidad de la ya existente aplicación “pycefrl”, con el fin de hacerla más accesible y cómoda de usar e interpretar para el usuario. Esta herramienta tiene como labor realizar análisis de código en el lenguaje Python para obtener un nivel de complejidad ajustado a los niveles CEFRL (Common European Framework of Reference for Languages), el cual se ha convertido en un estándar en Europa.

El desarrollo de este proyecto ha involucrado numerosas tecnologías, siendo las principales los lenguajes de programación Python, en lo referente al análisis del código, y JavaScript, para la visualización web de los resultados, junto con HTML y CSS. Han sido de vital importancia también el formato de datos JSON, para la manipulación de la información, y, por supuesto, GitHub para alojar tanto el código trabajado como el código a analizar.

Summary

The primary objective of this project is the redesign and extend de functionality of the existing “pycefrl” applicaction, with the aim of making it more accessible and user-friendly. This tool performas code analysis in the Python programming language to determine a complexity level aligned with the CEFRL (Common European Framework of Reference for Languages), which has become a standard in Europe.

The development of this project has involved numerous technologies, with the primary ones being Python for code analysis and JavaScript for web-based visualization of the results, along with HTML and CSS. The JSON data format has also been crucial for information manipulation, along with of course GitHub, which has been used to host both the worked-on code and the code to be analyzed.

Índice general

Índice de figuras

Capítulo 1

Introducción

El Marco Común Europeo de Referencia para las Lenguas (CEFRL) es el estándar ampliamente aceptado en Europa y otras regiones para evaluar las competencias lingüísticas de un hablante. En este marco las habilidades son clasificadas utilizando una escala de seis niveles, yendo desde A1 (básico) hasta C2 (maestría), permitiendo obtener una referencia clara sobre el progreso de los estudiantes en el dominio de un idioma. Este estándar ha sido popularizado no sólo en entornos educativos, sino también en el entorno laboral, donde a menudo resulta crucial contar con competencias de idiomas.

Partiendo de esto, el proyecto original que sirve de punto de partida para este trabajo buscaba aplicar este sistema de evaluación al popular lenguaje de programación Python. El objetivo era analizar distintas construcciones o patrones de código, asignándoles un nivel CEFR acorde a los elementos encontrados. Esto ofrecía una forma ágil de medir el nivel de competencia de un programador, brindando una herramienta útil para educadores o desarrolladores.

Mi contribución a este proyecto ha consistido en una extensión y rediseño del mismo que amplía su funcionalidad y mejora significativamente la experiencia de usuario. Mientras que el análisis de archivos Python dentro de un repositorio sigue siendo una parte clave de la herramienta, se han introducido mejoras que optimizan de manera notable su uso. Entre estas mejoras destacan una interfaz web más intuitiva, una visualización clara de los resultados a través de la consola y la inclusión de opciones configurables que permiten personalizar ciertos aspectos del proceso. Estas adiciones hacen que la herramienta sea sobre todo más accesible y flexible, adaptándose mejor a distintos entornos educativos y profesionales.

Además, las modificaciones implementadas en esta versión no sólo han mejorado la fun-

cionalidad actual, sino que también han facilitado futuras ampliaciones del proyecto. De este modo, la herramienta queda preparada para incorporar nuevas características o adaptaciones sin grandes esfuerzos de reestructuración.

Con estas mejoras mi trabajo ofrece una visión renovada del proyecto original, manteniendo su esencia de análisis detallado y preciso, pero con una usabilidad y experiencia de usuario muy superior. La herramienta se convierte así en una solución más completa para aquellos usuarios que busquen evaluar y entender el nivel de competencia en Python de manera estructurada y eficiente.

1.1. Estructura de la memoria

La memoria de este trabajo está estructurada de la siguiente manera:

- **Capítulo 1: Introducción.** Explicación del contexto del proyecto realizado y uuna brevemente introducción a los objetivos del mismo.
- **Capítulo 2: Objetivos.** Explicación del objetivo general del proyecto, así como el listado de los objetivos más específicos y la planificación seguida durante el desarrollo.
- **Capítulo 3: Estado del arte.** Explicación de las tecnologías y herramientas involucradas en el proyecto.
- **Capítulo 4: Diseño e implementación.** Explicación detallada del proceso de desarrollo del proyecto.
- **Capítulo 5: Validación y pruebas.** Explicación de las pruebas realizadas durante el desarrollo para validar los resultados obtenidos.
- **Capítulo 6: Resultados.** Explicación y muestra del estado final de la aplicación, incluyendo ejemplos de uso.
- **Capítulo 7: Conclusiones.** Reflexión sobre el estado final del proyecto, analizando el cumplimiento de los objetivos propuestos y planteando posibles futuras extensiones.
- **Capítulo 8: Manual de usuario.** Explicación de uso del proyecto, más concisa que la realizada en Resultados.

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo general del trabajo es realizar una revisión, reestructuración y extensión de funcionalidad del proyecto ya existente. Dicho proyecto se puede definir como una herramienta de análisis de código escrito en lenguaje Python, otorgando un nivel al código de acorde al estándar CEFR (Common European Framework of Reference).

2.2. Objetivos específicos

Los objetivos específicos de este trabajo se pueden definir de la siguiente manera:

- Optimizar el código existente para asegurar que el programa cumpla con los requisitos mínimos necesarios para su desarrollo futuro.
- Implementar una funcionalidad de "plug-and-play" que permita a cualquier usuario descargar y utilizar el programa con una configuración mínima.
- Incorporar un paquete de requisitos que incluya todas las dependencias y módulos necesarios para la correcta ejecución de la aplicación.
- Mejorar la visualización de los resultados en formato JSON, facilitando su análisis posterior.

- Desarrollar una interfaz web para la visualización de resultados, proporcionando una forma intuitiva de navegar entre los análisis realizados.
- Implementar la visualización de resultados en la consola al finalizar el proceso de análisis.
- Añadir registros detallados durante el proceso de análisis, para que el usuario pueda monitorizar el progreso y evitar la incertidumbre sobre el estado de ejecución en proyectos de larga duración.
- Reestructurar la arquitectura del código, reorganizando el árbol de directorios para asegurar una disposición lógica y coherente de los archivos, de acuerdo con los estándares recomendados en la industria.
- Eliminar redundancias en el código e incorporar documentación detallada.
- Ampliar el análisis con información adicional sobre el repositorio GitHub examinado, incluyendo datos relevantes sobre el autor y otros metadatos.
- Ampliar el análisis con nuevos elementos del lenguaje Python no considerados en la primera versión.

2.3. Planificación temporal

El tiempo total asignado para el desarrollo del trabajo ha sido de seis meses. Durante los primeros meses, debido a compromisos laborales y las asignaturas del curso universitario, el tiempo disponible para dedicar al proyecto fue limitado, concentrándose principalmente en los fines de semana.

Dado que el proyecto se basa en la extensión de una aplicación preexistente, la primera fase consistió en comprender la estructura y funcionalidad del código ya implementado. Este proceso requirió más tiempo del estimado inicialmente, debido a la cantidad de errores, la falta de organización y la ausencia de documentación adecuada en el código existente.

Concluida esta fase de análisis, la primera tarea fue implementar la funcionalidad “plug-and-play”. Sin embargo, este proceso se prolongó más de lo previsto debido a las dificultades para comprender el comportamiento del sistema durante la ejecución, entre otros motivos porque inicialmente presentaba fallos críticos que impedían su correcto funcionamiento.

Al finalizar el segundo mes de desarrollo, y tras haber establecido una base mínima operativa, pude realizar las primeras pruebas de ejecución, verificando los procesos a medida que los desarrollaba. Durante el tercer mes, me enfoqué en reescribir el código, eliminando una considerable cantidad de elementos redundantes, ineficientes e incluso obsoletos, garantizando además que tanto el código como la documentación cumplieran con los estándares de la industria.

Hacia mediados del cuarto mes, gran parte de la lógica de análisis ya había sido implementada. Aunque el desarrollo aún no estaba completamente finalizado, el progreso fue suficiente para iniciar pruebas del proceso en su totalidad, en lugar de continuar con las pruebas aisladas por partes realizadas previamente. A medida que comenzaba a obtener las primeras versiones de los resultados de los análisis, procedí a trabajar en la interfaz web y en la visualización de dichos resultados.

Durante el quinto mes, me enfoqué en finalizar la implementación del análisis y en realizar mejoras puntuales en diversas partes del proyecto que, aunque funcionales, podían optimizarse para mejorar tanto el rendimiento como la experiencia del usuario.

El sexto y último mes estuvo dedicado principalmente a la redacción de la memoria del proyecto. En cuanto al código, se realizaron ajustes menores para perfeccionar ciertos detalles, aunque ninguno de ellos representó un cambio significativo en el funcionamiento general del programa.

Capítulo 3

Estado del arte

3.1. Python

Python es un lenguaje de programación de alto nivel ampliamente utilizado en el desarrollo de software. Es un lenguaje referente para tareas relacionadas con el análisis de datos, y el reciente auge de la IA ha aumentado todavía más su popularidad. La simplicidad y legibilidad del código, añadida a su gran polivalencia, lo convierte en una excelente opción para desarrolladores de cualquier nivel.

Una de sus mayores virtudes es la presencia de una amplia biblioteca estándar y un ecosistema de paquetes más que robusto, el cual ofrece herramientas para prácticamente cualquier necesidad, desde desarrollo web hasta inteligencia artificial, doctrina muy popularizada hoy en día.

En el contexto de este proyecto Python ha sido utilizado como lenguaje principal para desarrollar la lógica del análisis, aprovechando su capacidad para manejar estructuras complejas de datos, así como su compatibilidad con bibliotecas de análisis, las cuales han sido cruciales para la tarea. La flexibilidad del lenguaje ha permitido una rápida iteración y mejora del código, facilitando también la integración con otras tecnologías.

3.2. HTML

HTML (HyperText Markup Language) [5] es el lenguaje de marcado estándar para la creación y estructuración de documentos en la web. Define la estructura básica de las páginas web

mediante el uso de etiquetas y atributos que organizan el contenido en elementos como encabezados, párrafos, listas, enlaces y multimedia. HTML proporciona el esqueleto de las páginas web, permitiendo a los navegadores interpretar y mostrar el contenido de manera coherente y estructurada.

En concreto, se utilizó HTML5, la versión más reciente del lenguaje, que introduce mejoras significativas con respecto a sus predecesores. HTML5 incluye nuevas etiquetas semánticas que mejoran la estructura del documento y la accesibilidad como `header`, `footer`, `article` y `section`. Estas etiquetas no sólo proporcionan una estructura más clara y lógica para los documentos, sino que también facilitan la optimización para motores de búsqueda (SEO) y la integración con tecnologías modernas. Además, HTML5 soporta de manera nativa elementos multimedia como `video` y `audio`, eliminando la hasta ahora necesaria incorporación de plugins adicionales y permitiendo una experiencia de usuario más rica y fluida.

En este proyecto, HTML ha sido utilizado para construir y organizar la interfaz web, a fin de ofrecer una presentación moderna y funcional de los resultados del análisis.

3.3. CSS

CSS (Cascading Style Sheets) es el lenguaje de hoja de estilos utilizado para describir la presentación de un documento escrito en HTML. CSS permite la separación de la estructura de una página web de sus estilo visual, facilitando el mantenimiento y escalabilidad del diseño. Con CSS es posible definir elementos como colores, fuentes, espaciado y la disposición de los componentes de la página, logrando un diseño responsivo que se adapta a diferentes dispositivos y resoluciones de pantalla.

En este proyecto, CSS ha sido empleado para mejorar la apariencia de la interfaz web, garantizando que los resultados visualizados fueran estéticamente atractivos y fáciles de interpretar. Además, CSS ha permitido personalizar el diseño para asegurar que la interfaz fuera totalmente responsiva, adaptándose a dispotivos móviles y de escritorio sin pérdida de funcionalidad o usabilidad.

3.4. JavaScript

JavaScript [6] es uno de los lenguajes de programación más extendidos a día de hoy. Es utilizado principalmente para el desarrollo de aplicaciones web, aunque no se limita únicamente a ese uso. En ese ámbito suele ir acompañado de HTML y CSS, donde HTML conforma el esqueleto de la página, CSS la define la apariencia y JavaScript la lógica de los elementos.

Durante décadas JavaScript ha impulsado la interactividad de las páginas web modernas y lo largo de los años ha ido evolucionado con frameworks y librerías que amplían sus funcionalidades y simplifican gran parte de los procesos, facilitando el desarrollo de aplicaciones complejas tanto en el lado del cliente como en el servidor.

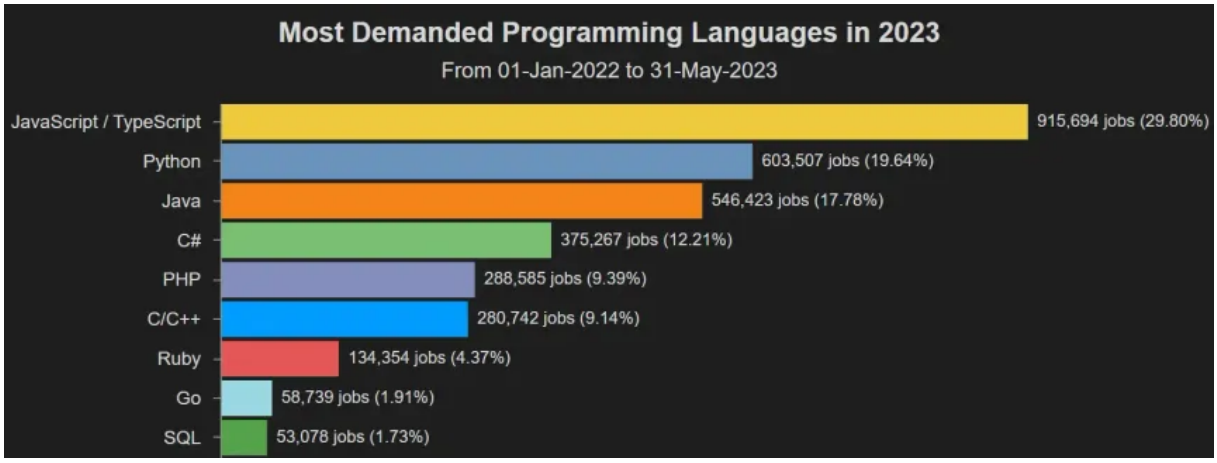


Figura 3.1: Lenguajes de programación más usados en 2023. Fuente: DevJobsScanner [2]

En este proyecto, JavaScript se ha utilizado para diversos propósitos:

1. **En el servidor:**

   - **Manejo de rutas y archivos:** Se ha utilizado el framework Express.js para controlar el manejo de rutas y la creación de un servidor web. Express facilita la configuración del middleware para procesar solicitudes HTTP y servir archivos estáticos. En el código proporcionado Express se usa para definir rutas que manejan solicitudes tanto para la API como para la entrega de archivos HTML y recursos estáticos, como imágenes, CSS y JavaScript.
   - **Manejo de archivos:** Se utilizan módulos nativos de Node.js para leer y escribir archivos. En el código se leen archivos JSON y HTML y se ayuda a resolver rutas

de archivos y directorios, asegurando la correcta ubicación de los archivos en el sistema.

- **API de resultados:** Se implementan rutas para servir resultados de análisis almacenados en archivos JSON. Esto incluye la lectura de directorios y el envío de datos JSON a través de una API RESTful, que permite que el cliente recupere información sobre los resultados de forma dinámica.

2. **En el cliente:**

   - **Manipulación del DOM:** JavaScript se usa para modificar dinámicamente el contenido y estilo de la página web según los datos recibidos del servidor. Por ejemplo, se crean y actualizan elementos HTML, se gestionan tablas y se aplican estilos en función de las propiedades de los datos.
   - **Interactividad:** Se implementan funcionalidades interactivas como la ordenación de tablas y la actualización dinámica de contenido. Estas tablas se pueden ordenar basándose en las interacciones del usuario y se ocultan o muestran elementos de la interfaz según los datos recibidos.
   - **Llamadas a la API:** El código del cliente realiza solicitudes HTTP a la API del servidor. Los datos devueltos se utilizan para iniciar o actualizar la interfaz de usuario, mostrando información relevante sobre los repositorios y sus resultados de análisis.

En resumen, el papel de JavaScript en este proyecto es crucial tanto en el lado del servidor como en el del cliente. La integración con Express y la manipulación dinámica del DOM (Document Object Model) permiten una experiencia de usuario fluida, cómoda y eficiente con los datos almacenados. Esta combinación de tecnologías ayuda a que la aplicación web sea interactiva, receptiva y funcional, cumpliendo con los estándares modernos de desarrollo web.

3.5. Node.js

Node.js [4] es un entorno de ejecución diseñado para ejecutar código JavaScript en el lado del servidor. Node.js ha ganado popularidad debido a su capacidad para crear aplicaciones escalables y su extenso ecosistema de paquetes, accesible a través de npm (Node Package Manager).

En este proyecto Node.js ha sido empleado como el entorno principal para el servidor frontend, permitiendo ejecutar el código de visualización de resultados y gestionar las interacciones con la interfaz web. Se utilizó para:

1. **Ejecución de código de frontend:** Node.js ejecuta el código que gestiona la lógica del servidor, incluyendo la carga de archivos y la configuración de la API.

2. **Interacción con Express.js:** Node.js permite la integración con Express.js para manejar las rutas y las solicitudes HTTP, ofreciendo una plataforma robusta para el desarrollo del servidor.

3. **Manejo de datos:** La arquitectura no bloqueante de Node.js es clave para manejar eficientemente grandes volúmenes de datos, optimizando el rendimiento del servidor.

3.6. Express

Express [3] es un framework web minimalista para Node.js, diseñado para crear aplicaciones web y APIs de manera sencilla y eficiente. Nos ofrece una estructura flexible y una serie de herramientas que nos permiten gestionar rutas, peticiones HTTP y el middleware necesario para agregar funcionalidades adicionales a una aplicación. Al ser ligero y altamente configurable Express resulta ideal para construir aplicaciones escalables y robustas en entornos de desarrollo modernos.

En el contexto de este proyecto Express ha sido fundamental para:

1. **Definir rutas y middleware:** Express.js ha facilitado la configuración de rutas para manejar tanto la API como la entrega de recursos estáticos y dinámicos, permitiendo una correcta visualización de los resultados analizados.

2. **Facilitar la comunicación cliente-servidor:** La integración de Express ha simplificado la gestión de las comunicaciones entre el cliente y el servidor, optimizando la entrega de datos y la actualización de la interfaz de usuario.

3.7. JSON

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos cuya fortaleza radica en que es fácil de leer y escribir para los humanos, así como sencillo de interpretar y generar por las máquinas. Su estructura basada en pares clave-valor lo hace ideal para transportar datos entre un servidor y una aplicación web, o incluso entre distintos componentes de un mismo sistema. JSON es ampliamente utilizado en APIs y servicios web debido a su simplicidad y compatibilidad con múltiples lenguajes de programación.

Por ejemplo, si quisiéramos almacenar la información de una persona podríamos utilizar una estructura similar a:

Figura 3.2: Ejemplo de objeto con formato JSON

En este proyecto JSON ha sido crucial en dos aspectos:

1. **Almacenamiento de los resultados:** Los resultados del análisis se almacenan en archivos JSON. Este formato ha permitido una organización clara y estructurada de los datos, facilitando su posterior manipulación y visualización.

2. **Intercambio de datos con la API de GitHub:** La información devuelta por las llamadas HTTP a la API de GitHub también está en formato JSON. Esto ha permitido integrar de manera eficiente los datos externos en el sistema, asegurando una comunicación fluida entre las distintas partes del proyecto.

La capacidad de JSON para representar objetos de manera sencilla ha sido esencial para la eficiencia y flexibilidad del proyecto.

3.8. VSCode

Visual Studio Code (VSCode) es un editor de código fuente desarrollado por Microsoft en abril de 2015. Desde entonces ha sido ampliamente adoptado por la comunidad de desarrolladores debido a su ligereza y versatilidad.

VSCode ofrece una experiencia de desarrollo robusta con características como resaltado de sintaxis, autocompletado inteligente, plena integración con Git y una muy amplia selección de extensiones para mejor la productividad y añadir funcionalidades.

En este proyecto VSCode ha sido empleado como el entorno principal de desarrollo debido a sus numerosas ventajas:

1. **Soporte para múltiples lenguajes:** A diferencia de otras herramientas VSCode tiene la capacidad de manejar tanto JavaScript, como Python y otros lenguajes clave del proyecto. Esta flexibilidad ha sido de vital importancia para facilitar el trabajo con el backend, frontend y la manipulación de datos.

2. **Extensiones y depuración:** Se han utilizado extensiones específicas para los lenguajes empleados, lo que ha facilitado y acomodado el proceso de desarrollo.

3. **Integración con Git:** La integración nativa de Git en VSCode ha permitido gestionar fácilmente el control de versiones del proyecto, permitiendo recuperar de manera sencilla trozos de código que han sido editados.

En definitiva, la flexibilidad y ligereza de VSCode lo convierte en una herramienta esencial para el desarrollo ágil y eficiente del proyecto, ayudando a mantener un flujo de trabajo organizado y estable.

3.9. GitHub

GitHub es una plataforma que facilita la gestión de proyectos utilizando Git, un sistema de versiones más extendido. Permite almacenar código en repositorios, donde varios desarrolladores pueden colaborar sin sobrescribir los cambios de otros mediante el uso de varias ramas, entre otras funcionalidades.

En este proyecto se ha utilizado GitHub para gestionar las versiones de código, teniendo la capacidad de retroceder, ver y recuperar cambios realizados con anterioridad. Además, puesto que el programa desarrollado tiene como objetivo realizar análisis de repositorios alojados en GitHub, hemos utilizado también la API que proporciona para obtener información relevante de los mismos.

3.10. Trello

Trello es una herramienta de gestión de proyectos basada en el navegador que utiliza la metodología Kanban, la cual permite organizar tareas de forma visual e intuitiva mediante un sistema de tableros, listas y tarjetas. Los usuarios pueden mover tarjetas entre columnas para reflejar el estado de una tarea, facilitando la visualización del progreso de proyectos.

En este proyecto Trello ha sido utilizado para organizar las diferentes etapas de desarrollo, permitiendo mantener un registro de todas las tareas a realizar y el estado de las mismas.

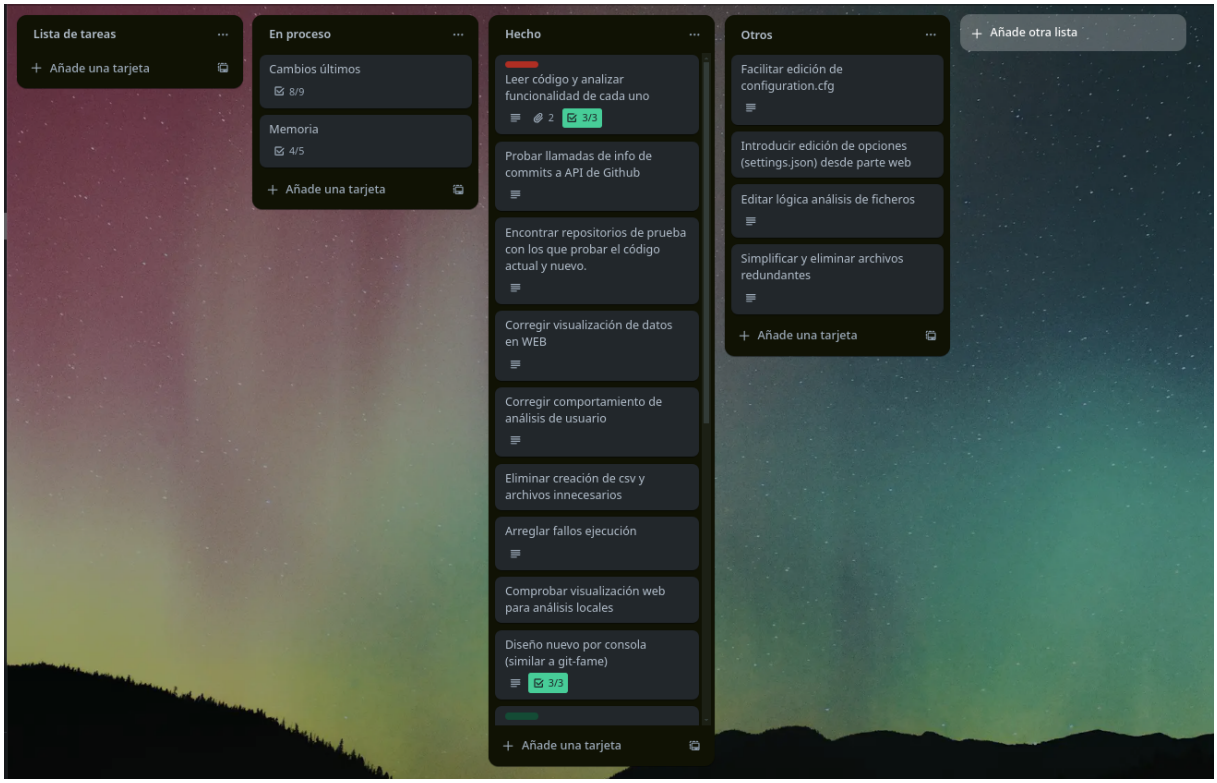


Figura 3.3: Tablero de Trello utilizado

En la figura 3.3 podemos ver el tablero de Trello empleado durante este proyecto, incluyendo

muchas de las tarjetas empleadas distribuir las tareas a realizar.

3.11. Postman

Postman [1] es una herramienta utilizada para probar y desarrollar APIs, que permite a los desarrolladores enviar peticiones HTTP a un servidor y verificar las respuestas de manera intuitiva. Su interfaz gráfica facilita la creación de peticiones `GET`, `POST`, `PUT`, `DELETE`, entre otras, y ofrece funcionalidades avanzadas como la creación de colecciones, pruebas automáticas y la posibilidad de compartir entornos de trabajo.

En el contexto de este proyecto, Postman ha sido utilizado para:

1. **Pruebas de APIs:** Se realizaron pruebas exhaustivas de las APIs desarrollados con Express y Node.js. Postman permitió verificar que las respuestas fueran correctas y que los datos se gestionaran adecuadamente en cada punto de la comunicación entre el cliente y el servidor.

2. **Depuración y optimización:** La herramienta facilitó la identificación de errores y la optimización del rendimiento de las APIs. Su uso fue fundamental para asegurar la robustez y la funcionalidad de los endpoints antes de su implementación final.

Postman ha sido de especial importancia para garantizar la calidad y fiabilidad del sistema al permitir una evaluación detallada de cada llamada HTTP realizada.

3.12. Pycefr

Pycefr es una herramienta de análisis de código Python, que permite realizar análisis de repositorios de código alojados en GitHub. Para ello descarga el código y recaba los elementos de Python existentes, otorgándoles una puntuación siguiendo el estándar CEFRL (Common European Framework of Reference for Languages). Los niveles de cada elemento fueron asignados mediante votaciones de usuarios, quienes otorgaban a cada uno de estos elementos de Python una dificultad entre A1 y C2.

Realizado el análisis, obtenemos varios archivos de resultados con el número de elementos de cada tipo y su nivel. Pycefr otorga también la posibilidad de crear archivos HTML con

el objetivo de facilitar la visualización de estos resultados, listándonos todos los elementos encontrados junto con su respectiva puntuación.

Pycefr sido de especial relevancia en este proyecto, pues el propósito de éste ha sido su rediseño y extensión.

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

El objetivo de este proyecto es corregir y extender la funcionalidad, así como añadir una visualización web cómoda e intuitiva de los resultados obtenidos. Para ello hemos seguido los siguientes pasos:

4.2. Obtención y comprensión del código existente

El primer paso es muy básico pero indispensable. Debemos realizar una copia del repositorio original para obtener así un código del que partir que podamos modificar sin alterar el original. Una tarea no complicada que quedó solventada rápidamente mediante el uso de Git y GitHub. Hecho esto, podemos ponernos manos a la obra y comenzar a trabajar en el que a partir de ahora será nuestro código.

La mejor manera de comprender rápidamente qué hace un programa es ejecutándolo e ir, mediante ingeniería inversa, extrayendo qué está ocurriendo a medida que avanza la ejecución. Sin embargo, al intentar esto nos encontramos con que el programa presentaba una gran cantidad de errores y ni siquiera realizaba la labor que supuestamente debía.

De este modo, hubo que virar el enfoque hacia una tarea diferente, volviéndose primordial el conseguir que el programa funcionase. Esto supuso un reto, porque en este punto todavía no estaba familiarizado con el código y desconocía qué debía hacer suponiendo que hiciese lo que debía, lo cual no era el caso.

4.3. Creación de un registro de tareas

El siguiente paso fue crear un nuevo tablero de Trello, del cual ya hemos hablado, para comenzar a recoger las tareas que iban apareciendo. Al principio del desarrollo estas tareas eran tan básicas como "Revisión y comprensión del código" y se volverían más específicas a medida que aumentara nuestro conocimiento sobre el proyecto.

Se realizaron diagramas de flujo para representar los proceso, con el fin de intentar entender y dejar anotado qué se realiza, dónde y cuándo.

4.4. Solución de problemas de ejecución

Comencé a editar pequeños fragmentos de código, a fin de intentar solucionar lo que pensaba en ese momento que eran los errores que impedían la correcta ejecución, pero a medida que unos se resolvían otros aparecían. Finalmente opté por rehacer el grueso del proyecto, dejando intacto de momento únicamente el proceso de análisis, es decir, la parte del programa que, dado cierto texto con código, lee y clasifica los elementos relacionados con Python que encuentra en el mismo.

De la parte del proyecto encargada se tomar y analizar los archivos sólo parecía funcionar (no daba error al instante) una de las tres opciones, la de analizar un repositorio dado un enlace de GitHub. Las otras dos, analizar un usuario de GitHub y analizar un directorio local, las dejé temporalmente de lado para centrarme en la primera. Si lograba hacer funcionar una de ellas, las otras dos no debían ser muy diferentes.

Toda esta parte conforma lo que en el proyecto he denominado "backend", dando a entender que es el código encargado de hacer el procesamiento y cálculos necesarios para obtener los resultados. La otra gran parte del proyecto, el llamado "frontend", sufría el mismo problema y daba errores de ejecución. De nuevo opté por rehacer el archivo principal que manejaba esa parte, obteniendo así unas primeras versiones ejecutables del código.

4.5. Reestructuración y mejoras preparativas

Una vez establecida una base ejecutable realicé unas últimas revisiones al código para asegurar el entendimiento del mismo. Además, antes de comenzar a implementar las nuevas fun-

cionalidades se realizaron múltiples mejoras para facilitar el trabajo futuro:

- Renombramiento y reelaboración de funciones para adecuarse a los estándares y simplificar su edición a futuros usuarios, utilizando nombres más representativos que expliquen rápidamente su finalidad.
- Añadida documentación.
- Reestructuración general del proyecto, diferenciando el código de backend y frontend en sus respectivas carpetas.
- Añadidos archivos de requisitos que el usuario puede utilizar para instalar rápidamente todo lo necesario para ejecutar el proyecto.
- Añadidos logs para que el usuario sepa en todo momento qué está realizando el programa. La demora del análisis en proyectos de gran tamaño pueden dar a entender al usuario que se ha parado cuando no es así. Estos logs vienen acompañados de mensajes de error descriptivos que indican al usuario por qué ha fallado el programa en caso de hacerlo.

4.6. Implementación de nuevas funcionalidades

Establecidas estas mejoras podemos comenzar a implementar desarrollos que extiendan las funcionalidades del proyecto actual. Puesto que el proyecto consta de dos grandes ramas, las agruparemos de acorde a cuál de éstas pertenecen:

4.6.1. Backend

En lo relativo al backend encontramos la parte del proyecto encargada de procesar el input del usuario con el fin de recabar los datos necesarios y realizar el análisis que convenga. Los cambios más relevantes en esta parte son los siguientes:

- **Extensión del análisis del repositorio** , incluyendo información relevante sobre el historial del mismo y de los usuarios que han participado en él. En caso de analizar un directorio local el programa comprobará antes si forma parte de un repositorio en GitHub. En caso de ser así, preguntará al usuario si prefiere analizar directamente el repositorio en

cuestión en lugar de los archivos locales. Las ventajas que tiene esto son (1) el directorio local puede no estar actualizado con los últimos cambios subidos al repositorio, de modo que al analizar directamente la versión de GitHub nos aseguramos de estar analizando la última versión compartida, y (2) nos añade a los resultados del análisis esa información referente a GitHub, en lugar de únicamente la del código.

- **Mejora del archivo resultante del análisis** para facilitar su posterior tratamiento. Originalmente se generaban dos archivos equivalentes en formatos JSON y CSV. Aunque ambos formatos son ampliamente utilizados para el análisis de datos, hemos optado por mantener únicamente el formato JSON debido a su mayor flexibilidad para estructurar y manipular información. A diferencia del CSV, que se ajusta mejor a datos tabulares con campos claramente definidos y fijos, el JSON permite cambios más dinámicos en la estructura de los datos sin la necesidad de realizar ajustes constantes para que los campos se adapten a un formato rígido. Esto hace que el JSON sea más adecuado para futuras reestructuraciones del archivo, evitando la complejidad asociada a modificar archivos CSV cuando la estructura de los datos cambia significativamente. Al prescindir del archivo CSV, simplificamos el proceso de manejo y actualización de los datos, permitiendo una mayor versatilidad en su tratamiento y reduciendo las dificultades asociadas al ajuste de formatos estáticos como el CSV. En la siguiente figura podemos ver la estructura del nuevo archivo JSON generado:

```
{
    "elements": [
        {
            "class": "Clase de elemento 1",
            "level": "Nivel de elemento 1",
            "numberOfInstances": Numero de instancias
        },
        {
            "class": "Clase de elemento 2",
            "level": "Nivel de elemento 2",
            "numberOfInstances": Numero de instancias
        },
        ...
    ],
    "repoInfo": {
        "data": {
            "name": "Nombre del repositorio",
            "url": "URL del repositorio",
            "description": "Descripción del repositorio",
            "createdDate": "Fecha y hora de creación",
            "lastUpdateDate": "Fecha y hora de última actualización",
            "owner": {
                "name": "Nombre del autor",
                "avatar": "Imagen de perfil del autor",
                "profile_url": "URL del perfil GitHub del autor"
            }
        },
        "commits": [
            {
                "name": "Nombre del autor",
                "github_user": "Nombre de usuario GitHub del autor",
                "loc": Líneas de código modificadas,
                "commits": Número de commits realizados,
                "total_hours": Número de horas invertidas (estimación),
                "total_files_modified": Número total de archivos modificados
            }
        ],
        "contributors": [
            {
                "name": "Nombre del contribuidor 1",
                "avatar": "Imagen de perfil del contribuidor 1",
                "profile_url": "URL del perfil GitHub del contribuidor 1",
                "commits": Numero total de commits del contribuidor 1
            },
            {
                "name": "Nombre del contribuidor 2",
                "avatar": "Imagen de perfil del contribuidor 2",
                "profile_url": "URL del perfil GitHub del contribuidor 2",
                "commits": Numero total de commits del contribuidor 2
            }
        ]
    }
}
```

Figura 4.1: Estructura del JSON generado

Como se aprecia en la figura 4.1, el archivo se divide en dos grandes categorías.

1. **elements:** Contiene una lista de los elementos encontrados. Se mencionan el tipo (class), nivel (level) y número de apariciones (numberOfInstancias) de cada uno.

2. **repoInfo:** Contiene la información relativa al repositorio. A su vez se divide en tres grupos:

   - **data:** Información general del repósitorio. Encontramos el nombre del repositorio (name), la URL (url), la descripción en caso de tenerla (description), la fecha de creación (createdDate), la fecha del último cambio (lastUpdateDate) y sobre el autor del repositorio el nombre (name), imagen de perfil (avatar) y URL de su perfil (url_profile).
   - **commits:** Información de los commits del repositorio. Incluye una lista de los autores que han realizado al menos un commit, dando de ellos su nombre (name), el nombre de usuario de GitHub (github_user), las líneas de código modificadas en total (loc), el número total de commits (commits), una estimación del tiempo invertido (total_hours) y el total de archivos modificados (total_files_modified). Cabe mencionar que la estimación del total de horas se realiza en función del número total de commits y del tiempo que haya pasado entre ellos, realizando una media. Este valor puede no ser preciso en caso de que el número de commits sea muy bajo o haya mucha variación de tiempo entre ellos. Por último, las propiedades *name* y *github_user* suelen tener el mismo valor, pero el usuario tiene la posibilidad de cambiar esto, de modo que es necesario añadir ambos campos.
   - **contributors:** Información de los usuarios GitHub que han participado en este proyecto. Acompaña al apartado interior, pero la información se extrae de manera distinta, es por ello que está separada. Incluye una lista de los usuarios que han realizado al menos un commit, dando de ellos su nombre (name), imagen de perfil (avatar), URL del perfil (profile_url) y número total de commits (commits). El número de elementos en este apartado suele coincidir con el de commits, pero puede ser diferente por varios motivos.

     El primero de ellos es que en este apartado, debido a la respuesta obtenida tras solicitar esta información, se utiliza como *name* lo que en *commits* es *github_user*. Esto tiene como consecuencia que si un mismo usuario de GitHub utilizara nombres (*name* en *commits*) diferentes, en *commits* habría dos entradas para este usuario, mientras que en contributors sólo una, pues es el mismo

usuario.

El segundo motivo es que ciertos procesos automáticos de GitHub, como la aprobación de un *pull request*, queda registrado como realizado por GitHub en lugar de por el usuario. Esto se hace usando un name distinto, pero manteniendo el *github_user* del autor.

En la figura 4.2, donde se muestra la sección "repoInfo" del archivo resultante de realizar el análisis en nuestro propio repositorio, se pueden ver estos dos casos. En el proyecto han realizado cambios tres usuarios: el autor del proyecto original, anapgh, el tutor de este proyecto, gregoriorobles y un servidor, PepeM112. En el apartado de *contributors* sí aparecen bien representados estos tres usuarios, mientras que en *commits* encontramos un total de cinco. Los dos usuarios extras son los que tienen como *name* los valores "jmatas" y "GitHub". El primero de éstos refleja que yo he realizado cambios utilizando varios name distintos, mientras que el segundo refleja el caso de las acciones que se registran como hechas por GitHub, a pesar de incluir el *github_user* del autor.

```
"repoInfo": {
        "data": {
            "name": "pycefr",
            "url": "https://github.com/PepeM112/pycefr",
            "description": null,
            "createdDate": "2024-05-02T10:26:27Z",
            "lastUpdateDate": "2024-10-01T15:46:24Z",
            "owner": {
                "name": "PepeM112",
                "avatar": "https://avatars.githubusercontent.com/u/129164725?v=4",
                "profile_url": "https://github.com/PepeM112"
            }
        },
        "commits": [
            {
                "name": "PepeM112",
                "github_user": "PepeM112",
                "loc": 217,
                "commits": 1,
                "total_hours": 2.0,
                "total_files_modified": 3
            },
            {
                "name": "jmatas",
                "github_user": "PepeM112",
                "loc": 20162,
                "commits": 81,
                "total_hours": 23,
                "total_files_modified": 59
            },
            {
                "name": "anapgh",
                "github_user": "anapgh",
                "loc": 21355,
                "commits": 227,
                "total_hours": 38,
                "total_files_modified": 32
            },
            {
                "name": "GitHub",
                "github_user": "anapgh",
                "loc": 2663,
                "commits": 19,
                "total_hours": 4,
                "total_files_modified": 19
            },
            {
                "name": "Gregorio",
                "github_user": "gregoriorobles",
                "loc": 1107,
                "commits": 8,
                "total_hours": 2,
                "total_files_modified": 8
            }
        ],
        "contributors": [
            {
                "name": "anapgh",
                "avatar": "https://avatars.githubusercontent.com/u/60195957?v=4",
                "profile_url": "https://github.com/anapgh",
                "commits": 246
            },
            {
                "name": "PepeM112",
                "avatar": "https://avatars.githubusercontent.com/u/129164725?v=4",
                "profile_url": "https://github.com/PepeM112",
                "commits": 82
            },
            {
                "name": "gregoriorobles",
                "avatar": "https://avatars.githubusercontent.com/u/842692?v=4",
                "profile_url": "https://github.com/gregoriorobles",
                "commits": 8
            }
        ]
    }
```


Figura 4.2: Sección ‘‘repoInfo” del análisis del repositorio pycefr

- **Añadida visualización de resultados desde la terminal**, en caso de que el usuario prefiera una vista rápida de los resultados sin tener que depender de la parte web.

```
pycefr ⚡ main
> python3 pycerfl.py -d .
A valid Git configuration has been detected. Would you like to analyse the origin repository? (Y/n) n
[✓] Analysing code
[✓] Saving data

|-------------------------------------------------------------------------|
|                              ANALYSIS                                   |
|-------------------------------------------------------------------------|
| Element                                                | Level |  Number |
|:-------------------------------------------------------|:-----:|--------:|
| Simple List                                            |  A1   |      40 |
| Simple Attribute                                       |  A2   |     714 |
| Simple Assignment                                      |  A1   |     370 |
| Import                                                 |  A2   |      24 |
| From                                                   |  A2   |      11 |
| If statements using → __name__ == '__main__'           |  B2   |       4 |
| Simple If statements                                   |  A1   |     179 |
| Function                                               |  A1   |       9 |
| Simple Tuple                                           |  A1   |       9 |
| 'range' call function                                  |  A2   |       4 |
| Files → 'open' call function                           |  A2   |      17 |
| Generator Expression                                   |  C1   |       8 |
| Assignment with sum (total = total + 1)                |  A1   |      15 |
| Simplified incremental Assignment with increase amount |  A2   |      49 |
| 'raise' exception                                      |  B1   |       1 |
| 'pass' statement                                       |  B1   |       8 |
| Simple For Loop                                        |  A1   |      37 |
| 1 Nested For Loop                                      |  A2   |       9 |
| Exception → try/except                                 |  B1   |      12 |
| With                                                   |  B1   |      18 |
| Function with Simple argument                          |  A1   |      78 |
| Return                                                 |  A1   |      34 |
| Simple Dictionary                                      |  A2   |      33 |
| 1 List Dictionary                                      |  B1   |       2 |
| 1 Nested Dictionary                                    |  B1   |       3 |
| Print                                                  |  A1   |      56 |
| 'enumerate' call function                              |  C2   |       1 |
| Files → 'read' call function                           |  A2   |       2 |
| Simple List Comprehension                              |  C1   |       3 |
| List Comprehension with 1 If statements                |  C1   |       2 |
| For Loop with Tuple as name                            |  A2   |       6 |
| While with Else Loop                                   |  B1   |       3 |
| 'break' statement                                      |  B1   |       3 |
| 'continue' statement                                   |  B1   |       3 |
| Recursive Functions                                    |  B2   |       1 |
| Function with Simple argument with Default argument    |  A2   |       3 |
| Lambda                                                 |  B1   |       2 |
| 1 Nested List                                          |  A2   |       2 |
| Import with 'as' extension                             |  B1   |       1 |
| Simple Class                                           |  B1   |       1 |
| Import 're' module                                     |  C1   |       1 |
|-------------------------------------------------------------------------|
| Total A1                                                        827 |
| Total A2                                                        874 |
| Total B1                                                         57 |
| Total B2                                                          5 |
| Total C1                                                         14 |
| Total C2                                                          1 |
|-------------------------------------------------------------------------|


Results file can be found in /home/pepe/Documentos/pycefr/results/pycefr_local.json
```

Figura 4.3: Vista de resultados por consola tras análisis de directorio local

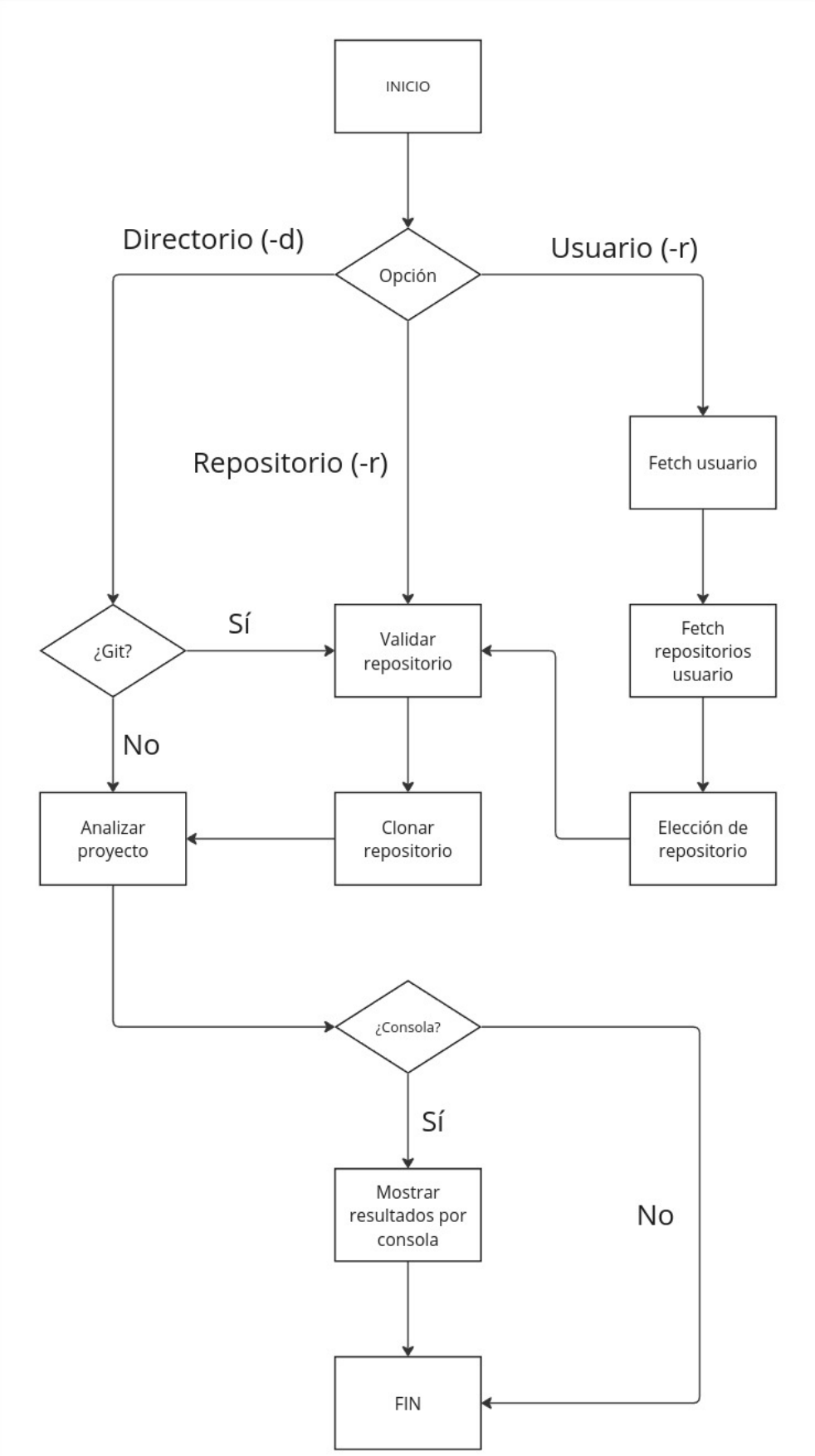


Figura 4.4: Flujo del backend (análisis)

En la figura 4.3 podemos ver un ejemplo de ejecución del análisis de un directorio local. Introducimos la opción *-d* para analizar un directorio local, y la dirección *","* para referirnos al repositorio en que nos encontramos actualmente, esto es, el de la propia aplicación.

Se nos muestra un primer mensaje indicando que se ha encontrado una configuración Git válida, preguntándonos si estamos interesados en analizar el repositorio original. Este mensaje se corresponde con el primer paso del flujo de ejecución mostrado en la figura 4.4. Se comprueba si el código contiene información relativa a Git, y en caso de ser así, da al usuario la opción de analizar el repositorio directamente de GitHub, en lugar de analizar el directorio local. En este ejemplo se ha indicado al programa que no, de modo que procede con el análisis del proyecto, mostrando los pasos realizados *.Analysing code"* y *"Saving data"*, y finalmente los resultados por consola.

Finalmente se añadieron también otras dos opciones de ejecución: La primera de éstas es

- **Consola (-c)** para imprimir por consola resultados ya existentes, sin necesidad de realizar de nuevo el proceso de análisis.
- **Lista (-l)** para listas los archivos de resultados ya existentes.

De este modo, en cualquier momento el usuario puede obtener un listado de los resultados ya existentes y visualizar por consola uno de ellos mediante la siguiente secuencia de comandos:

```
python3 pycefrl.py -l
python3 pycefrl.py -c <RESULTADOS>
```

4.6.2. Frontend

Utilizamos el término "frontend" para la parte del proyecto encargada de la web. En nuestro caso usamos la web para la navegación y visualización de los archivos producidos como resultado de los análisis.

La versión original del frontend es funcional, en el sentido de que cumple su labor, pero presentaba una estructura muy básica y poco modular. Todo el código relevante estaba contenido en un único archivo 'main.js', que generaba dinámicamente un archivo HTML a partir de unas plantillas e insertaba directamente toda la información en la página. Esto no solo dificultaba la escalabilidad del proyecto, sino que también limitaba la separación de responsabilidades entre

las diferentes partes del frontend y hacía más complicado el mantenimiento del código a largo plazo.

En este rediseño completo del frontend hemos implementado una estructura más profesional y escalable. Se han organizado los recursos del proyecto en carpetas separadas, como “public”, “assets”, “html”, “css” y “js”‘, siguiendo las buenas prácticas del desarrollo web moderno. Asimismo, se ha incluido un archivo ‘package.json‘, fundamental para gestionar las dependencias del proyecto y definir los scripts necesarios para su ejecución. El uso de ‘npm‘ como gestor de dependencias permite que el proyecto sea fácilmente reproducible en cualquier entorno, garantizando que todas las bibliotecas y módulos requeridos puedan ser instalados de manera automática y coherente.

Para el servidor, hemos adoptado “Express.js” como framework, creando un archivo ‘server.js‘ dedicado a cargar y gestionar el servidor web, lo que no existía en la versión anterior. También se han implementado rutas específicas para la API (“apiRoutes.js”) y para servir las páginas HTML (“htmlRoutes.js”), lo que proporciona una clara separación entre la lógica de backend y la presentación de los datos. Esto mejora el manejo de peticiones y la organización del código, haciendo más eficiente la gestión del servidor y la comunicación entre las distintas capas de la aplicación.

Además, el uso del “package.json” facilita la futura definición de scripts para tareas comunes como la ejecución del servidor en modo desarrollo, la instalación de dependencias y el despliegue. Esto asegura que todos los colaboradores del proyecto puedan utilizar los mismos comandos y configuraciones, evitando problemas derivados de configuraciones locales.

En resumen, este rediseño transforma el frontend en un proyecto más modular, mantenible y alineado con estándares profesionales, lo que facilita tanto su evolución futura como la colaboración en equipo. La inclusión de ‘Express.js‘, junto con el uso de “npm” y un “package.json” bien definido, asegura que la gestión de dependencias y la ejecución del proyecto sean eficientes y consistentes.

A continuación podemos observar el cambio significativo en la estructura del frontend tras el rediseño. La figura 4.5 muestra el árbol de archivos original, donde todo el código estaba contenido en un único archivo ‘main.js‘. Por el contrario, la figura 4.6 ilustra el nuevo diseño, que adopta una estructura modular y organizada en carpetas, incluyendo archivos específicos para las rutas de la API, los estilos, los scripts, y el uso de ‘Express.js‘ para la gestión del

servidor.

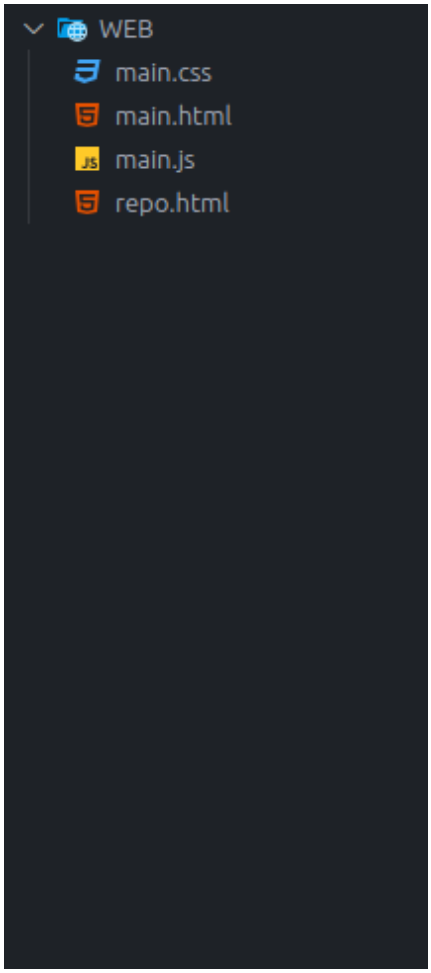


Figura 4.5: Estructura original del frontend

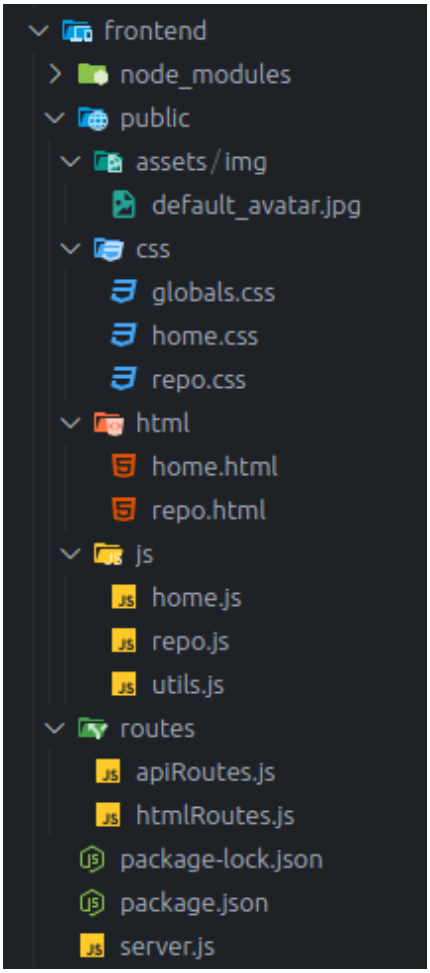


Figura 4.6: Nueva estructura del frontend tras el rediseño

Toda esta reestructuración implica naturalmente un cambio absoluto en el funcionamiento de esta sección. En las siguientes imágenes podemos ver una comparativa de ambos flujos de trabajo:

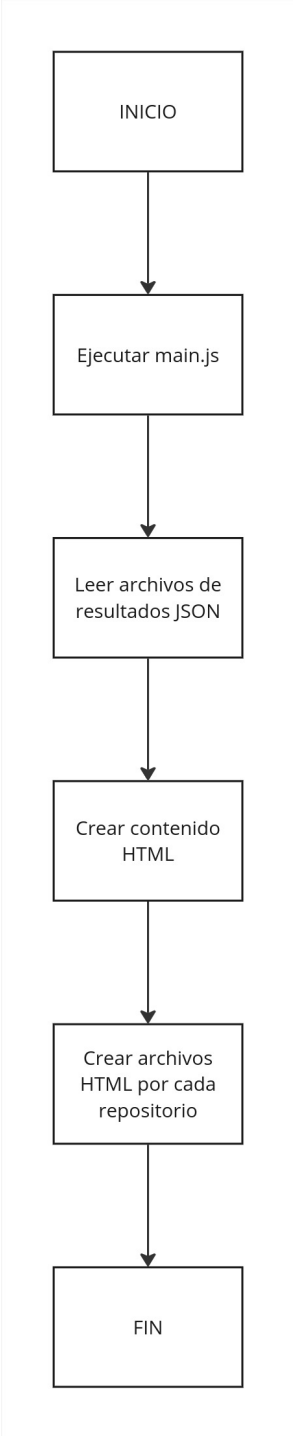


Figura 4.7: Flujo del frontend original

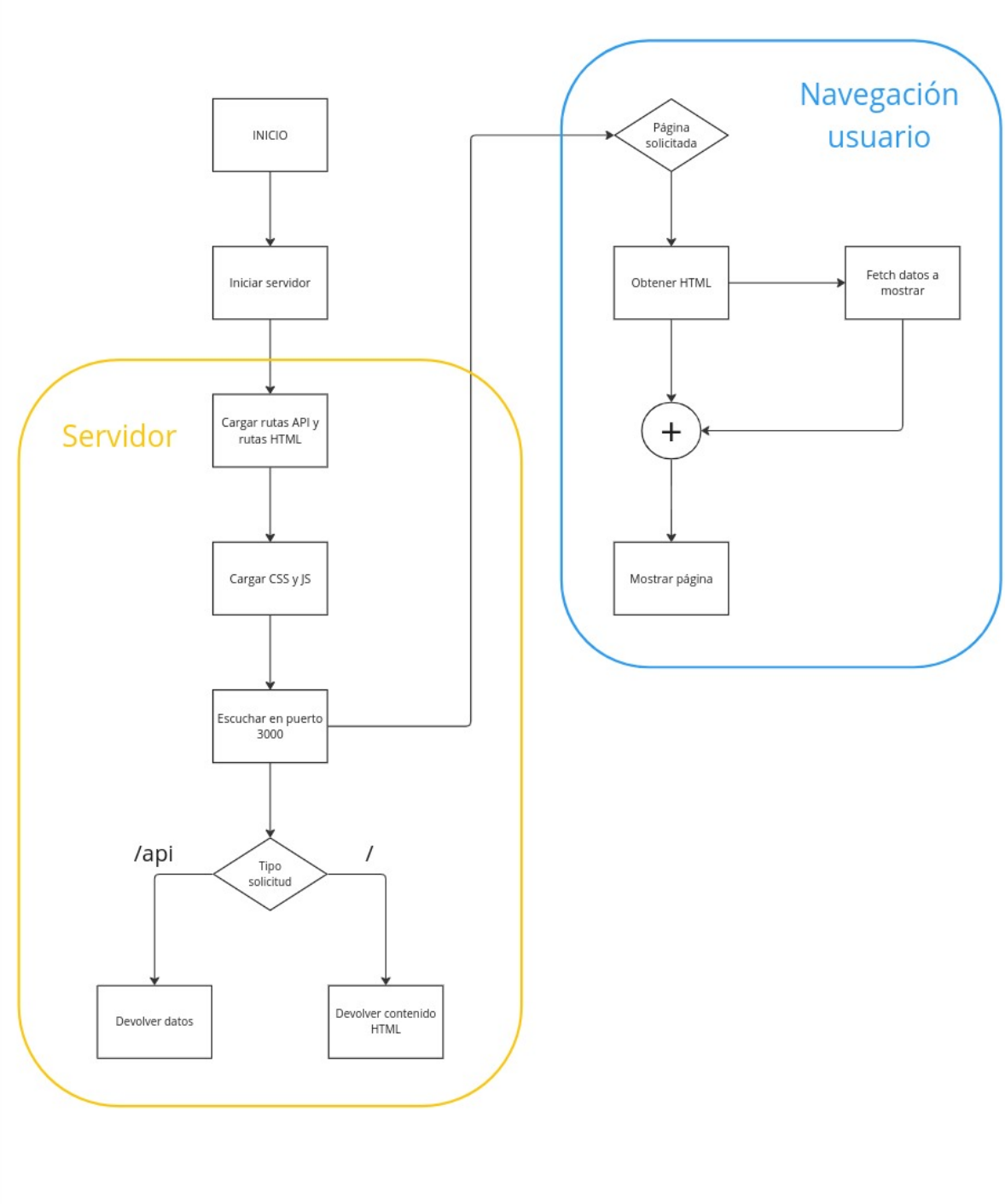


Figura 4.8: Flujo del frontend rediseñado

En el **flujo original**, que podemos ver en la figura 4.7, ejecutamos casi exclusivamente el archivo 'main.js', el cual realiza todo el trabajo. Éste busca en el sistema archivos de resultados JSON y extrae de ellos la información relevante a mostrar. Tras esto carga unas plantillas para crear dos archivos HTML:

- Un archivo 'index.html' que consta de un breve resumen de la información los análisis realizados. Desde aquí se puede acceder a cada análisis individual.

- Un archivo '*nombreDelRepositorioAnalizado*.html' que incluye todo el análisis realizado en el repositorio correspondiente.

Una vez creados podemos visualizarlos levantando un servidor local, aunque esto no es algo incluido en el proceso original, sino que hacerlo por nosotros mismos.

En **flujo actualizado** (ver figura 4.8) ha sido diseñado pensando en la flexibilidad y facilidad para una futura ampliación, añadiendo nuevos procesos y agrupando los ya existentes en archivos encargados de tareas comunes. Así, podemos definir los siguientes pasos en el proceso de ejecución:

1. Comenzamos ejecutando el archivo 'server.js', el cual crea y levanta un servidor que proporcionará todo el contenido necesario, tanto las páginas web como los resultados de los análisis.

2. El servidor se encarga de cargar la lógica de enrutamiento y la api, las cuales se encuentran cada una en sus respectivos archivos (htmlRoutes.js y apiRoutes.js), además de archivos encargados de los estilos y lógica de cada página web.

3. El servidor comienza a escuchar, esto es, esperar a recibir solicitudes. En nuestro caso encontramos dos tipos de solicitudes: **solicitudes de datos** (se piden los resultados de uno o más análisis) y **solicitudes de páginas** (se pide al servidor el contenido HTML a mostrar en un momento determinado).

4. El usuario puede ya comenzar a navegar. Al entrar en la dirección proporcionada por el servidor se dirigirá por defecto a la página principal, la cual incluye el resumen de los análisis existentes, incluyendo la información del repositorio GitHub en caso de incluirla.

5. Al seleccionar uno de los resultados podrá ver toda la información del análisis seleccionado, además de información relevante del repositorio.

En definitiva, uniendo ambas secciones, *backend* y *frontend*, obtenemos la siguiente arquitectura:

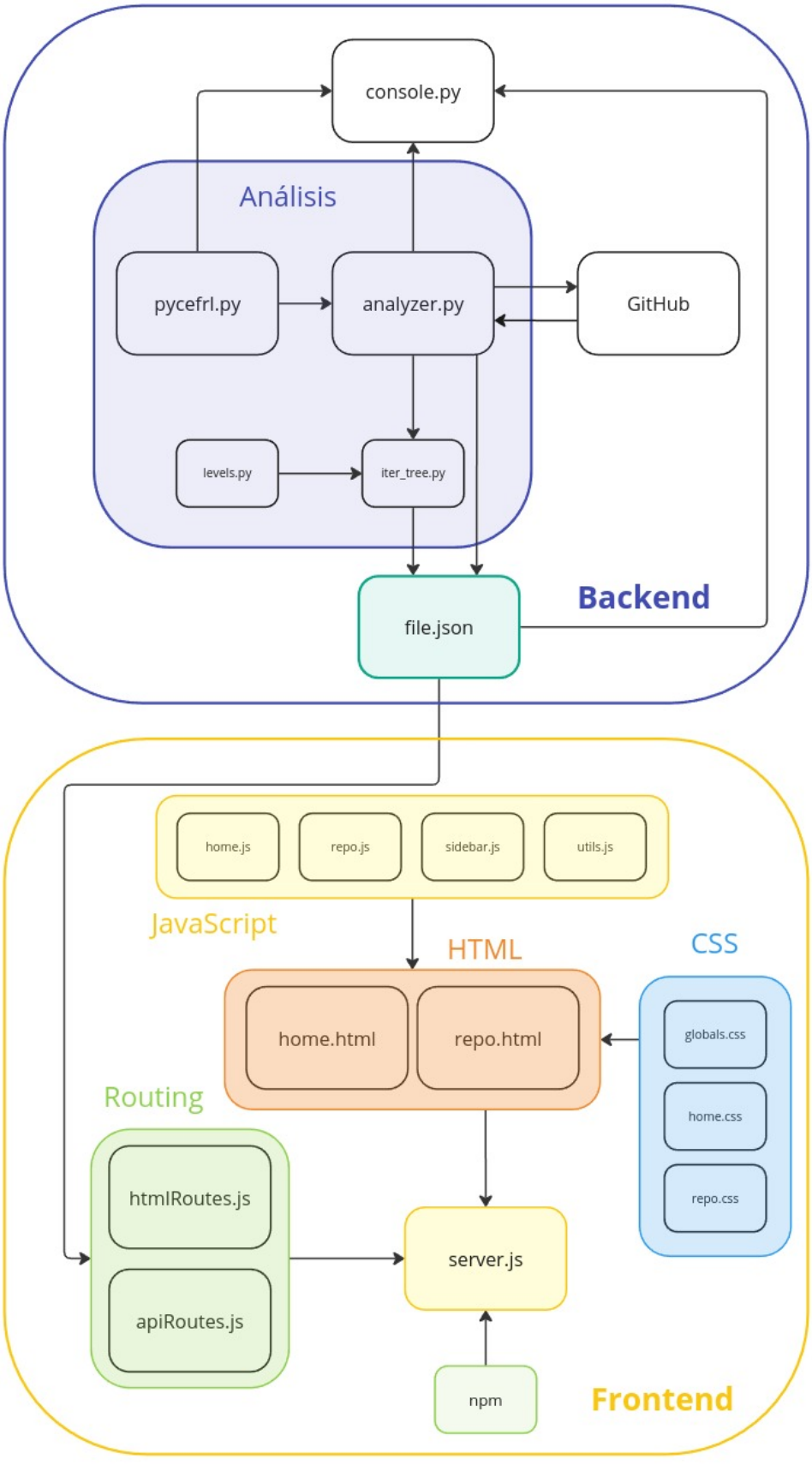


Figura 4.9: Arquitectura del proyecto

En la figura 4.9 podemos visualizar de manera esquemática la arquitectura del proyecto completo, agrupando los archivos involucrado no sólo por la sección a la que pertenecen sino por la tarea que cumplen.

El **backend** contiene los archivos utilizados para realizar el análisis del código. El archivo *pycefrl.py* procesa la entrada del usuario. Interacciona directamente con el archivo *console.py* en caso de que el usuario use el programa para visualizar datos ya existentes, puesto que éste es el archivo que procesa el mostrado de los resultados por consola. Sin embargo, durante su actividad más habitual, la del análisis de archivos, interactúa con *analyzer.py*, quien realiza la gestión principal de este proceso de análisis, sirviéndose primero de la API de GitHub para obtener información en caso de necesitar, y segundo del archivo *iter_tree.py*, para navegar por el código Python a analizar. Éste a su vez se sirve de *levels.py* para asignar los niveles correspondientes a los elementos Python encontrados, produciendo un archivo JSON con los resultados, cuyo nombre se corresponderá con el del proyecto analizado. Finalmente el *analyzer.py* añade al JSON la información relativa a GitHub.

En el **frontend** se incluyen todos los archivos involucrados en el funcionamiento del servidor web. El usuario debe simplemente ejecutar el archivo *server.js*, y éste se encarga de manera automática de utilizar los archivos necesarios para cada tarea. En esta parte podemos diferenciar varios grupos de archivos:

- **Routing:** Archivos encargados de procesar las solicitudes HTTP. *apiRoutes.js* permite al servidor procesar las solicitudes a API, normalmente devolviendo datos en la respuesta. El archivo *htmlRoutes.js* permite al servidor devolver el contenido HTML que mostrar en cada situación. Toda la información de este grupo de archivos depende de los archivos de resultados, los cuales son leídos para devolver la información pertinente.

- **JavaScript:** Archivos JavaScript que contienen la lógica utilizada por los archivos HTML.

- **CSS:** Archivos CSS que definen los estilos de las páginas. Son empleados también por los archivos HTML.

- **HTML:** Archivos HTML que conforman las plantillas de las páginas mostradas. Utilizan la lógica de los archivos JavaScript para obtener los datos y mostrarlos en la página, y aplican los estilos espeficiados en los archivos CSS indicados.

- **npm:** Incluye los módulos de Node.js utilizados en el proyecto.

Capítulo 5

Validación y pruebas

La validación del código añadido o editado se ha ido realizando en la mayor parte de los casos durante la propia codificación. A menudo se han ido probando los cambios a medida que se implementaban en la medida de lo posible. Naturalmente, en la mayoría de ocasiones estas pruebas se hacían por bloques, y no ha sido hasta la fase final del desarrollo (último mes, mes y medio) que se comenzaron a hacer pruebas del proceso completo. Esto se debe a que desde el principio se ha tenido una idea de producto final que presentaba muchas incompatibilidades (en cuanto a código se refiere) con el producto original, y resulta inconveniente, además de una más que ineficiente inversión del tiempo de desarrollo, asegurar la compatibilidad completa de las partes en todo momento.

Como ya ha sido mencionado el comienzo del desarrollo consistió en la creación de una versión estable, puesto que en un principio nos encontramos con múltiples errores de ejecución. Se decidió también, de cara a facilitar la compresión del código, refractorizar gran parte de éste. Esta versión estable se tomó tras realizar los cambios justos y necesarios para solucionar aquellos procesos que producían los errores, y para poder acceder en el futuro rápidamente a ella (para poder comprobar en todo momento el funcionamiento original) se añadió con Git una rama local bajo del nombre “original-fixed” a partir del commit en que se subieron esos últimos cambios. Esto estableció el primer ’ancla’ en el proyecto, proporcionándome un punto de referencia al que poder regresar si más adelante me encontrara con errores críticos difíciles de rastrear, además de permitirme realizar las primeras pruebas, aunque fuera del código original.

Tras esto los cambios se fueron probando poco a poco, a medida que se desarrollaban. Sin embargo, al introducir los cambios que incorporaban llamadas a la API de GitHub se me pre-

sentó la necesidad de probar dichas llamadas de algún modo. Ejecutar el análisis era una opción poco eficiente, dado que se perdería mucho tiempo ejecutando código cuando sólo estamos interesados en la llamada. En ocasiones muy concretas utilicé la herramienta `curl` desde la terminal, que me permite realizar solicitudes HTTP, pero resultaba incómoda cuando se trataba se llamadas complejas.

```
> curl -X GET https://api.github.com/users/PepeM112
{
  "login": "PepeM112",
  "id": 129164725,
  "node_id": "U_kgDOB7LltQ",
  "avatar_url": "https://avatars.githubusercontent.com/u/129164725?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/PepeM112",
  "html_url": "https://github.com/PepeM112",
  "followers_url": "https://api.github.com/users/PepeM112/followers",
  "following_url": "https://api.github.com/users/PepeM112/following{/other_user}",
  "gists_url": "https://api.github.com/users/PepeM112/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/PepeM112/starred{/owner}{/repo}",
  "subscriptions_url": "https://api.github.com/users/PepeM112/subscriptions",
  "organizations_url": "https://api.github.com/users/PepeM112/orgs",
  "repos_url": "https://api.github.com/users/PepeM112/repos",
  "events_url": "https://api.github.com/users/PepeM112/events{/privacy}",
  "received_events_url": "https://api.github.com/users/PepeM112/received_events",
  "type": "User",
  "site_admin": false,
  "name": null,
  "company": null,
  "blog": "",
  "location": null,
  "email": null,
  "hireable": null,
  "bio": null,
  "twitter_username": null,
  "public_repos": 5,
  "public_gists": 0,
  "followers": 0,
  "following": 0,
  "created_at": "2023-03-28T09:18:49Z",
  "updated_at": "2024-09-17T18:50:06Z"
}
```

Figura 5.1: Ejemplo de llamada con *curl*

Como podemos ver en la figura 5.1 la herramienta `curl` tiene como gran ventaja la agilidad que proporciona al poder ser utilizada desde la terminal, de modo que puede resultar de gran ayuda para consultas rápidas. Sin embargo, dado el número de llamadas que debía realizar a lo largo del proyecto realizar las pruebas de esta manera se volvía un proceso tedioso, sobre todo

para llamadas más complejas. Además, una necesidad común durante el desarrollo fue querer acceder de nuevo a las respuestas de llamadas ya probadas anteriormente, obligando a tener que repetir la llamada de nuevo. Es por ello que opté por usar **Postman**.

Postman me proporcionó una interfaz para desarrollar y probar tanto la API de GitHub como la desarrollada por mí para el frontend. Creé una nueva colección que contendría todas las llamadas que se realizan en el proyecto. Esta herramienta me permite añadir y editar fácilmente solicitudes que quiera realizar,con todo tipo de información. Además, permite guardar como ejemplos de ejecución de dichas llamadas, lo cual resolvió el problema de querer acceder de nuevo a los resultados de una llamada. Por ejemplo, a menudo necesitaba comprobar la estructura de la respuesta o el nombre del valor que quería extraer de ésta, y guardar de manera permanente estas respuestas me permitió satisfacer esta necesidad.

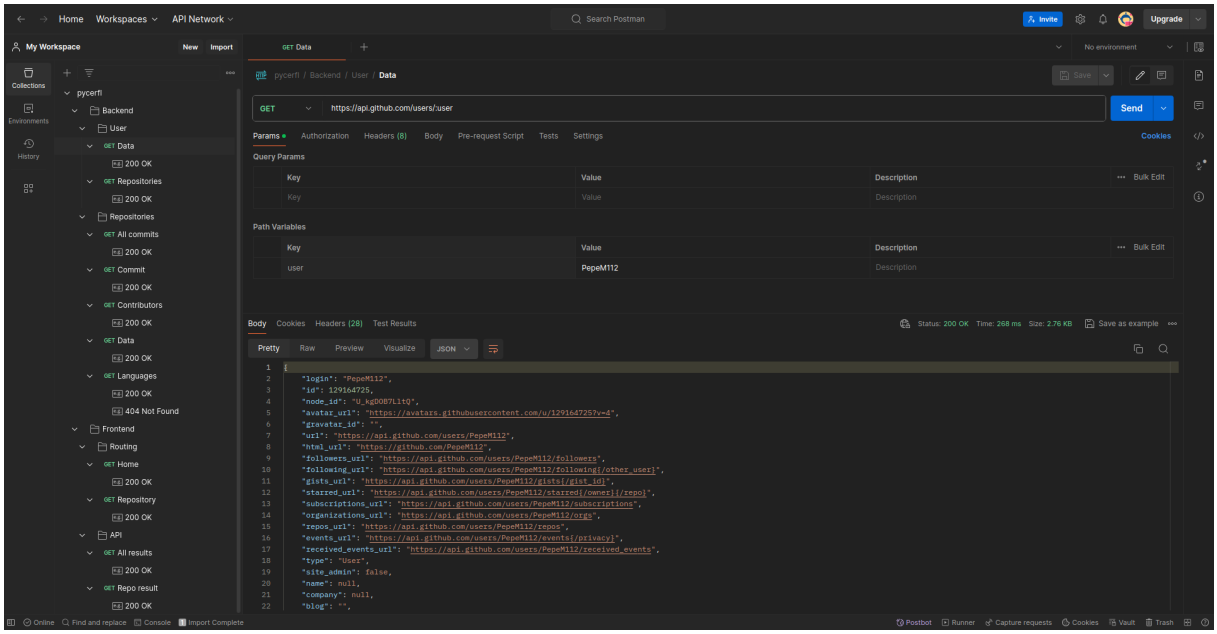


Figura 5.2: Ejemplo de llamada con Postman

En la figura 5.2 podemos ver la interfaz de Postman. En el lado izquierdo encontramos un árbol de la colección de llamadas utilizadas en el proyecto. Con esto podemos simplemente seleccionar la llamada que necesitamos, editarla con la información necesaria (bloque superior), hacerla y visualizar la respuesta (bloque inferior). Esto permitió obtener las respuestas a cualquier llamada de manera rápida y clara. Además, el formato empleado para las respuestas a las solicitudes es JSON, el cual, como ya hemos visto, permite una fácil manipulación de los datos.

En nuestro caso gran parte de las llamadas son a la API de GitHub, quien implementa medidas de seguridad que limitan las solicitudes desde una misma dirección IP si se detecta un número excesivo en un corto período de tiempo. Esto lo hace con el fin para prevenir abusos y proteger la plataforma de posibles ataques automatizados. Es por ello que se incluyó la opción de incluir un token en el archivo de ajustes. En caso de que el usuario exceda el límite de solicitudes se le presentará un mensaje de error indicando qué debe hacer para solventar este problema, como se muestra en la figura 5.3:

```
pycefr ⚡ main
> python3 pycerfl.py -r https://github.com/wohlert/semi-supervised-pytorch
[ ] Validating URL
ERROR: Looks like you've reached the limit of API requests.
To continue, you will need an API key. You can generate one at:
        https://github.com/settings/tokens
and add it to your settings.json file.
Also see: https://docs.github.com/rest/overview/resources-in-the-rest-api#rate-limiting
```

Figura 5.3: Ejemplo de error por exceso de llamadas a la API

En un principio se incluyeron únicamente las llamadas que ahora se encuentran bajo la carpeta *Backend*, pero en al retomar el desarrollo de la parte web surgió la necesidad de añadir las solicitudes realizadas aquí también. En este caso las llamadas son a nuestro propio servidor local que levantamos para mantener la página web, el cual funciona como una API REST y devuelve tanto los datos de los análisis en formato JSON como el HTML necesario para el renderizado de cada página.

Las pruebas finales consistieron en comprobar la correcta visualización de la parte web, lo cual consiste principalmente en levantar el servidor y navegar por las páginas, y finalmente comprobar el proceso completo, realizando primero un análisis y arrancando luego el servidor para comprobar que los datos se muestran en la web.

Capítulo 6

Resultados

En este capítulo procederemos a ver ejemplos de uso de la aplicación. Para ello hemos tomado el repositorio https://github.com/wohlert/semi-supervised-pytorch. Se trata de un repositorio público de GitHub de un tamaño entre pequeño y moderado con el que se han ido realizando numerosas pruebas a lo largo del desarrollo del proyecto.

6.1. Backend

El backend consiste esencialmente en la ejecución del archivo `pycefr.py`. Éste consta de 5 posibles métodos de ejecución, los cuales podemos obtener rápidamente añadiendo el argumento -h o –help:

```
pycefr ⚡ main
> python pycerfl.py -h
usage: pycerfl.py [-h] [-d DIRECTORY] [-r REPO] [-u USER] [-c CONSOLE] [-l]

A utility for analyzing Python level in local and Github repositories.

options:
  -h, --help            show this help message and exit
  -d DIRECTORY, --directory DIRECTORY
                        Run in directory mode
  -r REPO, --repo REPO  Run with the specified repository URL
  -u USER, --user USER  Run with the specified user identifier
  -c CONSOLE, --console CONSOLE
                        See results in console
  -l, --list            List available result files
```

Figura 6.1: Ejemplo de uso de argumento help

En la figura 6.1 se nos muestra el resultado de ejecutar el programa utilizando este argu-

mento -h para obtener ayuda sobre el uso de la aplicación. Comienza indicándonos el formato con que debemos ejecutarlo (usage). pycefrl.py es el archivo que queremos ejecutar, y cada uno de los corchetes siguientes indica un posible argumento a introducir, incluyendo en ocasiones segundos argumentos posibles, como es el caso de -d (DIRECTORY), -r (REPO), -u (USER) y -c (CONSOLE). Esto da a entender al usuario que en estos casos el programa espera un segundo argumento, el cual se corresponde en estos casos con el directorio (-d), repositorio (-r), usuario (-u) a analizar o el archivo de resultados a visualizar (-c).

Entendido esto, procederemos a explicar cada una de las opciones:

6.1.1. Repositorio (-r)

Utilizamos las opciones `-r <REPO_URL>` para realizar un análisis de un repositorio remoto alojado en GitHub.

Podemos ver el resultado de este análisis en la figura 6.2. Hemos introducido como segundo argumento el repositorio sobre el que vamos a realizar el análisis. Como en otros ejemplos hemos utilizado el proyecto *semi-supervised-pytorch* del usuario “wohlert”.

La consola nos muestra primero los mensajes de estado de cada proceso que realiza. Una vez terminado el proceso muestra por pantalla el resultado del análisis, lo cual se puede activar o desactivar editando el archivo de configuración *settings.json*.

```
pycefr ⚡ main
> python3 pycerfl.py -r https://github.com/wohlert/semi-supervised-pytorch
[✓] Validating URL
[✓] Cloning repository
[✓] Analysing code
[✓] Fetching data
[✓] Fetching commits
[✓] Fetching contributors
[✓] Saving data

|-----------------------------------------------------------------------------------------------------|
|                                              ANALYSIS                                               |
|-----------------------------------------------------------------------------------------------------|
| Element                                                                                | Level | Number |
|:---------------------------------------------------------------------------------------|:-------:|---------:|
| Simple List                                                                            |   A1    |       63 |
| Simple Tuple                                                                           |   A1    |       77 |
| 2 Nested Tuple                                                                         |   A2    |        4 |
| 'enumerate' call function                                                              |   C2    |        5 |
| Super Function                                                                         |   C2    |       20 |
| 'range' call function                                                                  |   A2    |       12 |
| Simple Attribute                                                                       |   A2    |      611 |
| Simple List Comprehension                                                              |   C1    |       12 |
| Simple Assignment                                                                      |   A1    |      301 |
| Assignment with sum (total = total + 1)                                                |   A1    |       20 |
| Simplified incremental Assignment with increase amount                                 |   A2    |       14 |
| Import                                                                                 |   A2    |       14 |
| Import with 'as' extension                                                             |   B1    |       15 |
| From                                                                                   |   A2    |       32 |
| Relative From                                                                          |   B1    |        9 |
| Simple If statements                                                                   |   A1    |       35 |
| Simple For Loop                                                                        |   A1    |       12 |
| For Loop with Tuple as name                                                            |   A2    |        7 |
| Function with Simple argument                                                          |   A1    |       56 |
| Return                                                                                 |   A1    |       55 |
| Simple Class using the constructor method → __init__                                   |   B1    |       11 |
| Inherited Class from VariationalAutoencoder using the constructor method → __init__   |   B1    |        2 |
| Inherited Class from DeepGenerativeModel using the constructor method → __init__      |   B1    |        3 |
| 6 Nested Tuple                                                                         |   A2    |        1 |
| 'map' call function                                                                    |   C2    |        1 |
| 'zip' call function                                                                    |   C2    |        2 |
| Function with Simple argument with Default argument                                    |   A2    |       21 |
| Lambda                                                                                 |   B1    |        6 |
| Inherited Class from object using the constructor method → __init__                    |   B1    |        2 |
| Simple Class                                                                           |   B1    |        1 |
| Inherited Class from Stochastic using the constructor method → __init__                |   B1    |        2 |
| Inherited Class from GaussianSample using the constructor method → __init__            |   B1    |        1 |
| Exception → try/except                                                                 |   B1    |        2 |
| Inherited Class from Dataset using the constructor method → __init__                   |   B1    |        2 |
| 2 Nested List                                                                          |   A2    |        1 |
| 4 Nested Tuple                                                                         |   A2    |        1 |
| Print                                                                                  |   A1    |        9 |
| Files → 'open' call function                                                           |   A2    |        2 |
| If statements using → __name__ == '__main__'                                           |   B2    |        2 |
| 1 Nested For Loop                                                                      |   A2    |        3 |
| 1 Nested List                                                                          |   A2    |        1 |
|-----------------------------------------------------------------------------------------------------|
| Total A1                                                                                            |      628 |
| Total A2                                                                                            |      724 |
| Total B1                                                                                            |       56 |
| Total B2                                                                                            |        2 |
| Total C1                                                                                            |       12 |
| Total C2                                                                                            |       28 |
|-----------------------------------------------------------------------------------------------------|


|-----------------------------------------------------------|
|                    AUTHOR INFORMATION                     |
|-----------------------------------------------------------|
| Author   |  Commits  |  Hours  |  LOC  |  Files Modified  |
|:---------|:---------:|:-------:|:-----:|:----------------:|
| wohlert  |    34     |    6    | 21326 |        65        |
|-----------------------------------------------------------|
| Total commits                                        34   |
| Total hours                                           6   |
| Total loc                                         21326   |
|-----------------------------------------------------------|

Results file can be found in file://home/pepe/Documentos/pycefr/results/semi-supervised-pytorch.json
```

Figura 6.2: Análisis completo de repositorio

6.1.2. Usuario (-u)

Las opciones `-u <USER>` nos permiten analizar un repositorio dado un nombre de usuario de GitHub.

En la figura 6.3 podemos ver este análisis. Hemos utilizado el mismo usuario de ejemplo, "wohlert". En este caso el programa nos lista los repositorios públicos de "wohlert" y nos pregunta cuál de ellos queremos analizar. Una vez seleccionado uno nos pide confirmar la acción, lo cual permite corregir la elección, y en caso de confirmar procede a realizar el análisis de dicho repositorio. Como en el caso anterior, se nos muestran mensajes indicando el estado del proceso y finalmente se imprimen los resultados por la consola.

```
pycefr  main *
> python3 pycerfl.py -u wohlert
[✓] Fetching user
[✓] Fetching repositories
wohlert has 6 public repositories:
  [1] atone
  [2] bayesian-saliency
  [3] generative-query-network-pytorch
  [4] mcmc-denoising
  [5] semi-supervised-pytorch
  [6] wohlert.github.io

Select which one you want to analyze (Enter [0] to exit): 6
Analyze [wohlert.github.io]? (Y/n) n

Select which one you want to analyze (Enter [0] to exit): 5
Analyze [semi-supervised-pytorch]? (Y/n) Y
[✓] Validating URL
[✓] Cloning repository
[✓] Analysing code
[✓] Fetching data
[✓] Fetching commits
[✓] Fetching contributors
[✓] Saving data

|-----------------------------------------------------------------------------------------------------|
|                                                ANALYSIS                                             |
|-----------------------------------------------------------------------------------------------------|
| Element                                                                                |  Level  |   Number |
|:---------------------------------------------------------------------------------------|:-------:|---------:|
| Simple List                                                                            |   A1    |       63 |
| Simple Tuple                                                                           |   A1    |       77 |
| 2 Nested Tuple                                                                         |   A2    |        4 |
| 'enumerate' call function                                                              |   C2    |        5 |
| Super Function                                                                         |   C2    |       20 |
| 'range' call function                                                                  |   A2    |       12 |
| Simple Attribute                                                                       |   A2    |      611 |
| Simple List Comprehension                                                              |   C1    |       12 |
| Simple Assignment                                                                      |   A1    |      301 |
| Assignment with sum (total = total + 1)                                                |   A1    |       20 |
| Simplified incremental Assignment with increase amount                                 |   A2    |       14 |
| Import                                                                                 |   A2    |       14 |
| Import with 'as' extension                                                             |   B1    |       15 |
| From                                                                                   |   A2    |       32 |
| Relative From                                                                          |   B1    |        9 |
| Simple If statements                                                                   |   A1    |       35 |
| Simple For Loop                                                                        |   A1    |       12 |
| For Loop with Tuple as name                                                            |   A2    |        7 |
| Function with Simple argument                                                          |   A1    |       56 |
| Return                                                                                 |   A1    |       55 |
| Simple Class using the constructor method → __init__                                   |   B1    |       11 |
| Inherited Class from VariationalAutoencoder using the constructor method → __init__    |   B1    |        2 |
| Inherited Class from DeepGenerativeModel using the constructor method → __init__       |   B1    |        3 |
| 6 Nested Tuple                                                                         |   A2    |        1 |
| 'map' call function                                                                    |   C2    |        1 |
| 'zip' call function                                                                    |   C2    |        2 |
| Function with Simple argument with Default argument                                    |   A2    |       21 |
| Lambda                                                                                 |   B1    |        6 |
| Inherited Class from object using the constructor method → __init__                    |   B1    |        2 |
| Simple Class                                                                           |   B1    |        1 |
| Inherited Class from Stochastic using the constructor method → __init__                |   B1    |        2 |
| Inherited Class from GaussianSample using the constructor method → __init__            |   B1    |        1 |
| Exception → try/except                                                                 |   B1    |        2 |
| Inherited Class from Dataset using the constructor method → __init__                   |   B1    |        2 |
| 2 Nested List                                                                          |   A2    |        1 |
| 4 Nested Tuple                                                                         |   A2    |        1 |
| Print                                                                                  |   A1    |        9 |
| Files → 'open' call function                                                           |   A2    |        2 |
| If statements using → __name__ == '__main__'                                           |   B2    |        2 |
| 1 Nested For Loop                                                                      |   A2    |        3 |
| 1 Nested List                                                                          |   A2    |        1 |
|-----------------------------------------------------------------------------------------------------|
| Total A1                                                                                            |      628 |
| Total A2                                                                                            |      724 |
| Total B1                                                                                            |       56 |
| Total B2                                                                                            |        2 |
| Total C1                                                                                            |       12 |
| Total C2                                                                                            |       28 |
|-----------------------------------------------------------------------------------------------------|


|-----------------------------------------------------------|
|                  AUTHOR INFORMATION                       |
|-----------------------------------------------------------|
| Author   |  Commits  |  Hours  |  LOC  |  Files Modified  |
|:---------|:---------:|:-------:|:-----:|:----------------:|
| wohlert  |    34     |    6    | 21326 |        65        |
|-----------------------------------------------------------|
| Total commits                                        34   |
| Total hours                                           6   |
| Total loc                                         21326   |
|-----------------------------------------------------------|

Results file can be found in file://home/pepe/Documentos/pycefr/results/semi-supervised-pytorch.json
```

Figura 6.3: Análisis de repositorio dado el usuario

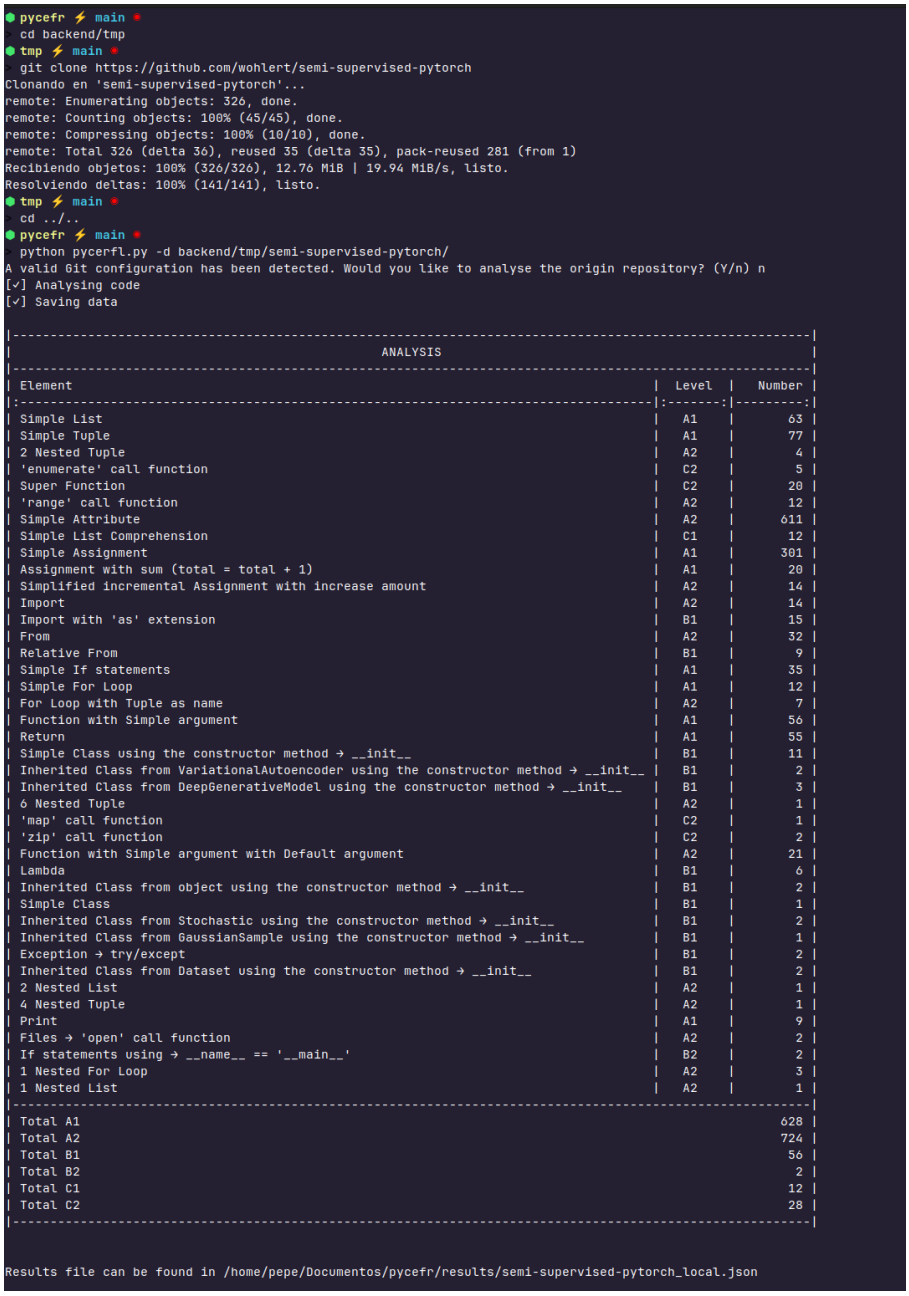
6.1.3. Directorio (-d)

El caso `-d <DIR>` realiza un análisis de un repositorio local, dada la ruta del mismo.

En este caso hemos clonado antes el repositorio que estamos empleando para las pruebas en la carpeta "tmp/", cuya finalidad es albergar archivos temporales cuya eliminación no altera la ejecución del programa.

La figura 6.4 nos muestra este proceso de clonado y el análisis del directorio. En este caso el programa nos dice que ha encontrado una configuración Git válida. Al tratarse de un directorio local es de esperar que no se incluya la información relativa al repositorio (puesto que no es un repositorio). Sin embargo, dado lo común que es trabajar con repositorios integrados en GitHub la aplicación comprueba si es el caso del directorio en cuestión. En caso de serlo, pregunta al usuario si prefiere analizar el repositorio remoto. Esto puede resultar ventajoso en caso de estar interesado en realizar un análisis con los cambios más recientes.

Si el usuario responde de manera afirmativa se procede a realizar el análisis del repositorio (igual que si usáramos el argumento -r), ignorando los archivos locales. En caso negativo, se analiza el directorio indicado anteriormente. El archivo resultante se generará añadiendo el sufijo "_local" al nombre del mismo. Esto da al usuario la oportunidad de realizar ambos análisis (local y del repositorio remoto) y no sobrescribir el archivo resultante (se toma el nombre del directorio o repositorio). Este comportamiento se puede activar o desactivar en el archivo de ajustes `settings.txt`, estableciendo en *false* la opción *addLocalSuffix*.



```
pycefr ⚡ main
 cd backend/tmp
tmp ⚡ main
 git clone https://github.com/wohlert/semi-supervised-pytorch
Clonando en 'semi-supervised-pytorch'...
remote: Enumerating objects: 326, done.
remote: Counting objects: 100% (45/45), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 326 (delta 36), reused 35 (delta 35), pack-reused 281 (from 1)
Recibiendo objetos: 100% (326/326), 12.76 MiB | 19.94 MiB/s, listo.
Resolviendo deltas: 100% (141/141), listo.
tmp ⚡ main
 cd ../..
pycefr ⚡ main
 python pycerfl.py -d backend/tmp/semi-supervised-pytorch/
A valid Git configuration has been detected. Would you like to analyse the origin repository? (Y/n) n
[✓] Analysing code
[✓] Saving data

|-----------------------------------------------------------------------------------------------------|
|                                              ANALYSIS                                               |
|-----------------------------------------------------------------------------------------------------|
| Element                                                                                |  Level  |   Number |
|:---------------------------------------------------------------------------------------|:-------:|---------:|
| Simple List                                                                            |   A1    |       63 |
| Simple Tuple                                                                           |   A1    |       77 |
| 2 Nested Tuple                                                                         |   A2    |        4 |
| 'enumerate' call function                                                              |   C2    |        5 |
| Super Function                                                                         |   C2    |       20 |
| 'range' call function                                                                  |   A2    |       12 |
| Simple Attribute                                                                       |   A2    |      611 |
| Simple List Comprehension                                                              |   C1    |       12 |
| Simple Assignment                                                                      |   A1    |      301 |
| Assignment with sum (total = total + 1)                                                |   A1    |       20 |
| Simplified incremental Assignment with increase amount                                 |   A2    |       14 |
| Import                                                                                 |   A2    |       14 |
| Import with 'as' extension                                                             |   B1    |       15 |
| From                                                                                   |   A2    |       32 |
| Relative From                                                                          |   B1    |        9 |
| Simple If statements                                                                   |   A1    |       35 |
| Simple For Loop                                                                        |   A1    |       12 |
| For Loop with Tuple as name                                                            |   A2    |        7 |
| Function with Simple argument                                                          |   A1    |       56 |
| Return                                                                                 |   A1    |       55 |
| Simple Class using the constructor method → __init__                                   |   B1    |       11 |
| Inherited Class from VariationalAutoencoder using the constructor method → __init__   |   B1    |        2 |
| Inherited Class from DeepGenerativeModel using the constructor method → __init__      |   B1    |        3 |
| 6 Nested Tuple                                                                         |   A2    |        1 |
| 'map' call function                                                                    |   C2    |        1 |
| 'zip' call function                                                                    |   C2    |        2 |
| Function with Simple argument with Default argument                                    |   A2    |       21 |
| Lambda                                                                                 |   B1    |        6 |
| Inherited Class from object using the constructor method → __init__                    |   B1    |        2 |
| Simple Class                                                                           |   B1    |        1 |
| Inherited Class from Stochastic using the constructor method → __init__                |   B1    |        2 |
| Inherited Class from GaussianSample using the constructor method → __init__            |   B1    |        1 |
| Exception → try/except                                                                 |   B1    |        2 |
| Inherited Class from Dataset using the constructor method → __init__                   |   B1    |        2 |
| 2 Nested List                                                                          |   A2    |        1 |
| 4 Nested Tuple                                                                         |   A2    |        1 |
| Print                                                                                  |   A1    |        9 |
| Files → 'open' call function                                                           |   A2    |        2 |
| If statements using → __name__ == '__main__'                                           |   B2    |        2 |
| 1 Nested For Loop                                                                      |   A2    |        3 |
| 1 Nested List                                                                          |   A2    |        1 |
|-----------------------------------------------------------------------------------------------------|
| Total A1                                                                                            |      628 |
| Total A2                                                                                            |      724 |
| Total B1                                                                                            |       56 |
| Total B2                                                                                            |        2 |
| Total C1                                                                                            |       12 |
| Total C2                                                                                            |       28 |
|-----------------------------------------------------------------------------------------------------|


Results file can be found in /home/pepe/Documentos/pycefr/results/semi-supervised-pytorch_local.json
```

Figura 6.4: Análisis de directorio local

6.1.4. List (-l)

La opción `-l` permite al usuario obtener el listado de los resultados existentes. Se nombran los archivos creados como consecuencia de los análisis realizados.

La figura 6.5 nos muestra este proceso, simple pero esencial para el uso del siguiente y último argumento disponible.

```
pycefr ⚡ main
> python pycerfl.py -l
Available results:
- semi-supervised-pytorch.json
- semi-supervised-pytorch_local.json
```

Figura 6.5: Listado de archivos de resultados

6.1.5. Consola (-c)

La última de las opciones disponibles, `-c <CONSOLE>` permite mostrar por consola resultados ya existentes, con el fin de evitar que tener que realizar de nuevo el análisis completo.

Tal como se presenta en la figura 6.6 hemos probado este argumento utilizando el archivo de resultados generado al comprobar el análisis de directorio local. Se nos muestra exactamente el mismo contenido, indicando primero los elementos encontrados, junto con su nivel y el número de instancias de cada uno, y finalmente agrupa por nivel todos los elementos, lo cual puede dar una mejor idea del nivel del proyecto.

```
pycefr ⚡ main
> python pycerfl.py -c semi-supervised-pytorch_local.json
```

ANALYSIS		
Element	Level	Number
Simple List	A1	63
Simple Tuple	A1	77
2 Nested Tuple	A2	4
'enumerate' call function	C2	5
Super Function	C2	20
'range' call function	A2	12
Simple Attribute	A2	611
Simple List Comprehension	C1	12
Simple Assignment	A1	301
Assignment with sum (total = total + 1)	A1	20
Simplified incremental Assignment with increase amount	A2	14
Import	A2	14
Import with 'as' extension	B1	15
From	A2	32
Relative From	B1	9
Simple If statements	A1	35
Simple For Loop	A1	12
For Loop with Tuple as name	A2	7
Function with Simple argument	A1	56
Return	A1	55
Simple Class using the constructor method → __init__	B1	11
Inherited Class from VariationalAutoencoder using the constructor method → __init__	B1	2
Inherited Class from DeepGenerativeModel using the constructor method → __init__	B1	3
6 Nested Tuple	A2	1
'map' call function	C2	1
'zip' call function	C2	2
Function with Simple argument with Default argument	A2	21
Lambda	B1	6
Inherited Class from object using the constructor method → __init__	B1	2
Simple Class	B1	1
Inherited Class from Stochastic using the constructor method → __init__	B1	2
Inherited Class from GaussianSample using the constructor method → __init__	B1	1
Exception → try/except	B1	2
Inherited Class from Dataset using the constructor method → __init__	B1	2
2 Nested List	A2	1
4 Nested Tuple	A2	1
Print	A1	9
Files → 'open' call function	A2	2
If statements using → __name__ == '__main__'	B2	2
1 Nested For Loop	A2	3
1 Nested List	A2	1
Total A1		628
Total A2		724
Total B1		56
Total B2		2
Total C1		12
Total C2		28

Figura 6.6: Mostrar resultados por consola

6.1.6. Apuntes

Es de especial relevancia mencionar el archivo de ajustes `settings.json`. Aunque ya hemos tratado algunas de las opciones disponibles falta mencionar la que probablemente sea la más importante de todas ellas.

A menudo los proyectos requieren de numerosos módulos y dependencias para su correcta ejecución. Éstos deben naturalmente estar instalados a la hora de ejecutar el proyecto, de lo contrario se producirían errores de ejecución. Por ejemplo, en este proyecto se está utilizando Python, y en este caso se requieren, entre otros, los módulos requests (utilizado para realizar solicitudes HTTP) o tabulate (para dar formato a los datos al mostrarlos por pantalla). Podríamos instalarlos en nuestra máquina, añadiéndolos a la versión de Python de ésta. Sin embargo, esto puede resultar problemático en caso de trabajar en varios proyectos que utilicen versiones distintas de un mismo módulo o dependencia.

La solución más ágil y común a este problema es la creación de un entorno virtual. Un entorno virtual es una herramienta que permite crear un espacio aislado en el que se pueden instalar y gestionar dependencias de proyectos sin interferir con otras instalaciones o proyectos en el mismo sistema. En nuestro caso hemos creado el entorno "myenv" para ello. Al crearse obtenemos un lienzo en blanco de Python al que añadir dependencias, lo cual hacemos mediante el uso del archivo `requirements.txt`. Esto nos sirve para el backend, escrito en Python; para el frontend, escrito en JavaScript, utilizamos una herramienta equivalente, llamada "npm".

En conclusión, estas carpetas de dependencias se almacenan dentro del proyecto y a menudo constan de un número muy elevado de archivos. Puesto que nuestro programa se dedica a analizar archivos, no nos conviene que se "entretenga" analizando estos numerosos (a veces incluso cientos), archivos. Es por ello en el archivo de ajustes hemos incluido una opción *"ignoreFolders"*, en el que podemos incluir las carpetas que queremos excluir del análisis. En la figura 6.7 mostramos un ejemplo de uso, en el que aparecen las carpetas que hemos decidido ignorar para la realización de todas las pruebas.

Figura 6.7: Ejemplo de uso de opción “ignoreFolders”

6.2. Frontend

Una vez visto el proceso para la creación del análisis podemos mostrar su visualización via web. Para ello nos posicionamos en la carpeta `frontend/` y levantamos el servidor ejecutando:

```
node server.js
```

Figura 6.8: Inicialización de servidor

Como se puede apreciar en la figura 6.8, el servidor nos indicará que está preparado para recibir solicitudes, añadiendo la dirección y el puerto por el que espera solicitudes. Por defecto se inicializará en el puerto 3000. Si vamos a esa dirección encontramos la página principal, la cual nos muestra un resumen de los análisis realizados.

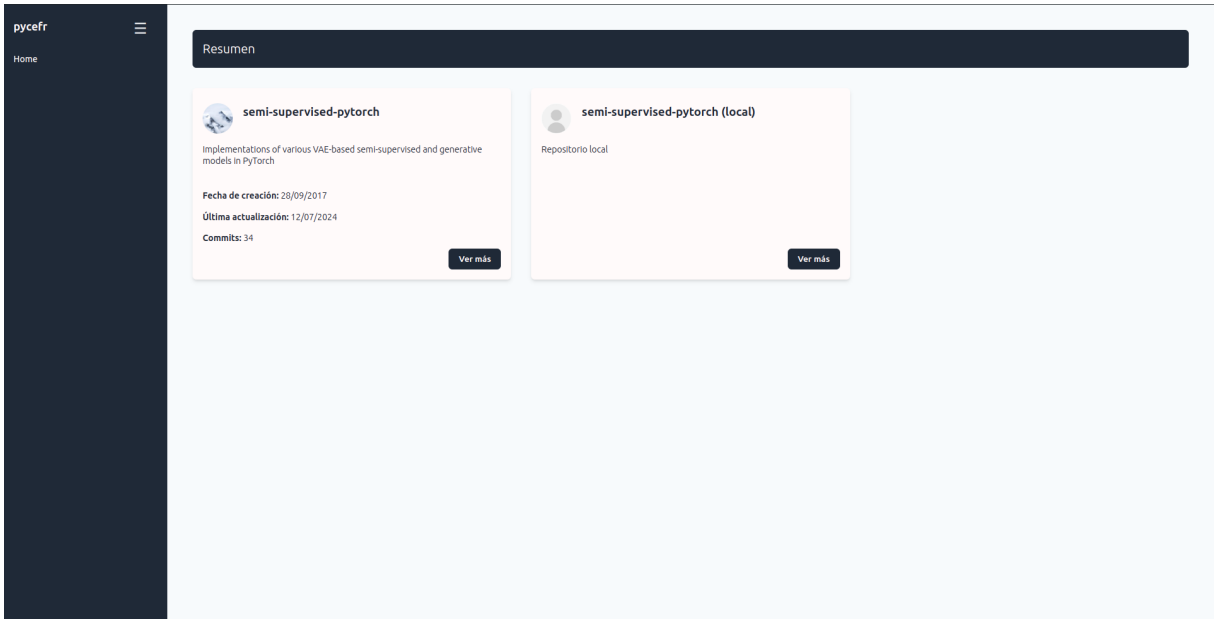


Figura 6.9: Vista web: Home

Como se aprecia en la figura 6.9, en este caso se nos muestran dos resultados en la página principal, referentes a los dos análisis que hemos realizado. Aquí no podemos ver más que un breve resumen, para dar al usuario una pequeña idea del repositorio o directorio analizado. En caso de tratarse de un repositorio este resumen mostrará algunos datos relevantes sobre el mismo.

Al hacer click en el botón "Ver más" nos dirigimos a la página del propio análisis (ver figura 6.10). En este caso seleccionaremos el primero de ellos.

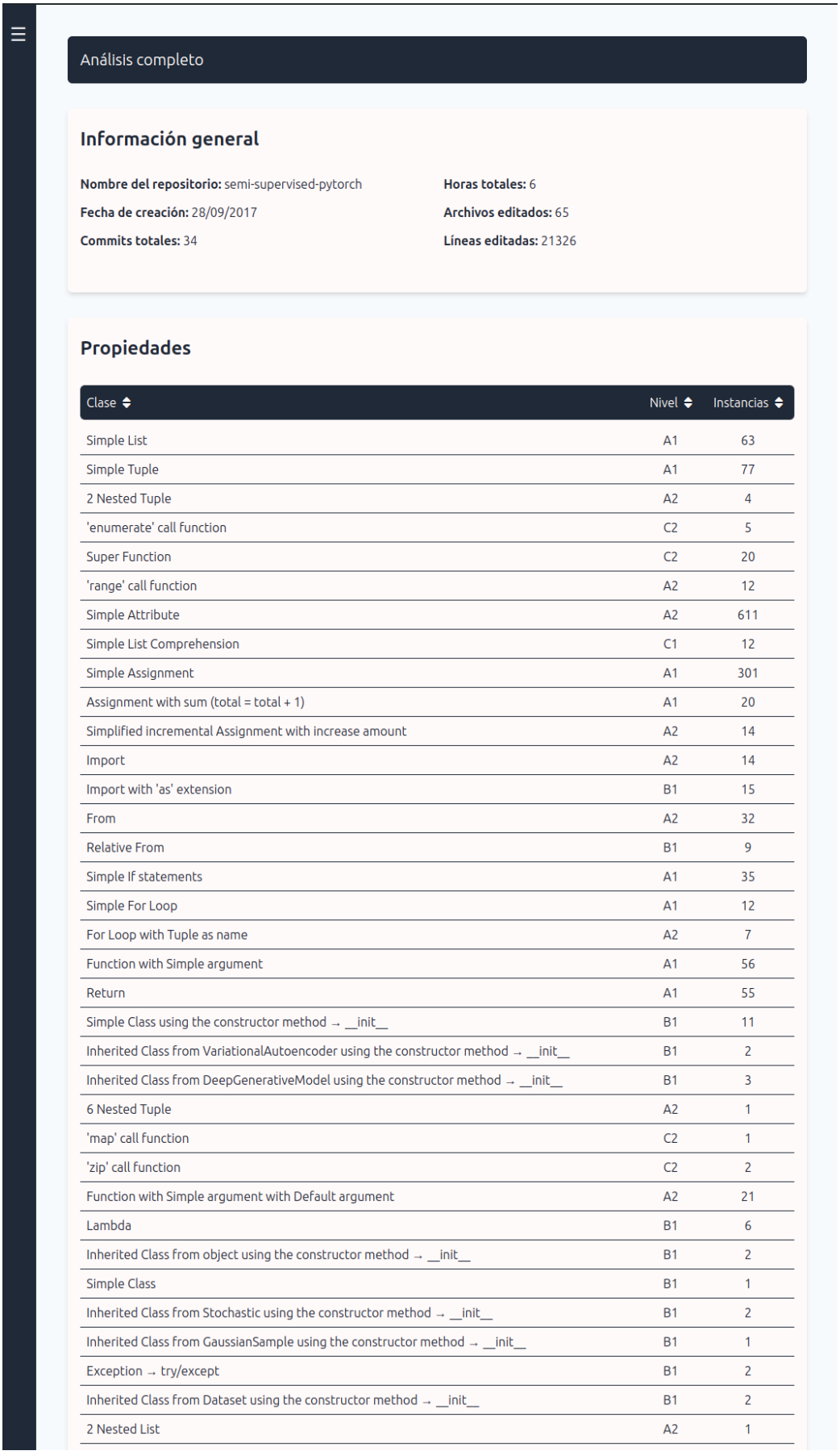
Diferenciamos en esta página dos bloques:

- **Información general:** Como su propio nombre indica muestra información general del proyecto. Entre estos datos se encuentran el nombre del proyecto, su fecha de creación, el número de commits realizados, una estimación del número de horas empleadas en el proyecto y el número total de archivos y líneas modificados en la historia del proyecto.

- **Propiedades:** En esta tabla vemos finalmente el resultado del análisis. Dado el tipo de información he encontrado más conveniente representarlo en formato tabla. Para mejorar la experiencia del usuario, se ha añadido la capacidad de ordenar la tabla por cada columna, en orden ascendente o descendente.

Conviene mencionar que en el caso de seleccionar un directorio local no se muestra el primer bloque, de **información general**, puesto que no hay manera de mostrar, entre otros, el tiempo empleado en su edición o el número de archivos editados. Es por ello que he considerado mejor opción excluirlo antes que dedicar únicamente este bloque a mostrar el nombre del proyecto. Dicho esto, en línea con la política general del proyecto, se han creado estructuras y páginas de estilos pensando ya en futuros cambios o ampliaciones, en caso de querer mostrar más datos o incluso más bloques.

Por último, en todas las páginas (incluida la home) encontramos en la parte izquierda un menu para navegar por la web. Este menú se puede ocultar o mostrar haciendo click en el botón "burguer" en la esquina superior derecha, y de momento tiene un único enlace navegable para redirigirse a la home. Dada la funcionalidad actual del proyecto no he visto necesario añadir más opciones, pero me parece conveniente dejar hecho el propio menú, de modo que de ser necesario simplemente haya que añadir enlaces sin preocuparse del comportamiento del menú.



Análisis completo

Información general

Nombre del repositorio: semi-supervised-pytorch
Fecha de creación: 28/09/2017
Commits totales: 34
Horas totales: 6
Archivos editados: 65
Líneas editadas: 21326

Propiedades

Clase	Nivel	Instancias
Simple List	A1	63
Simple Tuple	A1	77
2 Nested Tuple	A2	4
'enumerate' call function	C2	5
Super Function	C2	20
'range' call function	A2	12
Simple Attribute	A2	611
Simple List Comprehension	C1	12
Simple Assignment	A1	301
Assignment with sum (total = total + 1)	A1	20
Simplified incremental Assignment with increase amount	A2	14
Import	A2	14
Import with 'as' extension	B1	15
From	A2	32
Relative From	B1	9
Simple If statements	A1	35
Simple For Loop	A1	12
For Loop with Tuple as name	A2	7
Function with Simple argument	A1	56
Return	A1	55
Simple Class using the constructor method → __init__	B1	11
Inherited Class from VariationalAutoencoder using the constructor method → __init__	B1	2
Inherited Class from DeepGenerativeModel using the constructor method → __init__	B1	3
6 Nested Tuple	A2	1
'map' call function	C2	1
'zip' call function	C2	2
Function with Simple argument with Default argument	A2	21
Lambda	B1	6
Inherited Class from object using the constructor method → __init__	B1	2
Simple Class	B1	1
Inherited Class from Stochastic using the constructor method → __init__	B1	2
Inherited Class from GaussianSample using the constructor method → __init__	B1	1
Exception → try/except	B1	2
Inherited Class from Dataset using the constructor method → __init__	B1	2
2 Nested List	A2	1

Figura 6.10: Vista web: Repositorio

Capítulo 7

Conclusiones

7.1. Consecución de objetivos

Como hemos visto a lo largo de la sección de resultados se han cumplido la mayoría de las tareas propuestas inicialmente.

7.1.1. Mejora del código existente

La mejora del código existente forma parte de casi todos los objetivos planteados. Sin embargo, considero que conviene mencionarlo como uno independiente para destacar el trabajo de refractorización del código en aras de simplificar su comprensión y edición. La abundante documentación añadida sirve el mismo propósito: facilitar la futura edición del proyecto.

7.1.2. Implementación de funcionalidad “plug-and-play”

Personalmente éste era uno de los objetivos que primero me plantée y que consideraba prioritario. Me parecía de vital importancia dotar al usuario de la capacidad de ejecutar el programa nada más descargarlo. La inclusión de los archivos ’requisitos.txt’ y ’package.json’ para gestionar las dependencias del backend y del frontend, respectivamente, permiten a al usuario instalar en un par de segundos todo lo necesario. Esto junto con el cambio de rutas estáticas a rutas relativas en función del entorno de ejecución dan por cumplido este objetivo.

7.1.3. Mejora de la visualización de los resultados en formato JSON

Originalmente se generaban múltiples archivos que contenían a menudo la misma información pero de maneras distintas. Sin tener en cuenta todavía el uso de varios formatos, esto provocaba que incluso tras generar archivos de resultados había que crear nuevos archivos 'definitivos' de nuevo a partir de éstos. Se ha reducido a un solo archivo por análisis, facilitando al usuario el manejo de la información.

7.1.4. Desarrollo de interfaz web

El objetivo principal del proyecto era dotar al usuario de una mejor interfaz web para visualizar los datos. La original, aunque técnicamente funcional, era incómodo de usar para los estándares de hoy en día.

La implementación de una interfaz web navegable de manera cómoda y con la capacidad de ser expandida en el futuro, pues se ha diseñado de tal modo que esto resulte una tarea poco complicada, da por cumplido este objetivo.

7.1.5. Implementar visualización de resultados por consola

Además de la visualización web un objetivo importante era otorgar al usuario la capacidad de comprobar los resultados por consola, más rápido y ágil que la solución web. Se ha añadido esta vista en formato tabla, la cual considero más que intuitiva teniendo en cuenta el formato de información que estamos tratando, y se ha incluido en un archivo de ajustes la capacidad de activar o desactivar el mostrado automático de los datos tras finalizar el análisis. Del mismo modo, el usuario puede ejecutar el programa de tal manera que en lugar de realizar el análisis simplemente imprima por la consola en el mismo formato resultados ya existentes.

7.1.6. Añadido de registros durante el proceso de análisis

Aunque en el proyecto original había comentarios que explicaban qué tarea de estaba realizando, considero que los nuevos dan a entender al usuario de mejor manera qué está pasando en todo momento. Sin embargo, probablemente la mayor mejora se dé en los mensajes de error, antes ausentes, que ahora permiten al usuario saber qué está fallando exactamente, indicando

incluso en caso de que el usuario pueda solventarlo la manera de hacerlo.

7.1.7. Reestructuración de la arquitectura del código

Al abordar el proyecto por primera vez el principal problema que tuve fue entender qué realizaba o cómo funcionaba exactamente. Ya no sólo por los problemas de ejecución iniciales, los cuales de acabaron resolviendo en un tiempo razonable, sino por la propia estructura del proyecto. Éste, debido a la relativa pequeña cantidad de archivos que contenía, no constaba de una estructura práctica ni mucho menos escalable.

Se han agrupado los archivos en carpetas que contienen lógica relacionada, facilitando así la organización y el acceso a los componentes del proyecto, intentando seguir los estándares modernos.

7.1.8. Ampliación del análisis con información adicional

Desde un inicio se ha pretendido expandir el análisis de dos maneras. La primera de ellas, incluyendo la información relativa a GitHub, esto es, información de los usuarios involucrados, del propio repositorio... La segunda parte consistía en rehacer el proceso completo de análisis, de modo que los elementos presentes en el código se contasen de una manera diferente para facilitar, de nuevo, su futura edición o personalización (se consideró otorgar al usuario la capacidad de activar y desactivar los elementos del código que analizar). Mientras que la primera tarea se cumplió sin mayor problema, al abordar este rediseño del análisis de archivos, me di cuenta de que era necesario reconstruir el proceso casi desde los cimientos. Tras reevaluar la magnitud de esta tarea consideré que el esfuerzo no justificaba la teórica diferencia de resultados, lo que llevó a replantear la viabilidad de este enfoque, finalmente optando por desecharlo. Dicho esto, se han realizado mejoras en los textos de los resultados.

7.2. Aplicación de lo aprendido

La realización de este proyecto no habría sido posible de no ser por los conocimientos adquiridos en múltiples asignaturas del Grado de en Ingeniería en Tecnologías de la Telecomunicación. Al tratarse de un proyecto esencialmente de programación, no es de sorprender que

las asignaturas que considere más me han aportado sean:

- **Fundamentos de la programación:** La piedra angular de todo este proyecto. El conocimiento adquirido en esta asignatura, básico mas fundamental, es el que me ha permitido comenzar mi evolución como programador, sin la que no habría sido posible esta tarea.
- **Servicios y Aplicaciones Telemáticas:** Está asignatura me proporcionó por un lado el conocimiento de Python necesario para llevar a cabo este proyecto. Además, la elaboración de un gran proyecto en esta asignatura me hizo aprender a compaginar distintos lenguajes de programación en un mismo trabajo. Aunque no fuese un requisito indispensable de esta asignatura, fue ésta la que me impulsó a aprender en profundidad HTML y CSS, lenguajes que a día de hoy uso de manera extensa en el entorno laboral.
- **Desarrollo de Aplicaciones Telemáticas:** La asignatura que sin lugar a dudas considero que más me ha aportado en toda la carrera. Me permitió aprender JavaScript, uno de los lenguajes más utilizados a nivel mundial, a un ritmo personalizado, lo cual agradecí enormemente. Gracias a esta asignatura, junto con las anteriormente mencionadas, pude realizar mis labores más tarde en las prácticas externas.
- **Prácticas Externas:** El conocimiento adquirido en las asignaturas ya mencionadas me permitió tomar parte en una empresa como desarrollador web. Dejando a un lado el conocimiento adquirido en relación a los lenguajes empleados aquí aprendí sobre todo qué es el trabajo a nivel empresa, con todo lo que eso conlleva.

7.3. Lecciones aprendidas

Las lecciones aprendidas tras la elaboración de este proyecto son:

- Consolidación de conocimientos del framework Express.js, visto en la asignatura 'Desarrollo de Aplicaciones Telemáticas' y Node.js.
- Uso de la herramienta Postman para la realización de pruebas con API.
- Patrones de diseño y sintaxis de Python con los que no estaba familiarizado.
- Estándares de desarrollo web modernos.

7.4. Trabajos futuros

En el futuro podrían implementarse las siguientes funcionalidades:

- Edición del sistema de análisis. Permitir al usuario poder añadir o quitar elementos del análisis, así como poder estipular por sí mismo el nivel que cada elemento analizado.
- Funcionalidad completa del proyecto desde la página web, sin depender de ejecutar el proyecto anteriormente desde la consola. Es decir, poder realizar el análisis directamente desde la web.
- Interacciones entre varios archivos de resultados (comparar archivos, unirlos...).

Apéndice A

Manual de usuario

pycefr es una herramienta diseñada para evaluar el nivel de código Python 3 utilizando como inspiración el Marco Común Europeo de Referencia para las Lenguas (CERFL). Permite analizar repositorios de GitHub y directorios locales para proporcionar información valiosa sobre el nivel del código.

A.1. Requisitos Previos

Antes de comenzar, asegúrate de tener instalado Python 3 y Node.js. Asegúrate también de tener los permisos adecuados para acceder a los directorios que deseas analizar.

Puedes instalar los módulos de Python necesarios utilizando el archivo `requirements.txt` ejecutando el siguiente comando:

```
pip install -r requirements.txt
```

Además, para instalar los módulos de Node.js requeridos, navega a la carpeta `frontend/` y ejecuta:

```
npm install
```

A.2. Estructura del Proyecto

El proyecto incluye los siguientes componentes principales:

- **pycefr.py**: El ejecutable principal que realiza el análisis.
- **backend/**: Carpeta que contiene los archivos que realizan el análisis del código
- **frontend/**: Carpeta que contiene la interfaz de usuario basada en Node.js para visualizar los resultados.
- **results/**: Carpeta donde se almacenan los resultados del análisis.

Se incluyen también varios archivos secundarios, encargados de la configuración de la aplicación o de

- **settings.json**: Archivo de configuración para personalizar la herramienta.
- **pycefrl.postman_collection.json**: Colección de Postman con las llamadas involucradas en el proyecto.
- **requirements.txt**: Archivo utilizado para instalar los módulos de Python necesarios.

A.3. Cómo Usar pycefr

A.3.1. 1. Análisis de un Repositorio de GitHub

Para analizar un repositorio de GitHub, utiliza el siguiente comando:

```
python3 pycerfl.py -r <repo_url>
```

Este comando realiza un análisis completo en un repositorio válido de GitHub.

A.3.2. 2. Análisis de un Usuario de GitHub

Para obtener información sobre un usuario de GitHub y sus proyectos, usa:

```
python3 pycerfl.py -u <user_name>
```

El sistema te preguntará cuál de los proyectos públicos del usuario deseas analizar.

A.3.3. 3. Análisis de un Directorio Local

Para analizar un directorio local, ejecuta:

```
python3 pycerfl.py -d <dir_path>
```

Este comando realizará un análisis completo en el directorio especificado.

A.3.4. 4. Listar Archivos de Resultados

Para listar los archivos generados en la carpeta `results/`, puedes usar la opción:

```
python3 pycerfl.py -l
```

A.3.5. 5. Visualizar resultados por consola

Para visualizar un archivo de resultados por consola puedes usar la opción:

```
python3 pycerfl.py -c <results_file>
```

A.3.6. 6. Visualización de Resultados via web

Una vez completado el análisis, se generará un archivo JSON en la carpeta `results/`. Para visualizar estos resultados, puedes usar Node.js:

1. Navega a la carpeta `frontend/` y ejecuta:

   ```
   npm install
   ```

2. Luego, ejecuta el servidor local:

   ```
   node server.js
   ```

Abre tu navegador y dirígete a `http://localhost:3000` para ver los resultados.

A.3.7. Configuración de settings.json

El archivo `settings.json` permite personalizar la configuración de la herramienta:

Ajustes	Descripción
`ignoreFolders`	Lista de carpetas que serán excluidas del análisis.
`API-KEY`	Clave API de GitHub para evitar el límite en las llamadas a la API.
`addLocalSuffix`	Si se establece en `true`, añade el sufijo `_local` a los archivos de resultados.
`autoDisplayConsole`	Si se establece en `true`, muestra los resultados por consola tras finalizar el análisis.

Ejemplo de settings.json

```
{
    ``ignoreFolders'': [
        "node_modules/",
        "myenv/",
        ".git/",
        "__pycache__/"
    ],
    "API-KEY": "customapikey1234",
    "addLocalSuffix": true
    "autoDisplayConsole": true
}
```

A.3.8. Importación de colección Postman

En caso de querer probar las llamadas realizadas en el código tenemos la posibilidad de importar en Postman el archivo `pycerfl.postman_collection.json` y ver todas las solicitudes con ejemplos de uso. Para ello abrimos Postman y en seleccionamos la opción "Import" en la parte superior. Esto nos abrirá el siguiente panel:

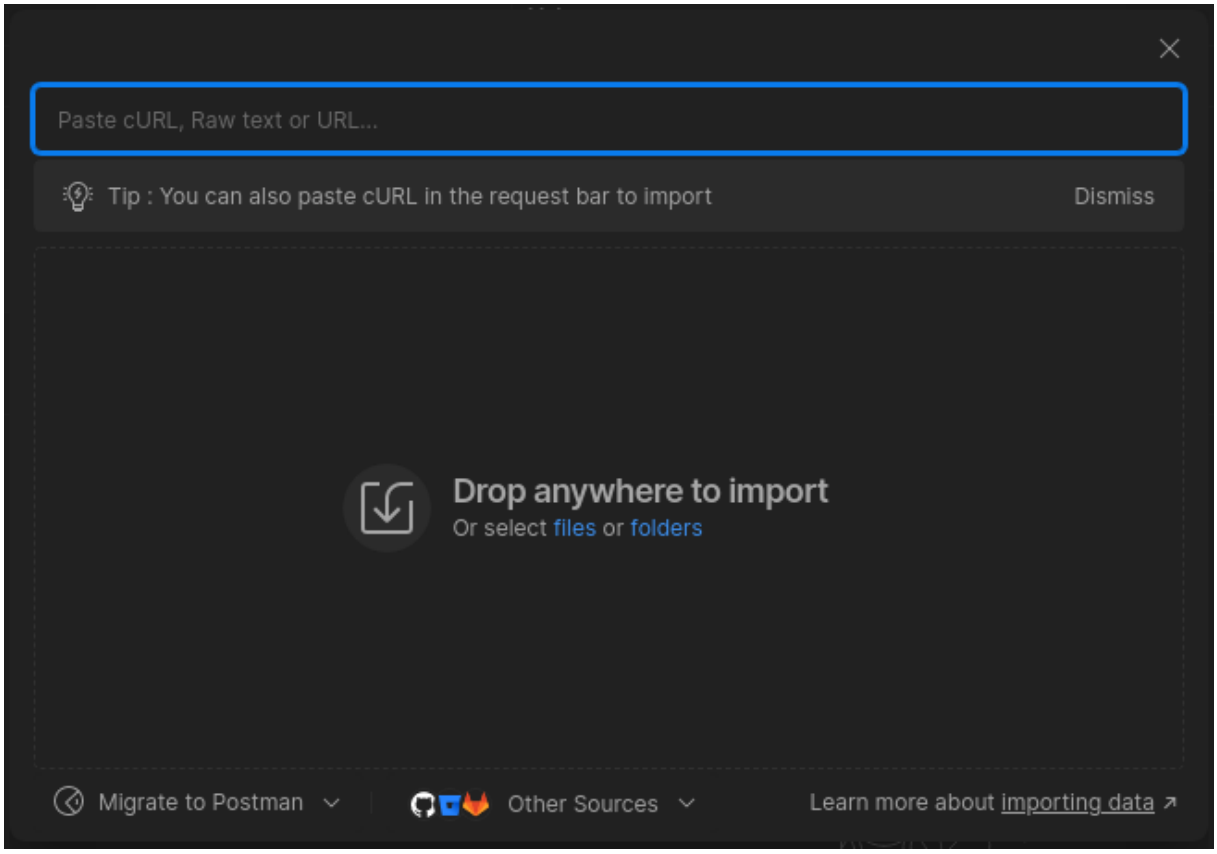


Figura A.1: Panel de importación en Postman

Añadimos el archivo ya sea haciendo click o arrastrándolo directamente. Tras esto ya tendremos acceso a todas las llamadas involucradas en el proyecto.

Bibliografía

[1] Postman api documentation. https://learning.postman.com/docs/publishing-your-api/documenting-your-api/.

[2] C. Anyadike. Top programming languages sought by employers in 2023 for software engineers. https://dev.to/chukwuma1976/top-programming-languages-sought-by-employers-in-2023-for-software-enc Sept. 2023. Dev.to.

[3] E. Brown. *Web Development with Node and Express*. O'Reilly Media, 2019.

[4] M. Casciaro and L. Mammino. *Node.js Design Patterns*. Packt Publishing, 2020.

[5] J. Duckett. *HTML y CSS: Diseño y construcción de sitios web*. Ediciones Pearson, 2015.

[6] M. Haverbeke. *Eloquent JavaScript: A Modern Introduction to Programming*. No Starch Press, 2018.