



ESCUELA DE INGENIERÍA DE FUENLABRADA

**GRADO EN INGENIERÍA EN SISTEMAS
AUDIOVISUALES Y MULTIMEDIA**

TRABAJO FIN DE GRADO

**ANÁLISIS DEL USO DE LENGUAJES DE PROGRAMACIÓN
MINORITARIOS EN PROYECTOS DE SOFTWARE LIBRE**

Autor : Raúl Gómez Cantador

Tutor : Dr. Gregorio Robles Martínez

Curso académico 2024/2025

Trabajo Fin de Grado/Máster

ANÁLISIS DEL USO DE LENGUAJES DE PROGRAMACIÓN MINORITARIOS EN PROYECTOS DE SOFTWARE LIBRE

Autor : Raúl Gómez Cantador

Tutor : Dr. Gregorio Robles Martínez

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

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente **Calificación:**

Fuenlabrada, a de diciembre de 2024



©2024 Raúl Gómez Cantador

Algunos derechos reservados

Este documento se distribuye bajo la licencia “Atribución-CompartirIgual 4.0 Internacional” de Creative Commons, disponible en

<https://creativecommons.org/licenses/by-sa/4.0/deed.es>

*Dedicado a
mi familia.*

Agradecimientos

Con este proyecto doy prácticamente el carpetazo final a mi paso por la universidad. Han sido seis años que por algunas épocas se han hecho muy cuesta arriba. Sin embargo, me quedo con todos aquellos momentos buenos que he pasado, que son infinitamente mayores que los malos. Quiero agradecer a todas las personas que me han ayudado a lo largo de todo este proceso y en especial a los amigos que he hecho durante el camino.

También, quiero agradecer a Gregorio, mi tutor, por ser tan flexible conmigo. Aunque por mi parte no haya sido muy organizado durante la realización del proyecto y me haya faltado constancia, siempre que he recurrido a ti para hacer cualquier tutoría, has sacado algún tiempo para ayudarme, incluso siendo algunas de ellas bastante tarde. ¡Muchas gracias, Gregorio!

Por último, quiero agradecer sobre todo, a mi familia. Siempre que los necesito están ahí y me apoyan en todo lo que hago. Gracias por decirme aquello que necesito oír, y no lo que quiero, siempre.

Resumen

Este proyecto consiste en la creación de una herramienta que permite realizar el análisis de uso de los lenguajes minoritarios en repositorios públicos de GitHub. Entendemos los lenguajes minoritarios por aquellos que representan un porcentaje pequeño de uso en un repositorio y que es usado por un grupo pequeño de usuarios. El objetivo que tiene es que el usuario que utilice esta herramienta pueda ver cómo es el uso de estos lenguajes minoritarios en el repositorio que escoja, los usuarios que han contribuido y en qué cantidad.

Para ello se ha realizado un programa en Python que se encarga de producir estos análisis recolectando los datos de la participación histórica de los repositorios a través de los commits mediante el uso de la librería Perceval y otros datos como el número de ficheros o las tecnologías usadas en los repositorios. Además, para hacer la representación de los análisis se ha realizado también una aplicación web cliente-servidor para que la visualización de estos sea mucho más cómoda y sencilla de usar para un usuario cualquiera. Esta aplicación se ha construido con HTML, CSS y JavaScript con uso de la librería Bootstrap y con la librería Flask para la gestión del servidor. Además se ha creado una base de datos SQLite3 donde se almacenan los datos de los repositorios analizados, la información de uso de todos los lenguajes y la información de todos los usuarios que hayan participado en alguno de los proyectos.

Puede ser un proyecto útil para desarrolladores que quieran conocer cómo funcionan los lenguajes con menor uso en repositorios, sobre todo, dirigido a aquellos repositorios muy grandes y con bastante tiempo detrás, donde es difícil obtener esta información.

Summary

This project consists of the creation of a tool that allows the analysis of the use of minority languages in public repositories on GitHub. We understand minority languages as those that represent a small percentage of use in a repository and that are used by a small group of users. The aim is that the user who uses this tool can see how these minority languages are used in the repository he/she chooses, the users who have contributed and in what quantity.

To do this, a Python program has been created that is responsible for producing these analyses by collecting data on the historical participation of the repositories through commits using the Perceval library and other data such as the total number of files or the technologies used in the repositories. In addition, a client-server web application has also been developed to represent the analyses so the visualisation of these analyses is much more comfortable and easy to use for any user. This application has been built with HTML, CSS and JavaScript using the Bootstrap library and the Flask library for server management. Also, a SQLite3 database has been created where the data of the analysed repositories, the usage information of all the languages and the information of all the users who have participated in any of the projects stored.

It can be a useful project for developers who want to know how the lesser-used languages work in repositories, especially for very large repositories with a long time behind them, where it is difficult to obtain this information.

Índice general

Lista de figuras	XI
1. Introducción	1
1.1. Estructura de la memoria	2
2. Objetivos	3
2.1. Objetivo general	3
2.2. Objetivos específicos	3
2.3. Planificación temporal	4
3. Estado del arte	7
3.1. Python	7
3.2. Perceval	8
3.3. Flask	8
3.4. Jinja2	9
3.5. Git	9
3.6. GitHub	9
3.7. SQLAlchemy	10
3.8. SQLite	10
3.9. HTML	11
3.10. CSS	11
3.11. Bootstrap	11
4. Diseño e implementación	13
4.1. Arquitectura general	13

4.2. Estructura del código del proyecto	15
4.3. Comenzando a usar la aplicación	18
4.4. Endpoint /userrepo	19
4.5. Endpoint /usersearch	29
4.6. Interacción con la base de datos	31
4.6.1. Tabla Repository	32
4.6.2. Tabla User_expertise	33
4.6.3. Tabla Languages	33
5. Experimentos y validación	35
5.1. Ejemplo de uso de la aplicación con el análisis de un repositorio	35
5.2. Análisis de más repositorios	42
6. Resultados	49
7. Conclusiones	65
7.1. Consecución de objetivos	65
7.2. Aplicación de lo aprendido	66
7.3. Lecciones aprendidas	67
7.4. Trabajos futuros	67
A. Manual de usuario	69
Bibliografía	73

Índice de figuras

2.1.	Diagrama de Gantt de la planificación temporal del proyecto	5
4.1.	Arquitectura general de la aplicación	14
4.2.	Página inicial de la aplicación	19
4.3.	Respuesta con el formulario a la petición con método GET	20
4.4.	Flujo de la aplicación para consultar si existe la última versión de un repositorio en la base de datos	21
4.5.	Tabla Repository	24
4.6.	Tabla User_expertise	25
4.7.	Tabla Languages	26
4.8.	Respuesta a la petición POST con la representación gráfica del análisis	27
4.9.	Análisis de los lenguajes minoritarios	28
4.10.	Contribuyentes al repositorio	29
4.11.	Respuesta a la petición GET al endpoint /usersearch	30
4.12.	Respuesta a la petición POST al endpoint /usersearch	31
4.13.	Diagrama entidad-relación de la base de datos	32
5.1.	Página principal de la aplicación	36
5.2.	Página que se muestra al clickar en ”Búsqueda por usuario	36
5.3.	Respuesta de la aplicación al buscar los repositorios del usuario netflix	37
5.4.	Repositorio Hystrix en la lista	37
5.5.	Log que se genera cuando Perceval obtiene la información de un repositorio	38
5.6.	Repositorio Hystrix en la tabla Repository	38

5.7. Repositorio Hystrix en la tabla Repository - 2	38
5.8. Repositorio Hystrix en la tabla User_expertise	39
5.9. Repositorio Hystrix en la tabla Languages	40
5.10. Representación gráfica del análisis del repositorio Hystrix - 1	40
5.11. Representación gráfica del análisis del repositorio Hystrix - 2	41
5.12. Representación gráfica del análisis del repositorio Hystrix - 3	41
5.13. Representación gráfica del análisis del repositorio Zuul	42
5.14. Representación gráfica del análisis del repositorio FlameScope	43
5.15. Representación gráfica del análisis del repositorio chaosmonkey	44
5.16. Representación gráfica del análisis del repositorio Eureka	44
5.17. Representación gráfica del análisis del repositorio Fenzo	45
5.18. Representación gráfica del análisis del repositorio Edda	46
5.19. Representación gráfica del análisis del repositorio Lipstick	46
5.20. Representación gráfica del análisis del repositorio Mantis	47
5.21. Representación gráfica del análisis del repositorio spectator	48
6.1. Estado de la tabla Repository después del análisis de 200 repositorios	49
6.2. Estado de la tabla Repository después del análisis de 200 repositorios - 2	50
6.3. Estado de la tabla User_expertise después del análisis de 200 repositorios	50
6.4. Estado de la tabla User_expertise después del análisis de 200 repositorios - 2	51
6.5. Estado de la tabla Languages después del análisis de 200 repositorios	51
6.6. Estado de la tabla Languages después del análisis de 200 repositorios - 2	52
6.7. Fórmula de Brown usada para el cálculo del coeficiente de Gini	56
6.8. Gráfica Coeficiente de Gini	57
6.9. Coeficiente de Gini YAML	58
6.10. Coeficiente de Gini VIM Help File	58
6.11. Coeficiente de Gini Shell	59
6.12. Coeficiente de Gini CSS	59
6.13. Coeficiente de Gini SVG	60
6.14. Coeficiente de Gini JSON	60
6.15. Coeficiente de Gini HTML	61

6.16. Coeficiente de Gini Markdown	61
6.17. Coeficiente de Gini JavaScript	62
6.18. Coeficiente de Gini TOML	62
A.1. Página inicial de la aplicación	69
A.2. Respuesta con el formulario a la petición con método GET	70
A.3. Respuesta con el análisis del repositorio	70
A.4. Respuesta a la petición GET al endpoint /usersearch	71
A.5. Respuesta con los repositorios del usuario	71

Capítulo 1

Introducción

Durante los últimos años, a través de plataformas de desarrollo de software como GitHub o GitLab, una gran cantidad de desarrolladores contribuyen diariamente a proyectos de software libre. Por supuesto, al haber tanta variedad de proyectos y desarrolladores, existe una gran diversidad de lenguajes de programación que se usan en los distintos repositorios de cada uno de los proyectos. Este trabajo de fin de grado tiene como objetivo realizar el análisis del uso de esos lenguajes de programación, dividiéndolos en dos clases: lenguajes mayoritarios y lenguajes minoritarios, si bien, nos centraremos en mayor medida en estos últimos. Definimos estos lenguajes minoritarios como aquellos que representan menos de un 5 % dentro de un repositorio donde se almacene el código de un programa, y que es usado por un número pequeño de contribuyentes. Estos lenguajes, si bien no son los más usados por la gran mayoría de desarrolladores, pueden llegar a cumplir roles específicos y críticos para el funcionamiento de una aplicación o proyecto.

Para llevar a cabo este análisis, se ha desarrollado una aplicación en Python cuyo objetivo es la recopilación y representación de datos provenientes de los repositorios de distintos proyectos para comprobar cómo es el uso de estos lenguajes minoritarios. Con todos estos datos, se ha creado una base de datos SQL que contiene la información de uso de los distintos lenguajes por usuario y repositorio de todos los proyectos que se han analizado a lo largo de la ejecución del trabajo de fin de grado.

Además, este proyecto cuenta con una aplicación web para poder facilitar la recolección y representación de los datos de manera mucho más fácil y cómoda. Esta herramienta se puede utilizar para hacer el análisis del uso de los lenguajes minoritarios dentro de un repositorio para

ver como se han gestionado estos a lo largo del tiempo. Además, permite conocer a los usuarios que han participado a los repositorios y con ello, poder hacerse una idea de los conocimientos de cada uno de ellos. Además, puesto que existe una base de datos que va almacenando todos estos análisis, en el futuro podría servir para seguir recopilando datos de repositorios distintos a los que se han usado para realizar este análisis y continuar investigando más adelante.

1.1. Estructura de la memoria

La memoria se estructura de la siguiente manera:

- **Capítulo 1: Introducción.** Se presenta la motivación y una breve descripción de lo que se afronta en este proyecto, además de informar de la estructura del mismo.
- **Capítulo 2: Objetivos.** Se describen los objetivos a alcanzar durante toda la consecución del proyecto, incluyendo tanto planificación temporal como objetivos técnicos.
- **Capítulo 3: Estado del arte.** Explicación de las distintas tecnologías usadas a lo largo del proyecto. Incluye la información sobre la aplicación web, y como se realiza la recopilación y guardado de datos en la base de datos.
- **Capítulo 4: Diseño e implementación.** Se realiza una explicación sobre el diseño en conjunto de todo el proyecto, y el por qué de como se ha implementado.
- **Capítulo 5: Experimentos y validación.** Se explica un caso de uso de la aplicación para ver las funcionalidades y comprobar que son correctas.
- **Capítulo 6: Resultados.** Se realiza el análisis final de los resultados obtenidos tras analizar 200 repositorios.
- **Capítulo 7: Conclusiones.** Incluye la conclusión final del proyecto.

Capítulo 2

Objetivos

2.1. Objetivo general

Mi trabajo de fin de grado consiste en obtener datos de los repositorios de proyectos de software libre que se encuentren disponibles en la plataforma de desarrollo software GitHub y analizar en profundidad los desarrolladores implicados en los mismos como los lenguajes de programación utilizados para llevarlos a cabo, principalmente aquellos que son minoritarios, pero de uso común.

2.2. Objetivos específicos

Se han seguido los siguientes objetivos específicos para llegar a cumplir el objetivo final. Cada objetivo específico cuenta con sus propios puntos:

■ Obtención de datos de los repositorios de plataformas Git

- Búsqueda de una herramienta para obtener los datos usando Python
- Investigación del módulo de python Perceval usado para la obtención de commits realizados en los repositorios
- Investigación de llamadas a la API de GitHub para obtener información adicional
- Guardado de datos en formato JSON para su posterior tratamiento

■ Tratamiento de los datos de los repositorios

- Organización de los datos obtenidos
 - Parsear ficheros JSON de las llamadas a la API de GitHub para guardar únicamente los datos relevantes para el proyecto:
 - Commits realizados sobre el repositorio
 - Usuario que ha participado en cada uno de los commits
 - Lenguaje de programación usado en cada commit
 - Total de usuarios que han participado en el repositorio
 - Creación de un fichero JSON para almacenar todos los datos resultantes del punto anterior:
- **Creación de una aplicación web para permitir un uso más fácil del aplicativo**
- Uso de Flask para construir la aplicación web
 - Conectividad con la base de datos creada
 - Permitir al usuario la obtención de los datos de un repositorio a su elección
 - Representación de los datos de los repositorios
- **Creación de una base de datos donde almacenar los análisis de los repositorios**
- Almacenar repositorios analizados
 - Almacenar usuarios contribuyentes a cada repositorio analizado
 - Almacenar usuarios con todas sus contribuciones a cada uno de los repositorios
 - Almacenar lenguajes usados en cada repositorio

2.3. Planificación temporal

El total de duración del proyecto ha sido de aproximadamente un año. Contacté por primera vez con Gregorio, mi tutor, en octubre del año 2023, con la intención de comenzar con este proyecto, dando inicio desde noviembre y terminándolo a lo largo del anterior curso académico 2023/2024. Sin embargo, no ha sido hasta este presente curso 2024/2025 en el que he podido finalizar el trabajo, sobre todo debido a algunos parones que hice durante algunos meses del

2.3. PLANIFICACIÓN TEMPORAL

5

curso anterior y a que, principalmente, se ha hecho en mi tiempo libre y durante los fines de semana. En la siguiente figura, se encuentra la planificación general final que ha seguido el proyecto a lo largo de todo este año en un diagrama de Gantt:



Figura 2.1: Diagrama de Gantt de la planificación temporal del proyecto

Capítulo 3

Estado del arte

En este proyecto se han utilizado múltiples tecnologías, siendo el lenguaje de programación Python el principal, y en el que se basan prácticamente todos los recursos que he utilizado.

3.1. Python

Python [7] es uno de los lenguajes de programación más utilizados durante los últimos años. Se debe, sobre todo, a su accesibilidad y facilidad que ofrece a los desarrolladores para realizar múltiples tareas.

Es un lenguaje de programación de código abierto y gratuito creado por Guido van Rossum a principios de la década de los noventa, pero no ha hecho nada más que evolucionar durante todo este tiempo para agregar nuevas características en cada una de las versiones. Para este trabajo se ha utilizado la versión 3.11 de Python, sin embargo, aún sigue actualizándose, siendo la última versión en el momento de escribir esta memoria la versión 3.13.

Las principales ventajas [10] que ofrece sobre otros lenguajes son:

- **Es un lenguaje interpretado**, es decir, no es necesario compilar el código para su ejecución, sino que en su lugar existe un intérprete que se encarga de la lectura del fichero y su ejecución.
- **Es un lenguaje multiplataforma**, por lo que su ejecución no está limitada a un sistema o software específico. Es muy flexible desde este punto de vista.

- **Facilidad de la sintaxis.** Es uno de los lenguajes preferidos para comenzar en el mundo de la programación debido a ello.
- **Existencia de un garbage collector** que permite la limpieza automática de memoria y facilita en gran cantidad evitar gastos de memoria inútiles.

En mi caso, lo he escogido en este proyecto ya que, además de ser el lenguaje que más he usado tanto a lo largo de la carrera universitaria como en el mundo laboral, y por tanto ser el lenguaje del que más conocimiento tengo, quería aprender más sobre él usándolo en otros ámbitos que no conocía antes, como la interacción con una base de datos, o la creación de la plataforma web usando el módulo Flask que ofrece el propio Python.

3.2. Perceval

Perceval [6] es la librería de Python que me recomendó mi tutor para poder conseguir la mayoría de información necesaria para realizar los análisis de este proyecto. Consiste en un módulo que consigue recuperar datos relacionados con el desarrollo software de varias posibles fuentes como pueden ser Confluence, Bugzilla o Slack, aunque en este trabajo solo se ha usado para obtener datos de repositorios Git.

Del modo que se ha usado Perceval en mi caso, ha sido para poder recuperar los commits de los distintos repositorios en formato JSON, lo que me ha permitido ver el historial de cambios de los repositorios y las personas que han contribuido a los mismos para realizar los análisis.

3.3. Flask

Flask [2] es un módulo de Python que consiste en un framework ligero para la creación de aplicaciones web. Está diseñado para ser sencillo y rápido, pero escalable a la construcción de aplicaciones complejas. Se usa en distintos ámbitos como aplicaciones web, como es el caso de este proyecto, pero también en microservicios, desarrollos de APIs...

Se ha escogido Flask sobre otros frameworks para Python como puede ser Django ya que ofrece más flexibilidad y me permitía elegir los demás módulos que fueran necesarios para la personalización de mi web y hacerla de la manera que he pensado.

3.4. Jinja2

Jinja2 [5] es el motor de plantillas que usa Flask. Permite la generación de contenido web dinámico en la aplicaciones de manera eficiente y estructurada. Una de sus características principales que se ha usado en este proyecto es la utilización de una estructura base que cualquier otra plantilla puede heredar, lo que permite ahorrar bastante tiempo de desarrollo HTML o CSS.

Su uso me ha permitido crear las páginas dinámicas en la que aparecen los análisis realizados de los repositorios, permitiéndome representarlos de manera sencilla y cómoda para mí como desarrollador, pero también, a imagen del usuario que podría visitar la web, fácil de entender.

3.5. Git

Git [3] es un sistema de control de versiones usado en proyectos de desarrollo de software. Realiza un seguimiento de los cambios en archivos de un proyecto, por lo que es muy útil en los casos donde un grupo de personas realizan estos cambios en los mismos ficheros al mismo tiempo.

Permite a los desarrolladores conocer todo el historial de modificaciones que se han realizado sobre el proyecto, lo que facilita la organización y el alineamiento entre los propios contribuyentes. Se utiliza tanto en proyectos de código abierto y gratuito como este, como en proyectos comerciales de empresas, que utilizan Git para la gestión de versiones del software que crean.

En este proyecto, además de para los repositorios Git que he escogido para realizar el análisis, se ha utilizado para hacer también, el control de versiones de mi propio proyecto, por lo que es fácil ver el progreso que ha seguido mi software a lo largo del tiempo, pues es posible ver el historial de cambios hechos durante este año.

3.6. GitHub

GitHub [4] es una plataforma para almacenar, compartir y trabajar junto a otros usuarios en proyectos de desarrollo. Sus principales características son:

- Compartir el trabajo junto a otras personas.

- Seguir los cambios en el código a lo largo del tiempo, para el control de versiones de un proyecto.
- Colaborar en proyectos con multitud de personas.
- Permitir a otros usuarios de la plataforma que puedan revisar el código y realizar sugerencias para su mejora.
- Creación y gestión de repositorios donde almacenar todo el código de un proyecto.

Todos los repositorios analizados en este proyecto forman parte de la propia plataforma de GitHub, incluyendo también mi propio trabajo de fin de grado.

3.7. SQLAlchemy

SQLAlchemy [8] es una librería de Python usada para la interacción con una gran variedad de bases de datos. Permite crear modelos de datos y consultas de una manera sencilla y cercana a las clases de Python.

Es un ORM (Object Relational Mapper), por lo que permite el mapeo de tablas de las bases de datos sin tener que escribir las consultas, sino que mediante los objetos de Python, podemos hacer esta función.

Lo he usado en mi implementación con la base de datos SQLite que se ha hecho y que contiene el análisis de todos los repositorios.

3.8. SQLite

SQLite [11] es una base de datos integrada de código abierto. Su principal característica y diferencia con la mayoría de bases de datos SQL es que no tiene un proceso independiente al de la aplicación, sino que se encuentra contenida en la propia aplicación.

Lo he usado por la facilidad que ofrece para la configuración de la propia base de datos, ya que, al estar dentro de la propia aplicación, no necesita configuración de red, lo que facilita bastante el uso de bases de datos para personas como yo, que no son expertos en ello.

3.9. HTML

HTML es un lenguaje de marcado que utiliza una serie de etiquetas con el objetivo de dar estructura a un contenido web. Representa el estándar de la visualización de páginas web y es utilizado por todos los grandes navegadores actuales.

Todas las páginas que forman la aplicación web se han creado y estructurado mediante ficheros HTML.

3.10. CSS

CSS es el lenguaje mayoritariamente utilizado para dar estilo a una página web. Este lenguaje permite vincular los documentos de texto en formato HTML con hojas de estilo que contiene la información topográfica de los elementos visuales de la página, y que además, permite separar completamente la estructura de los contenidos, del estilo que estos van a tener ya en la visualización en la web.

En este proyecto se ha utilizado CSS en todas las páginas para dar el estilo visual a los contenidos, utilizando además la librería Bootstrap para facilitar un diseño simple y que sirviera de plantilla para tener consistencia visual en toda la aplicación.

3.11. Bootstrap

Bootstrap [1] es un proyecto de código abierto que consiste en un framework de front-end que facilita el desarrollo de aplicaciones mediante un conjunto de herramientas y componentes pre-hechos como botones o formularios que permiten la personalización del diseño de una web.

Se ha usado Bootstrap en el diseño de todas las páginas incluyendo botones, formularios, o el encabezado de la página web.

Capítulo 4

Diseño e implementación

En este apartado de la memoria se detalla el diseño de la aplicación y de todos sus componentes.

4.1. Arquitectura general

Este proyecto sigue un modelo cliente-servidor. El cliente en este caso, para el uso que está pensado, sería la aplicación web que se ejecuta desde el navegador y que permite realizar las peticiones de una manera más cómoda, accesible y sencilla al servidor, aunque obviamente, se podrían hacer estas peticiones a mano sin utilizar la aplicación web, y seguiría siendo el cliente. El servidor, mientras tanto, espera a atender las peticiones que se envían a los endpoints que tiene definidos para finalmente dar la respuesta a los clientes.

Aunque en puntos posteriores entraremos más a fondo en cada uno de elementos de la aplicación, el flujo que sigue, a modo general es el siguiente:

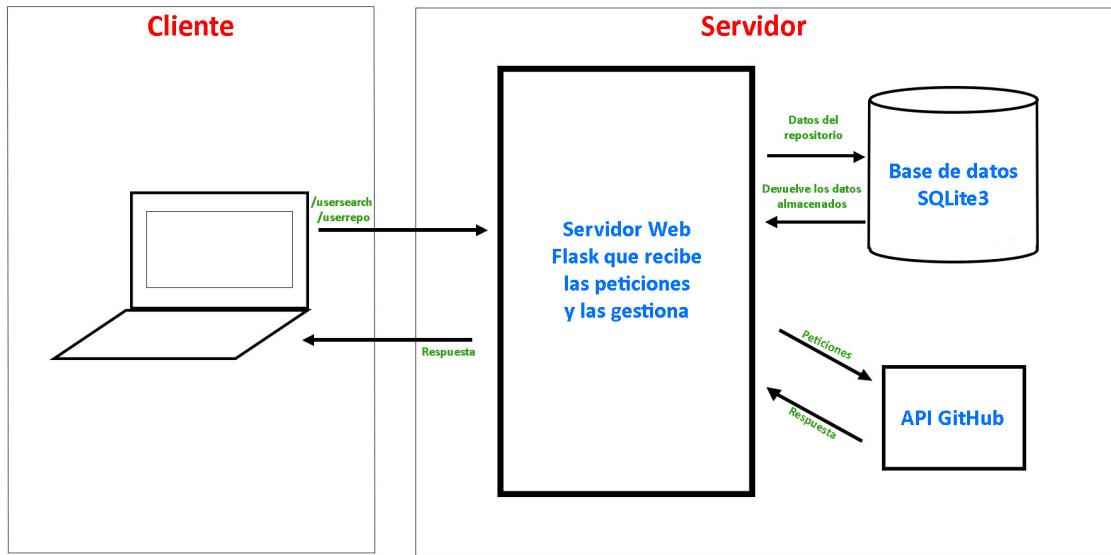


Figura 4.1: Arquitectura general de la aplicación

Un cliente cualquiera accede a la página web del proyecto, a la que he llamado OSSAnalyzer (viene de Open Source Software Analyzer), y haciendo click en el botón “Búsqueda de un repositorio concreto” en el encabezado de la página principal, provoca que el navegador lance una petición GET al endpoint principal de la aplicación (/userrepo). El servidor, después, ofrece en la respuesta un formulario donde el usuario deberá introducir el nombre del repositorio y el nombre del propietario del repositorio, con el objetivo de hacer una petición POST al servidor, también al endpoint /userrepo. Al llamar a este endpoint con estos datos y con el método POST, el servidor realiza toda la parte interna de recolección de datos usando Perceval, realiza también el análisis de los datos obtenidos y además, también realiza la representación de este análisis en la propia aplicación web. No hay que olvidar también la base de datos que forma parte de la aplicación; ya que los datos de los repositorios que se vayan a analizar, es decir, que se vayan a introducir en el endpoint /userrepo se van a guardar en la base de datos para tenerlos almacenados.

Además, está disponible también el endpoint /usersearch, que facilita la búsqueda de los repositorios perteneciente a un usuario que el cliente debe escoger. Esto se hace de manera que con una petición GET, el servidor devuelve al cliente un formulario que debe llenar con el usuario propietario, y al enviarlo, se realiza una petición POST, con la que el servidor, des-

pués de procesarla, le ofrece al cliente la lista de repositorios que posee ese usuario para que posteriormente pueda analizarlos.

4.2. Estructura del código del proyecto

Para hacer más sencilla la explicación de partes del código en puntos posteriores, hablaré de la estructura de código del proyecto:

```
OSSAnalyzer/
|---> main.py
|
|---> utils.py
|
|---> requests_to_github_api.py
|
|---> readme.md
|
|---> config/
|     |
|     |---> OSSAnalyzerConfig.json
|
|---> logs/
|     |
|     |---> OSSAnalyzer.log
|
|---> web/
|
|---> endpoints.py
|
|---> models.py
|
```

```
| ---> __init__.py
|
| ---> database.db
|
| ---> static/
|     |
|         | ---> mainpage.css
|     |
|         | ---> mainpage.js
|     |
|         | ---> userrepo.css
|     |
|         | ---> userrepo.js
|     |
|         | ---> formulario.css
|     |
|         | ---> formulario.js
|     |
|         | ---> images/
|             |
|                 | ---> logoaplicacion.png
|
| ---> templates/
|
|     | ---> base.html
|
|     | ---> mainpage.html
|
|     | ---> userrepoForm.html
|
|     | ---> userrepoResult.html
```

```
|  
| ---> usersearchForm.html  
|  
| ---> usersearchResult.html .
```

- **main.py:** Contiene la información necesaria para arrancar la aplicación.
- **utils.py:** Contiene todas las funciones usadas para la recolección de datos de los repositorios obtenidos usando la librería Perceval. También contiene métodos para calcular los lenguajes minoritarios y los contribuyentes que han participado en los mismos.
- **requests_to_github_api.py:** Contiene todas las funciones usadas para la recolección de datos extra mediante llamadas a la API de GitHub.
- **readme.md:** Es un fichero Markdown con una breve introducción a la aplicación. Se usa sobre todo para que al entrar en la página del repositorio en GitHub, cualquier usuario pueda ver fácilmente de qué trata el proyecto.
- **config/OSSAnalyzerConfig.json:** Es un fichero de configuración en formato JSON que se usa principalmente para guardar configuraciones que sean modificables fácilmente y no haya que cambiar más código si fuera necesario. Contiene información sobre la IP y el puerto donde se inicia la aplicación, configuración de los logs, el token que se usa en todas las llamadas contra la API de GitHub y un diccionario con una gran cantidad de extensiones y el lenguaje que representan.
- **logs/OSSAnalyzer.log:** En este fichero de log se escribe continuamente el estado de la aplicación, las peticiones que llegan y sus resultados, errores e información extra. Realmente este fichero tendría sentido si la aplicación estuviera corriendo en un servidor continuamente, ya que sirve para investigar posibles fallos.
- **web/models.py:** En este fichero se definen los modelos de las tablas con sus correspondientes columnas en la base de datos.
- **web/database.db:** En este fichero se almacena la base de datos SQLite3.

- **web/endpoints.py:** En este fichero se definen todos los endpoints de la aplicación y las acciones que realizan cada uno de ellos.
- **web/_init_.py:** Contiene la información para crear la aplicación y cargar todos los elementos, la base de datos, y los endpoints.
- **web/static:** En este directorio se almacenan las imágenes, y los ficheros CSS y JavaScript de las páginas.
- **web/templates:** Este directorio almacena los ficheros HTML con los templates que se modifican.

4.3. Comenzando a usar la aplicación

El primer paso para comenzar a usar la aplicación es, obviamente, acceder a la web desde un navegador. Al entrar se presenta una página de inicio en la que se informa al usuario de la finalidad del proyecto y una pequeña guía para facilitar el uso de la aplicación, aunque es bastante sencillo.

El usuario, debe escoger en ese momento entre dos opciones que se dan en el encabezado de la página:

- Búsqueda de un repositorio concreto: Llama al endpoint /userrepo de la aplicación usando un método GET.
- Búsqueda por usuario: Llama al endpoint /usersearch de la aplicación usando un método GET.

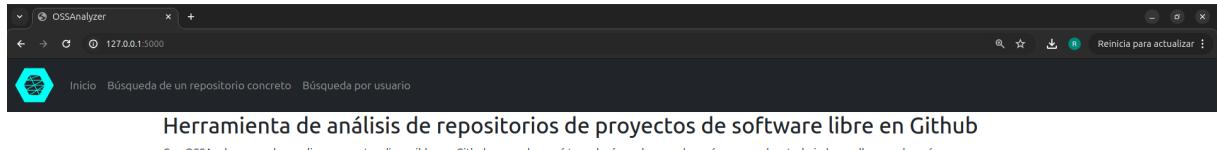
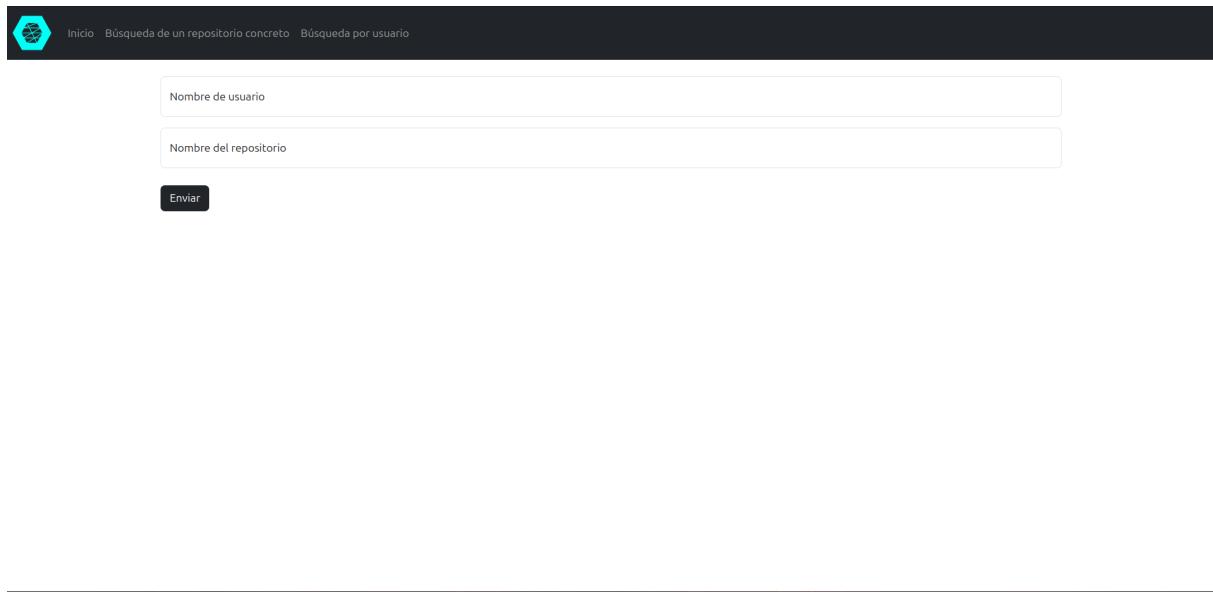


Figura 4.2: Página inicial de la aplicación

4.4. Endpoint /userrepo

En el endpoint /userrepo es donde se encuentra la gran mayoría de la lógica de la aplicación.

Al llamar a este endpoint con un método GET, el servidor responde a la petición con un formulario con dos campos que el cliente deberá llenar: el nombre del repositorio, y el nombre del propietario del repositorio. También, se entrega en la respuesta un botón de envío, que al pulsarlo, provoca que el navegador lance una petición al endpoint /userrepo, pero ahora usando un método POST. Al ser una petición POST, los parámetros se encuentran en el body, y en este caso son dos: username (nombre de usuario del propietario del repositorio) y repo (nombre del repositorio).



The screenshot shows a search interface for a concrete repository. At the top, there is a navigation bar with a logo, the text 'Inicio', 'Búsqueda de un repositorio concreto', and 'Búsqueda por usuario'. Below the navigation bar is a search form. The form contains two input fields: 'Nombre de usuario' (User name) and 'Nombre del repositorio' (Repository name). Below the input fields is a dark button labeled 'Enviar' (Send).

Figura 4.3: Respuesta con el formulario a la petición con método GET

Una vez que llega al servidor esa petición POST, comienza la lógica del análisis del repositorio que el usuario ha escogido:

La primera tarea que realiza el servidor es la obtención de la rama por defecto del repositorio y el identificador del último commit que se ha subido en esa misma rama. El sentido de esto tiene que ver con la implementación con la base de datos, ya que desde un punto de vista de rendimiento, no tendría sentido guardar de nuevo en la base de datos un repositorio cuya última versión ya tenemos almacenado. La comprobación que realiza el servidor para saber si ese repositorio ya está almacenado en la base de datos se hace mediante una consulta contra la base de datos. Si ese repositorio no está almacenado, o si la fecha del último commit es anterior a la que se ha obtenido anteriormente, se procede con la recolección de datos del repositorio.

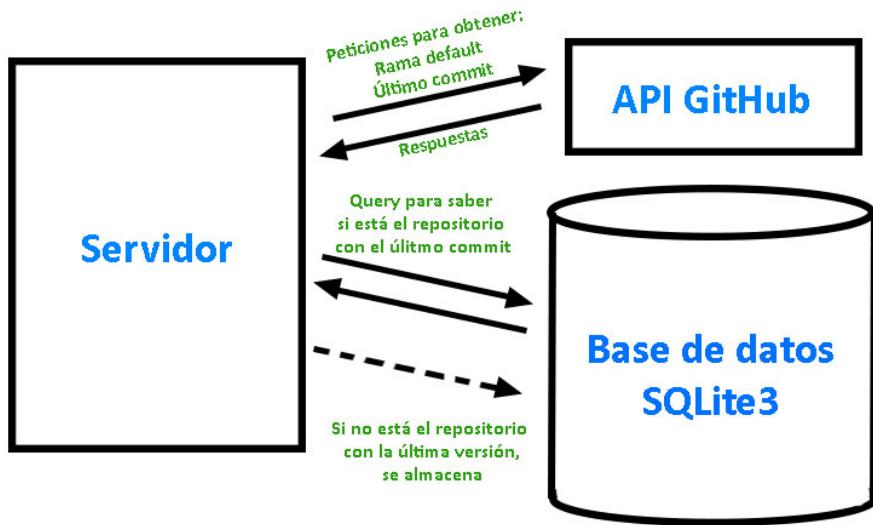


Figura 4.4: Flujo de la aplicación para consultar si existe la última versión de un repositorio en la base de datos

Como hemos comentado en puntos anteriores, la recolección de datos de los repositorios se ha hecho usando la librería Perceval. Perceval nos permite obtener todos los datos de los commits realizados sobre un repositorio Git en formato JSON, lo que facilita después la selección de datos que nos interesan, y descartar aquellos datos inservibles para nuestro análisis.

Los datos que podemos sacar del uso de Perceval y que nos interesan en nuestro análisis principalmente, son:

- Lista de usuarios que han realizado commits sobre el repositorio: para ello se ha creado la función `obtain_users` en `utils.py`, que genera esta lista accediendo al campo `Author` de cada uno de los objetos (cada uno de los objetos representa un commit) del JSON obtenido del uso de Perceval.
- Obtener los ficheros que ha modificado cada uno los usuarios de la lista creada anterior: usando la función `obtain_users_files` de `utils.py`, se genera un diccionario de listas con los archivos en los que ha trabajado cada usuario que ha participado en el repositorio.
- Obtener las extensiones de los ficheros que ha modificado cada usuario para conocer el lenguaje sobre el que se han hecho las modificaciones: con la función `obtain_files_extension`

del fichero utils.py se logra crear un diccionario con los lenguajes. Para ello se usa un diccionario llamado extensiones que se encuentra almacenado en un fichero de configuración JSON (OSSAnalyzerConfig.json). En este diccionario de extensiones se almacena como clave la extensión, y como valor el lenguaje de programación. Para la construcción de este diccionario se ha usado el fichero YAML languages.yml del repositorio Linguist de GitHub, que contiene una gran cantidad de lenguajes.

- Obtener el total de modificaciones realizadas sobre un mismo lenguaje de programación de cada uno de los contribuyentes al repositorio: para ello se ha creado la función counter_ext, almacenada también en utils.py y cuya salida consiste en un diccionario con la suma de modificaciones realizadas sobre todos los lenguajes que cada uno de los usuarios ha realizado.

Además para enriquecer algo más el análisis y ofrecer otros datos que no podemos obtener usando Perceval, se han incluido también algunas llamadas a la API de GitHub para obtener los siguientes datos:

- Porcentaje de uso de cada uno de los lenguajes usados en el repositorio. Mediante la función obtain_used_languages_on_repo del fichero requests_to_github_api.py se obtienen todos los ficheros almacenados en el repositorio. Una vez tenemos la lista con todos los ficheros, usando su extensión, se convierte a "lenguaje" de igual manera que se hace con los datos que hemos obtenido de Perceval anteriormente, es decir, usando el diccionario extensiones del fichero de configuración OSSAnalyzerConfig.json. Con ello ya tenemos el total de ficheros de cada lenguaje, por lo que es fácil obtener el porcentaje de cada uno de los lenguajes sobre el total.
- Número total de ficheros del repositorio. Se obtiene usando la función obtain_num_files del fichero requests_to_github_api.py.

Por último, para terminar nuestra recolección de datos, puesto que nos centramos en los lenguajes minoritarios, necesitamos saber qué lenguajes son minoritarios en un repositorio, y la información sobre los mismos. Entendemos que un lenguaje es minoritario en un repositorio si representa una cuota menor del 5 % del total de un repositorio. Por ejemplo:

Un repositorio contiene 5 lenguajes de programación: Python, HTML, CSS, JavaScript y un Dockerfile. Python representa el 80 % del repositorio, HTML un 10 %, CSS un 7 %, JavaScript, un 2 % y Dockerfile, el 1 % restante. En este caso, los lenguajes minoritarios serán JavaScript y Dockerfile.

Teniendo esto en mente, debido a que se obtiene el porcentaje de uso de cada uno de los lenguajes usado en los repositorios como se ha comentado anteriormente, se puede saber cuales de los lenguajes que forman parte de un repositorio son minoritarios. Ahora que tenemos los lenguajes minoritarios, buscamos el resto de información sobre estos: los commits hechos sobre esos lenguajes y quién de los usuarios contribuyentes han realizado estos commits.

Todo esto se hace de la siguiente manera:

- Obtención de los lenguajes minoritarios: con el diccionario de porcentajes de cada uno de los lenguajes de un repositorio, usando la función `obtain_min_languages` del fichero `utils.py` nos quedamos con una lista con solo los lenguajes que representan menos del 5 % que hemos definido.
- Número total de commits realizados sobre esos lenguajes minoritarios: se realiza un diccionario en el que se hace la suma total de commits realizados sobre cada lenguaje minoritario usando la lista anterior y los datos que hemos obtenido de los commits con Perceval con el total de modificaciones realizadas cada uno de los lenguajes por cada contribuyente al repositorio. Esta función la realiza el método `obtain_total_commits_min_languages` del fichero `utils.py`.
- Obtención de los contribuyentes y su participación en los lenguajes minoritarios: mediante la función `top_contribuyentes_por_lenguaje` del fichero `utils.py` se construye un diccionario con el número de commits realizados por los usuarios que han contribuido en estos lenguajes.

Ya se tienen todos los datos necesarios para nuestro análisis, por tanto, se procede a su almacenamiento en la base de datos.

En caso de que el repositorio no esté almacenado, se crea una fila en la tabla `Repository` de la base de datos con diferentes columnas con cada uno de los datos de interés para el análisis. El otro caso que se puede dar es que el repositorio que se haya analizado sea uno que ya existía

en la tabla Repository, pero de una versión posterior, por lo que en lugar de crear una nueva fila, se actualizan las columnas de la fila ya existente, de manera que no se gasta espacio innecesario en la base de datos.

Estructura Hoja de datos Editar pragmas Ejecutar SQL						
Tabla:	repository	Filtros	Filtrar en cualquier columna			
1	tiangolo	fastapi	["Sebasti\u00e1n Ram\u00edrez ", ...]	{"data": {"Sebasti\u00e1n Ram\u00edrez ": ...}}	("YAML": 2.52, "Python": 56.74, ...)	2441 ["YAML", "CSS", "SVG", "JavaScript", ...]
2	louisla	uptime-kuma	["LouisLam ", "Louis ", "Louis Lam ", ...]	{"data": {"LouisLam ": {"HTML": 1, ...}}	("JavaScript": 41.1, "YAML": 2.39, ...)	567 ["YAML", "Markdown", "Vim Help File", ...]
3	sindresor...	awesome	["Sindre Sorhus ", "Boris K ", ...]	{"data": {"Sindre Sorhus ": {"SVG": ...}}	("YAML": 6.67, "Shell": 6.67, ...)	23 []
4	microsoft	fluentui	["Eric Thompson ", "David Zearing ", ...]	{"data": {"Eric Thompson ": ...}}	("OASv3-json": 10.31, "Markdown": ...)	19722 ["YAML", "CSS", "HTML", "SVG", "MDX", ...]
5	rust-lang	mdBook	["Mathieu David ", "Kevin K ", ...]	{"data": {"Mathieu David ": {"Ignore": ...}}	("YAML": 2.5, "Markdown": 45.5, ...)	222 ["YAML", "TOML", "Shell", "SVG", ...]
6	fogleman	sdf	["Michael Fogelman ", "Marc ", ...]	{"data": {"Michael Fogelman ": ...}}	("Markdown": 7.69, "Go": 3.85, ...)	94 ["Go"]
7	vercel	vercel	["Guillermo Rauch ", "Tony Kovanc ", ...]	{"data": {"Guillermo Rauch ": ...}}	("Markdown": 26.0, "OASv3-json": ...)	11957 ["YAML", "CSS", "HTML", "Astro", ...]
8	dracula	dracula-theme	["Zeno Rocha ", "Vagner Santana ", ...]	{"data": {"Zeno Rocha ": {"Shell": 2, ...}}	("Markdown": 75.0, "YAML": 25.0)	7 []
9	adobe	react-spectrum	["Nate Ross ", "nross ", "Aaron Hard..., ...]	{"data": {"Nate Ross ": {"Markdown": ...}}	("JavaScript": 6.54, "CSS": 3.45, ...)	8129 ["CSS", "HTML", "YAML", "Markdown", ...]
10	framework...	framework7	["Vladimir Kharlampidi ", "Bittdeli ..., ...]	{"data": {"Vladimir Kharlampidi ": ...}}	("JavaScript": 34.45, "Markdown": ...)	1521 ["Markdown", "YAML", "OASv3-json", ...]
11	python-...	poetry	["S\u00f3nchez Basti\u00e1n Eustace ", "Igor ..., ...]	{"data": {"S\u00f3nchez Basti\u00e1n Eustace ": ...}}	("YAML": 2.35, "Markdown": 3.23, ...)	904 ["YAML", "Markdown", "Shell", ...]
12	miniflux	miniflux	["Fr\u00e1n\u00e9s Guillot ", "Eric Guillot ", ...]	{"data": {"Fr\u00e1n\u00e9s Guillot ": ...}}	("OASv3-json": 4.77, "YAML": 3.73, ...)	563 ["OASv3-json", "YAML", "Markdown", ...]
13	caddyserv...	caddy	["Matthew Holt ", "Matt Holt ", ...]	{"data": {"Matthew Holt ": {"Ignore": ...}}	("Markdown": 0.9, "YAML": 2.71, "Go": ...)	500 ["Markdown", "YAML", "HTML", "Shell", ...]
14	directus	directus	["olov ", "Max Glantzman ", "Ben ", ...]	{"data": {"olov ": {"Ignore List": 3, ...}}	("OASv3-json": 2.83, "YAML": 7.55, ...)	3437 ["OASv3-json", "JavaScript", ...]
15	getsetnry	symbolicator	["Markus Unterwaditzer ", "Anton ..., ...]	{"data": {"Markus Unterwaditzer ": ...}}	("YAML": 5.49, "OASv3-json": 2.2, ...)	323 ["OASv3-json", "Shell", "CSS", ...]
16	novnc	novnc	["Joel Martin ", "Kevin Chan ", ...]	{"data": {"Joel Martin ": {"HTML": ...}}	("Markdown": 5.11, "YAML": 3.41, "Vim ..., ...)	213 ["YAML", "Vim Help File", "CSS", ...]
17	darkreader	darkreader	["alexanderby ", "Alexander Shutov ", ...]	{"data": {"alexanderby ": {"Ignore": ...}}	("JavaScript": 5.13, "YAML": 1.41, ...)	927 ["YAML", "Markdown", "Ruby", "Vim ..., ...]
18	tootsuite	mastodon	["Eugen Rochko ", "Eugen ", "Yann ..., ...]	{"data": {"Eugen Rochko ": {"Ignore": ...}}	("YAML": 6.91, "OASv3-json": 1.38, ...)	8637 ["OASv3-json", "Vim Help File", ...]
19	xtermjs	xterm.js	["Christopher Jeffrey ", "Carson ..., ...]	{"data": {"Christopher Jeffrey ": ...}}	("OASv3-json": 15.08, "Markdown": ...)	671 ["Markdown", "YAML", "Dotenv", ...]
20	felangel	bloc	["Felix Angelov ", "alexey ", "Rody ..., ...]	{"data": {"Felix Angelov ": {"Ignore": ...}}	("YAML": 7.15, "Markdown": 5.17, ...)	2986 ["JavaScript", "OASv3-json", "SVG", ...]
21	projectdji	nuclei-...	["Ritesh Gohil ", "Ice3man ", ...]	{"data": {"Ritesh Gohil ": {"YAML": ...}}	("YAML": 97.14, "Python": 0.03, ...)	9887 ["Python", "Shell", "Vim Help File", ...]
22	cozy	cozy-stack	["Bruno Michel ", "Tristan Nitot ", ...]	{"data": {"Bruno Michel ": {"Ignore": ...}}	("YAML": 1.44, "OASv3-json": 0.58, ...)	1138 ["YAML", "OASv3-json", "CSS", ...]
23	oclif	oclif	["Jeff Dickey ", "heroku-cli ", "roo..., ...]	{"data": {"Jeff Dickey ": {"OASv3-...}}	("OASv3-json": 14.91, "Markdown": ...)	171 ["YAML", "Batchfile", "JavaScript"]
24	sebastian...	phpunit	["Sebastian Bergmann ", "Tobias ..., ...]	{"data": {"Sebastian Bergmann ": ...}}	("Markdown": 0.53, "YAML": 0.11, ...)	2682 ["Markdown", "YAML", "OASv3-json", ...]
25	graphql	graphql-...	["Tim Suchanek ", "timsuchanek ", ...]	{"data": {"Tim Suchanek ": {"Ignore": ...}}	("YAML": 2.32, "Markdown": 9.27, ...)	337 ["YAML", "SVG", "CSS", "HTML", ...]
26	prisma-...	graphql-...	["Johannes Schickling ", "Nilan ..., ...]	{"data": {"Johannes Schickling ": ...}}	("Markdown": 16.02, "YAML": 0.59, ...)	1550 ["YAML", "OASv3-json", "Vim Help ..., ...]
27	gridsome	gridsome	["Hans-J\u00fclgen Vedvik ", "Tommy ..., ...]	{"data": {"Hans-J\u00fclgen Vedvik ": ...}}	("JavaScript": 62.91, "Markdown": ...)	445 ["YAML", "HTML", "SVG", "GraphQL", ...]

Figura 4.5: Tabla Repository

También, además de almacenar cada repositorio en la tabla Repository, se guardan todos los usuarios que han participado en cada uno de los repositorios analizados junto a los repositorios que ha participado y los commits que ha realizado en los mismos. Esto nos permite ver el expertise de cada uno de los usuarios y conocer en qué lenguaje se especializa y qué número de lenguajes totales ha usado en todos los repositorios que haya participado y se hayan analizado previamente en la aplicación. Todo esto se almacena en la tabla User_expertise de la base de datos.

Estructura			
Hoja de datos			
Editar pragmas			
Ejecutar SQL			
Tabla:	user_expertise	Filtrar en cualquier columna	
	id	user	repositories_contribution
1	1	Sebastián Ramírez	["fastapi", "keras"]
2	2	Maria Camila Gutierrez	["fastapi"]
3	3	euri10	["fastapi"]
4	4	Ken Kinder	["fastapi"]
5	5	Kabir Khan	["fastapi"]
6	6	Zaar Hai	["fastapi"]
7	7	Matt Hegarty	["fastapi"]
8	8	Stratos Gerakakis	["fastapi"]
9	9	James Saunders	["fastapi"]
10	10	The Gitter Badger	["fastapi", ...]
11	11	yihuang	["fastapi", "redis"]
12	12	Daniel Hahler	["fastapi", "poetry"]
13	13	Alif Jahan	["fastapi"]
14	14	Alex Iribarren	["fastapi", ...]
15	15	Mohammed	["fastapi"]
16	16	Mostapha Sadeghipour Roudsari	["fastapi"]
17	17	Matthew McLeod	["fastapi"]
18	18	William Hayes	["fastapi", "poetry"]
19	19	hayata-yamamoto	["fastapi"]
20	20	Daniel Michaels	["fastapi"]
21	21	Christopher Dignam	["fastapi", ...]
22	22	Steve B	["fastapi"]
23	23	Derek J. Lambert	["fastapi"]
24	24	Ricardo Momm	["fastapi"]
25	25	zamiramir	["fastapi"]
26	26	Trim21	["fastapi", "poetry"]
27	27	Steinþor Palsson	["fastapi"]
28	28	James Kaplan	["fastapi"]

Figura 4.6: Tabla User_expertise

Por último, se almacenan los datos de los distintos lenguajes que han formado ese repositorio. Para ello, se crea un diccionario usando los datos obtenidos con Perceval cuyas claves son los lenguajes, y los valores son una lista con el número de commits totales realizados, el número de usuarios que ha hecho commits sobre ese lenguaje, y la lista de usuarios que hayan hecho los commits. Se almacenan en la tabla Languages.

Tabla: languages					
	id	lan_name	lan_num_users	lan_users	lan_num_commits
1	19	XML	11833	["Kerry ", "Carlos Ortiz Garcia ", ...]	687710
2	12	JavaScript	10398	["Joakim Riedel ", "Kerry ", "Willia..."]	469428
3	1	Python	21268	["", "Jake Tae ", "Tom Christie ", ...]	395461
4	6	YAML	9353	["", "Vlad Korobov ", "Ignacio ..."]	381840
5	14	OASv3-json	12094	["Kerry ", "Carlos Ortiz Garcia ", ...]	305591
6	8	Markdown	41307	["Bryan Luby ", "Kerry ", "Dmitry ..."]	290464
7	22	Go	3671	["Mos Roshanavand ", "Benjamin Schoc..."]	158410
8	96	C++	5049	["Dominik Spicher ", "nickjackolson ..."]	110348
9	7	SVG	1010	["Vlad Korobov ", "dshukertjr ", ...]	75317
10	32	MDX	2012	["Jake Tae ", "Neeraj-2307 ", ...]	62140
11	65	Objective-C	3133	["John L. Jegutanis ", "chendianqian..."]	58480
12	11	HTML	2874	["Rich Harris ", "bigbadcapers ", ...]	39618
13	26	PHP	1375	["Alexandre Borela ", "Miroslav ..."]	35579
14	9	Vim Help File	5064	["", "BigMoby ", "Gabor SZOLLOSI ", ...]	32198
15	30	Jest Snapshot	1013	["petdun2519 ", "Yamagishi Kazutoshi..."]	30868
16	16	Vue	1058	["Wasim Thoufiq ", "Brandon Scott ",...]	26193
17	10	CSS	1987	["Nikhil Dabas ", "Jim Frode Hoff ",...]	23401
18	1...	Verilog	689	["Kaiyin Zhong ", "mir.zhou ", ...]	22971
19	3	Shell	4989	["Brandon Weaver ", "Alan Quach ", ...]	21804
20	47	reStructured...	2416	["NarayanAdithya ", "danai-antoniu..."]	19903
21	62	C	1604	["chendianqiang ", "Ye Lin Aung ", ...]	19188
22	44	Ruby	881	["Binbin ", "Filippo Giunchedi ", ...]	14098
23	17	SCSS	1358	["Brian Van Eimeren ", "bigbadcapers..."]	13245
24	93	Dart	322	["dshukertjr ", "Cacing69 ", "Brando..."]	12356
25	25	Java	413	["Kumataro ", "Simon Heinen ", ...]	12099
26	1...	Zig	464	["ggobbe ", "Will Richards 2 ", ...]	10114
27	29	Less	609	["Billy Vong ", "Nikhil Dabas ", ...]	9691

Figura 4.7: Tabla Languages

Una vez almacenados los distintos datos, solo queda la representación de los mismos en la aplicación web. Se presenta de la siguiente manera:



Figura 4.8: Respuesta a la petición POST con la representación gráfica del análisis

Se informa al usuario del repositorio que se ha pedido analizar y el total de ficheros que tiene el repositorio.

Se presenta una gráfica circular con la representación de porcentaje de cada uno de los lenguajes usados junto a una leyenda para informar al usuario del número de porcentaje y el color que representa. La gráfica se genera a partir del diccionario con el porcentaje de uso de cada uno de los lenguajes usados en el repositorio que se ha obtenido en la recolección de datos usando la librería Matplotlib. Además, se presentan los lenguajes minoritarios del repositorio y el número de commits totales realizados sobre cada uno de ellos. Si se pulsa en cualquiera de ellos, muestra a los usuarios que han realizado los commits y la cantidad.



Figura 4.9: Análisis de los lenguajes minoritarios

Además, se muestran todos los usuarios que han contribuido, y al hacer clic sobre cualquiera de ellos, muestra los commits realizados sobre cada uno de los lenguajes del repositorio.

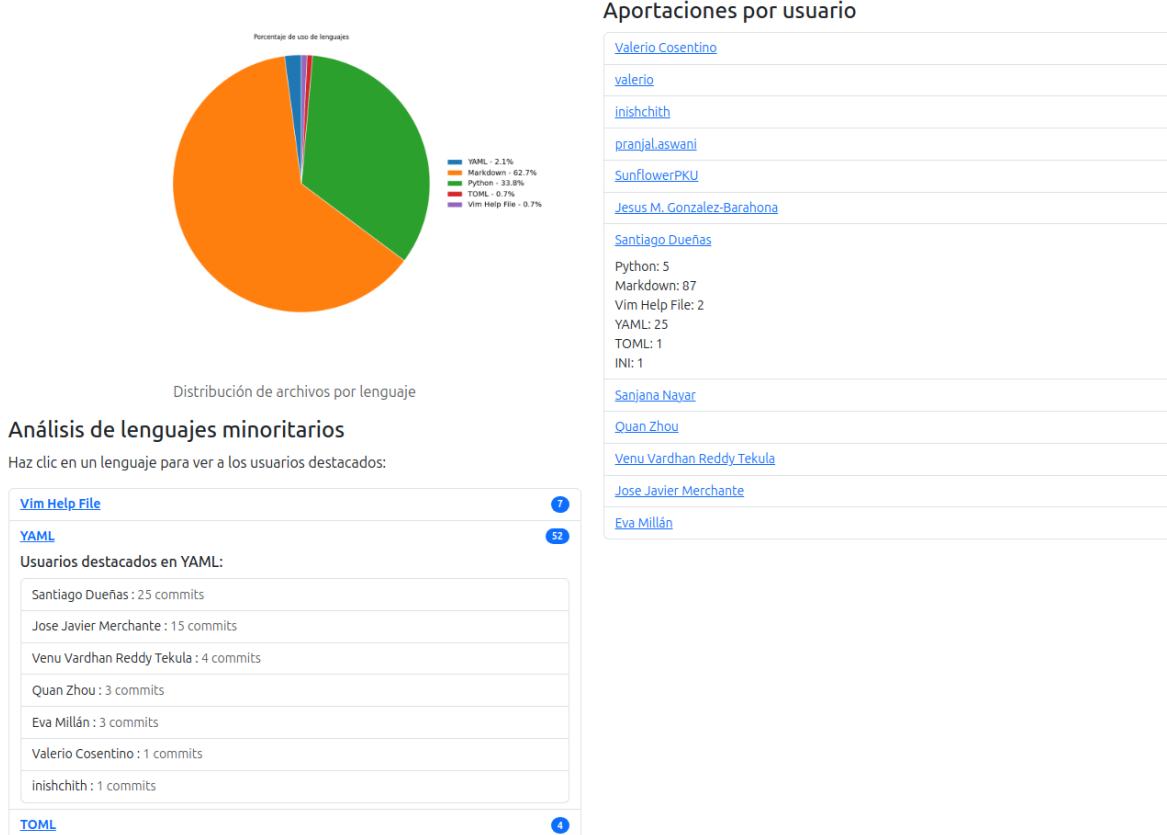


Figura 4.10: Contribuyentes al repositorio

Por otro lado, si el repositorio que el usuario pide analizar ya tiene los datos de la última versión almacenada en la base de datos, estos se recuperan mediante una consulta en la que se pide a la tabla Repository la fila correspondiente al repositorio, y con ella se recuperan de nuevo todos los datos para representarlos en la web.

4.5. Endpoint /usersearch

El objetivo del endpoint /usersearch es facilitar al usuario todos los repositorios públicos que pertenecen a un mismo propietario de manera que pueda localizar fácilmente los repositorios de un mismo usuario sobre los que hacer los análisis de lenguajes minoritarios. Sobre todo, está dirigido de cara a usuarios propietarios que consistan en un grupo de personas que suelen participar en los mismos proyectos.

El funcionamiento es de la siguiente manera:

- En caso de que llegue una petición al endpoint con método GET, se ofrece en la respuesta

un formulario donde el usuario debe indicar el nombre de usuario de GitHub del cual quiera consultar los repositorios junto a un botón de envío para provocar el envío de una petición al mismo endpoint /userrepo, pero en este caso, con método POST cuyo cuerpo contiene un parámetro llamado username y su valor es lo que haya introducido el usuario.

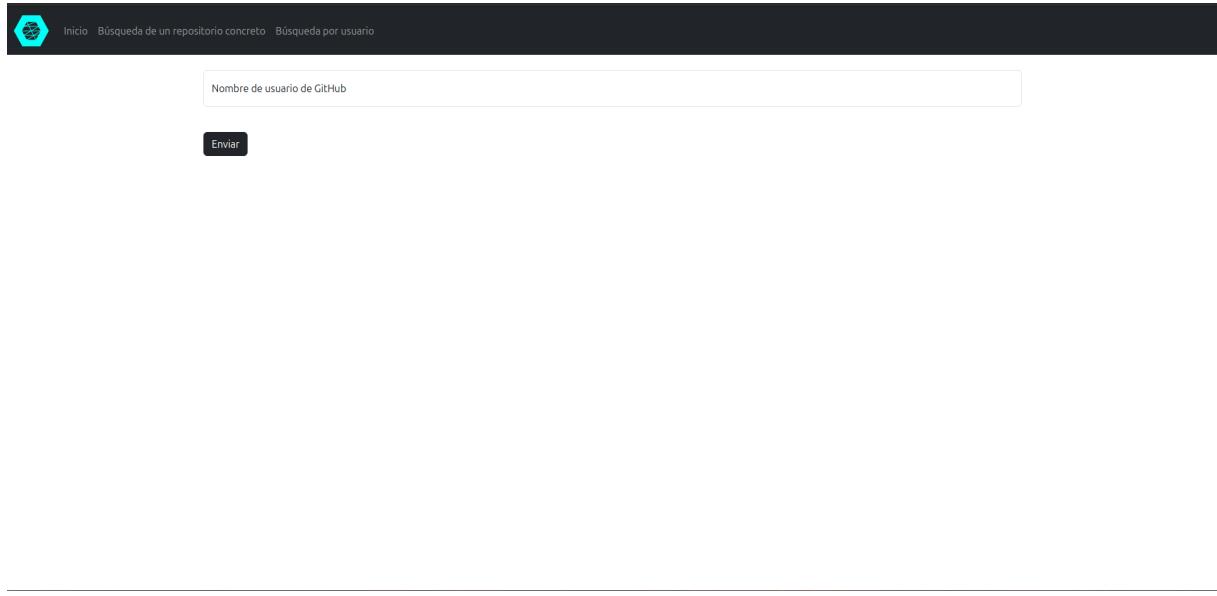


Figura 4.11: Respuesta a la petición GET al endpoint /usersearch

- Cuando llega esa petición POST, el servidor procede a buscar los repositorios públicos del usuario que ha llegado en el body de la petición. Esto se hace mediante la función obtain_user_repos del fichero requests_to_github_api.py, cuya salida es un diccionario con el nombre de todos los repositorios que posee ese usuario. Este diccionario es lo que da forma a la respuesta final del endpoint.

Al finalizar, se representa de la siguiente manera en la aplicación web:

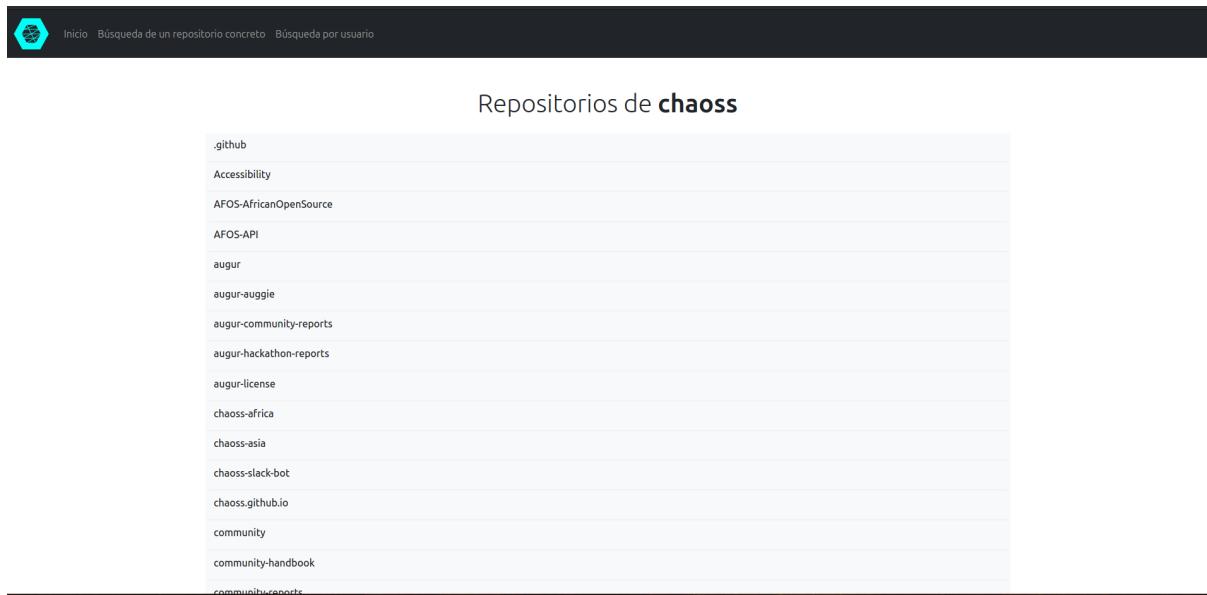


Figura 4.12: Respuesta a la petición POST al endpoint /usersearch

En la parte superior se informa al usuario del nombre de usuario que ha escogido.

En la parte inferior se muestran todos los repositorios públicos del usuario.

Lo que hace que facilite al usuario la experiencia de analizar varios repositorios de un mismo usuario es que, al clickar en cualquiera de los repositorios que aparecen en la lista, se realiza una llamada POST al endpoint /userrepo con el nombre de usuario y el nombre del repositorio. Como se ha explicado en el punto anterior, en el endpoint /userrepo es donde se realizan los análisis.

4.6. Interacción con la base de datos

Como se ha dicho anteriormente, es la base de datos está basada en SQLite3. Se han definido tres tablas en la base de datos:

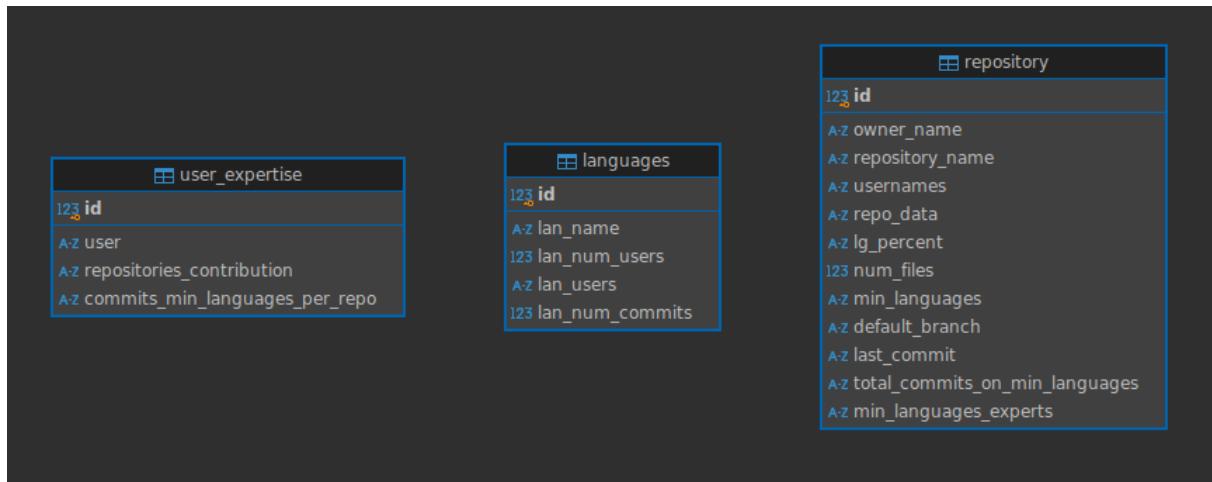


Figura 4.13: Diagrama entidad-relación de la base de datos

4.6.1. Tabla Repository

En la tabla Repository se almacenan los datos de los análisis de los repositorios. Se crea una fila por cada uno de ellos. Las columnas y datos que se guardan son:

- **id**: Identificador interno del repositorio en la base de datos.
- **owner_name**: Nombre de usuario propietario del repositorio.
- **repository_name**: Nombre del repositorio.
- **usernames**: Lista de los contribuyentes al repositorio.
- **repo_data**: Diccionario con los commits que ha realizado cada usuario sobre cada uno de los lenguajes.
- **lg_percent**: Diccionario con los porcentajes de uso de los lenguajes.
- **num_files**: Número total de ficheros del repositorio.
- **min_languages**: Lista con los lenguajes minoritarios.
- **default_branch**: Nombre de la rama por defecto del repositorio en GitHub.
- **last_commit**: Identificador del último commit realizado antes del análisis.

- **total_commits_on_min_languages:** Almacena un diccionario con el número de commits realizados sobre cada uno de los lenguajes minoritarios.
- **min_language_experts:** Diccionario con los lenguajes minoritarios, y la cantidad que ha contribuido cada uno de los usuarios que han realizado commits sobre los mismos.

4.6.2. Tabla User_expertise

Tiene como objetivo obtener información general de todos los usuarios, de cara a saber las tecnologías que conoce cada una de las personas y de qué manera ha participado en cada uno de los repositorios a los que ha contribuido. En la tabla User_expertise se almacenan todos los usuarios que han participado en alguno de los repositorios que se haya analizado. Cada fila presenta un usuario. Las columnas y datos que se guardan son:

- **id:** Identificador interno del usuario en la base de datos.
- **user:** Nombre de usuario.
- **repositories_contribution:** Lista de repositorios analizados en los que ese usuario ha contribuido.
- **commits_min_languages_per_repo:** Diccionario con el número de commits que ha realizado sobre cada uno de los lenguajes en cada repositorio.

4.6.3. Tabla Languages

En la tabla Languages se almacenan todos los lenguajes que han formado parte de alguno de los repositorios analizados, junto la suma total de commits, la lista de usuarios que han usado ese lenguaje y el número de usuarios que lo han usado. El objetivo de la tabla es tener la información de los distintos lenguajes, saber cuales son, en general, los lenguajes minoritarios, y de qué manera se usan. Los datos se guardan de la siguiente manera:

- **id:** Identificador interno del lenguaje en la base de datos.
- **lan_name:** Nombre del lenguaje.

- **lan_num_users:** Número de usuarios totales que han realizado commits sobre ese lenguaje.
- **lan_users:** Lista de usuarios que han realizado commits sobre el lenguaje.
- **lan_num_commits:** Número de commits totales que se han hecho sobre ese lenguaje.

Capítulo 5

Experimentos y validación

En este capítulo se explica un ejemplo del que podría ser el uso común de la aplicación. Para ello queremos a un usuario propietario de varios repositorios, cada uno de ellos con distintos lenguajes. Probaremos con la Netflix Open Source Pltaform, cuyo usuario es netflix. Este usuario posee 230 repositorios distintos. Analizaremos en profundidad uno de ellos, mientras que para otros 9, solo veremos el análisis final en la aplicación.

5.1. Ejemplo de uso de la aplicación con el análisis de un repositorio

Como hemos comentado en el punto anterior, nos ayudaremos de nuestra aplicación web para hacer una petición GET al endpoint /usersearch dando click en “Búsqueda por usuario”.

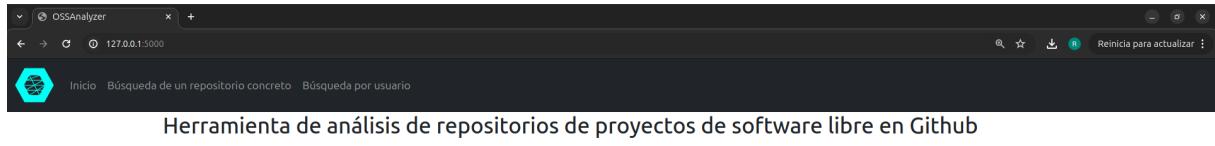


Figura 5.1: Página principal de la aplicación

Tras ello aparece el formulario donde deberemos indicar el nombre de usuario de github que es propietario de los distintos repositorios que vamos a analizar. En nuestro caso de prueba, netflix.

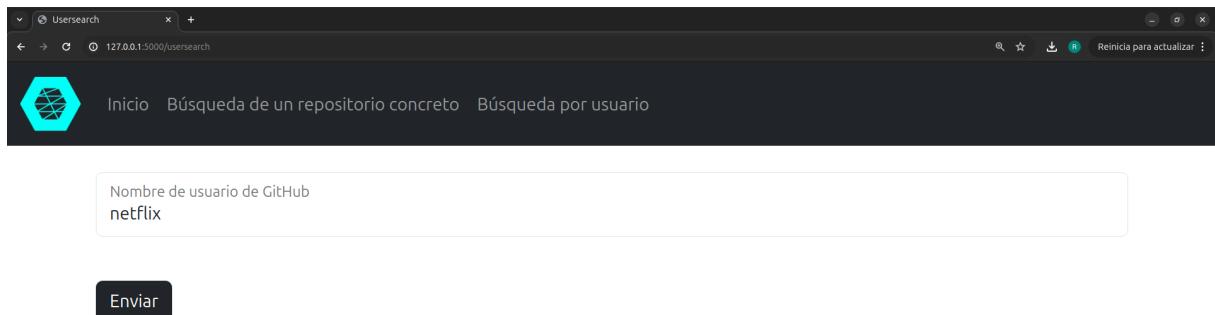


Figura 5.2: Página que se muestra al clickar en "Búsqueda por usuario"

Después de pulsar el botón de envío, se envía una petición POST al endpoint /usersearch

5.1. EJEMPLO DE USO DE LA APLICACIÓN CON EL ANÁLISIS DE UN REPOSITORIO 37

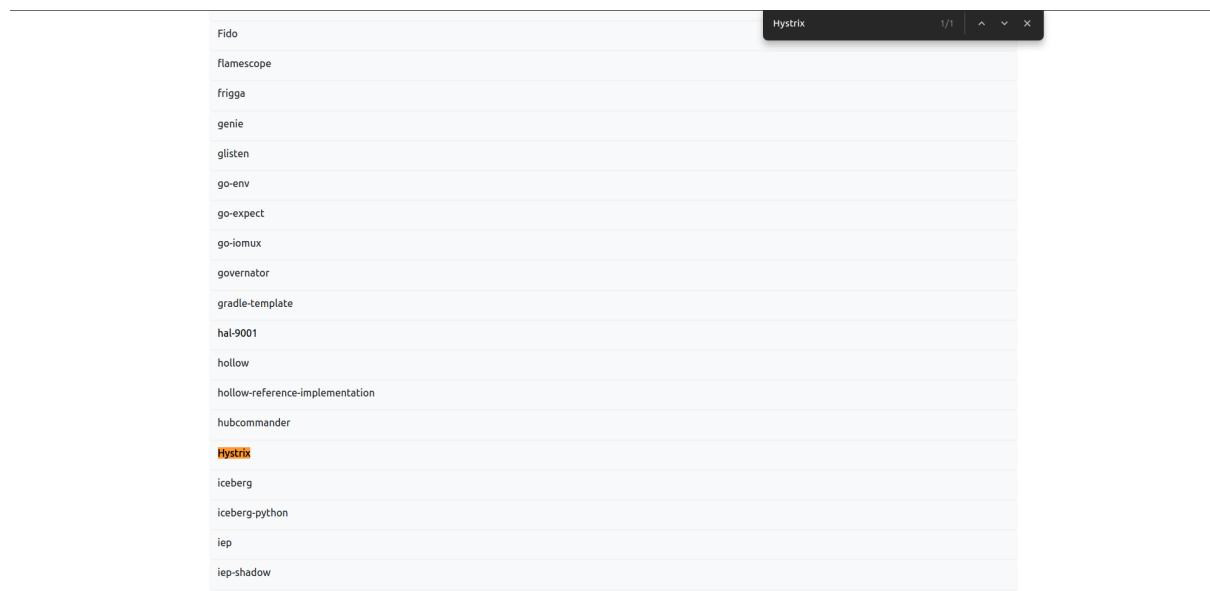
cuyo cuerpo lleva el parámetro username, con valor netflix. El servidor procesa la petición y realiza la lógica para buscar los repositorios del usuario netflix y mostrarlos en la página. Se muestra de la siguiente manera:



The screenshot shows a dark-themed web interface. At the top, there is a navigation bar with a logo on the left and three links: "Inicio", "Búsqueda de un repositorio concreto", and "Búsqueda por usuario". Below the navigation bar, the main content area has a title "Repositorios de netflix". The content is a list of repository names, each in its own row. The list includes: .github, aegisthus, af_tsa, aminator, archaius, asgard, astyanax, atlas, and atlas-docs.

Figura 5.3: Respuesta de la aplicación al buscar los repositorios del usuario netflix

Para realizar el análisis, usaremos su repositorio más popular, cuyo nombre es Hystrix. Lo encontramos entre los repositorios de la lista que hemos obtenido.



The screenshot shows a list of repository names in a table-like format. The header row contains "Fido", "flamescope", "frigga", "genie", "glisten", "go-env", "go-expect", "go-lomux", "governator", "gradle-template", "hal-9001", "hollow", "hollow-reference-implementation", "hubcommander", "Hystrix", "iceberg", "iceberg-python", "iep", and "iep-shadow". The cell for "Hystrix" is highlighted with a yellow background. The top right corner of the table has a status bar with the text "Hystrix" and "1/1".

Figura 5.4: Repositorio Hystrix en la lista

Al hacer click, se realizará una petición POST al endpoint /userrepo, con lo que el servidor comienza el análisis del repositorio. El análisis comienza con la obtención de datos de los commits del repositorio usando la librería Perceval. En el fichero de logs, se registra que se han cogido 2354 commits realizados:

```
2024-11-22 19:22:25,098 - Fetching commits: 'https://github.com/netflix/Hystrix.git' git repository from 1970-01-01 00:00:00+00:00 to 2100-01-01 00:00:00+00:00; all branches
2024-11-22 19:22:27,445 - Fetch process completed: 2354 commits fetched
```

Figura 5.5: Log que se genera cuando Perceval obtiene la información de un repositorio

Sobre estos datos, después de hacer el parseo correspondiente y las modificaciones necesarias, se obtiene un diccionario con los datos de cada uno de los usuarios que ha contribuido, los lenguajes sobre los que ha hecho esos commits, y la cantidad de los mismos.

Después, se almacenan los datos en la tabla Repository para almacenar ese repositorio, ya que suponiendo que no se ha analizado anteriormente el repositorio Hystrix del usuario netflix, se debe guardar en la base de datos. Se almacena de la siguiente manera:

Id	owner_name	repository_name	usernames ▲	repo_data	lg_percent	num_files	min_languages
1	netflix	Hystrix	[{"Justin Ryan", "Ben Christensen", ...}	{"data": {"Justin Ryan": ...	{"YAML": 0.65, "Markdown": 3.44, "Vi...	472	["YAML", "Markdown", "Vim Help File", ...]

Figura 5.6: Repositorio Hystrix en la tabla Repository

default_branch	last_commit	total_commits_on_min_languages	min_languages_experts
Filtro	Filtro	Filtro	Filtro

Figura 5.7: Repositorio Hystrix en la tabla Repository - 2

Además, para nuestro análisis personal final, guardaremos los datos de cada uno de los usuarios que ha participado en este repositorio, junto a las modificaciones sobre cada uno de los lenguajes y la lista de repositorios que hemos analizado en los que ha contribuido (Al ser este el primero que analizamos, solo tendrán en la lista este repositorio). Se almacenan así:

5.1. EJEMPLO DE USO DE LA APLICACIÓN CON EL ANÁLISIS DE UN REPOSITORIO39

<u>id</u>	<u>user</u>	<u>repositories_contribution</u>	<u>commits_min_languages_per_repo</u>
16	16 opuneet	["Hystrix"]	{"Hystrix": {"JavaScript": 1}}
17	17 Neeraj Joshi	["Hystrix"]	{"Hystrix": {"Java Properties": 2, ...}}
18	18 Benjamin Fagin	["Hystrix"]	{"Hystrix": {"Java": 1}}
19	19 mrhooray	["Hystrix"]	{"Hystrix": {"JavaScript": 2, "HTML": ...}}
20	20 Greg Orzell	["Hystrix"]	{"Hystrix": {"XML": 1}}
21	21 cgray	["Hystrix"]	{"Hystrix": {"CSS": 2, "HTML": 2, ...}}
22	22 Matt Jacobs	["Hystrix"]	{"Hystrix": {"Java": 521, "Gradle": ...}}
23	23 Joseph Wilk	["Hystrix"]	{"Hystrix": {"Clojure": 2}}
24	24 Raman Gupta	["Hystrix"]	{"Hystrix": {"Java": 1}}
25	25 Alexander Schwartz	["Hystrix"]	{"Hystrix": {"Gradle": 1, "CSS": 2, ...}}
26	26 dmgcodevil	["Hystrix"]	{"Hystrix": {"Markdown": 7, "Gradle": ...}}
27	27 Kolton Andrus	["Hystrix"]	{"Hystrix": {"Java": 10, "Gradle": ...}}
28	28 Zhang Yuhan	["Hystrix"]	{"Hystrix": {"Markdown": 1, "Java": ...}}
29	29 Daniel Sawano	["Hystrix"]	{"Hystrix": {"Java": 3}}
30	30 Sergey Krutsko	["Hystrix"]	{"Hystrix": {"Java": 2}}
31	31 Daniel Muino	["Hystrix"]	{"Hystrix": {"Gradle": 1, "Java": 1}}
32	32 Andrew Brampton	["Hystrix"]	{"Hystrix": {"Markdown": 1}}
33	33 BGehrels	["Hystrix"]	{"Hystrix": {"Java": 1}}
34	34 Sean Scanlon	["Hystrix"]	{"Hystrix": {"Markdown": 1}}
35	35 Tomasz Bak	["Hystrix"]	{"Hystrix": {"Gradle": 2, "Java": 4}}
36	36 Simon Irving	["Hystrix"]	{"Hystrix": {"Gradle": 1, "Java": 2}}
37	37 Tomasz Nurkiewicz	["Hystrix"]	{"Hystrix": {"Java": 6}}
38	38 Kevin van der Vlist	["Hystrix"]	{"Hystrix": {"Java": 9}}
39	39 Blake Carpenter	["Hystrix"]	{"Hystrix": {"Gradle": 3}}
40	40 pequinio3	["Hystrix"]	{"Hystrix": {"Gradle": 1}}
41	41 rcoomans	["Hystrix"]	{"Hystrix": {"Java": 1, "HTML": 2}}
42	42 dominictootell	["Hystrix"]	{"Hystrix": {"Java": 2}}
43	43 Andreas Kluth	["Hystrix"]	{"Hystrix": {"Markdown": 1}}

Figura 5.8: Repositorio Hystrix en la tabla User_expertise

Por último, se almacena en la tabla Languages cada uno de los lenguajes que forman el repositorio, la lista de usuarios de que ha realizado commits sobre ese lenguaje, el número total de usuarios que lo ha usado, y el número total de commits sobre el mismo. Queda así:

	id	lan_name	lan_num_users	lan_users	lan_num_commits
	F...	Filtro	Filtro	Filtro	Filtro
1	1	Markdo...	36	["Steve Hill ", "Carsten Otto ", "LuboVarga ", "Andreas Kluth ", "David Liu ", "pandeyabhi1987 ", "Jay Boyd ", "Josh Moore ",...]	70
2	2	Ignore...	3	["Justin Ryan ", "Ben Christensen ", "Geoff Denning "]	3
3	3	INI	1	["Justin Ryan "]	7
4	4	Gradle	35	["Diego Pacheco ", "Jay Anderson ", "David J. M. Karl森 ", "Jeff Beck ", "Neeraj Joshi ", "David Liu ", "Christian Schmitt "...]	191
5	5	XML	6	["Diego Pacheco ", "Greg Orzell ", "David Liu ", "Matt Jacobs ", "Justin Ryan ", "Ben Christensen "]	10
6	6	Java ...	17	["Roberto Perez Alcolea ", "builds ", "Howard Yuan ", "Neeraj Joshi ", "Justin Jose ", "Bob T Builder ", "Wladimir Schmidt ",...]	32
7	7	Batchf...	4	["Justin Ryan ", "Ben Christensen ", "Wladimir Schmidt ", "Matt Jacobs "]	4
8	8	Java	98	["Matthew Kavanagh ", "Clarke, Peter ", "Tim van Heugten ", "Daniel Muino ", "Jay Boyd ", "Ben Smith ", "tine2k ", "Raman ..."]	1442
9	9	CSS	9	["Fabian Hoffmann ", "Alexander Schwartz ", "cgray ", "Kennedy Oliveira ", "David Liu ", "Saravanakumar A. Srinivasan ", "Mat..."]	44
10	10	Java ...	3	["Ben Christensen ", "krrrr30 ", "Matt Jacobs "]	5
11	11	JavaSc...	15	["Alexander Schwartz ", "J\u00e1u00e9o e9\u00f3fame Mirc ", "Geoff Denning ", "Blake Caldwell ", "Jeremy Bull ", "Lucas Holmquist ",...]	41
12	12	HTML	20	["Diego Pacheco ", "David Liu ", "Antoine Toulme ", "dmgcodevil ", "exys666 ", "Csaba Palfi ", "Eunmin Kim ", "Kennedy ..."]	432
13	13	Vim ...	4	["David Liu ", "Ben Christensen ", "dmgcodevil ", "Matt Jacobs "]	14
14	14	Clojure	9	["dennis zhuang ", "Michael Blume ", "Robert Medeiros ", "Paul Bergeron ", "Matt Jacobs ", "Joseph Wilk ", "Dave Ray ", "Ben ..."]	18
15	15	YAML	9	["Roberto Perez Alcolea ", "Howard Yuan ", "Martin Chalupa ", "elandau ", "Matt Jacobs ", "Christoph Seibert ", "Ben ..."]	15
16	16	Shell	1	["elandau "]	2

Figura 5.9: Repositorio Hystrix en la tabla Languages

Ahora que se ha hecho la recolección y almacenaje de los datos, queda la representación gráfica para que el usuario que ha querido analizar el repositorio, pueda verlo de manera sencilla.

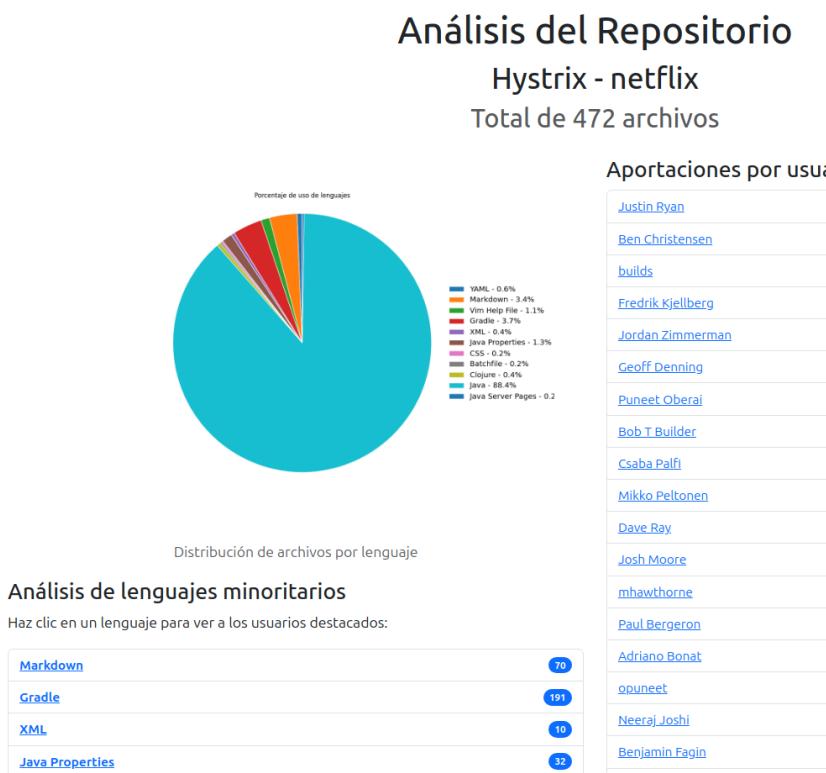


Figura 5.10: Representación gráfica del análisis del repositorio Hystrix - 1

5.1. EJEMPLO DE USO DE LA APLICACIÓN CON EL ANÁLISIS DE UN REPOSITORIO 41

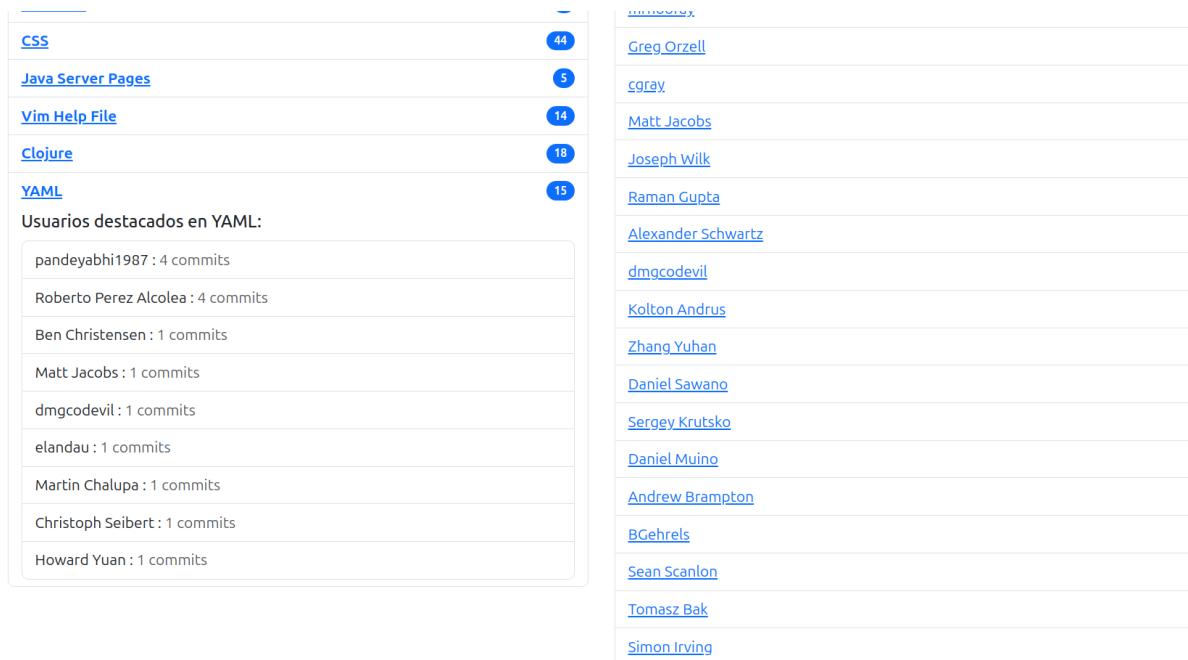


Figura 5.11: Representación gráfica del análisis del repositorio Hystrix - 2



Figura 5.12: Representación gráfica del análisis del repositorio Hystrix - 3

En las figuras 5.10, 5.11 y 5.12 se muestra el resultado del análisis. En este repositorio hay 472 ficheros en total, y hay 11 lenguajes. El lenguaje mayoritario es Java, con un 88,4 % de ocupación. Los lenguajes minoritarios son YAML, Markdown, VIM Help File (TXT), Grad-

le, XML, Java Properties, CSS, Batchfile, Clojure y Java Server Page, que representan menos del 5 %. Se muestra la información acerca de las contribuciones al repositorio en estos lenguajes minoritarios de parte de cada uno de los usuarios, y la lista de todos los usuarios con su participación en el repositorio.

5.2. Análisis de más repositorios

Para ver el análisis en más repositorios del usuario netflix, he escogido 9 repositorios. Muestro el análisis realizado de todos ellos.

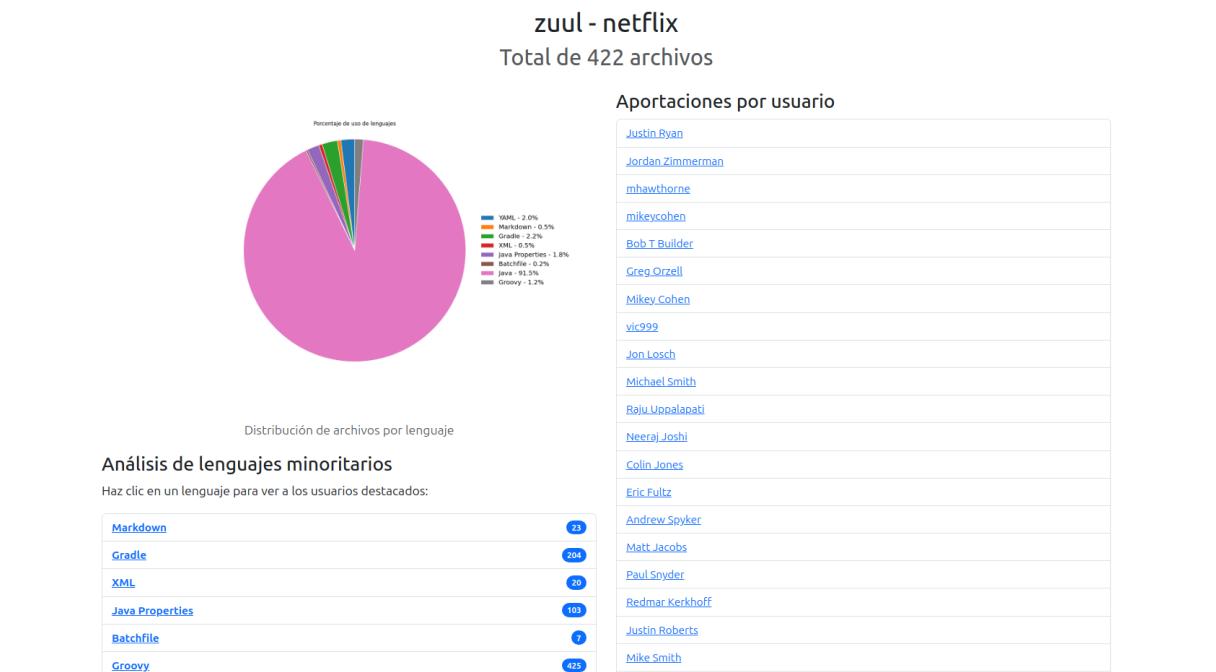


Figura 5.13: Representación gráfica del análisis del repositorio Zuul

En el repositorio Zuul, el lenguaje mayoritario es Java con un 91,5 %, mientras que YAML, Markdown, Gradle, XML, Java Properties, Batchfile y Groovy son lenguajes minoritarios. Se muestran las contribuciones de cada usuario a cada uno de los lenguajes minoritarios y la lista de todos los usuarios que han participado en el proyecto.

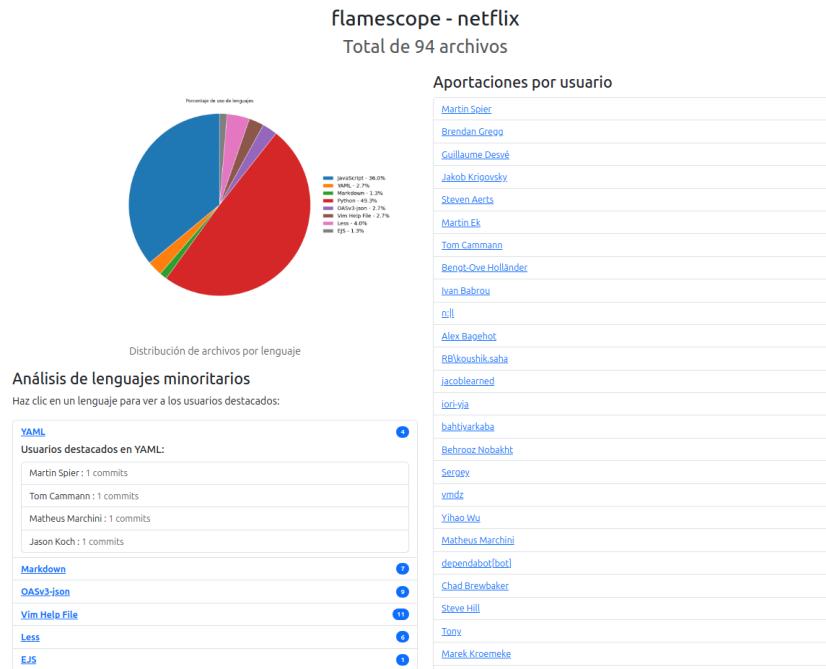


Figura 5.14: Representación gráfica del análisis del repositorio Flamescope

En el repositorio Flamescope los lenguajes mayoritarios se reparten más, y en este caso no es Java el lenguaje principal, sino Python, con un 49,3 % de ocupación. Le sigue JavaScript con un 36,0 %, que también es mayoritario. El resto, YAML, Markdown, JSON, VIM Help File (TXT), Less y EJS son los lenguajes minoritarios.

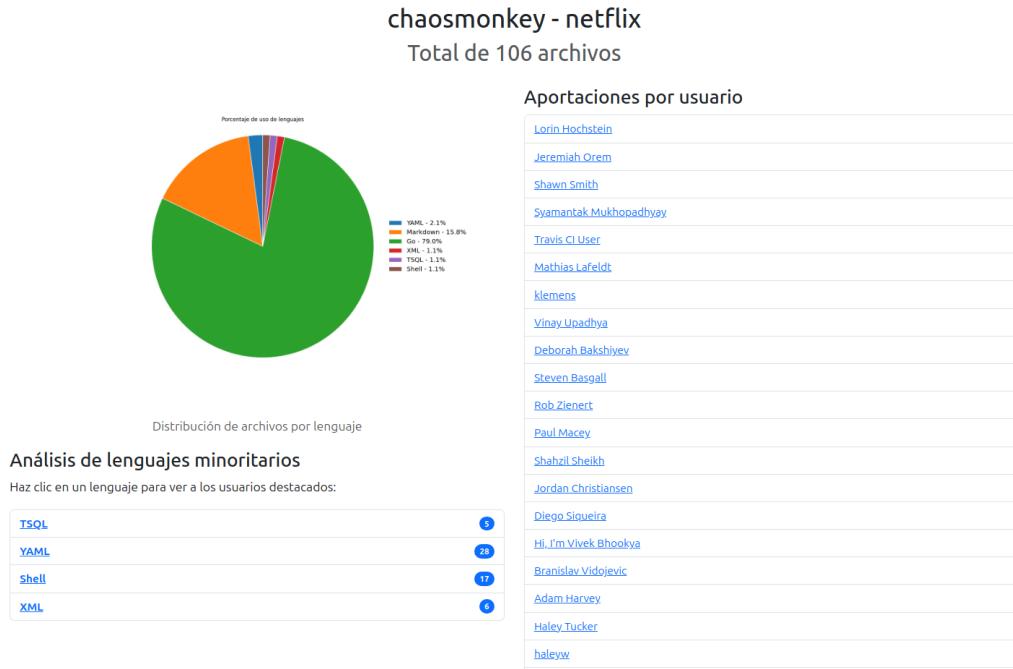


Figura 5.15: Representación gráfica del análisis del repositorio chaosmonkey

En esta ocasión, en el repositorio chaosmonkey, los dos lenguajes mayoritarios son Go con un 79 % y Markdown con un 15,8 %. YAML, XML, TSQL y Shell son minoritarios.

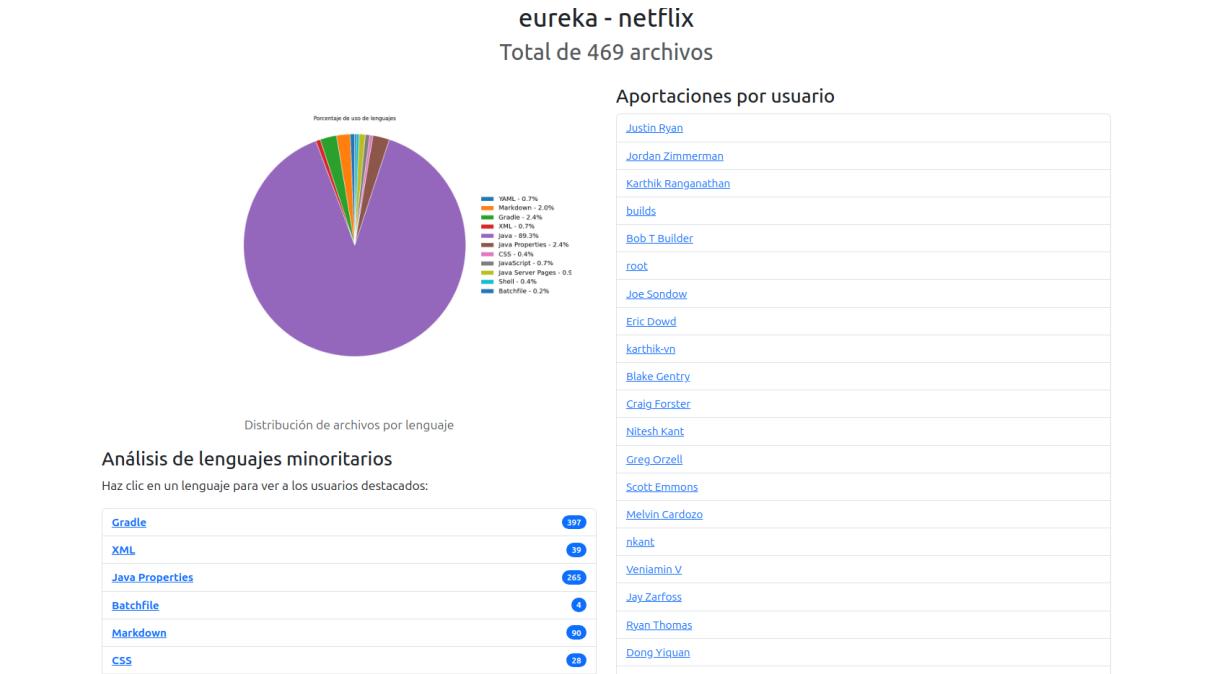


Figura 5.16: Representación gráfica del análisis del repositorio Eureka

Del repositorio eureka, Java es el lenguaje principal con un 89,3 % de uso total, mientras que

el resto de lenguajes que se han usado en el repositorio son todos minoritarios. Es un repositorio bastante grande, con 469 ficheros totales y un gran número de commits en algunos lenguajes minoritarios como Gradle o Java Properties.



Figura 5.17: Representación gráfica del análisis del repositorio Fenzo

En el repositorio Fenzo, los lenguajes minoritarios son Markdown, VIM Help File (TXT), Gradle, JSON, Java Properties, Groovy y Batchfile. En este caso, aunque haya 159 archivos, se han realizado pocos commits sobre los lenguajes minoritarios, a excepción de Gradle. Java es el único lenguaje mayoritario, con un 89,5 %.

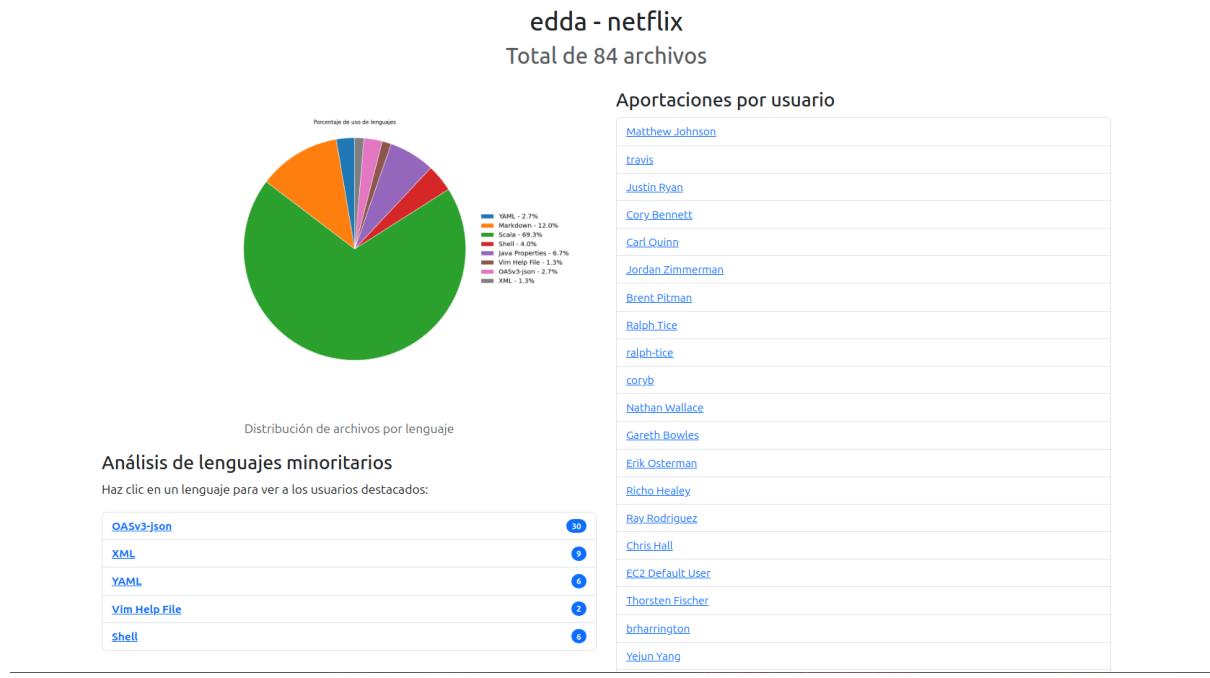
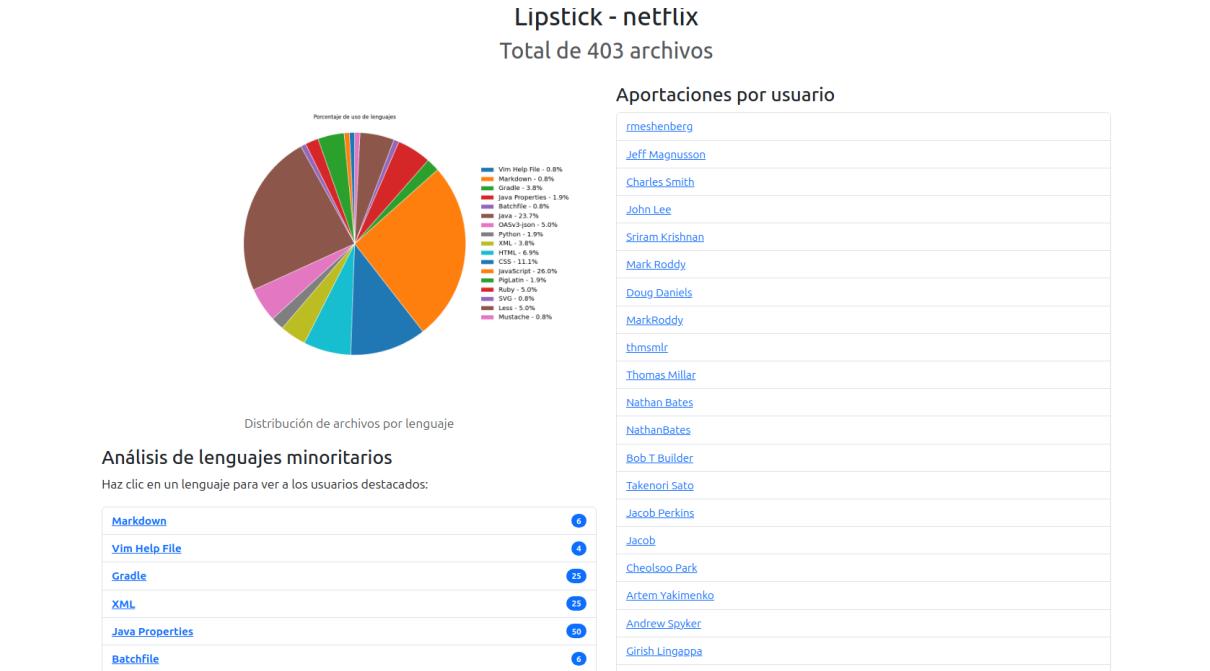


Figura 5.18: Representación gráfica del análisis del repositorio Edda

En este repositorio Scala es el lenguaje principal con un 69,3 % de uso, seguido de Markdown con un 12 %. No es un repositorio muy grande, cuenta con 84 archivos y los lenguajes minoritarios, a excepción de JSON, cuentan con pocas contribuciones.



El repositorio Lipstick tiene el uso de los lenguajes bastante repartidos y no hay un lenguaje que destaque en gran cantidad. Al estar bastante repartidos hay varios lenguajes minoritarios como Markdown o Gradle, que están siendo bastante recurrentes en los repositorios de este usuario.

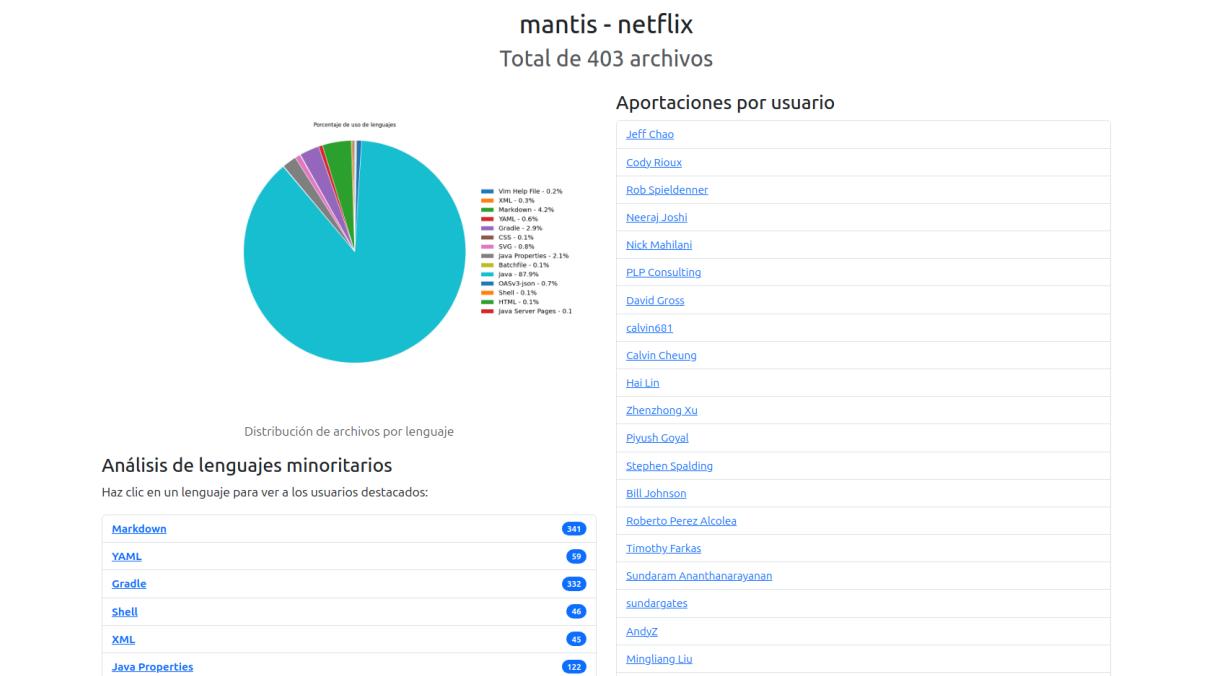


Figura 5.20: Representación gráfica del análisis del repositorio Mantis

Este repositorio, Mantis, vuelve a tener un lenguaje mayoritario principal, Java, con un 87,9 % de ocupación entre todos los ficheros. El resto, son todos lenguajes minoritarios con bastante participación.

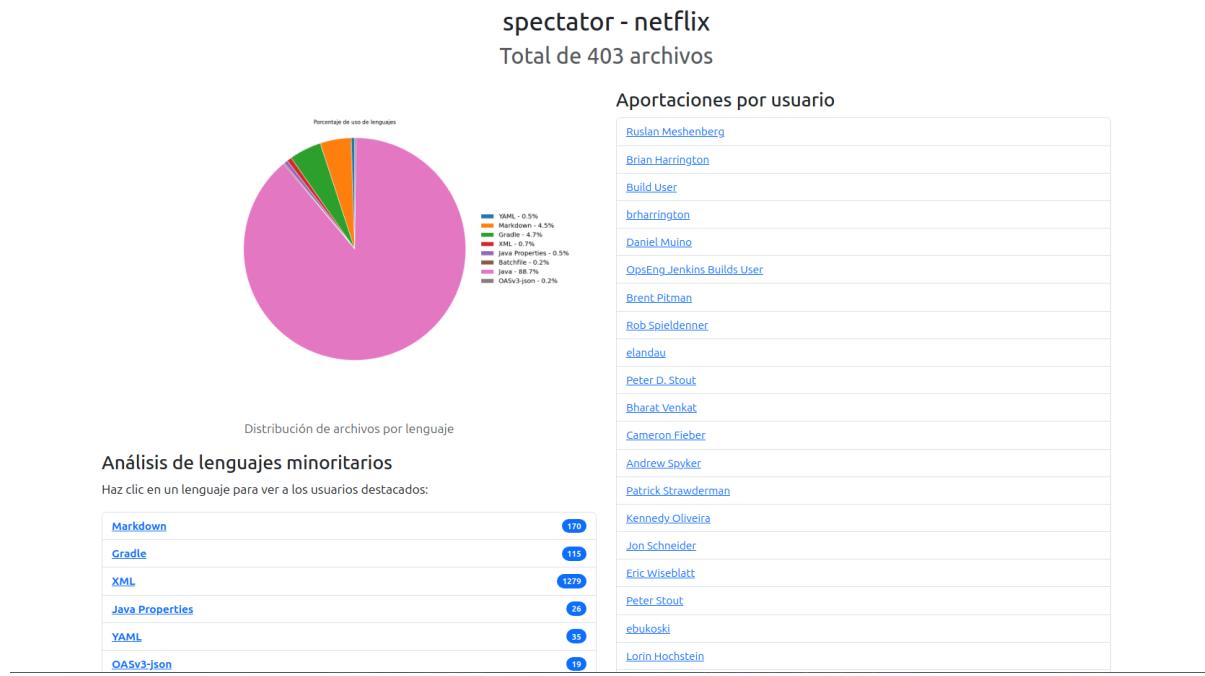


Figura 5.21: Representación gráfica del análisis del repositorio spectator

Spectator vuelve a ser un repositorio con Java como lenguaje principal, mientras que el resto son todos minoritarios. Destaca el número de contribuciones realizadas sobre ficheros XML con 1279 contribuciones entre todos los usuarios.

Capítulo 6

Resultados

Se ha realizado el análisis de 200 repositorios, con lo podemos obtener unos resultados representativos con bastantes lenguajes, usuarios y contribuyentes distintos.

En las figuras 6.1 y 6.2 se muestra el estado de la tabla Repository tras haber hecho el análisis de estos 200 repositorios. Aparecen las distintas filas y columnas de la tabla. Cada fila representa a uno de los repositorios, y las columnas contienen los distintos datos que se almacenan de cada uno de ellos.

	id	owner_name	repository_name	usernames	repo_data	lg_percent	num_files	min_languages
	F...	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	1	tiangolo	fastapi	["Sebasti\u00e1n Ram\u00edrez ", ...]	{"data": {"Sebasti\u00e1n Ram\u00edrez ": ...}}	2.52, "Python": 56.74, ...	2441	["YAML", "CSS", "SVG", "JavaScript", ...]
2	2	louisla...	uptime-kuma	["Louis Lam ", "Louis Lam ", ...]	{"data": {"Louis Lam ": {"HTML": 1, ...}}	"JavaScript": 41.1, "YAML": 2.39, ...	567	["YAML", "Markdown", "Vim Help File", ...]
3	3	sindresor...	awesome	["Sindre Sorhus ", "Boris K ", ...]	{"data": {"Sindre Sorhus ": {"SVG": ...}}	"YAML": 6.67, "Shell": 6.67, ...	23	[]
4	4	microsoft	fluentui	["Eric Thompson ", "David Zearing ", ...]	{"data": {"Eric Thompson ": ...}}	("OASv3-json": 10.31, "Markdown": ...)	19722	["YAML", "CSS", "HTML", "SVG", "MDX", ...]
5	5	rust-lang	mdBook	["Mathieu David ", "Kevin K ", ...]	{"data": {"Mathieu David ": {"Ignore": ...}}	"YAML": 2.5, "Markdown": 45.5, ...	222	["YAML", "TOML", "Shell", "SVG", "...]
6	6	fogelman	sdf	["Michael Fogelman ", "Marc ", ...]	{"data": {"Michael Fogelman ": ...}}	("Markdown": 7.69, "Go": 3.85, ...)	94	["Go"]
7	7	vercel	vercel	["Guillermo Rauch ", "Tony Kovancen ", ...]	{"data": {"Guillermo Rauch": ...}}	("Markdown": 26.0, "OASv3-json": ...)	11957	["YAML", "CSS", "HTML", "Astro", ...]
8	8	dracula	dracula-theme	["Zeno Rocha ", "Vagner Santam ", ...]	{"data": {"Zeno Rocha": {"Shell": 2, ...}}	("Markdown": 75.0, "YAML": 25.0)	7	[]
9	9	adobe	react-spectrum	["Nate Ross ", "nross ", "Aaron Hard...	{"data": {"Nate Ross": {"Markdown": ...}}	("JavaScript": 6.54, "CSS": 3.45, ...)	8129	["CSS", "HTML", "YAML", "Markdown", ...]
10	10	frameworks...	framework7	["Vladimir Kharlampidi ", "Bittdeli ...	{"data": {"Vladimir Kharlampidi": ...}}	("JavaScript": 34.45, "Markdown": ...)	1521	["Markdown", "YAML", "OASv3-json", ...]
11	11	python-...	poetry	["S\u00f9nchez Basti\u00e1n Eustace ", "Igor ...	{"data": {"S\u00f9nchez Basti\u00e1n Eustace": ...}}	("YAML": 2.35, "Markdown": 3.23, ...)	904	["YAML", "Markdown", "Shell", ...]
12	12	miniflux	miniflux	["Fr\u00f3nald\u00f3n Guillo ...	{"data": {"Fr\u00f3nald\u00f3n Guillo": ...}}	("OASv3-json": 4.77, "YAML": 3.73, ...)	563	["OASv3-json", "YAML", "Markdown", ...]
13	13	caddyserv...	caddy	["Matthew Holt ", "Matt Holt ", ...]	{"data": {"Matthew Holt": {"Ignore": ...}}	("Markdown": 0.9, "YAML": 2.71, "Go": ...)	508	["Markdown", "YAML", "HTML", "Shell", ...]
14	14	directus	directus	["olov ", "Max Glantzman ", "Ben ", ...]	{"data": {"olov": {"Ignore List": 3, ...}}	("OASv3-json": 2.83, "YAML": 7.55, ...)	3437	["OASv3-json", "JavaScript", ...]
15	15	getsetry	symbolicator	["Markus Unterwaditzer ", "Anton ...	{"data": {"Markus Unterwaditzer": ...}}	("YAML": 5.49, "OASv3-json": 2.2, ...)	323	["OASv3-json", "Shell", "CSS", ...]
16	16	novnc	novnc	["Joel Martin ", "Kevin Chan ", ...]	{"data": {"Joel Martin": {"HTML": ...}}	("Markdown": 5.11, "YAML": 3.41, "Vi...	213	["YAML", "Vim Help File", "CSS", ...]
17	17	darkreader	darkreader	["alexanderby ", "Alexander Shutov ...	{"data": {"alexanderby": {"Ignore": ...}}	("JavaScript": 5.13, "YAML": 1.41, ...)	927	["YAML", "Markdown", "Ruby", "Vim ...]
18	18	tootsuite	mastodon	["Eugen Rochko ", "Eugen ", "Yann ...	{"data": {"Eugen Rochko": {"Ignore": ...}}	("YAML": 6.91, "OASv3-json": 1.38, ...)	8637	["OASv3-json", "Vim Help File", ...]
19	19	xtermjs	xterm.js	["Christopher Jeffrey ", "Carson ...	{"data": {"Christopher Jeffrey": ...}}	("OASv3-json": 15.08, "Markdown": ...)	671	["Markdown", "YAML", "Dotenv", ...]
20	20	felangel	bloc	["Felix Angelov ", "alexey ", "Rody ...	{"data": {"Felix Angelov": {"Ignore": ...}}	("YAML": 7.15, "Markdown": 5.17, ...)	2986	["JavaScript", "OASv3-json", "SVG", ...]
21	21	projectdi...	nuclei-...	["Ritesh Gohil ", "Ice3man ", ...]	{"data": {"Ritesh Gohil": {"YAML": ...}}	("YAML": 97.14, "Python": 0.63, ...)	9887	["Python", "Shell", "Vim Help File", ...]
22	22	cozy	cozy-stack	["Bruno Michel ", "Tristan Nitot ", ...]	{"data": {"Bruno Michel": {"Ignore": ...}}	("YAML": 1.44, "OASv3-json": 0.58, ...)	1138	["YAML", "OASv3-json", "CSS", ...]
23	23	oclif	oclif	["Jeff Dickey ", "heroku-cl ", "roo...	{"data": {"Jeff Dickey": {"OASv3-...}}	("OASv3-json": 14.91, "Markdown": ...)	171	["YAML", "Batchfile", "JavaScript"]
24	24	sebastianu...	phunit	["Sebastian Bergmann ", "Tobias ...	{"data": {"Sebastian Bergmann": ...}}	("Markdown": 0.53, "YAML": 0.11, ...)	2682	["Markdown", "YAML", "OASv3-json", ...]
25	25	graphql	graphql-...	["Tim Suchanek ", "timsuchanek ", ...]	{"data": {"Tim Suchanek": {"Ignore": ...}}	("YAML": 2.32, "Markdown": 9.27, ...)	337	["YAML", "SVG", "CSS", "HTML", ...]
26	26	prisma-...	graphql-...	["Johannes Schickling ", "Nilan ...	{"data": {"Johannes Schickling": {"Ignore": ...}}	("Markdown": 16.02, "YAML": 0.59, ...)	1558	["YAML", "OASv3-json", "Vim Help ...]
27	27	gridsome	gridsome	["Hans-J\u00fclgen Vedvik ", "Tommy ...	{"data": {"Hans-J\u00fclgen Vedvik": ...}}	("JavaScript": 62.91, "Markdown": ...)	445	["YAML", "HTML", "SVG", "GraphQL", ...]

Figura 6.1: Estado de la tabla Repository después del análisis de 200 repositorios

	id	owner_name	repository_name	usernames	repo_data	lg_percent	num_files	min_languages
174	1...	junegunn	vim-plug	["Junegunn Choi ", "C.D. Clark III ", ...	{"data": {"Junegunn Choi ": ...	{"YAML": 33.33, "Markdown": 44.44, ...	18	[]
175	1...	colinhacks	zod	["Colin McDonnell ", "Lukasz ...	{"data": {"Colin McDonnell ": ("XML" ...	{"JavaScript": 2.38, "OASv3-json": ...	231	["JavaScript", "YAML", "Markdown", ...
176	1...	BVLC	caffe	["Yangqing Jia ", "Jeff Donahue ", ...	{"data": {"Yangqing Jia ": {"Ignore ...	{"Markdown": 15.99, "YAML": 0.33, ...	701	["YAML", "Vim Help File", "CMake", ...
177	1...	maybe--	maybe	["Josh Pigford ", "Rob Zolkos ", ...	{"data": {"Josh Pigford ": {"YAML": ...	{"Markdown": 0.66, "OASv3-json": ...	1119	["Markdown", "OASv3-json", "SVG", ...
178	1...	zenorocha	clipboard.js	["Zeno Rocha ", "Eduardo Lundgren ", ...	{"data": {"Zeno Rocha ": ...	{"OASv3-json": 14.89, "Markdown": ...	55	["XML"]
179	1...	goabstract	Awesome--	["Lisa Dziba ", "Valia Havruliuk ", ...	{"data": {"Lisa Dziba ": ...	{"Markdown": 8.0, "SVG": 42.67, ...	133	["CSS", "HTML", "OASv3-json"]
180	1...	helix-	helix	["Bla\u017e Hrastnik ", "Jan Hrastni...]	{"data": {"Bla\u017e Hrastnik ": ...	{"TOML": 14.99, "YAML": 0.6, ...	1206	["YAML", "Markdown", "CSS", "SVG", ...
181	1...	testerSun	12306	["MR.wen ", "wenxianping ", "wen ", ...	{"data": {"MR.wen ": {"XML": 8, ...	{"Markdown": 5.8, "Python": 86.96, ...	86	["YAML", "Shell", "Vim Help File"]
182	1...	penpot	penpot	["Andrey Antukh ", "Juan de la Cruz ...	{"data": {"Andrey Antukh ": {"Ignore ...	{"YAML": 0.35, "edn": 0.51, ...	4043	["YAML", "edn", "Markdown", "OASv3-...]
183	1...	shadowsocks	shadowsocks	["clownindy ", "Phue Lu ", "mitnk ", ...	{"data": {"clownindy ": {"Markdown": ...	{"Markdown": 100.0}	1	[]
184	1...	netty	netty	["Trustin Lee ", "Andy Taylor ", ...	{"data": {"Trustin Lee ": {"Vim Help ...	{"OASv3-json": 1.24, "Markdown": ...	3409	["OASv3-json", "Markdown", "YAML", ...
185	1...	rasbt	LLMs-from--	["rasbt ", "Sebastian Raschka ", ...	{"data": {"rasbt ": {"Ignore List": ...	{"Markdown": 23.78, "YAML": 4.32, ...	189	["YAML"]
186	1...	fxsyj	jieba	["Sun Junyi ", "fxsyj ", "Herman ...	{"data": {"Sun Junyi ": {"Ignore ...	{"Markdown": 1.33, "Vim Help File": ...	105	["Markdown"]
187	1...	Blankj	AndroidUtilCo...	["blankj ", "Mengjie Cai ", "cmj ", ...	{"data": {"blankj ": {"Markdown": 36...}	{"YAML": 0.32, "Markdown": 2.69, ...	704	["YAML", "Markdown", "Gradle", ...
188	1...	logseq	logseq	["Tienson Qin ", "Ed ", "Kamal ...	{"data": {"Tienson Qin ": ...	{"edn": 4.98, "Clojure": 34.24, ...	1461	["edn", "YAML", "JavaScript", "OASv3...]
189	1...	tw93	Pake	["W93 ", "liusihan ", "u4f91u591...]	{"data": {"W93 ": {"Ignore List": 2...}	{"OASv3-json": 13.33, "YAML": 14.67,...	146	["Python", "TOML", "desktop", "XML ...
190	1...	spacedrive...	spacedrive	["Jamie ", "Brendan Allan ", "Jamie ...	{"data": {"Jamie ": {"Ignore List": ...	{"TOML": 1.14, "Mustache": 0.86,...	2118	["Markdown", "Mustache", "TOML", "Vi...]
191	1...	Karanprat...	system-design	["Karan Pratap Singh ", "Ayushi Gupt...]	{"data": {"Karan Pratap Singh ": ...	{"YAML": 25.0, "Markdown": 75.0}	54	[]
192	1...	PKUAnonym	REKCARC-TSC-...	["unknown ", "Trinkle23897 ", "ne "...	{"data": {"unknown ": {"Ignore List": ...	{"Markdown": 4.59, "Batchfile": 0.29,...	14176	["Markdown", "Batchfile", "TeX", ...
193	1...	immutable	immutable.js	["Lee Byron ", "Bruno Heridot ", "Le...]	{"data": {"Lee Byron ": ...	{"OASv3-json": 5.07, "Markdown": 2.3...}	228	["Markdown", "YAML", "Jest Snapshot"]
194	1...	ultralytic...	ultralytics	["Glenn Jocher ", "Ayush Chaurasia ...	{"data": {"Glenn Jocher ": {"Ignore ...	{"YAML": 14.44, "Markdown": 54.86, ...	675	["Jupyter Notebook", "Vim Help File"]
195	1...	zxing	zxing	["(no author) ", "srowen ", ...	{"data": {"(no author) ": {}, "srowe...}	{"Markdown": 0.21, "YAML": 0.29, ...}	3768	["Markdown", "YAML", "XML", "CSS", ...]
196	1...	nushell	nushell	["Yehuda Katz ", "Jonathan Turner ", ...	{"data": {"Yehuda Katz ": {"Ignore ...	{"TOML": 2.73, "Markdown": 3.19, ...}	1876	["TOML", "Markdown", "YAML", "OASv3...]
197	1...	bilibili	ijkplayer	["Zhang Rui ", "bcaallen ", "Yrom ...	{"data": {"Zhang Rui ": {"Ignore ...	{"YAML": 0.17, "Roff Manpage": 0.17,...	734	["YAML", "Roff Manpage", "Vim Help ...
198	1...	Znoise	ChatTTS	["lich99 ", "CH.Li ", "anyvoice ", ...	{"data": {"lich99 ": {"Ignore List": ...	{"YAML": 6.67, "Python": 73.33, ...}	93	["OASv3-json", "Vim Help File", ...]
199	1...	solidjds	solidjds	["Ryan Carniato ", "Joe Pea ", ...]	{"data": {"Ryan Carniato ": {"Ignore ...	{"Markdown": 13.95, "OASv3-json": ...}	181	["YAML", "CSS"]
200	2...	aegisub	aegisub	["Niels Martin Hansen ", "Rodrigo ...	{"data": {"Niels Martin Hansen ": ...	{"XML": 5.14, "YAML": 0.07, ...}	2193	["YAML", "Microsoft Visual Studio ...

Figura 6.2: Estado de la tabla Repository después del análisis de 200 repositorios - 2

En total, se han registrado en la base de datos 92.816 usuarios que hayan contribuido a alguno o varios de entre los 200 repositorios que se han analizado. Se muestra en las figuras 6.3 y 6.4, con el estado de la tabla User_expertise. Cada fila presenta un usuario y las columnas los datos que se almacenan de cada uno de ellos.

	id	user	repositories_contribution	commits_min_languages_per_repo
1	1	Sebastián Ramírez	["fastapi", "keras", ...	{"fastapi": {"Python": 1772, "TOML": ...
2	2	Maria Camila Gutierrez	["fastapi"]	{"fastapi": {"Python": 2)}
3	3	euri10	["fastapi"]	{"fastapi": {"Python": 13, "TOML": 1...}
4	4	Ken Kinder	["fastapi"]	{"fastapi": {"Markdown": 1)}
5	5	Kabir Khan	["fastapi"]	{"fastapi": {"Python": 5, "Markdown": ...}
6	6	Zaar Hai	["fastapi"]	{"fastapi": {"Markdown": 1)}
7	7	Matt Hegarty	["fastapi"]	{"fastapi": {}})
8	8	Stratos Gerakakis	["fastapi"]	{"fastapi": {"Python": 1})}
9	9	Sanders	["fastapi"]	{"fastapi": {"Markdown": 1})}
10	10	The Gitter Badger	["fastapi", ...	{"fastapi": {"Markdown": 3), ...
11	11	yihuang	["fastapi", "redis"]	{"fastapi": {"Python": 1}, "redis": ...
12	12	Daniel Hahler	["fastapi", "poetry"]	{"fastapi": {"YAML": 1}, "poetry": ...
13	13	Alif Jahan	["fastapi"]	{"fastapi": {"TOML": 1}}
14	14	Alex Iribarren	["fastapi", ...	{"fastapi": {"Python": 1, "Markdown": ...}
15	15	Mohammed	["fastapi"]	{"fastapi": {"Python": 6})}
16	16	Mostapha Sadeghipour Roudsari	["fastapi"]	{"fastapi": {"Python": 4, "Markdown": ...}
17	17	Matthew McLeod	["fastapi"]	{"fastapi": {"Markdown": 1})}
18	18	William Hayes	["fastapi", "poetry"]	{"fastapi": {"Python": 7, "Markdown": ...}
19	19	hayata-yamamoto	["fastapi"]	{"fastapi": {"Markdown": 1})}
20	20	Daniel Michaels	["fastapi"]	{"fastapi": {"Markdown": 1})}
21	21	Christopher Dignam	["fastapi", ...	{"fastapi": {"Markdown": 4), ...
22	22	Steve B	["fastapi"]	{"fastapi": {}})
23	23	Derek J. Lambert	["fastapi"]	{"fastapi": {"Python": 1})}
24	24	Ricardo Momm	["fastapi"]	{"fastapi": {"Python": 1})}
25	25	zamiramir	["fastapi"]	{"fastapi": {"Shell": 1})}
26	26	Trim21	["fastapi", "poetry"]	{"fastapi": {"Python": 2}, "poetry": ...}
27	27	Steinþor Palsson	["fastapi"]	{"fastapi": {"Python": 5})}
28	28	James Kaplan	["fastapi"]	{"fastapi": {"Python": 16, ...}

Figura 6.3: Estado de la tabla User_expertise después del análisis de 200 repositorios

id	user	repository_contribution	commits_min_languages_per_repo
92789	92_darealshinji	["aegisub"]	{"aegisub": {"Objective-C": 71, ...}}
92790	92_Derek Buitenhuis	["aegisub"]	{"aegisub": {"C++": 1}}}
92791	92_djcj	["aegisub"]	{"aegisub": {"Gettext Catalog": 20}}}
92792	92_Oleksey Prytchyn	["aegisub"]	{"aegisub": {"Inno Setup": 3, ...}}
92793	92_torque	["aegisub"]	{"aegisub": {"C++": 1}}}
92794	92_kotobenko	["aegisub"]	{"aegisub": {"Gettext Catalog": 1}}}
92795	92_Khaled Hosny	["aegisub"]	{"aegisub": {"Objective-C": 1}}}
92796	92_line0	["aegisub"]	{"aegisub": {"Inno Setup": 9, ...}}
92797	92_9adefaf01e5bf6426d838cd20eae582d...	["aegisub"]	{"aegisub": {"C++": 2}}}
92798	92_Ryan Lucia	["aegisub"]	{"aegisub": {"Markdown": 2, "C++": ...}}
92799	92_Cirrus Wazza	["aegisub"]	{"aegisub": {"Gettext Catalog": 1}}}
92800	92_Eduard Ereza Martinez	["aegisub"]	{"aegisub": {"Gettext Catalog": 2}}}
92801	92_Yao	["aegisub"]	{"aegisub": {"Vim Help File": 1}}}
92802	92_Alex G.M	["aegisub"]	{"aegisub": {"Gettext Catalog": 1}}}
92803	92_Rodger Combs	["aegisub"]	{"aegisub": {"M4Sugar": 2, ...}}
92804	92_Rasmus Thomsen	["aegisub"]	{"aegisub": {}}}
92805	92_therealfun	["aegisub"]	{"aegisub": {}}
92806	92_Alexander Pozdnyakov	["aegisub"]	{"aegisub": {"M4Sugar": 6}}}
92807	92_Roxas Shadow	["aegisub"]	{"aegisub": {"Gettext Catalog": 1}}}
92808	92_sidneys	["aegisub"]	{"aegisub": {"C++": 2, "M4Sugar": 6}}}
92809	92_Marcin Kurczewski	["aegisub"]	{"aegisub": {"C++": 1}}}
92810	92_bkbkb	["aegisub"]	{"aegisub": {"XML": 1}}
92811	92_wangqr	["aegisub"]	{"aegisub": {"YAML": 1, "MoonScript": ...}}
92812	92_scx	["aegisub"]	{"aegisub": {"Shell": 1, "C++": 6, ...}}
92813	92_Yakauleu Uładzislau	["aegisub"]	{"aegisub": {"Gettext Catalog": 1}}}
92814	92_davste0816	["aegisub"]	{"aegisub": {"Inno Setup": 1, "C++": ...}}
92815	92_FichteFoll	["aegisub"]	{"aegisub": {"C++": 4, "Shell": 1}}}
92816	92_Myaamori	["aegisub"]	{"aegisub": {"Python": 1, "Vim Help ...}}

Figura 6.4: Estado de la tabla User_expertise después del análisis de 200 repositorios - 2

En la tabla Languages se han almacenado 262 lenguajes distintos. En las figuras 6.5 y 6.6 se muestra el estado de la tabla Languages tras analizar los 200 repositorios. Cada fila representa un lenguaje y las columnas, los datos que se almacenan de cada lenguaje.

Tabla: languages			
id	lan_name	lan_num_users	lan_users
1	1 Python	22281	["Andrew Newdigate ", "jvs ", ...]
2	2 TOML	2271	["HRUSHIKESH DOKALA ", "Jonah Snider...]
3	3 Shell	5377	["Henry Darnell ", "Matteo ", "Jared...]
4	4 Ignore List	3095	["Guy Korland ", "TENIOS ", "Dominik...]
5	5INI	997	["Andrew Newdigate ", "Maciej Bienie...]
6	6 YAML	10325	["", "Andrew Newdigate ", ...]
7	7 SVG	1179	["Josh Calder ", "Paul Emm. ...]
8	8 Markdown	46187	["Henry Darnell ", "Andrew Newdigate...]
9	9 Vim Help File	5669	["", "Andrew Newdigate ", ...]
10	10 CSS	2334	["Erik Verbeek ", "David Mosher ", ...]
11	11 HTML	3213	["Ben Cohen ", "Fatih Kadir Ak\u0131...]
12	12 JavaScript	11902	["Alex Galays ", "Elijah Manor ", ...]
13	13 Dotenv	230	["Daniel Vaz Gaspar ", "Vector-Hope ...]
14	14 OASv3-json	13000	["Alex Galays ", "Elijah Manor ", ...]
15	15 EditorConfig	460	["\u05e03\u05170\u067ef\u057fa ", "Dmitr...]
16	16 Vue	1107	["Wasim Thoufiq ", "Brandon Scott ",...]
17	17 SCSS	1513	["Olivia Mossberg ", "Matteo ", "Jan...]
18	18 TSQL	298	["AnnAngela ", "Joel Lee ", "Jan ...]
19	19 XML	14888	["Alex Galays ", "Stephen Kempin ", ...]
20	20 PowerShell	444	["Artem Zinenko ", "Joe Tauke ", ...]
21	21 Dockerfile	734	["Maiko Bossuyt ", "jamespcole ", ...]
22	22 Go	3750	["Ben Cohen ", "Asbj\u00f8rn Hanse...]
23	23 Smalltalk	241	["HRUSHIKESH DOKALA ", "Romfos ", ...]
24	24 Microsoft _	106	["HRUSHIKESH DOKALA ", "srown ", ...]
25	25 Java	1538	["wildma ", "seatrain ", ...]
26	26 PHP	1376	["Alexandre Borela ", "Miroslav ...]
27	27 HTTP	3	["Louis Lam ", "Kosta Petan ", ...]
28	28 JSON5	42	["Nick Schonning ", "Renaud Chaput ..."]

Figura 6.5: Estado de la tabla Languages después del análisis de 200 repositorios

Tabla: languages

id	lan_name	lan_num_users	lan_users	lan_num_commits
235	Nearley	1	["Bogdan "]	7
236	Pickle	1	["noptuno "]	5
237	Fortran	1	["noptuno "]	14
238	Csound	1	["noptuno "]	4
239	HTML+PHP	3	["Swk ", "swisskyrepo ", "Swissky "]	11
240	ASP.NET	4	["boldanssec ", "Swk ", "chivato ", ...]	9
241	Classic ASP	3	["boldanssec ", "Swk ", "Swissky "]	3
242	SystemVerilog	2	["Alexander Medvednikov ", "Delyan ..."]	2
243	Wavefront ..	3	["LeRoyce Pearson ", "Andrew Kelley ..."]	32
244	Wavefront ..	2	["Delyan Angelov ", "penguindark "]	3
245	LLVM	2	["LeRoyce Pearson ", "Andrew Kelley ..."]	625
246	OpenEdge ABL	5	["ZoeyYoung ", "Dingyuan Wang ", ...]	29
247	Bicep	5	["Reuben Bond ", "Kosta Petan ", ...]	132
248	RPC	3	["David Lazar ", "John MacFarlane ",...]	3
249	Muse	6	["John MacFarlane ", "Albert ..."]	11
250	Dhall	1	["John MacFarlane "]	1
251	Pod 6	1	["Jess Robinson "]	2
252	Nushell	107	["Reilly Wood ", "Yethal ", "Dongjia..."]	494
253	Elvish	2	["Tobias Hunger ", "Alexander Breivig..."]	2
254	World of ..	2	["Sun Junyi ", "miao.lin "]	2
255	PostCSS	1	["maxichrome "]	1
256	Gnuplot	1	["Lee Byron "]	1
257	Pan	2	["srowen@gmail.com ", "ftylitak "]	2
258	Scilab	2	["srowen@gmail.com ", ...]	2
259	GDB	2	["srowen@gmail.com ", ...]	2
260	E-mail	4	["JT ", "Adam Shirey ", "Chris ..."]	5
261	Visual Basic..	2	["Rodrigo Braz Monteiro ", "Amar ..."]	2
262	MoonScript	3	["Thomas Goyne ", "tophf ", "wangqr ..."]	22

Figura 6.6: Estado de la tabla Languages después del análisis de 200 repositorios - 2

Con estos datos, ahora podemos comprobar como es el uso de los lenguajes en los repositorios. Para ello nos haremos las siguientes preguntas:

■ **¿Cuáles son los 10 lenguajes sobre los que más usuarios han realizado commits en los distintos repositorios analizados?**

- Viendo la tabla Languages, y ordenando según la columna lan_num_users:
 - Markdown: 46187 usuarios
 - Python: 22281 usuarios
 - XML: 14888 usuarios
 - JSON: 13000 usuarios
 - JavaScript: 11092 usuarios
 - YAML: 10325 usuarios
 - Vim Help File (TXT): 5669 usuarios
 - C++: 5587 usuarios
 - Shell: 5377 usuarios
 - Go: 3750 usuarios.

■ **¿Cuáles son los 10 lenguajes sobre los que se han realizado más commits en total?**

- Viendo la tabla Languages, y ordenando según la columna lan_num_commits:
 - XML: 795957
 - JavaScript: 703486
 - Python: 412025
 - YAML: 388705
 - JSON: 339749
 - Markdown: 330496
 - Go: 163826
 - HTML: 141931
 - C++: 126958
 - Objective-C: 97321.

■ **¿Cuáles son los 10 lenguajes que en más repositorios han sido minoritarios?**

- Haciendo la siguiente query contra la base de datos:

```

WITH RECURSIVE split_languages AS (
  SELECT
    id,
    TRIM(REPLACE(REPLACE(SUBSTR(min_languages, 1,
      INSTR(min_languages || ',', ',') - 1), '[', '')',
      , ']')) AS language,
    SUBSTR(min_languages, INSTR(min_languages
      || ',', ',') + 1) AS remaining_languages
  FROM repository
  WHERE min_languages IS NOT NULL
  UNION ALL

  SELECT
    id,
  
```

```

    TRIM(REPLACE(REPLACE(SUBSTR(remaining_languages
    , 1, INSTR(remaining_languages || ',', ',')
    - 1), '[', ''), ']', '')),
    SUBSTR(remaining_languages,
    INSTR(remaining_languages || ',', ',') + 1)
FROM split_languages
WHERE remaining_languages != ''
)
SELECT
    language,
    COUNT(*) AS frequency
FROM split_languages
WHERE language != ''
GROUP BY language
ORDER BY frequency DESC;

```

Obtenemos que los 10 lenguajes que más se han repetido como minoritarios han sido:

- YAML: 130 veces
- Vim Help File (TXT): 115 veces
- Shell: 96 veces
- CSS: 92 veces
- SVG: 88 veces
- JSON: 85 veces
- HTML: 85 veces
- Markdown: 70 veces
- Javascript: 63 veces
- TOML: 58 veces.

- **¿Si nos fijamos en los lenguajes minoritarios que más se repiten en los repositorios, cuál es la tendencia de uso por parte de los contribuyentes?**

- Ayudándonos de la tabla Languages, vamos a ver si estos lenguajes se han usado por muchos usuarios, y la cantidad de commits que estos han realizado.
 - YAML: Entre 10325 usuarios se han realizado 388705 commits.
 - VIM: Entre 5669 usuarios se han realizado 37867 commits.
 - Shell: Entre 5377 usuarios se han realizado 24034 commits.
 - CSS: Entre 2334 usuarios se han realizado 26619 commits.
 - SVG: Entre 1179 usuarios se han realizado 83344 commits.
 - JSON: Entre 13000 usuarios se han realizado 339749 commits.
 - HTML: Entre 3213 usuarios se han realizado 141931 commits.
 - Markdown: Entre 46187 usuarios se han realizado 330496 commits.
 - JavaScript: Entre 11902 usuarios se han realizado 703486 commits.
 - TOML: Entre 2271 usuarios se han realizado 10026 commits.
- **¿Cómo es el uso de los lenguajes minoritarios dentro de los repositorios? ¿Se centralizan en usuarios concretos? ¿Qué tipo de lenguajes suelen ser?**

- En la mayoría de los repositorios que se han analizado, la tendencia que se sigue es que, si bien no se suelen centralizar del todo en uno o un grupo pequeño de usuarios, si que suele ocurrir que uno o varios de ellos destacan más en el uso de alguno de los lenguajes. Por ejemplo, con el repositorio prettier/prettier que se ha analizado, más de 20 personas han realizado commits sobre ficheros de lenguaje YAML, pero la mayoría de ellos han realizado menos de 3 commits, mientras que la persona con más commits ha realizado 1081 commits, y el segundo, 351 commits. También tiene que ver con el número de commits que se realiza sobre cada uno de los lenguajes, ya que la tendencia que he comentado se suele dar sobre todo para aquellos lenguajes sobre los que se han realizado muchos commits, mientras que para los lenguajes con menos contribuciones, están más repartidos y no hay ningún usuario que destaque demasiado por encima del resto.

Los lenguajes que suelen ser minoritarios en la mayoría de repositorios se utilizan para realizar configuraciones o despliegues de aplicaciones y que suelen ser flexibles de manera que la mayoría de lenguajes de programación pueden usar. Por ello son

tan importantes a pesar de que no sean los lenguajes con un mayor uso dentro de los repositorios, ya que suelen ser clave para el funcionamiento de una aplicación, un programa, etc.

- **¿Cómo es el uso general de los lenguajes minoritarios en los repositorios públicos?**

- Para ver el uso general de los lenguajes minoritarios vamos a calcular el coeficiente de Gini de los lenguajes minoritarios más usados en los repositorios que hemos analizado. El coeficiente de Gini [9] es una medida de desigualdad que normalmente se suele usar para calcular la desigualdad de ingresos o de población. Sin embargo, se puede usar para medir cualquier distribución desigual. El coeficiente de Gini puede tener un valor entre 0 y 1, siendo 0 la perfecta igualdad, mientras que el 1 es la desigualdad total. Se calcula con la siguiente fórmula matemática:

$$G = \left| 1 - \sum_{k=1}^{n-1} (X_{k+1} - X_k)(Y_{k+1} + Y_k) \right|$$

Figura 6.7: Fórmula de Brown usada para el cálculo del coeficiente de Gini

donde:

- G es el Coeficiente de Gini
- X suele ser la proporción de población
- Y suele ser la proporción de ingresos.

Y se suele representar de la siguiente manera, con la línea de máxima igualdad, y la curva de Lorenz:

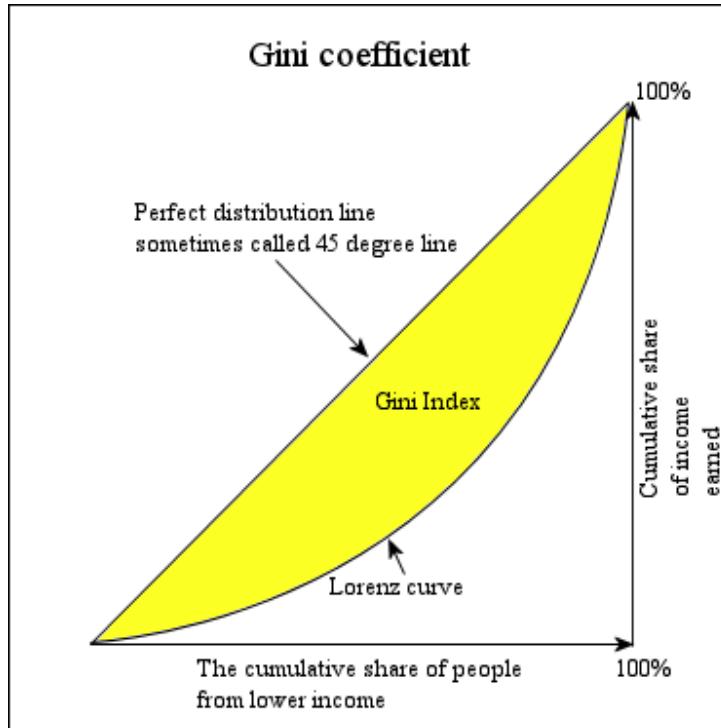


Figura 6.8: Gráfica Coeficiente de Gini

En nuestro caso, al analizar la participación en los lenguajes teniendo en cuenta el número de contribuyentes y el número de commits, cambiaremos la proporción de población e ingresos por nuestros datos. Por tanto, si el coeficiente de Gini que obtenemos del uso de un lenguaje es cercano o igual a 0, significa que el uso es muy parecido entre los contribuyentes, mientras que si se acerca al 1 quiere decir que hay usuarios que contribuyen en mucha mayor cantidad que otros.

Con ello, los datos que hemos obtenido en los 10 lenguajes minoritarios que más se repiten son:

YAML: Coeficiente de Gini 0,739

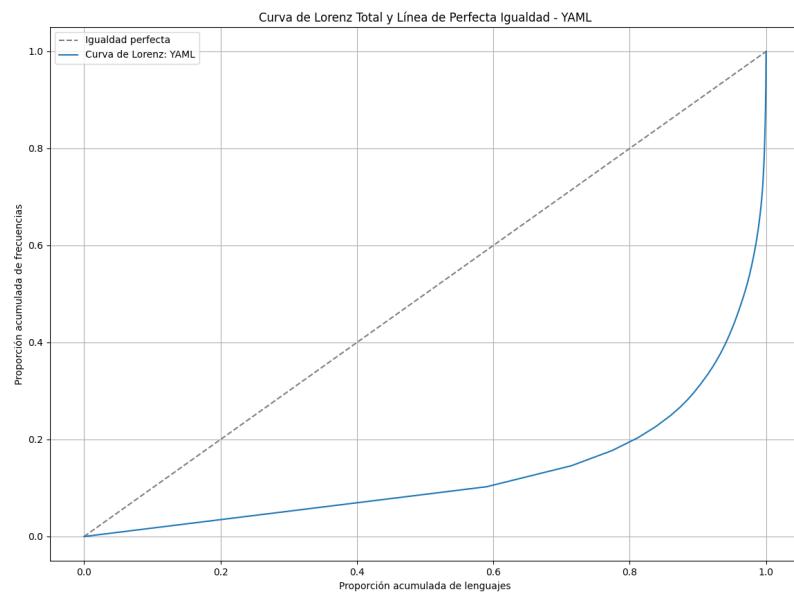


Figura 6.9: Coeficiente de Gini YAML

VIM Help File: Coeficiente de Gini 0,750

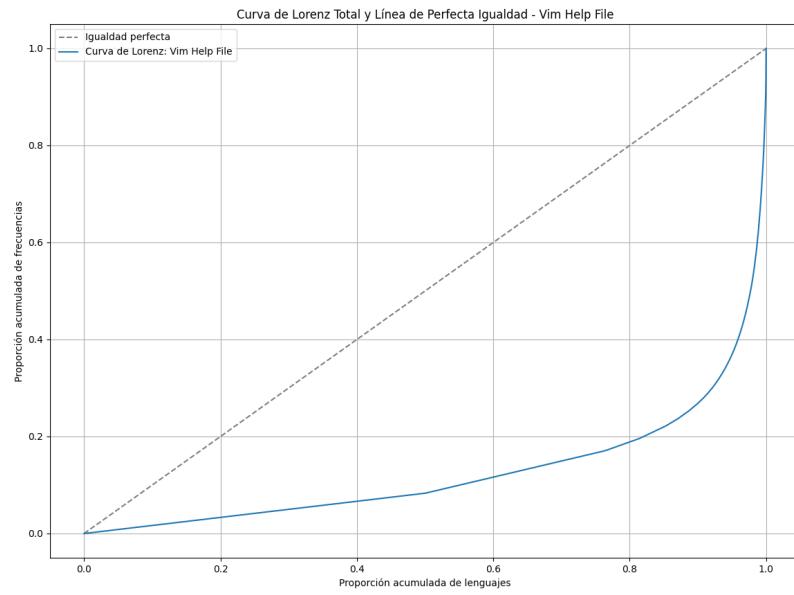


Figura 6.10: Coeficiente de Gini VIM Help File

Shell: Coeficiente de Gini 0,699

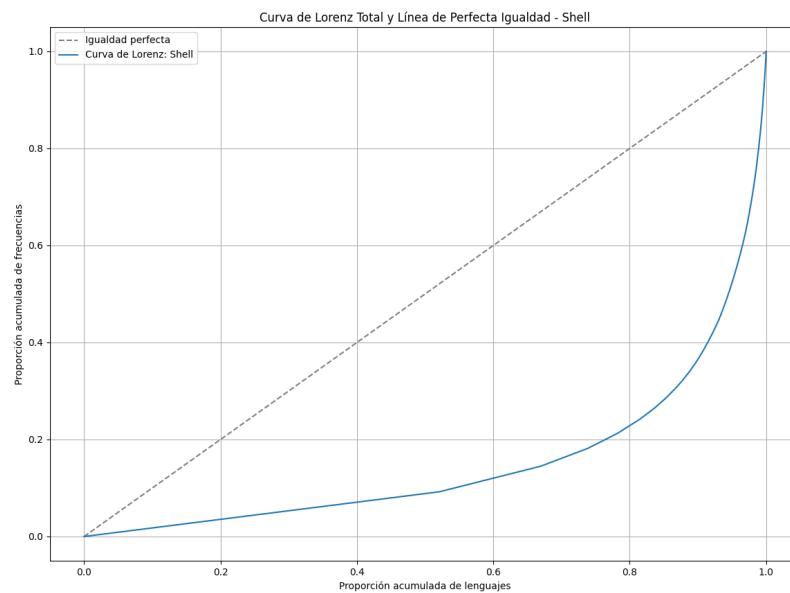


Figura 6.11: Coeficiente de Gini Shell

CSS: Coeficiente de Gini 0,814

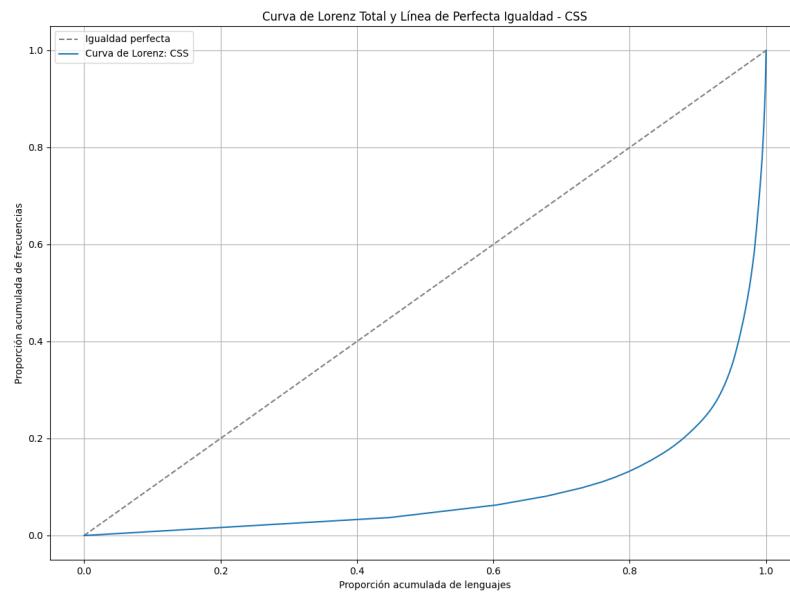


Figura 6.12: Coeficiente de Gini CSS

SVG: Coeficiente de Gini 0,814

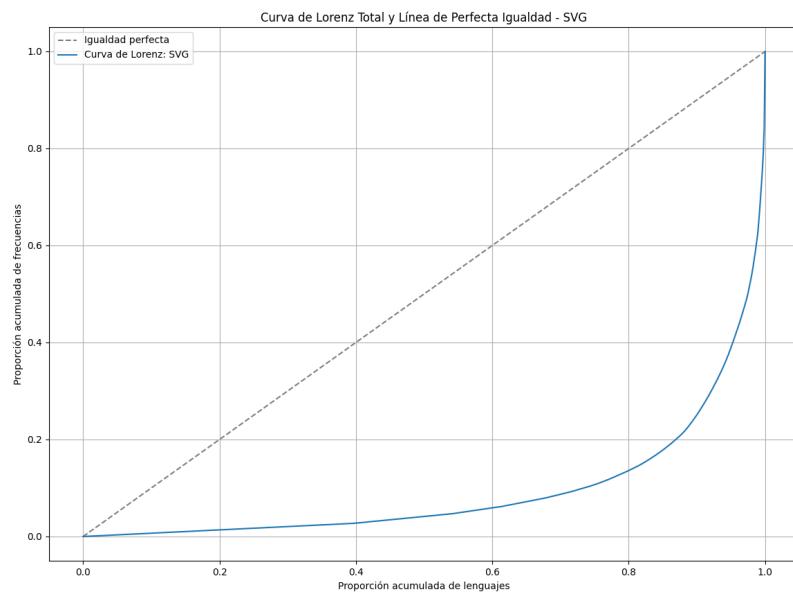


Figura 6.13: Coeficiente de Gini SVG

JSON: Coeficiente de Gini 0,879

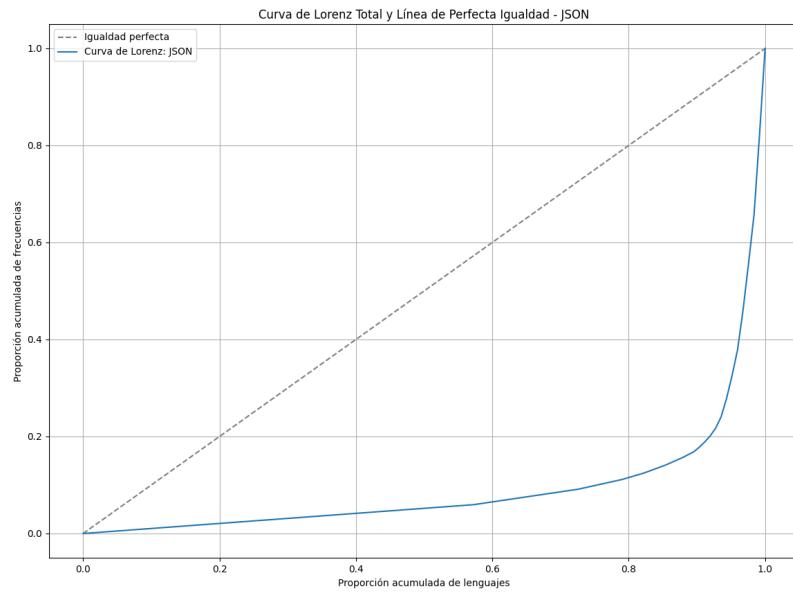


Figura 6.14: Coeficiente de Gini JSON

HTML: Coeficiente de Gini 0,960

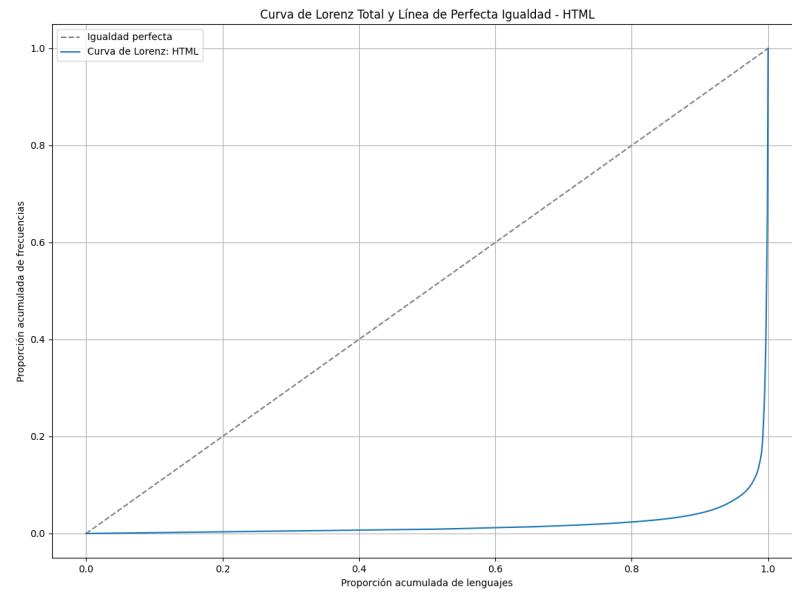


Figura 6.15: Coeficiente de Gini HTML

Markdown: Coeficiente de Gini 0,828

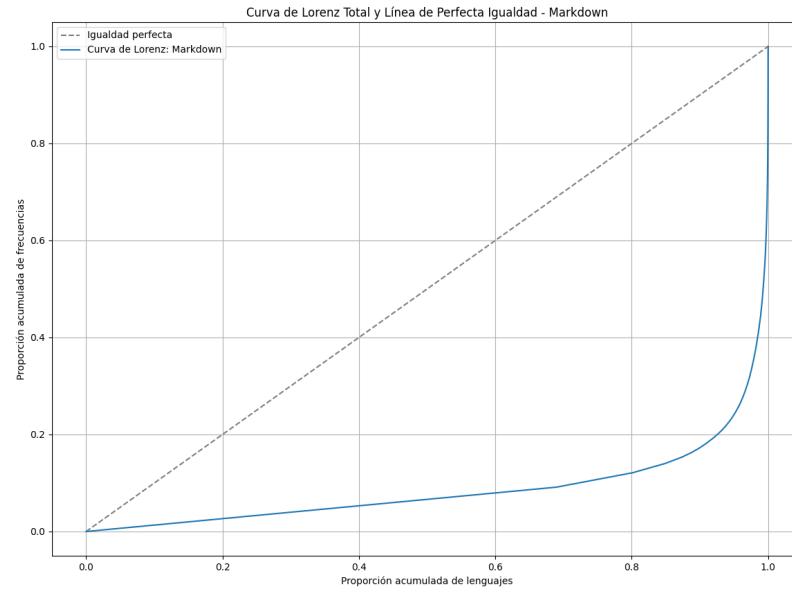


Figura 6.16: Coeficiente de Gini Markdown

JavaScript: Coeficiente de Gini 0,896

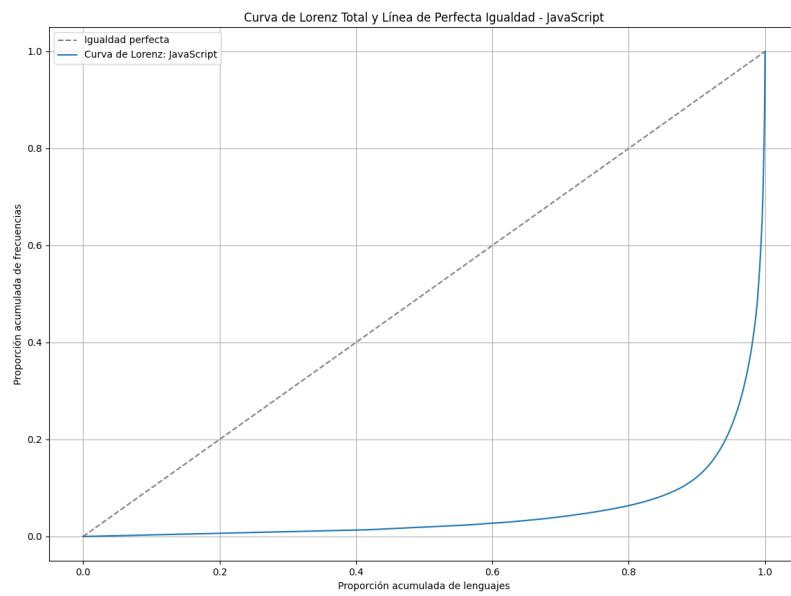


Figura 6.17: Coeficiente de Gini JavaScript

TOML: Coeficiente de Gini 0,674

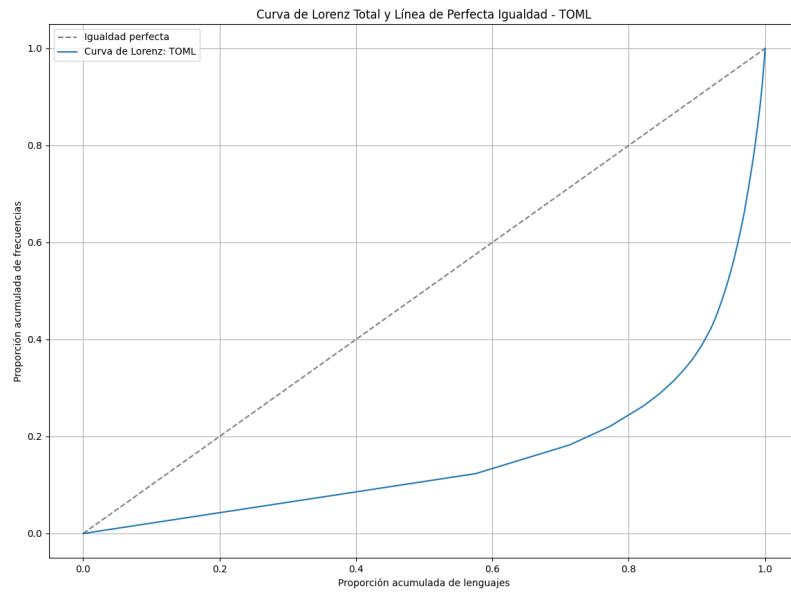


Figura 6.18: Coeficiente de Gini TOML

Con esto se comprueba que la mayoría de lenguajes minoritarios de los repositorios

que hemos analizado tienen un uso bastante desigual ya que todos se acercan bastante al 1. Esto quiere decir que en estos lenguajes suelen aportar bastante más unos pocos usuarios que el resto que contribuyen a estos.

Capítulo 7

Conclusiones

7.1. Consecución de objetivos

El objetivo general se ha conseguido a base de la cumplimentación de los objetivos específicos que se pusieron al comenzar el proyecto.

- **Obtención de datos de los repositorios de plataformas Git:** Este objetivo se ha logrado cumplir gracias, sobre todo, al uso de la librería Perceval, que ha permitido la recolección de los datos de los commits de los distintos repositorios y en la cual se basa la aplicación para hacer los análisis. Además, también, mediante llamadas a la API de GitHub se obtienen datos adicionales sobre los repositorios como el número de ficheros del repositorio o los lenguajes usados en él.
- **Tratamiento de los datos de los repositorios:** Los datos que recolectamos tanto con Perceval como con nuestras propias llamadas a la API de Github se reciben en formato JSON. La aplicación adapta las respuestas a los formatos que necesitamos para realizar los análisis y guardarlos en la base de datos según estaba definido.
- **Creación de una aplicación web para permitir un uso más fácil del aplicativo:** Mediante el uso del módulo de Flask de Python, se ha podido crear la aplicación web, formada sobre todo con HTML, CSS, JavaScript y usando la librería Bootstrap. Se ha conseguido que la experiencia del usuario que entra a la web sea más sencilla para el uso de la aplicación y que pueda ver gráficamente los análisis de los repositorios que escoja.

- **Creación de una base de datos donde almacenar los análisis de los repositorios:** Se ha conseguido cumplir este objetivo gracias a la base de datos SQLite3 que se ha creado. Además, logramos almacenar todos los datos que buscábamos para realizar los análisis finales. Quizá este sea el punto que, si bien se ha conseguido, se podría mejorar ya que la clasificación de las diferentes tablas y sus propias columnas que se ha hecho durante el proyecto, es mejorable de cara a poder sacar los datos y poder analizarlos de una manera más fácil a posteriori.

Debido a la consecución de todos estos objetivos, damos por cumplido el objetivo general del proyecto mediante nuestro programa en Python que procesa los datos obtenidos de los repositorios y que analiza el proyecto, junto a la aplicación web que permite visualizar fácilmente el análisis realizado.

7.2. Aplicación de lo aprendido

Para la realización de este trabajo de fin de grado he aplicado conocimientos adquiridos a lo largo de todas la carrera universitaria, sobre todo de aquellas asignaturas dirigidas a la programación.

1. **Informática 1:** Con esta asignatura comencé a conocer los fundamentos de la programación. Si bien el lenguaje que practiqué fue Picky, me ayudó a saber los conceptos más básicos.
2. **Informática 2:** En esta asignatura, ya más centrada en programación más avanzada, pude ampliar mi conocimiento del lenguaje Python usado en este proyecto.
3. **Protocolos para la Transmisión de Audio y Vídeo:** Fue la primera asignatura en la que comencé a conocer Python y a aplicarlo. Probablemente, sea la que más me ha ayudado con este proyecto, ya que también hice una aplicación cliente-servidor, por lo que ya algo de experiencia.
4. **Construcción de Servicios y Aplicaciones Audiovisuales en Internet:** En esta asignatura comencé a tratar como funciona el front-end de una aplicación web, practicando con HTML, CSS y JavaScript, con lo que se ha construido el cliente de mi web.

5. **Laboratorio de Tecnologías Audiovisuales en la web:** En esta asignatura conocí como funciona el back-end de una plataforma web, con lo que a la hora de hacer el servidor, pude aplicar algunos de los conceptos que aprendí.

7.3. Lecciones aprendidas

Durante la realización de este trabajo de fin de grado he aprendido:

1. Uso de la librería Perceval para la obtención de datos de repositorios Git.
2. Uso de la librería Flask para la creación de la aplicación web.
3. Uso de distintas librerías como matplotlib para la representación del análisis de los repositorios
4. Creación de una base de datos SQLite3.
5. Manejo de una base de datos y la interacción con la aplicación Python.
6. El uso de distintos lenguajes en repositorios de GitHub.
7. Organización de proyectos en cuanto a tiempo y esfuerzo.

7.4. Trabajos futuros

Tras finalizar el proyecto y hacer un análisis general de lo que se puede mejorar, cambiar o añadir a la aplicación, tengo las siguientes propuestas a futuro:

- **Mejorar la clasificación de la base de datos:** Como he comentado en un punto anterior, este es el principal punto de mejora que le encuentro a la aplicación. Me ha sido bastante complicado hacer el análisis final del uso de los lenguajes minoritarios en los repositorios debido a como he almacenado los datos en la tabla Repository. Creo que sería mejor almacenar los datos en más columnas, y sin usar formatos JSON o listas.
- **Añadir en la aplicación páginas que permitan ver los repositorios que ya están en la base de datos:** Creo que sería útil para el usuario que entre a la página, que pueda

acceder al análisis de los repositorios sin tener que buscarlos antes, es decir, una página que muestre una tabla con todos los repositorios que están en la base de datos y los pueda ver fácilmente.

- **Mejorar los tiempos de recolección de datos:** Perceval quizá es un poco lento a la hora de generar los ficheros JSON con los datos en repositorios que tienen una gran cantidad de commits realizados. Puede que haya una mejor alternativa para mejorar la eficiencia de este punto.

Apéndice A

Manual de usuario

Al entrar en la aplicación se muestra esta pantalla:



Figura A.1: Página inicial de la aplicación

Al clicar en Búsqueda de un repositorio concreto se muestra esta pantalla:

Figura A.2: Respuesta con el formulario a la petición con método GET

Se debe introducir el nombre de usuario de GitHub y el nombre del repositorio a analizar y al enviarlo se muestra la siguiente pantalla:



Figura A.3: Respuesta con el análisis del repositorio

Al hacer clic sobre los lenguajes o los usuarios aparece la información detallada acerca de ellos.

Si en la pantalla principal se pulsa en "Búsqueda por usuario" se muestra la siguiente pantalla:

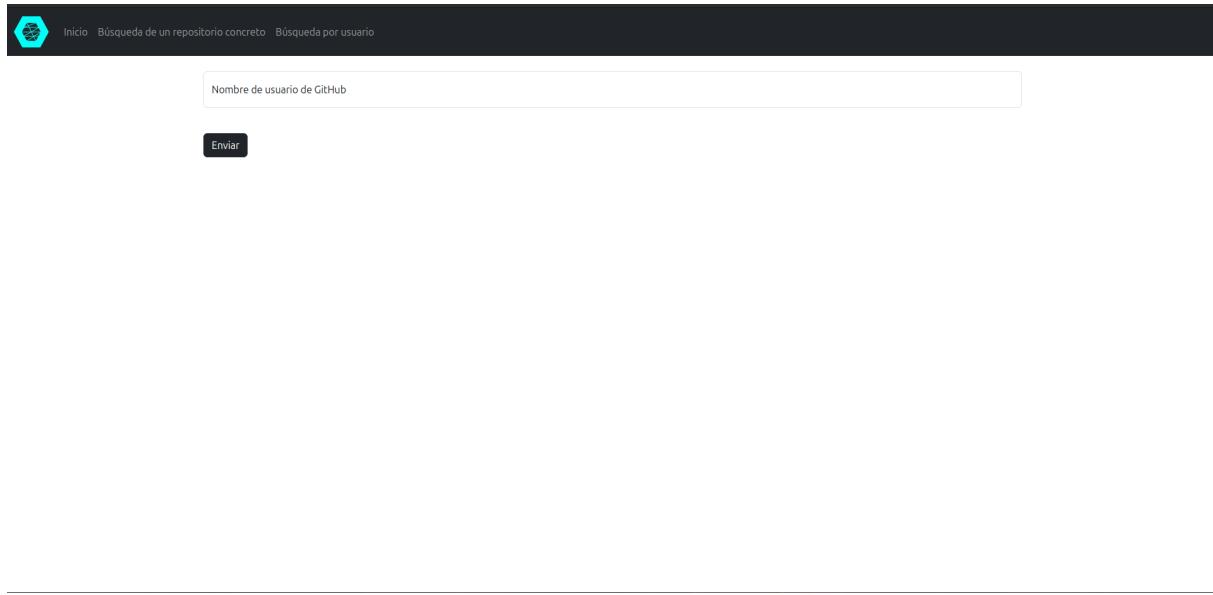


Figura A.4: Respuesta a la petición GET al endpoint /usersearch

Se debe introducir el nombre de usuario de GitHub, y se muestra una lista de los repositorios de ese usuario:



Figura A.5: Respuesta con los repositorios del usuario

Al clicar en cualquier repositorio de la lista se pasa a su análisis.

Bibliografía

[1] Bootstrap docs.

[https://getbootstrap.com/docs/5.3/getting-started/introduction/.](https://getbootstrap.com/docs/5.3/getting-started/introduction/)

[2] Flask documentation.

[https://flask.palletsprojects.com/en/stable/.](https://flask.palletsprojects.com/en/stable/)

[3] Git.

[https://git-scm.com/.](https://git-scm.com/)

[4] Github.

[https://github.com/.](https://github.com/)

[5] Jinja documentation.

[https://jinja.palletsprojects.com/en/stable/.](https://jinja.palletsprojects.com/en/stable/)

[6] Perceval read the docs.

[https://perceval.readthedocs.io/en/latest/.](https://perceval.readthedocs.io/en/latest/)

[7] Python.

[https://www.python.org/.](https://www.python.org/)

[8] J. M. . R. Copeland. *Essential SQLAlchemy*. O'Reilly Media, Inc., 2016.

[9] . P. L. Lorenzo Giovanni Bellù. *Inequality Analysis*. 2016.

[10] A. F. Montoro. *Python 3 al descubierto*. Alfaomega, 2013.

[11] M. P. P. S.T.Bhosale, Miss. Tejaswini Patil. Sqlite: Light database system. *International Journal of Computer Science and Mobile Computing*, 4(Issue 4), 2015.