UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería de la Salud

Aplicación web de sincronización entre Open Project y Github

Web App for Open Project and Github Synchronization

Realizado por
Juan Carlos Ruiz Ruiz

Tutorizado por
Cristóbal Barba González y
José Francisco Aldana Martín

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, junio de 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DE LA SALUD

Aplicación web de sincronización entre Open Project y Github

Web App for Open Project and Github Synchronization

Realizado por
Juan Carlos Ruiz Ruiz

Tutorizado por
Cristóbal Barba González y José Francisco Aldana Martín

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2023

Fecha defensa: junio de 2023

Resumen

En todas las empresas de desarrollo software con un número suficiente de empleados es necesaria la organización mediante diferentes metodologías, en concreto, una de las más utilizadas es la metodología SCRUM. Para el buen funcionamiento de esta u otra metodología ágil es necesario el uso de aplicaciones donde manejar las tareas de cada desarrollador. Este tipo de aplicaciones se usan en conjunto a gestores de versiones donde alojar el proyecto desarrollado. Dos aplicaciones muy utilizadas en estos campos son Open Project y GitHub, y a raíz del uso extendido de ambas aplicaciones aparece la necesidad de combinarlas y utilizarlas simultáneamente. Es en ese caso donde GHOPI toma importancia. GHOPI es un servicio web de código abierto a disposición de los grupos de investigación para poder sincronizar Open Project con GitHub, permitiendo así el intercambio de información entre ambas y mejorando la productividad del grupo a través de la eficiencia.

Palabras clave: Open Project, GitHub, Ingeniería del software, API

Abstract

In all software development companies with a sufficient number of employees, it is necessary to organize through different methodologies, one of the most commonly used is the SCRUM methodology. For the proper functioning of this or any other agile methodology, it is necessary to use applications to manage the tasks of each developer. These types of applications are used in conjunction with version control systems where the developed project is hosted. Two applications widely used in these fields are Open Project and GitHub, and due to the extended use of both applications, the need arises to combine and use them simultaneously. This is where GHOPI becomes important. GHOPI is an open-source web service available to research groups to synchronize Open Project with GitHub, allowing the exchange of information between both and improving group productivity through efficiency.

Keywords: Open Project, GitHub, Software Engineering, API

Índice

1 Introducción

1.1. Contexto y motivación

En el grupo de investigación Khaos Research [18], centrado en la gestión, integración y análisis de datos y del Big Data, se ha detectado un problema en lo referido a la gestión de tareas dentro del mismo. En dicho grupo, trabajan mediante la metodología SCRUM haciendo *sprints* semanales, durante los cuales los desarrolladores imputan las horas que realizadas y las tareas completadas. Al final de cada iteración, los *scrum masters* deben revisar cada una de las tareas finalizadas manualmente, comprobando los cambios realizados y creando o modificando los repositorios de GitHub. Además, debido al número de personas dentro del grupo, no pueden cumplir el principio de mínimo privilegio[1] de forma correcta, ya que tendrían que hacerlo manualmente, lo que genera un problema de seguridad durante el desarrollo de los proyectos.

Ante este dilema, en el grupo de investigación se ha propuesto crear una aplicación que automatice todos estos procesos y ayude así al desarrollo de los proyectos y al trabajo de los revisores de las tareas. Si bien es cierto que existe una extensión de Open Project para integrar GitHub, este módulo no cumple las necesidades del grupo y es por ello que se pretende crear una que satisfaga sus requisitos. Se pretende que con esta aplicación se afiancen buenas prácticas de producción dentro del grupo, como la separación de las tareas dentro del repositorio, la comunicación entre los desarrolladores y los revisores o la coherencia del método de trabajo entre los diferentes proyectos, entre otras.

Para lograrlo se quiere sincronizar Open Project (la aplicación utilizada para el reparto de tareas) con GitHub (la aplicación que se utiliza para el control de versiones en la nube) y así aumentar la eficiencia de los grupos de desarrollo software y aligerar la carga de trabajo de los *scrum masters* para que puedan centrarse en la revisión de las tareas exclusivamente o en

[1] El principio de mínimo privilegio consiste en dar, cuando sea posible, a un usuario no más que los permisos de acceso mínimos necesarios para desempeñar sus funciones laborales [19].

el desarrollo de proyectos si además se dedicasen a ello.

1.2. Objetivos del proyecto

El objetivo principal de este proyecto será el desarrollo de GHOPI[2], una aplicación basada en tecnologías web que consiga sincronizar diferentes procesos entre Open Project y GitHub. Esta aplicación deberá poder dar y quitar permisos de escritura en los repositorios de GitHub en función de las tareas de Open Project de cada desarrollador, generar mensajes informativos en las tareas de Open Project, crear nuevas ramas en el repositorio del proyecto para cada tarea creada, modificar el estado de las tareas de Open Project en función del estado de la Pull Request de su rama de GitHub asociada, entre otras funcionalidades. Todas ellas con el objetivo de mejorar la producción de proyectos software y reducir el trabajo repetitivo de los *scrum masters* del grupo.

Se pretende que la aplicación sea de código abierto y de uso público. Así, cualquier grupo de desarrollo pueda desplegar su propia instancia en su dispositivo, sin que interfiera con el resto de instancias que existan.

Por otra parte, se busca que el uso de la aplicación sea fácil e intuitivo, por lo que será necesario crear una interfaz de usuario informativa y fácil de utilizar. Cabe destacar que esta aplicación, por el contexto en el que se va a utilizar, será usada por personas experimentadas en el manejo de nuevas tecnologías, por ello la interfaz de usuario debe crearse teniendo en cuenta esta información.

1.3. Estructura del documento

Este documento recogerá la información necesaria para comprender el desarrollo y funcionamiento del proyecto para sincronizar Open Project y GitHub, cumpliendo los objetivos descritos en el apartado 1.2.

En primer lugar, en la sección 2, se describirá el estado del arte actual, explicando qué son Open Project y GitHub, además de otras tecnologías parecidas y cuáles son sus perspectivas futuras y sus predecesores. También se hará una explicación de la tecnología detrás de las API, su historia y funcionamiento, y nos centraremos en los *webhooks*, tecnología muy presente en

[2] *GitHub and Open Project Interaction*

el desarrollo del proyecto.

En la sección 3 se explicará la metodología utilizada durante el desarrollo de todo el proyecto y el porqué de la misma. Además de la planificación utilizada para el mismo, con el objetivo de clarificar cómo se ha llevado a cabo la creación de esta aplicación.

Posteriormente, en la sección 4 se explicará el proceso de diseño y la arquitectura de la solución propuesta siguiendo los requisitos obtenidos del grupo de investigación Khaos Research.

Más adelante, en la sección 5 se presentará la implementación de esta solución, explicando el porqué se han seleccionado las tecnologías empleadas y describiendo los detalles de la implementación de los diferentes componentes, entre los que encontramos la parte de la API que comunicará con Open Project o la implementación de la interfaz de usuario. Además, para finalizar dicha sección se evaluará la aplicación creada y se explicarán algunos métodos de verificación que se han utilizado.

Por último, en la sección 6, se realizará una conclusión sobre todo el proyecto, discutiendo si se han cumplido los requisitos definidos para el mismo y resumiendo los resultados obtenidos. Además, se hará una reflexión sobre posibles líneas futuras de investigación alrededor del proyecto y posibles mejoras o ampliaciones del mismo.

Además, tenemos el apéndice A en el que se describe una breve guía de uso de la aplicación, que incluye cómo descargarla y utilizarla.

1.4. Tecnologías utilizadas

En este proyecto se han usado diversas tecnologías para su desarrollo. Para la programación del *backend* de la aplicación se ha utilizado Go, un lenguaje de programación desarrollado por Google y que pretende tener una sintaxis simple sin sacrificar eficiencia, además que permite de forma muy sencilla la creación de concurrencias [8]. Por otra parte, se ha utilizado HTML5, CSS y JavaScript para la implementación de la interfaz de usuario de la aplicación. Todo el desarrollo se ha realizado utilizando Visual Studio Code, una aplicación para programar en múltiples lenguajes. Además, se ha utilizado Docker para el despliegue de Open Project.

2

Estado del arte

2.1. Ingeniería del software

El término "programación" se acuña aproximadamente durante la década de las 50 del siglo XX, con la aparición de los primeros lenguajes de programación. Anteriormente, solo algunas grandes instituciones y universidades poseían ordenadores con los que investigar, sin embargo, en la década de los 50 los ordenadores comienzan a utilizarse también en diversas empresas, comenzando así a ser una tecnología cada vez más utilizada. Como resultado de esta generalización de los ordenadores aparece la nueva profesión de programador y múltiples compañías comienzan a contratar este tipo de personal. Conforme la década de los 50 finalizaba, aparecían los primeros lenguajes de programación y la programación estaba cada vez más asentada.

Es en 1963 cuando aparece el primer ordenador de tiempo compartido, desarrollado por el MIT, que es capaz de soportar multitarea y el uso simultáneo de la máquina por varios usuarios. Esta nueva tecnología hizo que las grandes desarrolladoras comenzaran a crear sistemas basados en ordenadores de tiempo compartido. No obstante, el salto a la multitarea fue más difícil de lo previsto, debido a la falta de experiencia con los nuevos conceptos, y provocó grandes retrasos en el desarrollo de los nuevos sistemas, e incluso algunos no pudieron completarse. A este suceso se le conoce como "la crisis del software". Durante esta crisis, el comité científico NATO[3] patrocinó una conferencia en la que se discutieron nuevas formas de abordar este problema del desarrollo software [20], fue entonces cuando surgió el término "Ingeniería del software" y supuso un gran avance hacia nuevas metodologías de trabajo que se utilizan hoy en día en los procesos de desarrollo software. [29]

La ingeniería del software, según el IEEE[4] es "la aplicación de un enfoque sistemático, dis-

[3] *North Atlantic Treaty Organization*

[4] *Institute of Electrical and Electronics Engineers*

ciplinado y cuantificable al desarrollo, operación y mantenimiento de software“ [16]. Gracias al avance en ingeniería del software y el desarrollo de nuevas metodologías para la gestión de proyectos software, se han conseguido grandes avances en este campo y a día de hoy la digitalización está muy extendida en la vida cotidiana de las personas. Esto es debido a que mediante la buena organización de los grupos de desarrollo y la comunicación correcta con los clientes se han conseguido desarrollar múltiples tecnologías que han llegado al usuario medio.

2.1.1. Metodologías en la gestión de proyectos software

La ingeniería del software está estrechamente relacionada con la gestión de proyectos software. Es la disciplina que estudia la forma de gestionar este tipo de proyectos y de buscar la mejor manera de realizarlos. Para ello, a lo largo de la historia se han ideado múltiples metodologías mediante las cuales afrontar el desarrollo de software.

La más primitiva de ellas es la metodología en cascada, de la que parten la mayoría de metodologías estáticas [2]. Esta se basa en los requisitos recavados previamente para llevar a cabo el proyecto y realizar las validaciones pertinentes, esto de manera secuencial y lineal, tal y como se observa en la figura 1. A partir de esta metodología, surgen otras muchas estáticas que se han utilizado a lo largo del tiempo, entre las que encontramos el modelo en espiral o el modelo incremental. Estos modelos, pese a ser distintos del modelo en cascada, mantienen la misma secuencia que este durante el desarrollo del proyecto, pero adaptado a cada metodología. En general, los métodos tradicionales de la gestión de proyectos software carecen de una buena comunicación cliente-desarrollador, ya que toda la información se recava al inicio del proyecto y este se lleva a cabo sin tener en cuenta al cliente durante el desarrollo del mismo. Este es el problema que, desde el año 2000, pretenden solucionar las metodologías ágiles.

Las metodologías ágiles nacen en el año 2001 con el manifiesto ágil[5], en este manifiesto se defiende el desarrollo de software más útil, una mayor comunicación entre el equipo de desarrollo y el cliente, y una mayor adaptabilidad al cambio; creándose así nuevas metodologías como XP (eXtreme Programming), Kanban o SCRUM. Cada una de ellas tiene su propio enfoque y prácticas específicas, pero todas comparten algunos principios fundamentales, como la iteración continua, la retroalimentación constante y la adaptación al cambio, en pos de cumplir las ideas de una metodología ágil [7].

La más utilizada de ellas es SCRUM, en múltiples variantes. Esta forma de trabajar se cen-

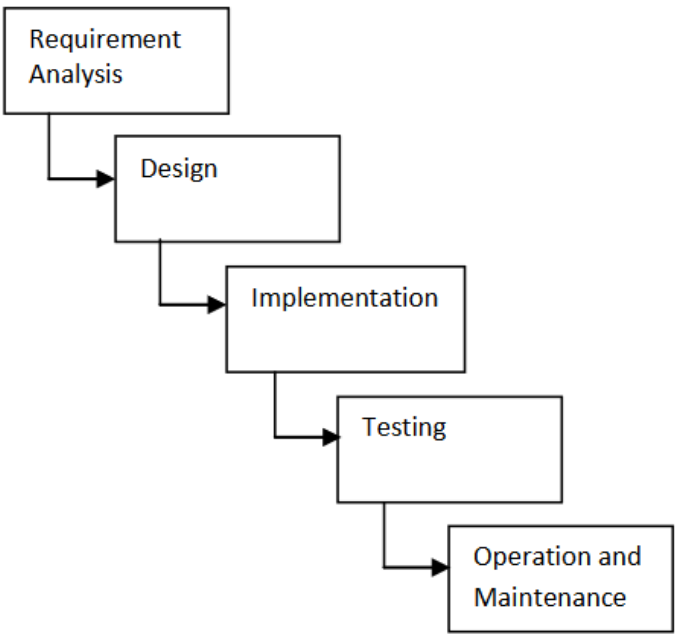

Figura 1: Fases de la metodología en cascada. Fuente: [2]

tra en "*sprints*" que son intervalos de tiempo cortos o medios (de entre una semana hasta un mes, aproximadamente) en los cuales los desarrolladores del proyecto realizarán una serie de tareas, que al final del mismo se evaluarán y discutirán con el cliente para comprobar su validez y asignar o modificar tareas, tal y como podemos observar en la figura 2. En un grupo que utilice SCRUM se definen tres roles: el propietario del producto (*product owner*), el equipo de desarrollo (*development team*) y el *Scrum Master*. El propietario del producto es responsable de definir y priorizar los requisitos del proyecto en forma de elementos de trabajo, llamados "elementos del *backlog* del producto" (*product backlog items*). El equipo de desarrollo es responsable de crear el incremento de trabajo en cada sprint, y el *Scrum Master* es el encargado de asegurar que se sigan las prácticas y principios de SCRUM y de eliminar cualquier obstáculo que impida al equipo avanzar en el proyecto. SCRUM ha demostrado ser una metodología muy eficaz en la gestión de proyectos software y satisface tanto al cliente como al grupo de desarrollo en mayor medida que los métodos tradicionales, ya que ofrece una forma de trabajo mucho más coherente y consistente [24].

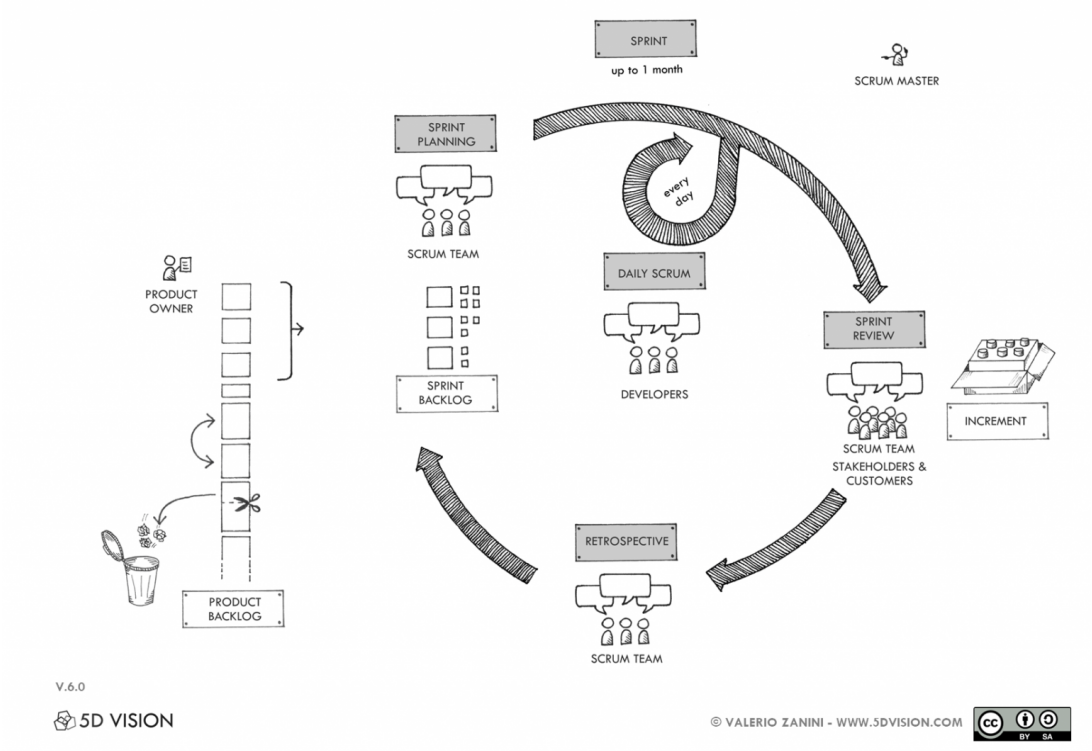

Figura 2: Gráfico que muestra el ciclo de desarrollo con la metodología SCRUM. Fuente: [28].

2.2. Open Project

Para el correcto funcionamiento de las metodologías ágiles es necesaria una buena comunicación dentro del equipo de desarrollo y con el *Scrum Master*. Tanto para diseñar y repartir las tareas, como para comunicar nuevos cambios o actualizaciones del proyecto.

Para ello, existen múltiples aplicaciones para ayudar a la gestión de proyectos software tales como, MS Teams, Trello o Slack. Sin embargo, nos centraremos en Open Project, una aplicación gratuita y de código abierto, que sirve para gestionar y organizar grupos de desarrollo. Está orientada hacia grupos de desarrollo software, más concretamente, y por ello contiene múltiples funcionalidades que son útiles para este tipo de proyectos, por ejemplo, permite la creación de *user stories* para proyectos en los que se utilicen, permiten la personalización completa de tareas y usuarios, etc [22].

2.2.1. Características principales

Al ser una aplicación de gestión de proyectos software, esta contiene las características de este tipo de aplicaciones. Permite la creación de diferentes proyectos donde englobar las

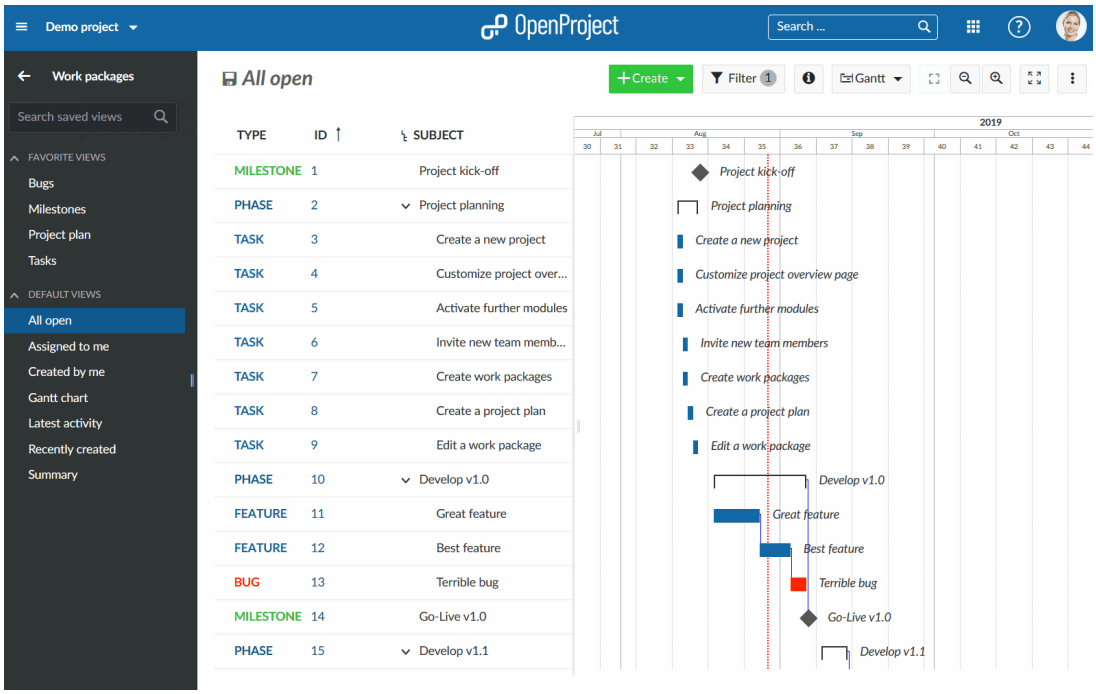

Figura 3: Vista del diagrama de Gantt en la aplicación Open Project. Fuente: [11].

tareas de cada uno de ellos. Cada proyecto puede tener asignados diferentes usuarios e incluso jerarquías dentro del mismo, también permite la creación de paquetes de trabajo que pueden ser desde hitos en un proyecto hasta la más simple de las tareas. Además, puedes observar estos paquetes en forma de diagrama de Gantt para hacer una planificación mucho más sencilla del proyecto, tal y como observamos en la figura 3. También permite el conteo temporal y de materiales necesarios durante el proyecto, esto es muy útil para controlar el estado del proyecto y gestionar mejor los recursos. No obstante, una de las características más utilizadas es la personalización mediante atributos personalizados, roles de usuarios e incluso flujos de trabajo para automatizar determinados procesos. No obstante, para nuestra aplicación lo más útil será la API que Open Project nos ofrece para utilizar su aplicación.

Todas estas características la hacen una aplicación perfecta para la gestión de proyectos software mediante metodologías ágiles como SCRUM o Kanban, entre otras.

2.2.2. Herramientas similares en el mercado

No obstante, Open Project no es la primera ni la única herramienta de este estilo en el mercado y cada una presenta ventajas y desventajas sobre Open Project. Esto se debe al auge

de las metodologías ágiles a principios de siglo, ya que se necesitaron herramientas para hacer un buen seguimiento de las tareas en el grupo.

Una de las herramientas más populares es Trello. Esta plataforma se basa en tarjetas y permite a los equipos organizar y priorizar tareas en tableros personalizados. Los usuarios pueden arrastrar y soltar tarjetas en diferentes listas para hacer seguimiento del progreso. Trello posee una versión gratuita, sin embargo, al contrario que Open Project, no permite crear proyectos ilimitados en la versión gratuita, por lo que para un grupo de trabajo real no sería suficiente dicha versión [27].

Asana es otra herramienta de gestión de proyectos que ofrece funcionalidades de colaboración y seguimiento. Permite a los equipos crear y organizar tareas, establecer hitos, asignar responsabilidades, hacer seguimiento del tiempo y el presupuesto, y colaborar en documentos. Asana está más centrado en equipos de marketing y otro tipo de negocios, no obstante, permite el uso para todo tipo de grupos de trabajo [3].

Basecamp es otra herramienta popular que permite a los equipos comunicarse, colaborar y hacer seguimiento del progreso en un solo lugar. Ofrece funcionalidades como la gestión de tareas, la programación de hitos, el seguimiento del tiempo y los archivos compartidos. Esta herramienta no es de código libre ni tiene una versión gratuita, sin embargo, sí ofrece un período de prueba [4].

Jira es una herramienta de gestión de proyectos desarrollada específicamente para equipos de desarrollo de software. Permite a los equipos crear y gestionar tareas, programar hitos, hacer seguimiento de errores y problemas, y colaborar en la planificación y el desarrollo de software [17].

Finalmente, Microsoft Project es una herramienta de gestión de proyectos ampliamente utilizada en diferentes sectores. Permite a los usuarios crear y programar tareas, establecer hitos, asignar recursos, hacer seguimiento del presupuesto y generar informes y análisis. Esta herramienta ofrece una integración total con las herramientas de Microsoft [21].

Cualquiera de estas, u otras, aplicaciones es completamente válida para la gestión de proyectos y en el contexto del desarrollo software ofrecen todo lo que un grupo puede necesitar. Sin embargo, se ha decidido utilizar Open Project debido a su alta personalización y el ofrecer la aplicación prácticamente al completo en su versión gratuita.

2.3. GitHub

GitHub es una plataforma en línea diseñada para alojar y compartir código fuente de proyectos de software. Fundada en 2008, GitHub ha crecido hasta convertirse en uno de los repositorios más grandes y populares de código abierto en el mundo, con más de 100 millones de repositorios.

La plataforma permite a los desarrolladores colaborar y trabajar en equipo de manera efectiva al ofrecer herramientas de seguimiento de problemas, control de versiones y administración de proyectos. Además, GitHub es un lugar para aprender, compartir conocimientos y descubrir nuevas tecnologías, ya que permite a los usuarios explorar y contribuir a proyectos de software de todo el mundo y promueve mediante su política de repositorios públicos el intercambio de información entre proyectos [12].

2.3.1. Características principales

GitHub posee una gran gama de características, por lo que se ha convertido en uno de los gestores de versiones en remoto más utilizados del mundo. Podemos ver en la figura 4 GitHub centra sus características en los repositorios de los usuarios, donde se implementa la gran mayoría de cambios.

La plataforma permite a los desarrolladores alojar y compartir sus proyectos de software a través de la creación de repositorios públicos o privados. Con Git, un sistema de control de versiones distribuido, GitHub realiza un seguimiento de los cambios realizados en un proyecto a lo largo del tiempo. Esto permite a los usuarios colaborar en cambios, revertir cambios no deseados y fusionar diferentes versiones del proyecto.

Además, GitHub ofrece una amplia gama de características que facilitan la colaboración y la gestión de proyectos. Las ramas permiten a los usuarios trabajar en diferentes características o solucionar diferentes problemas sin afectar la versión principal del proyecto, y las solicitudes de extracción permiten a los usuarios proponer cambios y solicitar su inclusión en la versión principal del proyecto. Las herramientas de seguimiento de problemas y los wikis permiten a los usuarios gestionar la documentación y discutir temas relacionados con el proyecto. También permite la creación de organizaciones para unificar todos los proyectos de un grupo de desarrollo en un espacio de la herramienta concreto, lo que además permite diferentes flujos

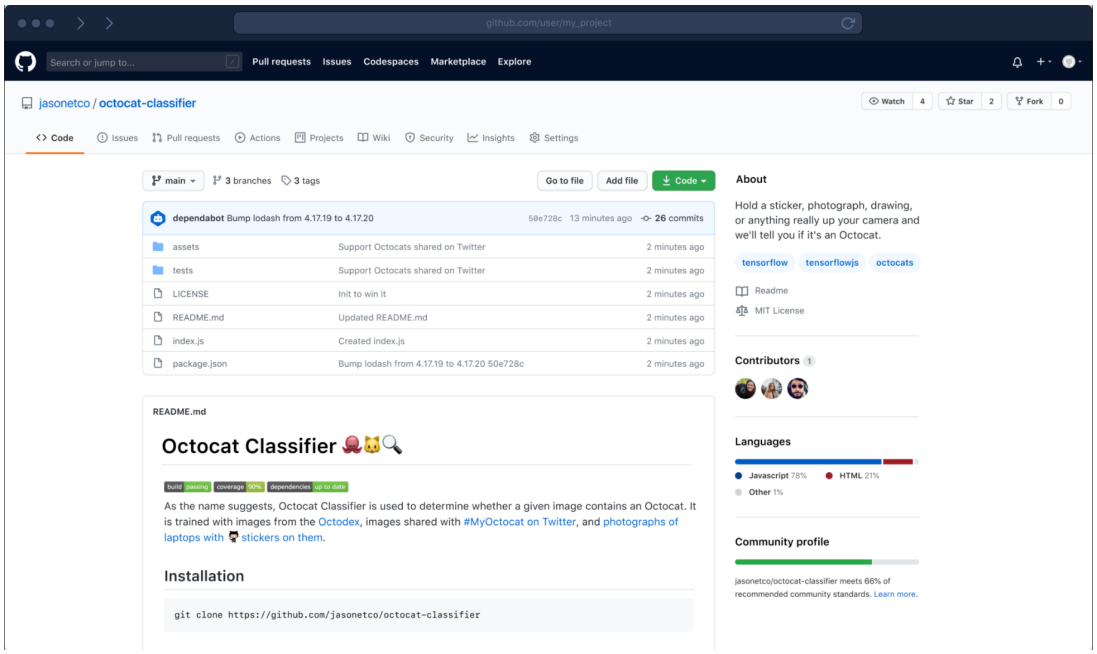

Figura 4: Interfaz de usuario de GitHub. Ventana del repositorio. Fuente: [13].

de trabajo [12].

GitHub también se integra con una amplia gama de herramientas y servicios, incluyendo servicios de integración continua, herramientas de análisis de código y plataformas de alojamiento web. Además, permite la creación de flujos de trabajo que permitan automatizar determinados procesos como realizar pruebas automáticas. En general, GitHub es una herramienta esencial para los desarrolladores de todo el mundo que buscan alojar, compartir y colaborar en proyectos de software de manera efectiva.

2.3.2. Herramientas similares en el mercado

En el mercado existen varias herramientas similares a GitHub que ofrecen funcionalidades similares para la gestión de proyectos de software. Entre las herramientas más populares se encuentran GitLab, Bitbucket y SourceForge.

GitLab es una plataforma en línea que ofrece alojamiento de repositorios de código fuente, control de versiones y herramientas de gestión de proyectos similares a GitHub. Además, GitLab ofrece herramientas adicionales para la gestión de *DevOps*, como herramientas de integración continua y entrega continua (CI/CD), lo que permite a los equipos de desarrollo automatizar el ciclo de vida completo de las aplicaciones [14].

Bitbucket es una plataforma de alojamiento de repositorios de código fuente de Atlassian, que también ofrece herramientas de seguimiento de problemas y control de versiones similares a GitHub. A diferencia de GitHub, Bitbucket es más conocido por su integración con otros productos de Atlassian, como Jira y Confluence, lo que facilita la gestión de proyectos para equipos que ya utilizan otras herramientas de Atlassian [6].

SourceForge es una plataforma de alojamiento de repositorios de código fuente fundada en 1999 por VA Software, y ofrece herramientas de gestión de proyectos y control de versiones similares a GitHub. Además, SourceForge ofrece alojamiento gratuito para proyectos de software de código abierto y una amplia gama de herramientas de colaboración, incluyendo wikis, foros y herramientas de seguimiento de problemas [1].

Si bien GitHub es una de las herramientas más populares y ampliamente utilizadas para la gestión de proyectos de software, existen varias alternativas similares que ofrecen funcionalidades similares. Cada herramienta tiene sus propias ventajas y características, por lo que es importante evaluarlas cuidadosamente para encontrar la mejor solución para las necesidades específicas de cada proyecto.

2.4. API

API significa “Interfaz de Programación de Aplicaciones“ en inglés y se refiere a un conjunto de protocolos, herramientas y estándares que permiten que diferentes aplicaciones y sistemas se comuniquen y trabajen juntos de manera eficiente y efectiva.

En términos simples, una API es una interfaz que permite que diferentes sistemas interactúen entre sí de manera programática. Las API proporcionan una forma estandarizada y estructurada de comunicación entre aplicaciones, lo que significa que los desarrolladores pueden utilizarlas para integrar diferentes sistemas y servicios en sus propias aplicaciones de software. Las API se utilizan en una amplia variedad de contextos, desde la integración de plataformas de redes sociales en aplicaciones móviles hasta la integración de sistemas de pago en sitios web de comercio electrónico. También se utilizan en la automatización de procesos empresariales, la integración de servicios de terceros y la recopilación de datos de diversas fuentes. Las API pueden ser públicas o privadas, y pueden ser utilizadas por desarrolladores de software para crear aplicaciones que aprovechan la funcionalidad proporcionada por los sistemas y servicios que ofrecen las API. Las API también pueden ser utilizadas por organizaciones para crear

sistemas integrados y automatizar procesos empresariales complejos [15].

La idea de las API no es nueva y ha existido desde los primeros días de la informática. En los primeros sistemas informáticos, las API se utilizaban para que los desarrolladores pudieran crear aplicaciones para interactuar con el hardware subyacente del sistema. Sin embargo, la noción moderna de las API como interfaces estandarizadas para la comunicación entre sistemas se desarrolló a finales de la década de 1990 y principios de la década de 2000, cuando se popularizó la arquitectura orientada a servicios (SOA) y la web comenzó a crecer en popularidad. Con la aparición de la web, las empresas comenzaron a darse cuenta del valor de las API para permitir que los desarrolladores crearan aplicaciones que pudieran interactuar con sus sistemas y servicios en línea. Al hacerlo, las empresas podían ofrecer una experiencia de usuario mejorada y más personalizada para sus clientes, y también podían integrarse fácilmente con otros sistemas y servicios de terceros. En los últimos años, la popularidad de las API ha aumentado aún más con el crecimiento de la nube y los servicios en línea. Las empresas están utilizando cada vez más las API para ofrecer acceso a sus servicios y datos, lo que permite a los desarrolladores crear aplicaciones innovadoras y personalizadas utilizando una variedad de tecnologías y plataformas.

2.4.1. Webhooks

Los webhooks son un tipo de API que permite que las aplicaciones se comuniquen en tiempo real a medida que ocurren eventos en un sistema en línea. A diferencia de una API tradicional, donde una aplicación debe solicitar datos de un servidor de manera activa, con los webhooks, el servidor envía datos a una aplicación en tiempo real tan pronto como se produce un evento.

El proceso de funcionamiento de un webhook es el siguiente: una aplicación o servicio en línea puede registrar una URL como “destino“ para un evento determinado. Cuando ese evento ocurre, el servicio en línea envía una solicitud HTTP POST a la URL registrada, que contiene información sobre el evento en sí. La aplicación receptora de la URL puede entonces procesar la información y tomar cualquier acción necesaria [26].

En la figura 5 se observa el funcionamiento de los webhooks en comparación a la estrategia clásica de preguntas constantes. Tradicionalmente, se ha creado una función que pida una determinada información cada cierto tiempo, esto asegura la obtención de información

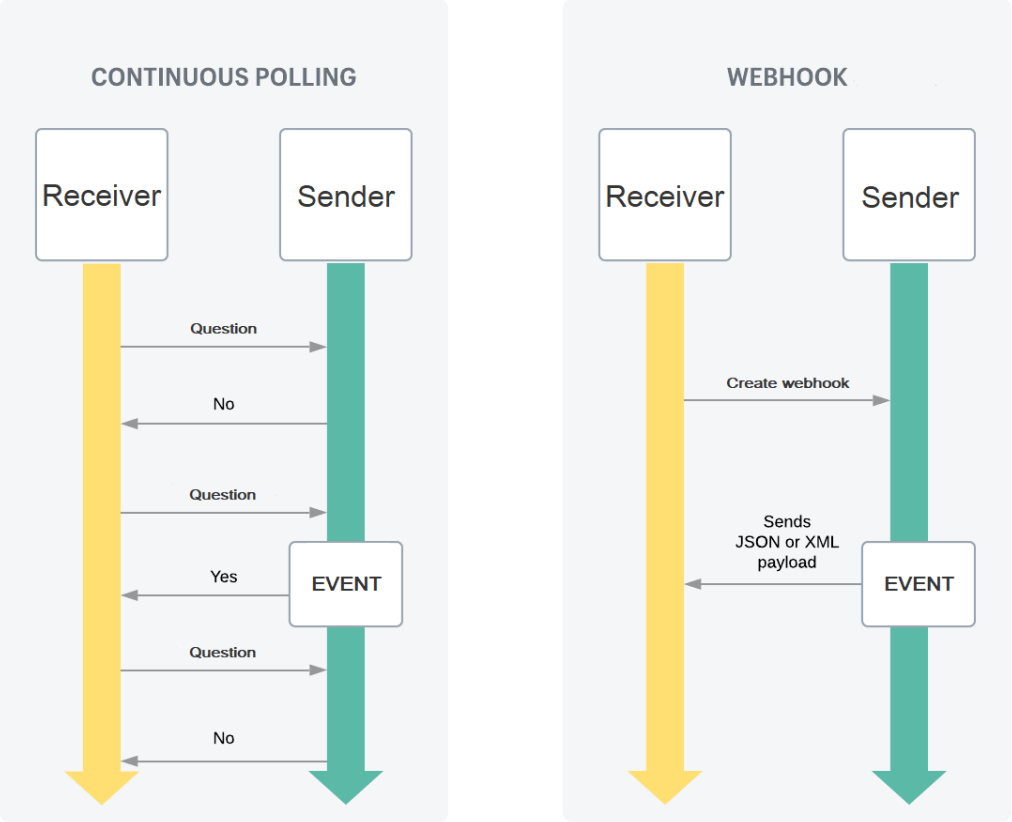

Figura 5: Esquema de la diferencia entre *continuous polling* y los *webhooks*. Esquema adaptado de https://shopify.dev/ .

de una API y es más fácil de implementar. Sin embargo, esto supone un coste que, para determinados casos, no es asumible, ya sea por eficiencia o porque la API no pueda soportar tanta cantidad de llamadas. Los webhooks, por otro lado, no se lanzan desde el lado del receptor sino desde el emisor y envían un mensaje JSON o XML cuando se produzca el efecto que se haya configurado.

2.4.2. Casos de uso

Existen múltiples casos de uso en los que utilizar los *webhooks*. Son muy útiles a la hora de enviar información en tiempo real, para ahorrar recursos y son extremadamente útiles para integrar aplicaciones en línea o con cambios reiterados.

Los *webhooks* son especialmente útiles en situaciones en las que se requiere una respuesta en tiempo real. Por ejemplo, una aplicación de comercio electrónico puede utilizar un *webhook*

para enviar una notificación a una aplicación de envío en tiempo real tan pronto como se realiza un pedido. De esta manera, la aplicación de envío puede comenzar a preparar el pedido de inmediato, en lugar de esperar a que se solicite la información.

Los *webhooks* también pueden utilizarse para integrar diferentes aplicaciones y servicios en línea. Por ejemplo, una aplicación de seguimiento de tareas puede utilizar un *webhook* para recibir actualizaciones en tiempo real sobre cambios en un proyecto de gestión de proyectos. O una aplicación de análisis de redes sociales puede utilizar *webhooks* para recibir actualizaciones sobre menciones de marca en Twitter.

3 Metodología

3.1. Selección de la metodología

Para el desarrollo del proyecto se han planteado diferentes metodologías con las que regir el mismo. Al tratarse de un proyecto individual, las metodologías ágiles como SCRUM deben adaptarse, ya que no existe un grupo de trabajo al uso. En un principio se planteó utilizar un método de trabajo en cascada o de prototipado, ya que es lo más simple de adaptar a trabajos individuales. Sin embargo, al ser un proyecto para un grupo de investigación, era necesario mantener un contacto constante para conocer las necesidades y adaptar la aplicación a sus hábitos dentro del equipo. Es por esto que se descartó el utilizar la metodología en cascada, además evitaríamos la entrega de un producto no deseado, lo que alargaría el tiempo de producción y, en consecuencia, dificultaría la entrega. Por otra parte, el modelo de prototipado, era una opción buena, no obstante, y debido a la naturaleza del proyecto, no se podía llevar a cabo, ya que el grupo de investigación Khaos Research no podía arriesgar la integridad de su plan de trabajo lanzando una aplicación incompleta.

Finalmente, se ha decidido utilizar una variante de SCRUM para grupos de trabajo pequeños [24]. En esta variante el mismo desarrollador es el encargado de planificar las tareas y de contabilizar el tiempo. Mientras que, el cliente, en este caso los tutores del proyecto, supervisaban el avance del mismo. Se realizan *sprints* de dos semanas, tras las cuales los tutores mantienen una reunión con el alumno para observar el estado del proyecto y sugerir modificaciones, tal y como se haría en un modelo de SCRUM al uso. Además, se han ido planificando las tareas en cada uno de esos intervalos y así puedan ser supervisadas por los tutores. Por otra parte, no existían las reuniones diarias que sí existirían en un grupo de desarrollo que utilice SCRUM. Lo que ha permitido una mayor eficiencia y libertad a la hora de desarrollar el proyecto, evitando además conflictos en los repositorios o entre los diferentes cambios que se

han hecho a lo largo del desarrollo.

Utilizando este modelo de trabajo se ha conseguido realizar el proyecto con facilidad y obteniendo un resultado más que satisfactorio para el cliente. Además de cumpliendo los plazos estipulados por los mismos y permitiendo un tiempo de pruebas en el grupo antes de poner la aplicación en producción.

3.2. Planificación del proyecto

Cada dos semanas se han ido realizando tareas acordes al estado del proyecto y siguiendo una planificación concreta. Comenzando con la búsqueda de información y documentación del desarrollador. Más adelante procediendo con el diseño e implementación del mismo, y finalmente realizando una serie de pruebas de validación del proyecto, tanto de integración como pruebas beta dentro del grupo. Durante todo este período se ha documentado correctamente el código de la aplicación y se han creado guías para clarificar el uso de la misma.

En la tabla 1 podemos observar claramente una planificación aproximada del proyecto. Este se divide en 4 fases generales: investigación, modelado, implementación y verificación y pruebas. Dentro de cada una de estas fases se definen una serie de tareas generales que durante cada uno de los *sprints* se subdividirán en tareas simples que se irán realizando durante el desarrollo. Además, cada una de estas tareas tiene asociada una cantidad de horas aproximada con la que calcular el tiempo necesario para el desarrollo del proyecto.

Fase	Tarea	Tiempo estimado
Investigación	Selección de tecnologías	15 h
Investigación	Estudio del lenguaje Go	20 h
Investigación	Estudio de las API y tecnologías web	10 h
Investigación	Estudio de la API de GitHub	10 h
Investigación	Estudio de la API de Open Project	10 h
Modelado	Análisis de requisitos	20 h
Modelado	Abstracción de la estructura	10 h
Modelado	Abstracción de las interacciones	10 h
Implementación	Desarrollo de la API para interactuar con Open Project	30 h
Implementación	Desarrollo de la API para interactuar con GitHub	30 h
Implementación	Desarrollo de la interfaz web	30 h
Verificación y pruebas	Pruebas unitarias y de integración	20 h
Verificación y pruebas	Refactorización del código	15 h
Verificación y pruebas	Pruebas beta	20 h
Verificación y pruebas	Entrega del prototipo final	-

Cuadro 1: Planificación por horas estimadas del proyecto dividiendo por fases y tareas generales

4

Diseño y arquitectura de la solución

4.1. Análisis de requisitos

El análisis de requisitos es, según el IEEE, "el proceso de estudio de las necesidades del usuario para llegar a una definición de los requisitos del sistema, hardware o software" [16]. Este proceso es sumamente importante para hacer un buen diseño de la aplicación, no obstante, gracias a la metodología planteada para el proyecto, estos requisitos o verse modificados durante el desarrollo.

En este caso se van a recoger los requisitos más importantes para entender cuál es el objetivo principal de la aplicación en un inicio. Se tendrán en cuenta los siguientes parámetros: El requisito en cuestión, el nivel de prioridad del mismo (si es necesario o deseable) y, si es posible, definir un criterio de aceptación para el mismo.

En las tablas 2 y 3 se observan los requisitos funcionales y no funcionales, respectivamente, recavados al inicio del proyecto, estos serán trabajados durante el desarrollo del mismo en función de su prioridad. En la tabla se muestra la identificación por la cual podremos referirnos al requisito en cuestión, una descripción breve del mismo, el criterio de aceptación que se usará y el nivel de prioridad que tiene cada requisito (se dividirán en prioritario, necesario y deseable). Esta tabla servirá de guía para el desarrollo del proyecto.

ID	Descripción	Criterio de aceptación	Nivel de prioridad
RF-0	Todo desarrollador que no tenga una tarea asociada a un repositorio no tendrá permisos de escritura a ese repositorio	Las personas sin tareas en el proyecto no pueden tener permiso de escritura	Prioritario
RF-1	Todo desarrollador que tenga una tarea asociada a un repositorio tendrá permisos de escritura a ese repositorio	Las personas con tareas en el proyecto deberán tener permiso de escritura	Prioritario
RF-2	La aplicación debe crear ramas en un repositorio de GitHub por cada tarea de Open Project	Por cada tarea marcada en Open Project existirá una rama en su repositorio de GitHub	Prioritario
RF-3	Se debe notificar en Open Project los cambios en las ramas de dicha tarea	Cuando se produzca un cambio en su rama asociada	Deseable
RF-4	Debe aparecer un mensaje en Open Project redireccionando a la creación de una *Pull Request*	Tras crear una tarea en Open Project aparecerá un mensaje enlazando con la *Pull Request* asociada	Deseable
RF-5	Debe aparecer un mensaje en Open Project cuando cambie una *Pull Request*	Aparecerá un mensaje en Open Project al abrir o cerrar una tarea	Necesario
RF-6	Debe poder sincronizarse manualmente ambas aplicaciones mediante un botón o comando	Existirá un comando o botón con el que realizar los cambios automáticamente	Prioritario
RF-6	La aplicación debe poder configurarse fácilmente	Tiene que existir una interfaz de usuario reconocible y utilizable	Prioritario

Cuadro 2: Tabla de requisitos funcionales (RF-nn) del proyecto

ID	Descripción	Criterio de aceptación	Nivel de prioridad
RNF-0	Debe proteger la seguridad de las credenciales de Open Project y GitHub	No se podrá acceder a las credenciales de las aplicaciones mediante ningún comando de la API	Necesario
RNF-1	Debe ser seguro iniciar sesión en GitHub u Open Project	Deben usarse protocolos de seguridad para evitar filtraciones durante el inicio de sesión	Necesario
RNF-2	La interfaz de usuario deberá ser accesible desde un navegador web	Se utilizará el *framework* adecuado para crear una interfaz web	Necesario
RNF-3	La interfaz de usuario seguirá tendrá un estilo coherente y agradable a la vista	-	Deseable

Cuadro 3: Tabla de requisitos no funcionales (RNF-nn) del proyecto

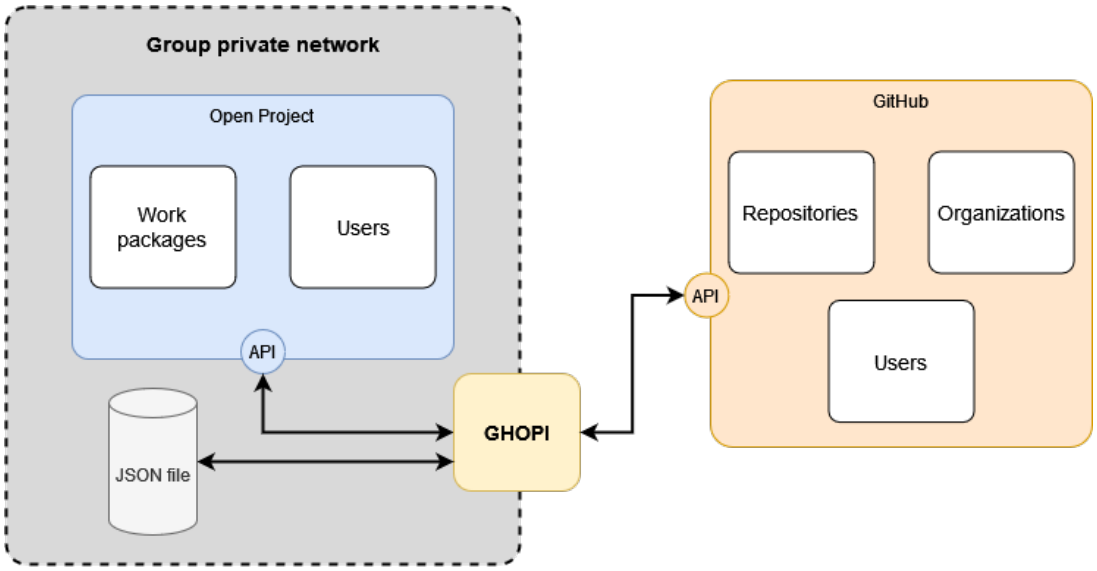

Figura 6: Diseño de la arquitectura de la aplicación

4.2. Arquitectura de la solución propuesta

Teniendo en cuenta los diferentes requisitos recavados para el proyecto. Se pretende crear una aplicación que se pueda levantar localmente en una máquina y que acceda a ambas aplicaciones a través de las API de cada una de ellas. En la gran mayoría de grupos de investigación, el acceso a Open Project está restringido a una red local privada, por lo que la solución propuesta debe poder acceder a esta red. Esto será posible, ya que, GitHub sí es una aplicación pública con lo que podremos acceder a esta a través la red local privada mencionada.

En la figura 6 se representa la arquitectura que se propone para la aplicación. Esta se levantará en el límite de la red privada del grupo de investigación. Es decir, se levantará dentro de la propia red, pero podrá ser accesible desde aplicaciones públicas, como GitHub. La aplicación deberá ser capaz de hacer llamadas a ambas aplicaciones, así como estas aplicaciones hacer llamadas a la aplicación. Cabe destacar que, ya que la información que necesitamos persistir no es suficiente para tener que mantener un servidor de base de datos dedicado, se ha optado por utilizar un archivo JSON para guardar los datos de forma estructurada y así mejorar el rendimiento de la aplicación, además de facilitar el uso de la aplicación.

4.2.1. Descripción de los componentes

Una vez estudiada la arquitectura externa de la aplicación, debemos entender la estructura interna de la misma. La aplicación posee tres componentes principales: la interfaz de usuario, la API para Open Project y la API para GitHub.

La interfaz es la componente más simple de las tres y, a su vez, esencial para el proyecto. Esta componente será la encargada de relacionar al usuario con la aplicación y de proporcionar información sobre el uso de la misma. A través de la interfaz el usuario podrá acceder a sus cuentas de Open Project y GitHub desde la aplicación para poder dar los permisos necesarios. Además, existirá la posibilidad de utilizar un botón para actualizar la sincronización entre ambas aplicaciones por si el usuario lo deseara. Por otro lado, la interfaz de usuario también proporcionará una forma simple de visualizar los cambios que se han ido produciendo y los errores que hayan podido aparecer, además de contener un manual de usuario para facilitar el uso de la misma.

La API de Open Project contendrá todas las funciones necesarias para realizar cambios en dicha aplicación, así como los *endpoints* que recibirán la información desde GitHub y que deberá ser trasladada a Open Project. Entre otras cosas, esta componente deberá ser capaz de hacer las llamadas necesarias a la API de Open Project para realizar los cambios pertinentes y modificar o filtrar los datos enviados desde GitHub para que se añadan a Open Project. Además, a través de esta componente se realizará el inicio de sesión en la cuenta de Open Project que otorgue los permisos a la aplicación para la modificación de los diferentes proyectos del grupo de desarrollo, todo ello a través de un protocolo *OAuth 2.0* [5].

Por último, la API REST de GitHub será la encargada de realizar los cambios en esta aplicación, además, recibirá toda la información desde Open Project que deberá ser trasladada a GitHub. Esta componente tendrá la capacidad de filtrar los datos enviados desde el gestor de tareas para obtener la información que se deba enviar a GitHub, además de obtener a partir de esta la información del repositorio destino. También utilizará el protocolo *OAuth 2.0* para iniciar sesión en la cuenta administradora de GitHub que deberá proporcionarle los permisos necesarios a la aplicación. Por último, esta componente deberá contener *endpoints* que permitan enviar información desde Open Project para que sea trasladada a Github.

[5] *Open Authorization* 2.0

Las tres componentes están intrínsecamente relacionadas, ya que, las tres son imprescindibles para el correcto funcionamiento de la aplicación. La gran mayoría de relaciones ocurren entre las API y la interfaz de usuario, ya que es en esta donde se van a ver reflejados los cambios entre ambas aplicaciones. Utilizando estas tres componentes y gracias a las relaciones entre ellas, se creará la aplicación capaz de sincronizar Open Project y GitHub y así conseguir el objetivo final del proyecto.

5 Implementación

5.1. Selección de tecnologías y herramientas

Para la implementación se han discutido diferentes tecnologías con las que abordar el problema. Lo primero que había que decidir era el lenguaje con el que desarrollar la aplicación. Se estudiaron dos opciones: Python y Go.

Python es un lenguaje de programación interpretado, interactivo y orientado a objetos; generalmente utilizado en el desarrollo de aplicaciones de software. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. Python se destaca por su sintaxis clara y legible, lo que lo hace muy fácil de aprender y comprender. Python se ha vuelto extremadamente popular en los últimos años debido a su enfoque en la legibilidad del código y a su amplia gama de aplicaciones. Es utilizado en una variedad de campos, como desarrollo web, análisis de datos, inteligencia artificial, aprendizaje automático, automatización de tareas, desarrollo de juegos y mucho más. Una de las características más destacadas de Python es su amplia biblioteca estándar, que proporciona módulos y funciones predefinidos que facilitan tareas comunes. Además, Python cuenta con una gran comunidad de desarrolladores que contribuyen con bibliotecas y *frameworks* de código abierto, lo que amplía aún más sus capacidades y funcionalidades, y lo hace un lenguaje muy versátil tanto para desarrolladores nóveles como para los más experimentados[23]. Los *frameworks* que ofrece Python para el desarrollo de aplicaciones web y API son muy comunes actualmente en la industria, y se encuentran ampliamente documentados y con una comunidad activa. En concreto, se barajaron la posibilidad de utilizar *Flask*[10] o *FastAPI*[9], dos *frameworks* de Python cuyo objetivo es facilitar la creación de aplicaciones webs y API, justo lo que necesitamos para nuestro proyecto. *FastAPI* ofrece un marco de posibilidades más amplio, además de un mejor rendimiento que *Flask*, sin embargo, durante el grado hemos adquirido cierta experiencia con este *framework* con lo que también es una buena posibilidad,

ya que no sería necesario el aprendizaje desde cero de este.

Por otro lado, Go es un lenguaje de programación de código abierto desarrollado por Google en 2007. Go se creó con el objetivo de combinar la eficiencia de la programación de sistemas con la facilidad de uso y la productividad de la programación de lenguajes más modernos. Se caracteriza por ser un lenguaje rápido, eficiente y escalable, con un enfoque en la concurrencia y la paralelización. Al igual que Python, utiliza una sintaxis sencilla y clara que facilita la legibilidad del código y la colaboración entre desarrolladores. Cuenta con una amplia biblioteca estándar que incluye herramientas y paquetes para multitud de tareas. Go está específicamente desarrollado para ser eficiente y está muy enfocado en el desarrollo web y la creación de API gracias a sus bibliotecas nativas para esto, tales como *http* o *Gin*. Además, la facilidad con la que se pueden utilizar concurrencias y el tipado estático lo hacen el lenguaje perfecto para la aplicación que queremos desarrollar, ya que, nos ofrece eficiencia y robustez en el código sin necesidad de utlizar *frameworks*. Además, utilizar un nuevo lenguaje que, además, está aumentando su popularidad en la última década, ampliará en gran medida el impacto que tendrá la aplicación al público y aumentará las capacidades del alumno para un futuro.

Por todo ello, y a pesar de necesitar aprender un lenguaje de programación desde cero, se ha decidido utilizar Go como la tecnología que se utilizará para la implementación de la aplicación. También se hará uso de otras herramientas para el desarrollo web como HTML5, CSS y JavaScript; todo ello para que la aplicación sea lo más completa posible.

5.2. API REST para la comunicación con Open Project

Una vez seleccionadas las herramientas con las que se va a trabajar y habiendo diseñado la aplicación, debemos implementar en código la solución propuesta. Para ello, y siguiendo la metodología descrita en la sección 3, se han implementado todas las componentes necesarias para el correcto funcionamiento de la aplicación. Para mejorar la comprensión de la implementación se ha decidido explicar en primer lugar las funciones relacionadas con las API de Open Project.

5.2.1. Inicio de sesión en Open Project

En primer lugar, explicaremos la implementación de las funciones para lograr la comunicación con Open Project. Esta componente de la API deberá permitir el acceso a Open Project desde la aplicación y también, al recibir información desde GitHub, la procesará y enviará a Open Project.

Para poder acceder a las cuentas, tanto de Open Project en este caso como de GitHub, se ha utilizado un patrón de diseño *Factory*, con el que definir las funciones que deben existir para cada uno de los componentes, pero sin definir su comportamiento, tal y como se observa en el cuadro de código 1. En esta clase abstracta se han definido una serie de funciones que han de existir en cada implementación del protocolo OAuth 2.0 que necesitemos para una aplicación. Las funciones que realizarán el protoclo son: `LoginHandler`, que es la función que se utiliza para iniciar el proceso de acceso y se encarga de hacer la llamada correspondiente a la aplicación que estemos usando; `CallbackHandler`, que es la función a la que la aplicación destino enviará la información necesaria para obtener el código de acceso y es en esta función donde se hará uso de `getAccessToken` y `getData` para procesar los datos y obtener información importante como la clave de acceso; por último, se utiliza la función `LoggedinHandler`, que se encarga de guardar todos los datos necesarios y redireccionar al *endpoint* correspondiente.

Código 1: Implementación de la interfaz para definir las funciones del protocolo OAuth 2.0 para una aplicación cualquiera

```
1  package oauths
2
3  import "net/http"
4
5  type Application interface {
6      LoggedinHandler(http.ResponseWriter, *http.Request, map[string]string,
       string)
7      LoginHandler(http.ResponseWriter, *http.Request)
8      getAccessToken(string, string) string
9      getData(string) map[string]string
10     CallbackHandler(http.ResponseWriter, *http.Request)
11 }
12
```

```
13 type AbstractApplication struct {
14     Application
15 }
```

En el caso de Open Project, el destino al que deben hacerse las llamadas es variable, ya que cada grupo de desarrollo tendrá levantada una instancia del mismo en su red privada. Es por ello, que se ha implementado un sistema en el que podemos guardar el dominio de nuestra instancia de Open Project, utilizando la interfaz de usuario o insertándolo manualmente en el archivo de configuración, y que a través de la función `GetOPuri` se pueda utilizar dicha información. No obstante, para prevenir posibles fallos, esta URL será por defecto `http://localhost:8080`, ya que es la dirección de Open Project cuando es levantado en la misma máquina.

5.2.2. Filtrado de datos desde GitHub para Open Project

Es importante que la información que recibimos desde GitHub se filtre correctamente, ya que todos los datos se reciben en el mismo *endpoint*. Al recibir una llamada desde GitHub, se guarda en memoria el cuerpo del mensaje en formato JSON[6] y este se analiza a través de la función `OpenProjectOptions`, cuyo nombre hace referencia a todas las opciones que se podrán realizar en Open Project con la información recibida desde GitHub.

En dicha función lo primero que se hará será descodificar los datos guardados en bytes para poder leerlos correctamente. En concreto, vamos a buscar si se trata de una *Pull Request* o si se trata de una rama eliminada. En el primer caso, se hará un segundo filtro en función de cuál haya sido el cambio que se ha producido, en cualquier caso se enviará un mensaje a la tarea relacionada con la rama en la que se produce el cambio informando de este, no obstante, cuando se sincronice la rama con la principal se cambiará el estado de la tarea a "cerrado" además de notificar el cambio. En el segundo caso, cuando se cierre una rama, se notificará este suceso en la tarea de Open Project correspondiente.

Por otro lado, también se creará un mensaje automático cuando se creen nuevas tareas en Open Project, donde se genere la URL para crear una *Pull Request* de la rama con los datos correspondientes a la tarea insertados en la *Pull Request*. Así se facilita el trabajo a los desarrolladores y aumenta en gran medida las probabilidades de que dicha petición de revisión se

[6] *JavaScript Object Notation*

cree correctamente y siguiente los estándares de calidad impuestos por el grupo de desarrollo.

5.2.3. Funciones secundarias para Open Project

Una vez visto como se clasifican los datos en la sección 5.2.2, tenemos que conocer como se implementan las funciones que llevan a cabo los procesos explicados anteriormente. Concretamente, hablaremos de las funciones encargadas de escribir los mensajes en Open Project y de la capaz de modificar los estados de las tareas.

La función encargada de enviar mensajes a Open Project funciona de manera que se envíe un *POST* a la instancia de Open Project indicando el identificador de la tarea que se puede obtener desde la información proporcionada por GitHub y en el cuerpo se introduce el mensaje que se desea enviar. Utilizando la clave de acceso que habíamos obtenido mediante el inicio de sesión (sección 5.2.1), se consigue enviar cualquier mensaje que deseemos.

Por otro lado, para modificar el estado de una tarea utilizaremos la siguiente función: `openprojectChangeStatus`. Para conseguir este resultado, de nuevo se utiliza la API de Open Project para hacer la solicitud correspondiente a esta acción. Previamente, necesitaremos conocer la versión de la tarea, por lo que usaremos la función `getLockVersion` que devuelve la identificación de la versión de la tarea. Una vez conocemos la versión, enviaremos la petición escribiendo en el cuerpo del mensaje el identificador del estado que queremos asignar.

Con todo ello, se ha conseguido implementar la componente de la aplicación capaz de interaccionar con Open Project, realizar cambios en las tareas y filtrar los datos que se deben utilizar.

5.3. API REST para la comunicación con GitHub

A continuación, se describirán los procesos que se han seguido en el desarrollo para implementar la componente encargada de crear las conexiones con GitHub, filtrar los datos enviados desde Open Project y modificar los diferentes repositorios del grupo para así conseguir que la aplicación automatice los procesos solicitados por el cliente.

En el caso del inicio de sesión en GitHub se ha seguido una metodología similar a la que se ha explicado en la sección 5.2.1, ya que se vuelve a utilizar la interfaz creada anteriormente para la implementación. No obstante, al contrario que en Open Project no es necesario tener

almacenada la dirección de la instancia, ya que GitHub es un servicio web público. Además, cabe destacar que en GitHub no es posible crear *webhooks* a repositorios o usuarios concretos, es por ello que para la aplicación se utilizarán organizaciones, esto también es coherente, ya que en el grueso de grupos de desarrollo software, son este tipo de grupos los que se utilizan para englobar todos los proyectos en un mismo lugar. Por esta razón es necesaria la implementación de un protocolo *OAuth 2.0* con el que acceder a GitHub con una cuenta que permita administrar estas organizaciones.

Es importante conocer que, para la correcta implementación de muchos procesos, es necesario crear diversos campos personalizados en Open Project en los que almacenar información acerca de las cuentas de GitHub de cada uno de los usuarios, los repositorios asociados a cada tarea o las ramas asociadas. Es por ello, que se utiliza una estructura constante que permite acceder a los campos personalizados, revisados previamente en la aplicación.

Además, durante el desarrollo de esta componente, se han presentado algunos problemas con la API de Open Project, ya que el tiempo de envío de los cambios producidos en la plataforma es bastante elevado y no es posible modificarlo. Por lo que, el tiempo de respuesta de esta componente está directamente relacionado con este insalvable problema.

5.3.1. Filtrado de datos desde Open Project para GitHub

En la función encargada de filtrar la información que se envía desde Open Project se da especial atención a la creación y modificación de tareas, ya que es la forma de asociar cada proyecto a las diferentes ramas que pertenezcan al repositorio.

En la función se recoge en primer lugar la acción que se ha realizado. Cuando esta acción sea la creación de una tarea, en primer lugar se creará la rama en el repositorio en cuestión, siempre que la tarea tenga uno asociado. Posteriormente, se utiliza una rutina para dar permisos de escritura al usuario que realice dicha tarea, mientras que se genera un link que se escribirá en la tarea de Open Project redirigiendo a la página de creación de *Pull Requests* de esa tarea, este link además contiene la información mínima necesaria para que la solicitud de revisión se adecúe a los estándares de buenas prácticas del grupo de desarrollo.

En el caso de que se trate de una modificación de una tarea ya existente, se revisará si la actualización se produce en el estado de la misma, para así modificar los permisos de escritura de los usuarios cuando la tarea se cierre.

Por último, en el caso de que no se encontrase repositorio asociado, estos protocolos no serían necesarios, por lo que simplemente se notificará al usuario el cambio mediante la consola de la aplicación.

5.3.2. Creación automática de ramas

Una de las funciones esenciales de la aplicación es la creación de nuevas ramas para cada una de las tareas de Open Project que lo necesiten. Es entonces, donde es necesaria una función que se encargue de ello. Esta se divide en dos partes. Una primera función que sirve de antesala y se encarga de desgranar los datos de Open Project para utilizar únicamente los necesarios y una segunda función que usa estos datos para realizar la llamada a la API de GitHub.

En esta función se reciben como parámetros: la clave de acceso, el repositorio, la organización, el nombre de la rama deseado y el de la rama objetivo. En primer lugar, se obtiene el identificador de la rama objetivo que será la base para la nueva rama. A continuación, se crea el cuerpo del mensaje indicando el nombre de la nueva rama y desde la que se crea, cuyo identificador hemos obtenido mediante la función `getLastcommit`. Finalmente, se realiza el envío de información a la API de GitHub que se encarga de crear la nueva rama en el repositorio asignado.

Se devolverá el entero que identifica el estado de la respuesta de la API de GitHub para controlar que todo el proceso haya salido correctamente.

5.3.3. Control automático de permisos

La segunda función esencial del proyecto es la asignación automática de permisos para mantener el principio de mínimo privilegio. De nuevo este proceso utiliza una primera función que obtendrá los datos y una segunda que se encarga de realizar la llamada.

En la primera función se obtienen los datos de la organización de GitHub asociada, el repositorio asignado, el identificador del usuario de GitHub asociado a la tarea al que se le aplicará la actualización, el cambio que se desea realizar y la clave de acceso. Todos estos datos los recibirá la segunda función y los insertará en la llamada a la API de GitHub. Cabe destacar que se ha utilizado un enumerado para estas funciones denominado `permission` y que contiene las cadenas de caracteres que GitHub interpreta como permisos de escritura o lectura.

Finalmente, se devuelve el estado de la respuesta de la API de GitHub y se escribe un mensaje en la consola de la aplicación indicando el resultado del proceso.

5.3.4. Creación automática de *webhooks*

Otra funcionalidad que se ha querido implementar en la aplicación es la creación de *webhooks*. Al contrario que con la API de Open Project, la de GitHub sí que nos permite crear la conexión automáticamente. Es por esto que mediante un botón de la interfaz o el *endpoint* "`/github/webhook`" en la aplicación, se puede crear un webhook con la organización de GitHub que se desee. En la interfaz de usuario se solicitará insertar el nombre de la organización, mientras que en el *endpoint* se envía en el cuerpo de la llamada.

La función encargada de este proceso es `githubWebhook`. En esta función se reciben las llamadas que se reciben en el *endpoint* mencionado anteriormente y del que también hace uso la interfaz de usuario. Al llegar el mensaje, se recogen los datos de este y se crea el cuerpo con el que se indicará a GitHub como debe crearse el *webhook*. Una vez hecho esto, se realiza la solicitud a la API de GitHub y se recoge la respuesta de esta. Esta respuesta se mostrará por pantalla cuando se trate de la interfaz web y además dejará un rastro en la consola de la aplicación indicando el resultado de la operación.

5.4. Sincronización manual

Otra funcionalidad que ofrece la aplicación es realizar una sincronización manual, en lugar de depender de cuando se produzcan cambios en las aplicaciones conectadas. Este proceso es importante para permitir corregir errores o acelerar un proceso cuando sea necesario. A esta función se puede acceder a través de la interfaz de usuario o utilizando el *endpoint* "`/api/refresh`".

El proceso comienza accediendo a través de un *proxy* que se encarga de revisar la última fecha y hora en la que se realizó la sincronización manual, que está guardada en el archivo JSON, donde se almacenan los datos necesarios para la aplicación. Esto es así para mejorar el rendimiento de la función, evitando revisar elementos cuya fecha de modificación sea anterior a la que se ha recogido, es decir, reduciendo el número de tareas a revisar por la aplicación. Si nunca se ha realizado una sincronización, la aplicación hará una primera revisión de todas las

tareas.

Una vez se conoce la fecha desde la que parte la sincronización, se traslada esta información a la función que será la encargada de llevar a cabo el proceso de revisión.

En primer lugar, la función comprueba que los campos personalizados de Open Project existen y son correctos. A continuación, se recoge la lista de tareas de Open Project que hayan sido modificadas tras la fecha de la última sincronización.

Posteriormente, se recorre la lista de tareas una a una, cada vez que una de ellas tenga un repositorio asociado, se eliminarán los permisos de escritura de todos los participantes del mismo; cabe destacar que esto solo ocurre la primera vez que se revisa dicho repositorio, el resto de veces simplemente se dará permiso de escritura a los usuarios que tengan tareas asociadas. Además, si se descubre alguna tarea con repositorio asociado pero que no tenga una rama asociada, se creará dicha rama en el repositorio en cuestión. Algunos de los procesos que se producen durante la sincronización, tales como la obtención de los usuarios de un repositorio, se han subdividido en funciones privadas para facilitar la lectura y actualización del código.

Finalmente, se obtiene la fecha actual y se sustituye la fecha de última modificación por esta, finalizando el proceso de sincronización enviando un mensaje notificando el resultado del proceso.

Todo este proceso se encuentra descrito en el pseudocódigo 2, donde podemos ver el orden de realización de los diferentes procesos y donde se puede analizar el impacto de la función sobre el rendimiento de la aplicación.

Gracias a esta función se consigue que el usuario pueda hacer una sincronización total entre ambas aplicaciones rápidamente y siempre que lo necesite.

Código 2: Función que realiza la sincronización entre Open Project y GitHub

```
1 func Refresh(lastRefresh time.Time) {
2   CheckCustomFields() // Función que revisa los campos personalizados en
     Open Project
3   . . .
4
5   // ====== Obtención de la lista de paquetes desde la última
     sincronización ======
6
```

```
7     . . .
8   page_size := 1000
9   req_url := OP_url + // URL de Open Project
10    `/api/v3/work_packages?filters=%5B%7B%22updatedAt%22:%7B%22operator
    %22:%22%3C%3Ed%22,%22values%22:%5B%22` + // Transcripción a lenguaje
    natural -> /api/v3/work_packages?filters=[{"updatedAt":{"operator":"<>d
    ","values":["
11    lastRefresh.Format("2006-01-02T15:04:05Z") + // Fecha de última
    sincronización
12    `%22,%20%22` + // Transcripción a lenguaje natural -> ", "
13    time.Now().Format("2006-01-02T15:04:05Z") + // Fecha actual
14    `%22%5D%7D%7D%5D&pageSize=` + // Transcripción a lenguaje natural ->
    "]}}]&pageSize=
15    strconv.Itoa(page_size) // Número de paquetes mostrados por página, por
     defecto se muestran 1000 (máx)
16
17  [Definición de cabeceras]
18  response, _ := http.Do(request)
19  if response.StatusCode is ERROR {
20    return ERROR
21  }
22    . . .
23  if nº de paquetes == 0 {
24    return SINCRONIZACIÓN COMPLETADA
25  }
26
27  // ====== Para cada paquete dar o quitar permisos para los usuarios que
     lo necesiten ======
28
29  for lista_paquetes {
30        [Obtener repositorio del paquete]
31
32    If Repository == "" {
33
34        [Continuar al siguiente]
35
36    } else {
```

```
37      r := strings.Split(repoURL, "/")
38      repoName := r[len(r)-1]
39      org := r[len(r)-2]
40      if Repositorio no se ha actualizado {
41
42        [Obtener colaboradores del repositorio]
43
44        for colaboradores {
45                        . . .
46          givePermissionREAD(colaborador.user)    // Dar permiso de
    escritura a todos los colaboradores del repositorio no modificado
47        }
48
49        [Guardar el repositorio actualizado]
50      }
51            . . .
52      // If task branch does not exist, create a new one
53      if rama no existe {
54        go createBranch(gh_token, repoName, org, source, target) // Crear
    una nueva rama
55      }
56
57      [Obtener el usuario asignado a la tarea]
58
59      if usuario existe {¡
60                . . .
61        givePermissionWRITE() // Dar permisos de escritura a los usuarios
    con tarea
62      }
63    }
64  }
65
66    [Guardar la fecha y hora actual como fecha y hora de última
    sincronización]
67    return SINCRONIZACIÓN COMPLETADA
68 }
```

5.5. Interfaz de usuario

Para la interfaz de usuario se ha optado por hacer una web de configuración donde sea más sencillo para el usuario configurar la aplicación y visualizar los cambios que se produzcan. Para facilitar el uso de la interfaz se ha utilizado una disposición en ventanas donde la configuración para cada aplicación se encuentra en una página diferente. Además, la sincronización manual es de fácil acceso desde cualquier ventana, ya que se sitúa en la cabecera de la interfaz, al igual que un pequeño menú donde poder copiar automáticamente las direcciones más utilizadas.

En la figura 7 se observa la ventana de configuración de Open Project. En esta se puede iniciar sesión en la cuenta de Open Project que deseemos utilizando el botón designado y, además, existe un campo de texto para insertar la URL de la instancia de Open Project que queramos, por defecto, esta tomará el valor `http://localhost:8080` dirección que tomaría Open Project en local. Esto se consigue gracias a la función `saveOPurl` que recibe la llamada que se hace desde la interfaz y guarda la URL insertada en el archivo correspondiente.

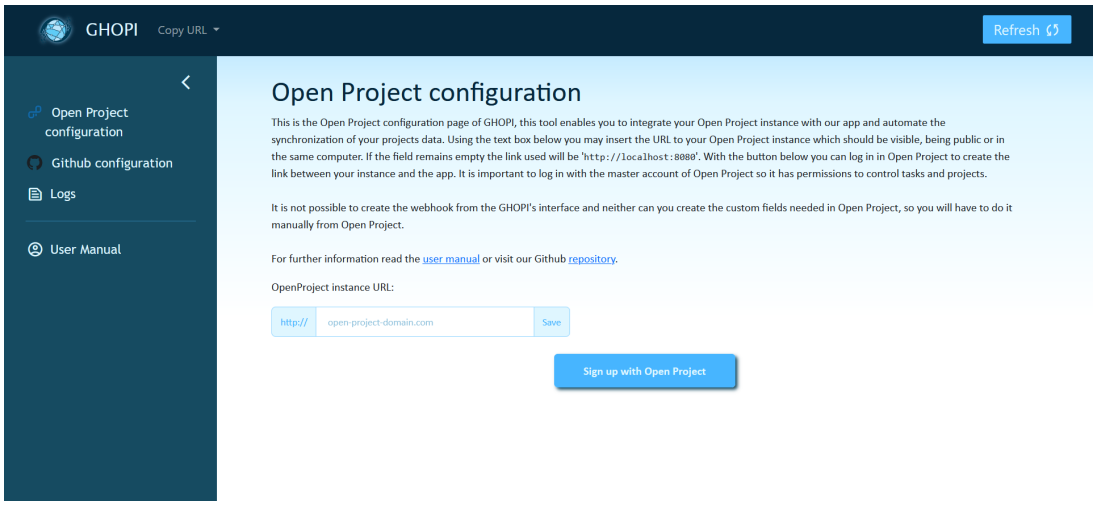

Figura 7: Interfaz de configuración de Open Project en GHOPI

Por otra parte, en la figura 8 observamos la ventana de configuración de GitHub en la que podemos ver tanto el botón de inicio de sesión, como el botón para crear un nuevo *webhook*, ambos ya explicados en la sección 5.3. Al utilizar, el segundo botón aparece una ventana de diálogo donde se nos permitirá insertar el nombre de nuestra organización y así crear el enlace.

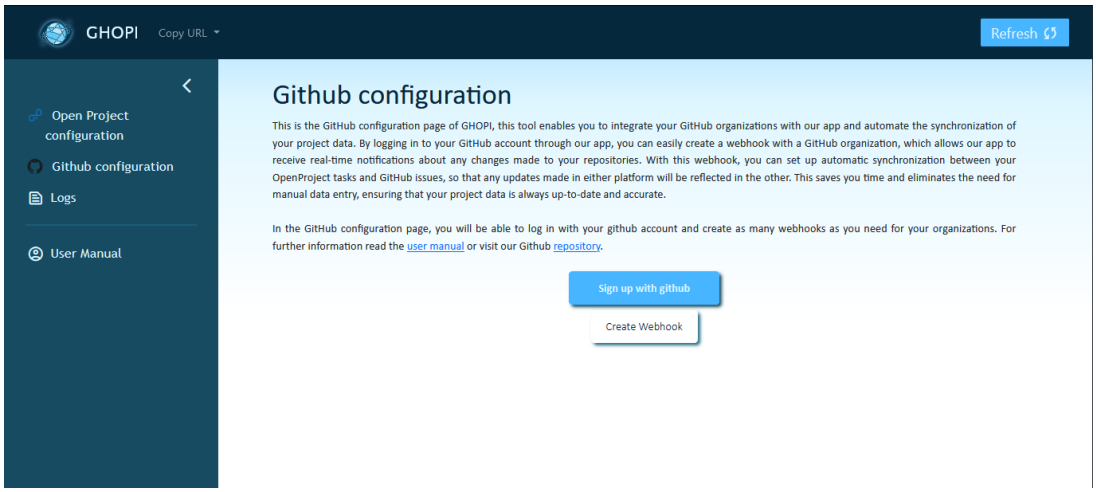

Figura 8: Interfaz de configuración de Github en GHOPI

Por último, en la figura 9 se observa como en la interfaz web se puede visualizar la consola de la aplicación, donde se ven todos los mensajes y errores que han podido suceder. Además, mediante la interfaz se permite hacer un filtro de dicha consola, permitiendo así ver únicamente los mensajes que nos interesen. Por ejemplo, si estamos buscando los fallos que han sucedido durante un día concreto del año, podemos filtrar por errores e indicar la fecha del día en cuestión. Una vez que se quiera deshacer los filtros se puede utilizar el botón de reseteo designado a ello para que todo vuelva al estado predeterminado.

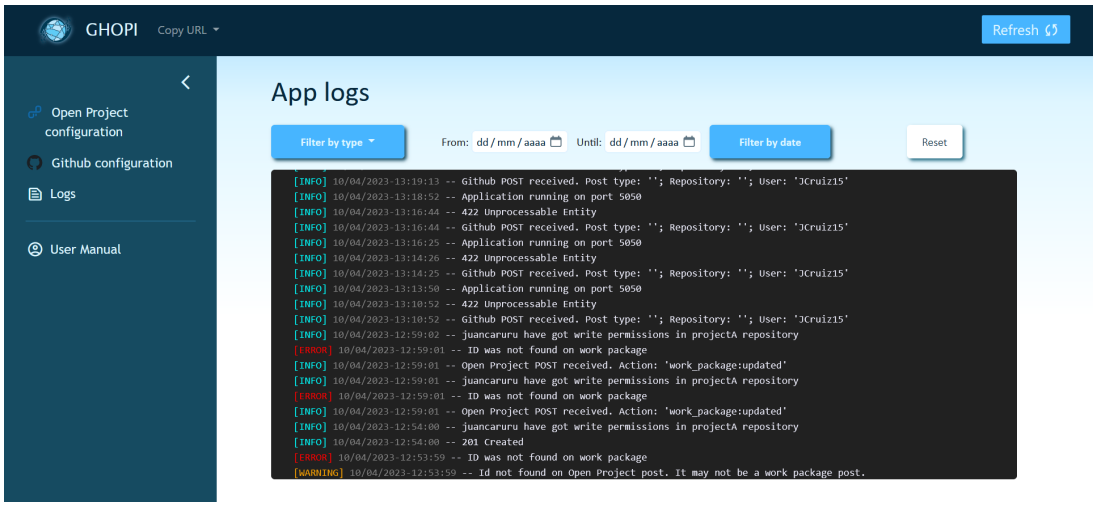

Figura 9: Interfaz de visualización de la consola de GHOPI

5.6. Evaluación de la aplicación resultante

Una vez completada la implementación de la aplicación, es útil hacer una verificación del código implementado. Durante el desarrollo de la misma se han realizado múltiples pruebas de integración, comprobando que todas las componentes que se han producido interactúan correctamente entre sí. Entre otras pruebas, se ha revisado que la interacción entre la creación de tareas y la creación de ramas se realiza correctamente de manera que una acción desencadene la siguiente.

También se han realizado diversas pruebas unitarias para comprobar que las funciones individualmente están correctamente implementadas. Debido a la planificación del proyecto, estas revisiones han sido pruebas de "caja blanca", ya que se han realizado tras haber completado el desarrollo del código. Se han utilizado estas pruebas para comprobar que las funciones encargadas de tareas menores funcionasen correctamente, por ejemplo, se han realizado pruebas sobre una función que se encarga de realizar la comprobación de los errores mientras la aplicación está en uso y de notificarlos en la consola de la aplicación, revisando así que ante cualquier situación la función responda como esperamos. Gracias a este tipo de pruebas, se consigue que el código sea más robusto y además, permite detectar flaquezas en la implementación que pueden arreglarse haciendo una refactorización del mismo.

Una vez realizadas estas pruebas individuales, se ha contactado con el grupo de investigación Khaos Research [18] para que realizaran las pruebas *alpha* del proyecto. Esto quiere decir, que utilizaron la aplicación en una copia de su instancia de Open Project y dedicaron algunos sprints a utilizar con algunos desarrolladores esta copia, probando así la usabilidad de la aplicación. Con este tipo de pruebas se consigue información mucho más valiosa acerca de la aplicación, ya que se trata de un entorno real de trabajo donde se pueden aprovechar las funcionalidades de la misma. Gracias a esta información se puede ampliar la funcionalidad de la aplicación, corregir errores que puedan aparecer y mejorar la experiencia de uso utilizando los comentarios recibidos.

Finalmente, con las pruebas realizadas se puede concluir la implementación de la aplicación y, por tanto, el desarrollo de la misma estaría completo. Todo el código desarrollado se encuentra disponible en el repositorio público de GitHub con el mismo nombre que la aplicación [25].

6

Conclusiones y Líneas Futuras de investigación

6.1. Conclusiones y resumen de los resultados

Podemos concluir el proyecto como un éxito, ya que se han cumplido los requisitos solicitados por el grupo de investigación Khaos Research [18]. La aplicación resultante es completamente funcional y se encuentra disponible en su repositorio de GitHub asociado [25]. La sincronización entre ambas plataformas se realiza de manera eficiente y efectiva, lo que ayuda a los desarrolladores a centrarse en sus tareas, obviando así la necesidad de mantener ambas aplicaciones actualizadas. Por otro lado, la automatización de procesos repetitivos permite realmente aumentar la productividad de los grupos de investigación, además de prevenir la duplicación de datos o peticiones incorrectas de revisión.

También, gracias a este proyecto se ha conseguido aprender un nuevo lenguaje con el que es posible no solo crear API o aplicaciones web, sino además algoritmos eficientes que utilicen la concurrencia. Por otro lado, el uso de tecnologías web, tales como las API, ha permitido conocer en gran profundidad el funcionamiento del internet actual y así permitir una mayor adaptabilidad en la programación de este campo.

Cabe destacar, sin embargo, que durante el desarrollo del proyecto se han presentado multitud de problemas a raíz de la naturaleza del mismo. Tales como la inexistencia de determinadas direcciones en la API de Open Project que hubieran facilitado la implementación de algunos procesos o incluso no lo han permitido, como puede ser el caso de la creación del webhook automáticamente con esta plataforma; o el alto tiempo de respuesta que ofrece la API de Open Project a la hora de enviar paquetes de información. Por otro lado, la metodología utilizada,

aun siendo eficaz, no ha sido eficiente, ya que la planificación de tareas individualmente requería más tiempo del necesario y algunas ocasiones no se definían correctamente los intervalos de tiempo necesarios para cada una de ellas. Todos estos problemas no han supuesto un gran impedimento para la correcta implementación de las componentes del proyecto, ya que en muchos casos, aun siendo insalvables, no comprometían ninguno de los requisitos definidos en la sección 4.1.

En comparativa, esta aplicación cubre más áreas que las que ofrece la extensión de Open Project mencionada en el apartado 1.1. Este complemento simplemente notifica a los usuarios los cambios producidos en GitHub, sin embargo, GHOPI va más allá y consigue modificar los repositorios que sean necesarios para cada tarea e incluso gestionar los permisos individuales de cada usuario. Esto no solo mejora la productividad, sino que permite el correcto cumplimiento de las buenas prácticas del desarrollo software.

6.2. Líneas Futuras de investigación

Este proyecto, pese a ser un producto finalizado y utilizable, aún puede verse actualizado siguiendo diferentes líneas de investigación.

En primer lugar, se puede hacer un análisis exhausto del impacto de esta aplicación en los grupos de desarrollo para estudiar el aumento de productividad de los mismos. Esto sería útil, ya que nos permite conocer el verdadero impacto sobre la eficiencia en el desarrollo y así avanzar aún más en el desarrollo de este tipo de tecnologías de apoyo.

Otra línea de investigación que podría continuar sería la implementación de nuevas funcionalidades que permitan una mayor conectividad entre las aplicaciones. Inclusive, se podría ampliar las funcionalidades añadiendo nuevos gestores de versiones, tales como GitLab o Bitbucket. La implementación de nuevas funcionalidades en un futuro aseguraría el mantenimiento de la aplicación y aumentaría las posibilidades de se extienda su uso en más grupos de desarrollo software.

También se podrían estudiar nuevas formas de optimizar el rendimiento y la escalabilidad del proyecto. Aplicando técnicas de refactorización y de optimización para asegurar el correcto incluso con una gran cantidad de datos y usuarios.

Asimismo, el proyecto aún puede continuar en el tiempo, ya que permite multitud de nuevas líneas de investigación que podrían mejorar el resultado ya obtenido.

Referencias

[1] *About SourceForge*. URL: https://sourceforge.net/about.

[2] Adetokunbo Adenowo, Adetokunbo AA Adenowo y Basirat A Adenowo. «Software Engineering Methodologies: A Review of the Waterfall Model and Object-Oriented Approach». En: *International Journal of Scientific and Engineering Research* 4.7 (2013). ISSN: 2229-5518. URL: http://www.ijser.org.

[3] *Asana*. URL: https://asana.com/es.

[4] *Basecamp*. URL: https://basecamp.com/.

[5] Kent Beck et al. *Manifesto for Agile Software Development*. Inf. téc. 2001. URL: http://agilemanifesto.org/.

[6] *Bitbucket Overview*. URL: https://bitbucket.org/product/guides/getting-started/overview#a-brief-overview-of-bitbucket.

[7] Torgeir Dingsøyr et al. «A decade of agile methodologies: Towards explaining agile software development». En: *Journal of Systems and Software* 85.6 (2012), págs. 1213-1221. ISSN: 01641212. DOI: 10.1016/J.JSS.2012.02.033.

[8] *Effective Go - The Go Programming Language*. URL: https://go.dev/doc/effective_go.

[9] *FastAPI documentation*. URL: https://fastapi.tiangolo.com/.

[10] *Flask Documentation (2.3.x)*. URL: https://flask.palletsprojects.com/en/2.3.x/.

[11] *Gantt charts*. URL: https://www.openproject.org/docs/user-guide/gantt-chart/.

[12] *GitHub*. URL: https://github.com/.

[13] *Github for Dummies: What Is GitHub and How To Leverage Its Toolkit*. URL: https://www.springboard.com/blog/software-engineering/github-for-dummies/.

[14] *GitLab Documentation*. URL: https://docs.gitlab.com/.

[15] Daniel Glez-Pen‹ et al. «Web scraping technologies in an API world». En: (). DOI: 10.1093/bib/bbt026. URL: http://hc..

[16] IEEE. *Systems and software engineering — Vocabulary*. Inf. téc. IEEE, sep. de 2017. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8016712&tag=1.

[17] *Jira*. URL: https://www.atlassian.com/es/software/jira.

[18] *Khaos research Group*. URL: https://khaos.uma.es/.

[19] Xiaopu Ma et al. «Specifying and enforcing the principle of least privilege in role-based access control». En: *Concurrency and Computation: Practice and Experience* 23.12 (ago. de 2011), págs. 1313-1331. ISSN: 15320634. DOI: 10.1002/CPE.1731.

[20] Robert M Mcclure. «NATO SOFTWARE ENGINEERING CONFERENCE 1968». En: (2001).

[21] *Microsoft Project*. URL: https://www.microsoft.com/es-es/microsoft-365/project/project-management-software.

[22] *Openproject API documentation*. URL: https://www.openproject.org/docs/api/.

[23] *Python documentation*. URL: https://docs.python.org/3/.

[24] Linda Rising y Norman S. Janoff. «Scrum software development process for small teams». En: *IEEE Software* 17.4 (jul. de 2000), págs. 26-32. ISSN: 07407459. DOI: 10.1109/52.854065.

[25] Juan Carlos Ruiz Ruiz. *GHOPI*. 2023. URL: https://github.com/JCruiz15/GHOPI.

[26] SendGrid Team. *What's a Webhook? | SendGrid*. URL: https://sendgrid.com/blog/whats-webhook/.

[27] *Trello*. URL: https://trello.com/es.

[28] *What is Scrum? - 5D Vision - Agile Product Innovation*. URL: https://www.5dvision.com/post/what-is-scrum/.

[29] Niklaus Wirth. «A Brief History of Software Engineering». En: *IEEE Annals of the History of Computing* 30.3 (2008), págs. 32-39. ISSN: 10586180. DOI: 10.1109/MAHC.2008.33.

Apéndice A
Guía de instalación

Esta es la guía de instalación de GHOPI. Tendrá como objetivo explicar cómo descargar y configurar las diferentes componentes de la aplicación, además de la configuración de GitHub y Open Project para levantar la aplicación. En la sección A.1 se comentarán los requisitos necesarios para utilizar la aplicación y en la sección A.2 se explicará la instalación de la aplicación paso a paso en los sistemas operativos más utilizados.

Una vez realizada la instalación de la aplicación, en la propia interfaz web se podrá acceder un manual de usuario donde se explica la usabilidad de la interfaz y la configuración de algunas funcionalidades de la aplicación.

A.1. Prerrequisitos

- Versión 1.19.1 o superior de Go.
- Software capaz de hacer visible la dirección de la aplicación, véase *ngrok* o *Heroku* entre otros.

A.2. Guía de instalación

A.2.1. Windows

Para instalar esta aplicación en Windows, simplemente descargue el archivo ejecutable `GHOPI.exe` y complete el archivo `.env` según se explica en la sección A.2.3.

Luego, utilice el software seleccionado para levantar la aplicación de forma pública.

A.2.2. Linux y macOS

Para instalar esta aplicación en computadoras Unix, clone el repositorio de GHOPI[25] en tu computadora y ejecuta el siguiente comando:

```
1 go build main.go
```

Lo cual creará un archivo ejecutable para usar la aplicación. Una vez completado el archivo de credenciales (`.env`) explicado en la sección A.2.3, utiliza el software seleccionado para levantar la aplicación de forma pública.

A.2.3. Creando el archivo .env

Para que esta aplicación funcione correctamente, primero se debe configurar un archivo `.env`, que debe incluir el ID de cliente y el ID secreto de GitHub y Open Project para poder iniciar sesión en ellos. Este archivo `.env` debe estar en la misma carpeta que la aplicación GHOPI.exe o en la carpeta raíz del proyecto.

Open Project En su instancia de Open Project, vaya a *Administración >Autenticación >Aplicaciones OAuth*, donde se puede agregar una nueva aplicación presionando el botón "*Add*".

En la página de configuración abierta, complete los espacios con la información de GHOPI. En la URI de redireccionamiento, asegúrese de escribir *https://tu-url-de-la-aplicacion/op/login/callback*, donde *https://tu-url-de-la-aplicacion* es la ruta de GHOPI que **debe** ser pública.

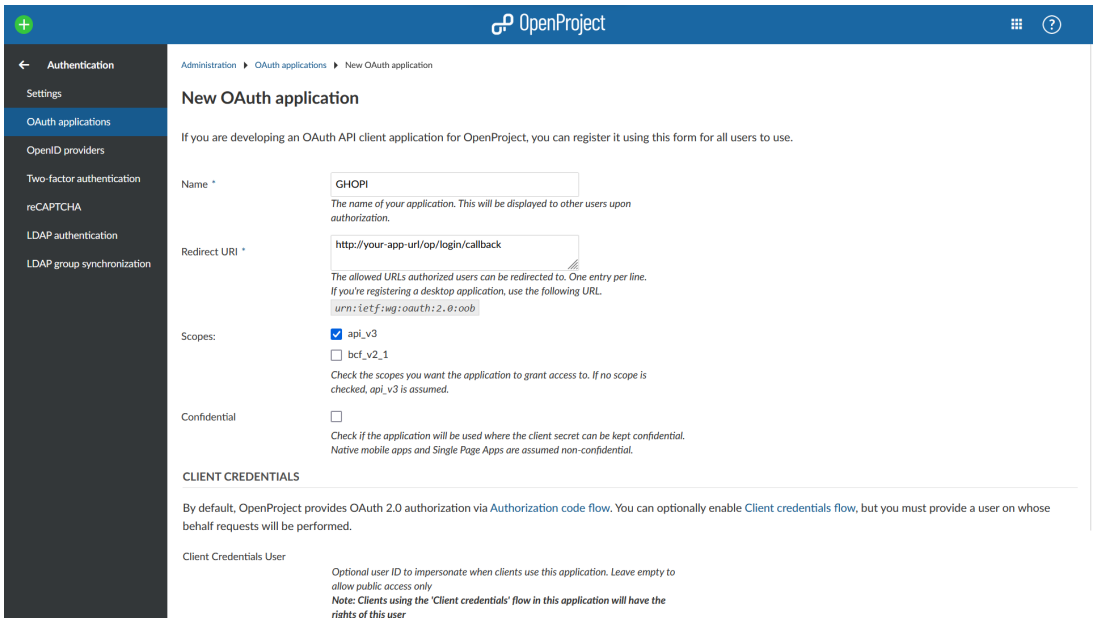

Figura 10: Creación de OAuth app en Open Project

Una vez que se haya creado, se mostrarán las credenciales, y deberá guardar el ID de cliente (*Client ID*) y el secreto de cliente (*Client secret*) en el archivo `.env` con los nombres: `OPENPROJECT_CLIENTID` y `OPENPROJECT_SECRETID`.

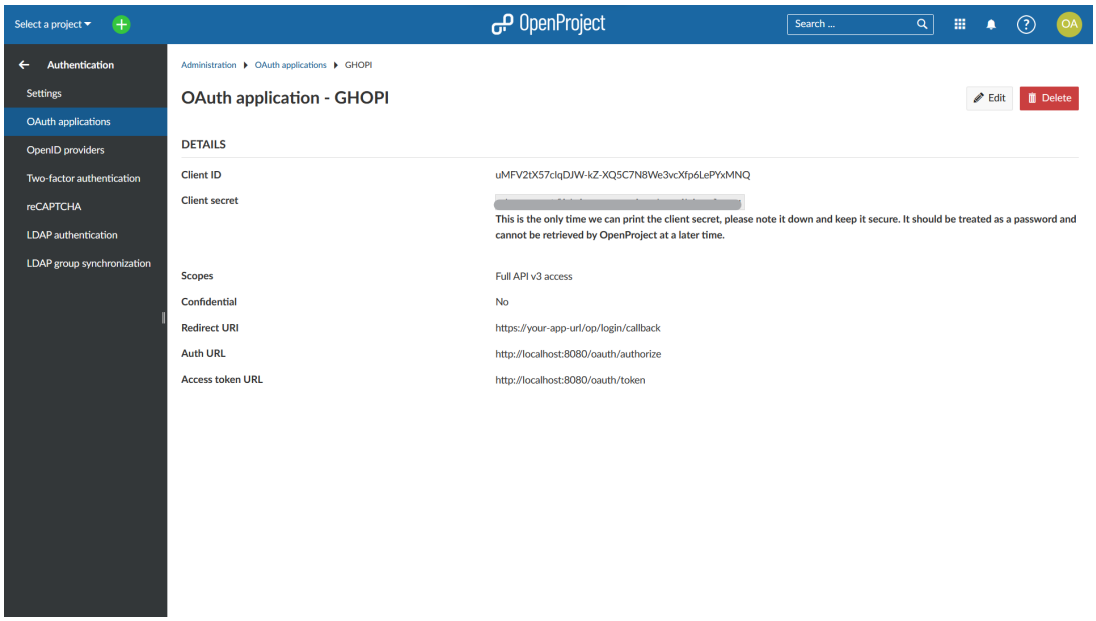

Figura 11: Credenciales del OAuth Open Project

GitHub Para configurar la autenticación OAuth de GitHub, vaya a *Configuración >Configuración de desarrollador >Aplicaciones OAuth* y haga clic en el botón "*New OAuth app*".

En la página de configuración, debe completar el campo "URL de inicio" (*Homepage URL*) con *https://tu-url-de-la-aplicacion* y el campo "URL de devolución de autorización" (*Authorization callback URL*) con *https://tu-url-de-la-aplicacion/github/login/callback*, donde *https://tu-url-de-la-aplicacion* es la ruta de GHOPI que **debe** ser pública.

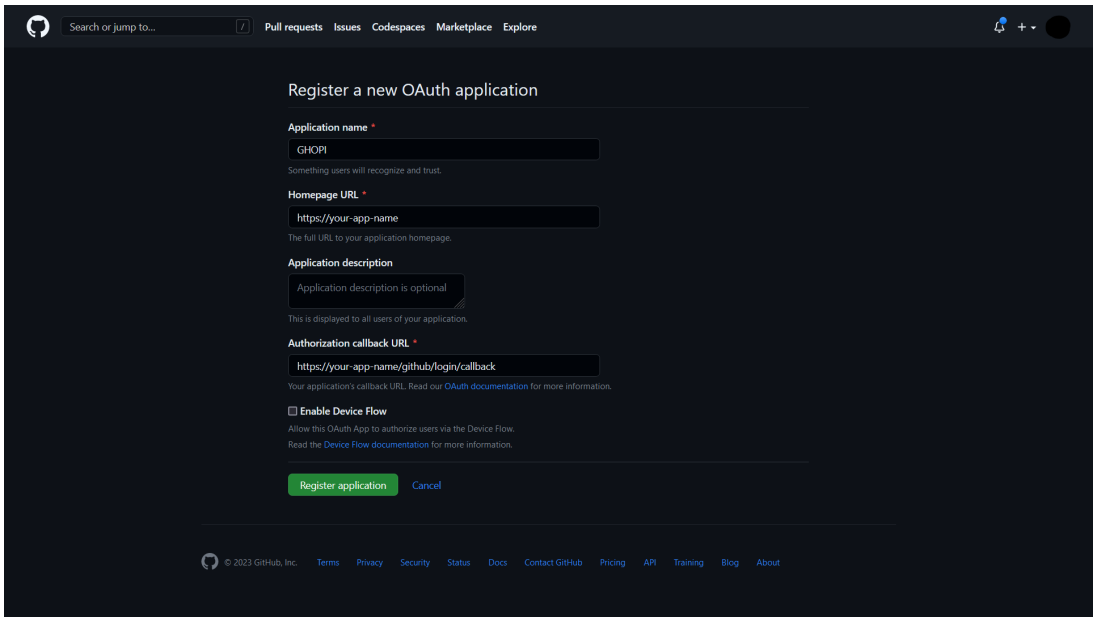

Figura 12: Creación de OAuth app en GitHub

Una vez que se haya creado la aplicación, se generarán nuevos *Client ID* y *Secret client* que deberá guardar en el archivo `.env` con los nombres: `GITHUB_CLIENTID` y `GITHUB_SECRETID`.

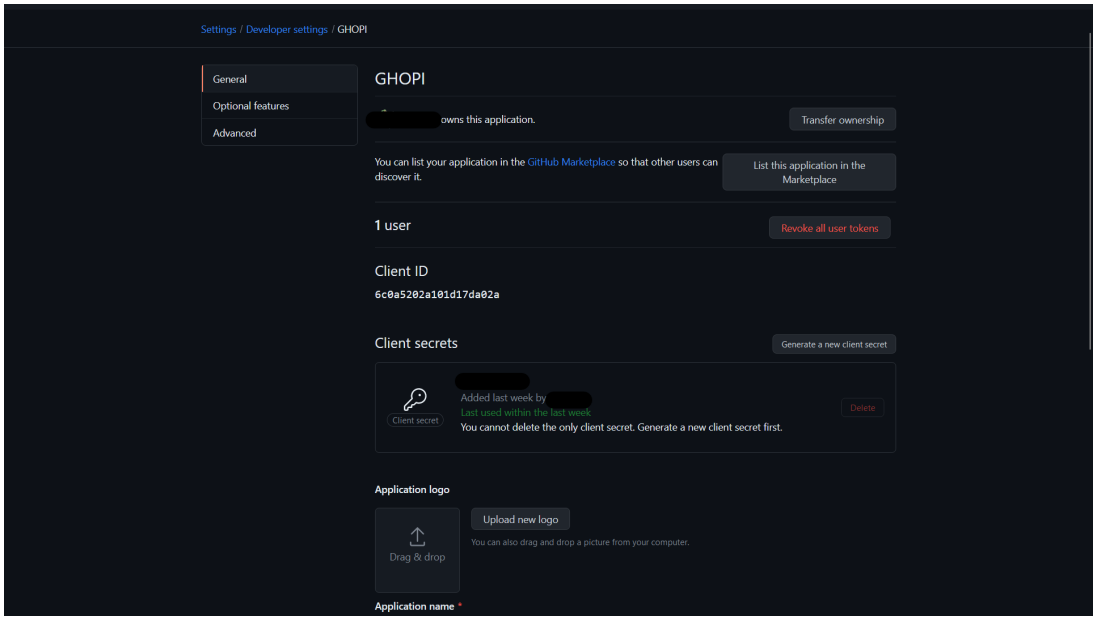

Figura 13: Credenciales del OAuth GitHub

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA