UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

Trabajo final del GºIng.Informática:

Evalúa y compara la actividad del repositorio en GitHub

Presentado por Roberto Luquero Peñacoba
en junio de 2019
Tutor Carlos López Nozal

Índice de contenido

Índice de ilustraciones

Índice de tablas

I - Apéndice A. Plan de Proyecto Software

1. Introducción

Todo proyecto que se quiera llevar a cabo debe de iniciarse mediante una planificación adecuada que permita definir prioridades sobre las funcionalidad que ha de incluir, el asignamiento de prioridades a las mismas, etc. y establecer una estrategia que logre mejorar la calidad del producto final.

Es en esta fase donde también debe analizarse cada parte del proyecto y realizar un cálculo de los recursos necesarios para su desarrollo, así como los costes previstos.

En este proyecto esta fase se ha dividido en dos partes:

- **Planificación temporal:** en el que se detalla cada uno de los sprints planeados, las tareas asignadas a esos sprints y el tiempo estimado en completar cada una de ellas.
- **Estudio de viabilidad:** donde se va a analizar tanto la viabilidad económica como la legalidad derivada de utilizar las distintas librerias y herramientas para su desarrollo.

2. Planificación temporal

Desde el inicio del proyecto se decidió aplicar la metodología ágil SCRUM, para permitir un continuo desarrollo del proyecto en el que estar informados acerca de los cambios aplicados y que se dispusiera de suficiente flexibilidad para poder dar con soluciones que solventaran los posibles problemas que se encontrasen.

A pesar de decidir utilizar esta metodología, no se ha podido aplicar completamente debido a la naturaleza educativa del proyecto, y a ser de carácter individual, es decir, no se han realizado reuniones diarias comentando el trabajo del día anterior o no se ha realizado entre varias personas.

Con todo ello, se ha seguido la línea de una metodología ágil mediante el desarrollo iterativo e incremental del código del proyecto.

Se determinó que la duración de cada uno de los sprints iba a ser de aproximadamente dos semanas y tras finalizar ese periodo de tiempo, se realizaría una nueva reunión en la que comentar y comprobar el trabajo realizado, declarar las nuevas tareas a realizar, estimar los tiempos dedicados a cada una mediante la herramienta ZenHub y finalmente asignar prioridades a cada una de ellas.

En líneas generales, esta asignación de prioridades se ha basado en tres puntos:

- Si una nueva tarea compartía proceso con la última realizada, se le asignaba una prioridad alta, para continuar con las mismas ideas aplicadas a la tarea del sprint anterior.
- Si la tarea era independiente o el proceso mental necesario para llevarla a cabo variaba mucho de las últimas tareas realizadas se le asignaba una prioridad baja.
- Finalmente, entre las resultantes se escogían por orden de tiempo estimado, de menos a más tiempo.

Como escala de tiempo estimado se han tomado los story points que ofrece ZenHub, asignando a cada punto una duración aproximada de 30 minutos.

Los diferentes sprints realizados han sido los detallados a continuación.

2.1. Sprint 1

Este primer sprint se basó principalmente en una explicación de lo que se quería lograr con el proyecto y en plantear una serie de posibles herramientas a utilizar para alcanzar ese objetivo. [1]

El sprint comenzó el día 2 de febrero y terminó el 17 del mismo mes, pero no fue hasta ese mismo día que se creo la milestone que deja constancia del mismo en GitHub debido a que olvidé la creación del mismo y sus issues.

La documentación sobre las distintas herramientas y la valoración sobre si utilizar unas u otras es en lo que se empleó todo el tiempo.

La estimación de tiempo fue de 3 story points por cada uno de los 4 issues, lo que da un total de 6 horas, aunque finalmente se emplearon 5 debido a que una de las valoraciones estuvo clara desde el principio (utilizar JavaFX frente a Java Swing).

2.2. Sprint 2

En este segundo sprint se realizó la instalación del proyecto de forma local en el entorno de desarrollo Eclipse, debido a que ha sido el utilizado a lo largo de la carrera. También se comenzó con la integración continua. [2]

El sprint comenzó el día 17 de febrero (aunque de nuevo no se creó la milestone hasta el día 20 debido a la realización de otras tareas externas al proyecto) y terminó el 27 de frebrero, por lo que constituye el sprint más corto realizado en el proyecto.

El tiempo dedicado al proyecto durante este sprint se basó primeramente en la correcta instalación del proyecto en Eclipse y el comenzar con la integración continua, que se logró parcialmente (únicamente se logró con Codacy y Ant en vez de con los 4 previstos) debido a un problema en la estructura del proyecto. El resto del tiempo se dedicó a, gracias a la integración conseguida con Codacy y su detección de defectos, refactorizar parte del código de Activiti-Api que iba a permanecer en el proyecto.

La estimación de tiempo fue de 15 story points, es decir, 7 horas y media. Finalmente se acabó empleando alrededor de 14 horas, debido a los problemas que aparecieron al realizar la integración continua.

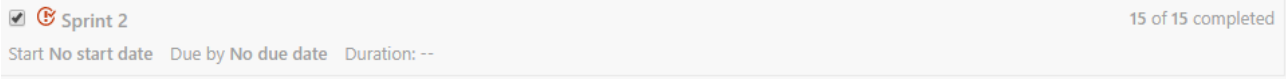


Ilustración 1: Story points estimados para el sprint 2.

2.3. Sprint 3

El tercer sprint tuvo de nuevo poco progreso en cuanto al código del proyecto y fue utilizado de nuevo para evaluar una nueva alternativa e intentar terminar la integración continua. [3]

Comenzó el día 27 de febrero y termino el 13 de marzo, una duración de 14 días.

Durante este sprint se dedicó la mayor parte del tiempo a realizar una serie de pequeños proyectos con Vaadin, para terminar de decidir si crear una aplicación web o una de escritorio. Debido a que la mayor parte de la experiencia obtenida durante la carrera ha consistido en el desarrollo de software de escritorio, se decidió continuar por ese camino.

El resto de tareas consistió en el inicio de la Wiki del proyecto (existente en el propio repositorio GitHub) y la correción de los distintos bugs encontrados en la programación del build.xml de Apache Ant.

Se estimó que las tareas llevarían alrededor de 10 horas y media (21 story points) y finalmente se emplearon 14, debido principalmente a la dificultad encontrada para comprender el funcionamiento de Vaadin y el uso de sistemas distribuidos.

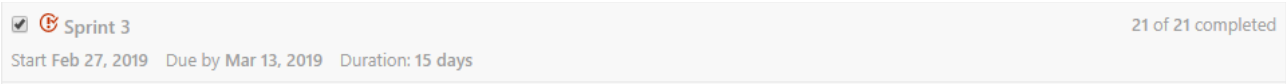


Ilustración 2: Story points estimados para el sprint 3.

2.4. Sprint 4

De nuevo, otro sprint basado en la integración continua y la generación de documentación sobre el proyecto, pero que también amplió la cobertura de los tests creando algunos nuevos. [4]

Este sprint comenzó el 13 de marzo y finalizó el 28 del mismo mes.

Una gran parte del sprint se dedicó a la correción de los bugs encontrados en la integración continua, recayendo la mayor parte en la reestructuración del proyecto en GitHub y la correcta programación de los nuevos test y de la tarea para ejecutar los tests automáticamente en Ant.

Fue en este sprint también donde se decidió no utilizar codecov para utilizar únicamente Codacy, puesto que también tiene una parte en la que mostrar la cobertura de los tests. Además de lo mencionado, también se dedico algo de tiempo a continuar con la documentación sobre el proyecto escrita en la wiki, así como la generación de una nueva distribucción del proyecto para comprobar que las refactorizaciones realizadas sobre el código inicial no había modificado su funcionalidad.

En total se estimó que se necesitaría alrededor de 14 horas (28 story points) para la realización de la tareas que en realidad duraron alrededor de 12 puesto que algunas de ellas resultaron ser más fáciles de completar que lo esperado.

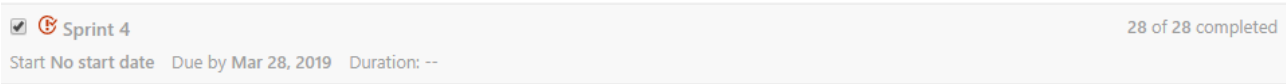


Ilustración 3: Story points estimados para el sprint 4.

2.5. Sprint 5

El quinto sprint fue dedicado a la modificación de la información disponible en la wiki, al aprendizaje de Zotero, a la correción del javadoc existente en Activiti-Api que presentaba una gran cantidad de warnings y errores y finalmente a comenzar con el desarrollo de la nueva funcionalidad a implementar: la comparación con la base de datos. [5]

El sprint comenzó el día 28 de marzo y finalizó el día 10 de abril, siendo otro de los sprints más cortos del proyecto debido principalmente a que la carga de trabajo era mayormente de modificación de la documentación.

Se comenzó aprendiendo a utilizar Zotero, gracias a una explicación del tutor Carlos López y fue en las fechas iniciales de este sprint cuando se añadieron a Zotero muchas de las páginas que se había consultado hasta ese momento (los tutoriales para la realización de diversos ejercicios fueron añadidos casi al final del proyecto, para tener una referencia de lo realizado en los pequeños proyectos para evaluar las distintas alternativas).

Los cambios en la documentación existente en la wiki fueron otro de los aspectos presentes, que al no tratarse de una gran cantidad, se realizaron en poco tiempo. Otro aspecto de la documentación fue el inicio del cambio aplicado a javadoc, que se finalizaría en sprints posteriores.

Finalmente, se dedicó gran cantidad de tiempo a implementar la nueva funcionalidad, necesitando documentarse sobre la generación de los percentiles y la lectura de un archivo .csv, pero aunque en un principio la idea era la de programar la obtención de los cuartiles, se decidió utilizar la libería Apache Commons Math, que simplificó el proceso.

Se calculó que el tiempo necesario para completar estas tareas era de 15 story points, es decir, 7 horas y media, y finalmente fue de 7 horas.

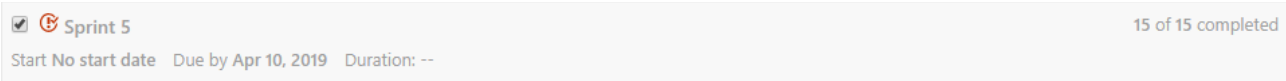


Ilustración 4: Story points estimados para el sprint 5.

2.6. Sprint 6

El sexto sprint supuso el comienzo del desarrollo de la interfaz gráfica y la mejora de la documentación, tanto de los conceptos teóricos de la wiki, como de lo realizado en cada uno de los test existentes y el javadoc. En cuanto al código existente, se realizó la factorización de los métodos utilizados en las comparaciones, tanto con la base de datos como entre informes. [6]

El sprint se inició el día 10 de abril y terminó el 26 del mismo mes.

En los primeros días se corrigió el javadoc restante del código existente, para que dejase de contener errores o warnings. Tras ello, se añadieron string a los assert de los test, indicando que comprobación realizada cada uno, para que en caso de error, se detectase fácilmente que comprobación era la que fallaba (aunque los assert en estos momentos funcionaban correctamente, el orden de entrada de datos de los mismos estaba cambiado, el resultado esperado estaba como segundo parámetro mientras que los resultados en el primero).

Tras ello se refactorizaron ambos métodos de comparación, para reducir el número de líneas de código usadas, la duplicidad existente en ellos y su complicidad.

En cuanto a la funcionalidad, se añadió el guardado de los datos de las métricas en la base de datos .csv, así como la modificación de esa misma base de datos para contener únicamente los campos que se iban a utilizar. Este guardado, al no realizarse mediante el formato UTF-8 presentó problemas a la hora de leer y escribir ciertos caracteres.

La última tarea realizada fue la de comenzar con la interfaz gráfica, creandola con JavaFX en vez de Java Swing para que tuviese mejor aspecto y fuese más intuitiva. Se crearon las principales clases de escenas (que en realidad son los paneles de distribución de los elementos, no escenas) y se encontró el primer problema de utilizar JavaFX: la biblioteca JavaHelp únicamente tenía compatibilidad con JavaSwing, que se solucionó añadiendo un botón Java Swing.

Lo ultimo que se hizo fue generar una nueva distribucción que mostrase los cambios introducidos en la funcionalidad y la interfaz.

Se estimó que el total de la tareas podría completarse en 36 story points, es decir, 18 horas aunque finalmente se trabajó algo más de 24 debido principalmente a la creación de la interfaz y las diversas pruebas realizadas en proyectos externos para dar con una interfaz interesante para aplicar al proyecto.

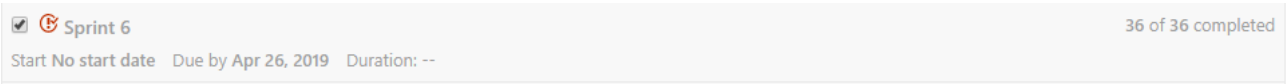


Ilustración 5: Story points estimados para el sprint 6.

2.7. Sprint 7

El séptimo sprint fue dedicado casi en su totalidad a continuar el desarrollo de la interfaz, rediseñando ciertas pantallas, cambiando colores y fuentes, añadiendo pequeños cambios como hacer la aplicación redimensionable o establecer las carpetas iniciales de los FileChooser y aplicando el patrón de diseño asistente (wizard), para mejorar la facilidad de uso de la aplicación. El resto del tiempo se dedicó a actualizar los conceptos teóricos existentes en la wiki. [7]

El sprint comenzó el día 26 de abril y se terminó el 10 de mayo, siendo uno de los sprints más largos y que más cambios supuso.

Los primeros trabajos realizados fueron los pequeños cambios aplicados para mejorar la apariencia de la interfaz, como el cambio de fuente, el cambiar la evaluación del desarrollo del proyecto en la patalla de comparación con la base de datos, limitar los decimales mostrados en la misma o establecer la carpeta inicial y extensiones para los FileChooser (aunque el establecimiento de las extensiones presentó un pequeño defecto que hacía que no se añadiesen correctamente, que sería corregido más adelante).

Tras ésto, se decidió añadir a la Wiki la información de los conceptos teóricos sobre las métricas de evolución utilizadas y las refactorizaciones.

De nuevo el trabajo que continuó fueron cambios en el diseño de diversas pantallas de la interfaz aplicando el patrón asistente (wizard), se hizo la aplicación redimensionable aunque manteniendo un tamaño mínimo para que los distintos elementos no colapsasen, se decidió guardar el estado de la conexión utilizada en la última operación realizada (tanto modo desconectado como conectado) y finalmente se separó la búsqueda entre repositorios y forks.

Se estimó que todos estos cambios se podrían completar en 57 story points, es decir, 28 horas y media aunque se sobreestimo el tiempo necesario para realizar distintas tareas como la creación

del menú o cambiar los diseños de la interfaz. Finalmente se destinaron algo menos de 32 horas a la realización de las tareas.

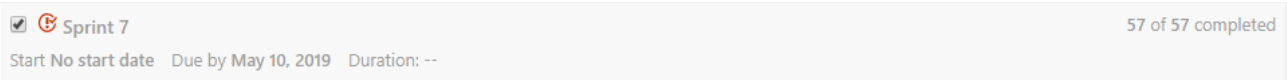


Ilustración 6: Story points estimados para el sprint 7.

2.8. Sprint 8

En general este sprint, aunque contó con varias tareas, no supuso una gran carga de trabajo, debido a que todas ellas eran pequeños cambios en la interfaz para hacerla más agradable a la vista, corregir pequeños defectos de la aplicación del patrón asistente (wizard) o modificar la exportación de la distribución del proyecto. [8]

La fecha de inicio del sprint fue el 10 de mayo y su finalización el 23 del mismo mes.

Lo primero que se realizó fue la modificación a la forma de distribuir el proyecto, debido principalmente a un error que no permitía guardar nuevas métricas en la base de datos desde un .jar (por estar modificándose a sí mismo), por lo que se decidió distribuir la base de datos de forma externa al .jar.

Una vez conseguido ésto, se realizaron diversos pequeños cambios, como organizar correctamente el menú, cambiar el título del proyecto para dejar de utilizar Activiti-Api (aunque se ha conservado en GitHub por ser un fork), ordenar los distintos botones de acuerdo al patrón utilizado o cambiar los colores, en particular los de las comparaciones, disminuyendo su saturación para que resultasen más agradables a la vista.

Tras ello se organizó el apartado técnicas y herramientas de la wiki y se pasó a reestructurar la pantalla about (que cambió el contenido que tenía completamente) y la pantalla de resultados, para hacerla mucho más atractiva.

La ultima tarea consistió en generar una nueva distribución con todos los pequeños cambios aplicados.

La estimación del tiempo para las tareas de este sprint fue de 21 story points (10 horas y media) y su realización se completo en 10 horas.

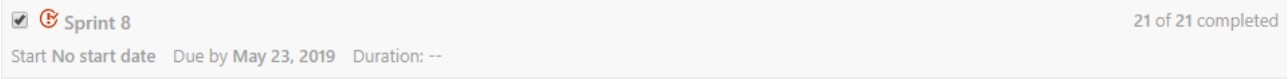


Ilustración 7: Story points estimados para el sprint 8.

2.9. Sprint 9

El sprint 9 es el último realizado y ha consistido tanto en solventar pequeños defectos como a hacer la aplicación más accesible. [9]

El sprint se inició el 23 de mayo y su fecha de finalización será el mismo día de la entrega, el 6 de junio.

El sprint se inició cambiando de nuevo la información de la pantalla About para dejar la que

finalmente se presenta y algunos cambios en la pantalla de selección de repositorio (añadir los botones para eliminar la selección de un repositorio o fork).

Tras ello se arregló el bug existente en la aplicación de la extensión de archivo a los FileChooser y el guardado de los archivo para evitar que se añadiesen dos extensiones y en añadir una notificación sobre la conexión utilizada cuando se está buscando un repositorio, debido a que únicamente se guardaba sin notificar al usuario.

La mayor parte del trabajo ha recaído en los cambios en las tablas de comparación, ordenando las métricas utilizadas y modificando las tablas para hacerlas más vistosas.

Una de las últimas tareas realizadas ha sido una de las más costosas del sprint: la internacionalización, debido a la necesidad de modificar la estructura de las escenas para poder cambiar el idioma dinámicamente.

Lo último que se ha realizado en el proyecto (exceptuando la generación de la documentación de la memoria, anexos y vídeos) es la modificación de la ayuda para que se adaptase a las pantallas existentes (aunque únicamente existe la ayuda en español).

Las horas estimadas para la realización del trabajo asignado fueron 21 y media (43 story points) y realmente se han dedicado 26 (exceptuando las dedicadas a la realización de la memoria, anexos y vídeos).

Ilustración 8: Story points estimados para el sprint 9.

2.10. Resumen

Finalmente, la tabla resumen con las horas empleadas en cada uno de los apartados es la siguiente, aunque hay que tener en cuenta que varios issues comparten label puesto que tenían parte de distintos apartados, por lo que los asignaré al apartado del que más trabajo ha dado de entre ambos.

En el apartado otros incluiré la duración de las reuniones de los sprints o el tiempo dedicado a la realización de la memoria, anexos, vídeos del proyecto o búsquedas de información sobre la resolución de distintos bugs o issues que no han quedado reflejados en los mismos (por haberse realizado sin un issue).

Otro aspecto a aclarar es que parte del testeo lo ha realizado el tutor en el entorno Linux, por lo que el tiempo dedicado a ese testeo no ha corrido por mi cuenta.

Apartado	Issues	Tiempo (en horas)
Bug	6	6,5
Desarrollo	33	84,5
Distribución	3	1
Documentación	11	19,5
Evaluación de alternativas	4	6
Instalación e integración	8	20
Testeo	3	4
Otros	.	80
Total	68	221,5

Tabla 1: Tabla resumen de horas empleadas

3. Estudio de viabilidad

En este apartado voy a evaluar la viabilidad económica y legal del proyecto, de haber sido creado con el fin de obtener beneficios.

3.1. Viabilidad económica

Los costes derivados del desarrollo del proyecto habrían sido los siguientes:

3.1.A. Costes hardware

Únicamente existiría el coste del ordenador utilizado para el desarrollo, que no necesitaría ser demasiado potente.

La amortización de los equipos informáticos y el software está estipulada en 6 años. [10]

6 años = 72 meses.

493,05 / 72 = 6,84 € por mes.

6,84 x 2 = 13,68 € en dos meses.

Concepto	Coste amortizado mensual (€)
Ordenador [11]	6,84
Total	13,68

Tabla 2: Tabla de costes hardware

3.1.B. Costes software

En cuanto a coste del software, se necesitaría una licencia Windows, puesto que la realización del proyecto ha sido realizada en este sistema. Para probar en otro sistema operativo como

Linux, bastaría con una máquina virtual, por lo que sería gratuito.

De nuevo, la amortización del software es también de 6 años.

259 / 72 = 3,60 € por mes.

3,60 x 2 = 7,2 € en dos meses.

Concepto	Coste amortizado mensual(€)
Windows 10 [12]	3,6
Total	7,2

Tabla 3: Tabla de costes software

3.1.C. Costes personal

Como el proyecto ha sido desarrollado por una persona, únicamente habría que valorar su salario. Dadas las horas empleadas en el proyecto, se podría completar su desarrollo en dos mes trabajando a tiempo completo en jornadas de 40 horas semanales. El salario supuesto tras comprobar las ofertas de puestos para ingenieros informáticos presentes en infojobs será de 18000€ brutos anuales.

Concepto	Coste(€)
Salario [13]	1500
Total	3000

Tabla 4: Tabla de costes de personal

3.1.D. Otros costes

Aquí se valoran los costes de alquilar una oficina para desarrollar el proyecto o el coste de internet, necesario para la aplicación.

Para la oficina se han tenido en cuenta el precio del alquiler encontrado en idealista.

Para internet se ha estimado un coste de 30 euros mensuales.

Concepto	Coste(€)
Alquiler oficina [14]	300
Internet	30
Total	660

Tabla 5: Tabla de costes varios

3.1.E. Coste total

Concepto	Coste(€)
Costes hardware	13,68
Costes software	7,2
Costes personal	3000
Otros costes	660
Total	3680,68

Tabla 6: Tabla de coste total

3.1.F. Beneficios

En principio existen varias posibilidades a la hora de obtener beneficios:

- Poner precio a la aplicación: que aunque es una posibilidad, el precio no debería ser demasiado alto, debido a que una gran cantidad cantidad de las herramientas con propósitos similares son gratuitas.
- Introducir publicidad: como se puede apreciar al utilizar la aplicación, existe una gran cantidad de espacio en muchas escenas, por lo que el añadir publicidad en esos espacios podría suponer una forma de monetizar la aplicación.
- Establecer un pago por servicios: este método requeriría el mudar la base de datos a un sistema online y que este servicio únicamente estuviese disponible para aquellos que pagasen por él.

3.2. Viabilidad legal

La legalidad del producto depende en su totalidad de las librerias utilizadas para que éste pueda realizar el trabajo. Las librerias utilizadas y sus licencias son las siguientes:

- Apache Commons Math: Apache License 2.0 [15]
- Gson: Apache License 2.0 [16]
- Hamcrest: BSD License [17]
- JaCoCo: Eclipse Public License 1.0 [18]
- JavaHelp: existe algo de confusión con lo referente a la licencia de esta librería, debido a la compra de la misma por empresas. A día de hoy en su página consta que es CDDL 1.0 [19]
- JUnit: Eclipse Public License 1.0 [20]

Como la licencia más restrictiva de todas es CDDL, que obliga a los productos que incluyan una librería con esta licencia a tener como mínimo esta misma licencia, la licencia con la que se ha de distribuir el proyecto es CDDL 1.0.

Aclarar que todas ellas permiten el uso en productos con fines comerciales siempre y cuando no hagan responsables del producto a los propietarios de las librerías, por lo que se podría comercializar el proyecto.

II - Apéndice B. Especificación de requisitos

1. Introducción

Un requisito es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. [21]

En esta parte del documento voy a describir cada uno de los requisitos funcionales existentes en el proyecto, así como los requisitos no funcionales que se quisieron alcanzar desde el inicio.

2. Objetivos Generales

Los objetivos generales pueden dividirse en dos partes, el propio código y el aspecto gráfico. En el código las mejoras son las siguientes:

- Refactorización del código y mejora del mantenimiento del proyecto.
- Añadir la funcionalidad de comparar un proyecto con los guardados en la base de datos mediante percentiles.

Y la parte gráfica consta de:

- Eliminar completamente la interfaz anterior, hecha con Java Swing, y crear una totalmente nueva utilizando Java FX.
- Mejorar la facilidad de uso de la aplicación, para que sea más intuitiva.

3. Catalogo de requisitos

Los diferentes requisitos presentes en el proyecto, debido a que previamente existían y se han decidido conservar o porque se han incluído durante la realización del proyecto son los siguientes:

3.1. Requisitos funcionales

- **RF1 Eliminar la conexión:** el usuario ha de poder eliminar el modo de conexión guardado por defecto en la última operación realizada.
- **RF2 Analizar un repositorio GitHub:** el usuario ha de poder buscar un repositorio disponible públicamente en GitHub y analizarlo.
  - **RF2.1 Log in:** la aplicación ha de permitir loguearse con el usuario existente en GitHub.
  - **RF2.2 Buscar repositorios de un usuario:** el usuario ha de poder realizar la búsqueda de los repositorios públicos de un usuario.
  - **RF2.3 Guardar informe:** el usuario ha de poder guardar los resultados de las métricas obtenidas durante el análisis.

  - **RF2.4 Evaluar con base de datos:** el usuario ha de poder comparar los resultados de las métricas obtenidas en su búsqueda con las existentes en la base de datos.
  - **RF2.5 Guardar en la base de datos:** el usuario ha de poder guardar los resultados de las métricas obtenidas en su búsqueda en la base de datos.
- **RF3 Importar informe:** el usuario ha de poder importar los resultados de un proyecto previamente exportado.
- **RF4 Comparar informes:** el usuario ha de poder comparar dos ficheros de resultados previamente exportados.
- **RF5 Configurar idioma:** el usuario ha de poder escoger y cambiar entre los idiomas disponibles.
- **RF6 Consultar ayuda:** el usuario ha de poder consultar la ayuda con el contenido de cada pantalla de la aplicación.
- **RF7 Consultar about us:** el usuario ha de poder consultar la información de la aplicación y sus autores.

3.2. Requisitos no funcionales

- **RNF1 Facilidad de uso:** la aplicación ha de presentar una usabilidad sencilla, para poder ser utilizada por usuarios sin una gran experiencia.
- **RNF2 Rendimiento:** la aplicación ha de soportar grandes cargas de datos debido a la existencia de repositorios enormes en GitHub y obtener los datos en una cantidad de tiempo considerablemente baja.
- **RNF3 Seguridad:** la aplicación no debe mostrar datos sensibles, como la contraseña de GitHub necesaria para realizar la conexión usuario.
- **RNF4 Portabilidad:** la aplicación ha de poder ejecutarse tanto en Windows como en Linux.
- **RNF5 Internacionalización:** la interfaz de aplicación debe soportar varios idiomas, incluyendo el inglés para poder optar a un público mayor.

4. Especificación de requisitos

En esta parte se va a mostrar el diagrama de casos de uso con los requisitos funcionales antes mencionados y una tabla para cada uno que los detalle.

4.1. Diagrama de casos de uso

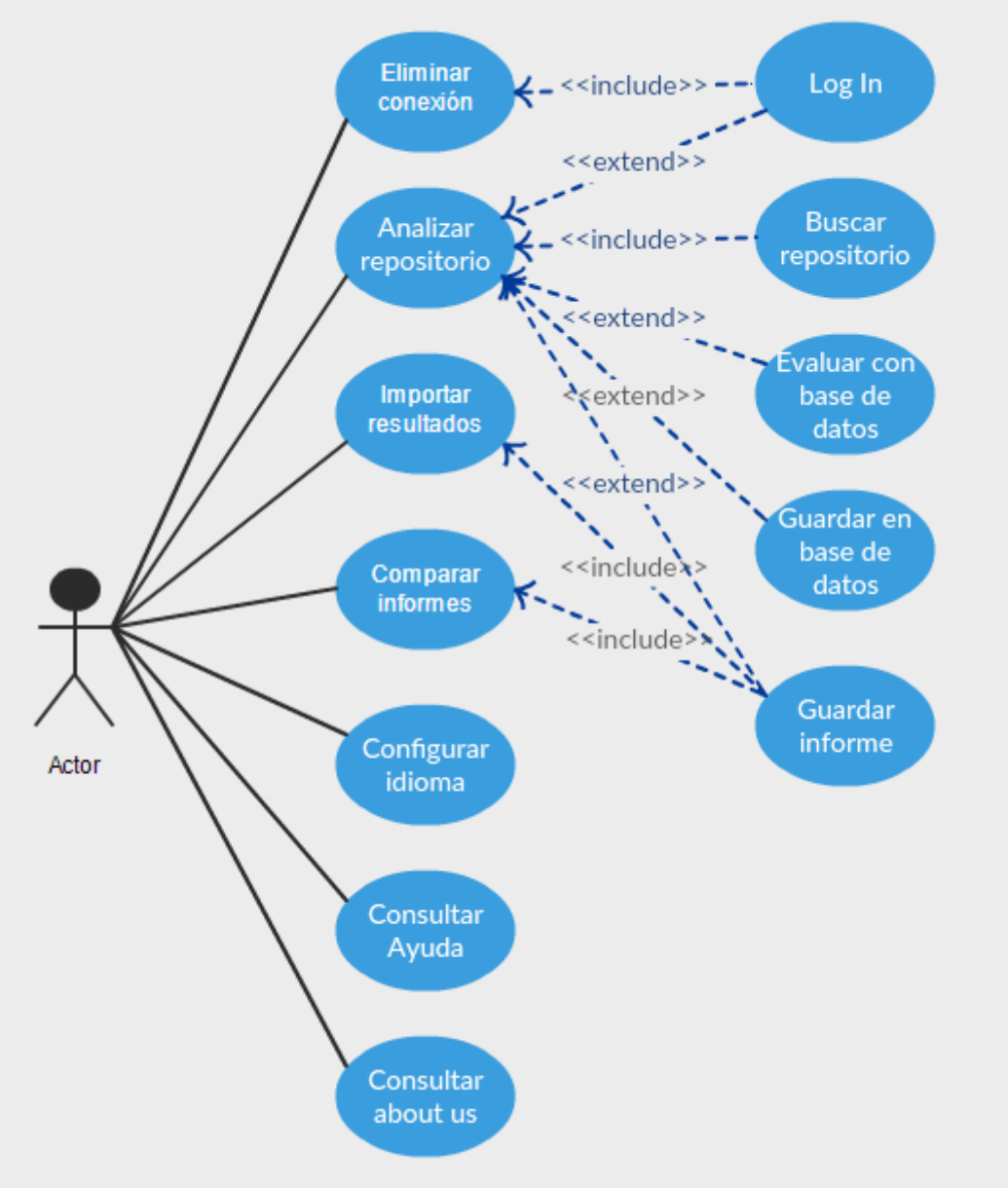


Ilustración 9: Diagrama de casos de uso de la aplicación.

4.2. Actores

La aplicación únicamente contará con un actor, que el usuario que quiera realizar cualquiera de las funciones disponibles.

4.3. Casos de uso

4.3.A. Eliminar conexión

Nombre	Eliminar conexión
Descripción	Eliminar la conexión guardada por defecto tras realizar una operación.
Actor	Usuario
Requisitos relacionados	RF2.1
Precondiciones	-
Flujo principal	1. Elimina la conexión (asigna el lector como null). 2. Se cambia a la pantalla de inicio.
Flujo secundario	-
Postcondiciones	Se elimina la conexión guardada.
Excepciones	-

Tabla 7: Caso de uso Eliminar conexión.

4.3.B. Analizar repositorio

Nombre	Analizar repositorio
Descripción	Obtener los resultados y gráficos del repositorio.
Actor	Usuario
Requisitos relacionados	RF2.1, RF2.2, RF2.3, RF2.4, RF2.5
Precondiciones	-
Flujo principal	1. Se selecciona el modo de conexión. 2. Se busca un repositorio. 3. Se obtienen los datos y se calculan las métricas y obtienen los gráficos. 4. Se separan las métricas para su presentación. 5. Se cambia a la pantalla de resultados.
Flujo secundario	-
Postcondiciones	Aparece la pantalla con los resultados de las métricas, los distintos gráficos y las opciones de comparación y guardado con la base de datos.
Excepciones	• Errores derivados de los casos de uso pertinentes.

Tabla 8: Caso de uso Analizar repositorio.

4.3.C. Log In

Nombre	Log In
Descripción	Crear una conexión.
Actor	Usuario
Requisitos relacionados	RF1, RF2
Precondiciones	No hay conexión guardada.
Flujo principal	1. Se selecciona el modo de conexión. 2. Se piden usuario y contraseña. 3. Se crea la conexión.
Flujo secundario	1. Si se selecciona el modo desconectado se muestra warning. 2. Se crea el modo desconectado.
Postcondiciones	Se crea la conexión o el modo desconectado y se pasa a la pantalla de búsqueda de repositorio.
Excepciones	• Error de datos (usuario o contraseña erróneos)

Tabla 9: Caso de uso Log In.

4.3.D. Buscar repositorio

Nombre	Buscar repositorio
Descripción	Obtener un repositorio para poder analizar.
Actor	Usuario
Requisitos relacionados	RF2
Precondiciones	-
Flujo principal	1. Se selecciona el usuario a buscar. 2. Se selecciona un repositorio. 3. Se obtienen los datos para calcular las métricas y gráficos. 4. Se separan las métricas para su presentación.
Flujo secundario	1. Se selecciona el usuario a buscar. 2. Se selecciona un fork. 3. Se obtienen los datos para calcular las métricas y gráficos. 4. Se separan las métricas para su presentación.
Postcondiciones	Aparece la pantalla con los resultados de las métricas, los distintos gráficos y las opciones de comparación y guardado con la base de datos.
Excepciones	• Error de datos (usuario erróneo). • Número de peticiones agotadas.

Tabla 10: Caso de uso Buscar repositorio.

4.3.E. Guardar informe

Nombre	Guardar informe
Descripción	Guardar los resultados de las métricas obtenidas.
Actor	Usuario
Requisitos relacionados	RF2
Precondiciones	Repositorio o fork encontrado.
Flujo principal	1. Se selecciona el destino del archivo. 2. Se selecciona el nombre del archivo. 3. Si el nombre no tiene la extensión .txt se le añade. 4. Si existe un archivo con el mismo nombre se pide confirmación apra sobreescribirlo. 5. En caso afirmativo se crea un archivo .txt con los resultados de las métricas.
Flujo secundario	1. Se cancela el guardado y se cierra la pantalla de selección de destino.
Postcondiciones	Se crea un archivo .txt con los resultados de las métricas calculados.
Excepciones	• Archivo con el mismo nombre.

Tabla 11: Caso de uso Guardar informe.

4.3.F. Evaluar con base de datos

Nombre	Evaluar con base de datos.
Descripción	Evaluar los resultados obtenidos con los existentes en la base de datos.
Actor	Usuario
Requisitos relacionados	RF2
Precondiciones	-
Flujo principal	1. Se selecciona la base de datos a utilizar en formato .csv. 2. Se cargan los datos existentes. 3. Se calculan los percentiles. 4. Se muestra la tabla de comparación.
Flujo secundario	1. Se cancela la comparación y se vuelve a la pantalla de resultados.
Postcondiciones	Aparece la pantalla con los resultados de la comparación, los valores recomendados y una evaluación.
Excepciones	• Error de apertura de archivo.

Tabla 12: Caso de uso Evaular con base de datos.

4.3.G. Guardar en la base de datos

Nombre	Guardar en la base de datos
Descripción	Guardar los resultados de las métricas obtenidos en la base de datos .csv.
Actor	Usuario
Requisitos relacionados	RF2
Precondiciones	-
Flujo principal	1. Se selecciona la base de datos a utilizar. 2. Se abre el archivo. 3. Si el proyecto ya existe se modifica únicamente esa fila, de lo contrario se añade una nueva al final.
Flujo secundario	1. Se cancela el guardado y se vuelve a la pantalla de resultados.
Postcondiciones	Aparece la pantalla con los resultados de las métricas, los distintos gráficos y las opciones de comparación y guardado con la base de datos.
Excepciones	• Error de apertura de archivo.

Tabla 13: Caso de uso Guardar en la base de datos.

4.3.H. Importar informe

Nombre	Importar informe
Descripción	Cargar un informe previamente guardado con los resultados de las métricas.
Actor	Usuario
Requisitos relacionados	RF2.3
Precondiciones	Existe un informe guardado previamente.
Flujo principal	1. Se selecciona el informe a importar. 2. Se obtienen los datos que contiene. 3. Se genera una conexión para obtener los resultados y gráficos. 4. Se pasa a la pantalla de resultados.
Flujo secundario	-
Postcondiciones	Aparece la pantalla con los resultados de las métricas, los distintos gráficos y las opciones de comparación y guardado con la base de datos.
Excepciones	• Error de apertura de archivo.

Tabla 14: Caso de uso Importar informe.

4.3.I. Comparar informes

Nombre	Comparar informes
Descripción	Comparar dos informes previamente guardados.
Actor	Usuario
Requisitos relacionados	RF2.3
Precondiciones	Existen dos informes de distintos repositorios previamente guardados.
Flujo principal	1. Se selecciona el primer informe a comparar. 2. Se selecciona el segundo informe a comparar. 3. Se obtienen los datos de los informes. 4. Se genera la tabla de la comparación y se pasa a la pantalla de resultado de la comparación.
Flujo secundario	-
Postcondiciones	Aparece la pantalla con los resultados de la comparación de los dos informes.
Excepciones	• Error de apertura de archivo. • Error de dos informes no seleccionados. • Error de informe similar.

Tabla 15: Caso de uso Comparar informes.

4.3.J. Configurar idioma

Nombre	Configurar idioma
Descripción	Cambiar el idioma usado en la interfaz de la aplicación.
Actor	Usuario
Requisitos relacionados	-
Precondiciones	-
Flujo principal	1. Se selecciona la opción del menú Opciones y se muestra un menú desplegable. 2. En el apartado idioma se muestra un submenú con los idiomas disponibles. 3. Tras seleccionar uno de ellos, se cargan los archivos .config de cada escena que contienen el texto a mostrar en los elementos de cada una de ellas. 4. Se aplican esos textos a los elementos.
Flujo secundario	-
Postcondiciones	El idioma de la interfaz se cambia por el seleccionado.
Excepciones	• Error de apertura de archivos.

Tabla 16: Caso de uso Configurar idioma

4.3.K. Consultar ayuda

Nombre	Consultar ayuda.
Descripción	Mostrar la ayuda de la aplicación disponible en español.
Actor	Usuario
Requisitos relacionados	-
Precondiciones	-
Flujo principal	1. Se selecciona la opción de menu Ayuda y se muestra un submenú. 2. Se selecciona la opción Ver ayuda del submenú. 3. Se realiza click mediante un evento en el botón Java Swing que abre la ayuda. 4. Se abre una nueva ventana que contiene la ayuda.
Flujo secundario	-
Postcondiciones	Aparece la nueva ventana con la ayuda.
Excepciones	-

Tabla 17: Caso de uso Consultar ayuda.

4.3.L. Consultar about us

Nombre	Consultar about us
Descripción	Consulta la página que contiene información de la universidad, los autores y el proyecto.
Actor	Usuario
Requisitos relacionados	-
Precondiciones	-
Flujo principal	1. Se selecciona la opción del menú About. 2. Se selecciona la opción del submenú Ver about us. 3. Se cambia a la pantalla con información sobre el proyecto, los autores y la universidad.
Flujo secundario	-
Postcondiciones	Aparece la pantalla con la información.
Excepciones	-

Tabla 18: Caso de uso Consultar about us.

III - Apéndice C. Especificación de diseño

1. Introducción

En esta parte del documento se explican los detalles más importantes de las decisiones tomadas en cuanto a diseño se refiere. Incluye el diseño de los datos utilizados, el diseño procedimental y el diseño arquitectónico.

2. Diseño arquitectónico

En este apartado voy a mostrar los diagramas de clases más relevantes en el desarrollo del proyecto.

Si se quiere obtener más información, se puede consultar el Javadoc generado en el repositorio del proyecto.

2.1. Diagrama de clases del paquete lector

Uno de los diagramas más relevantes es el del lector que permite crear una conexión y obtener los datos del repositorio para calcular las métricas que se utilizarán en el proyecto.

- La clase FabricaConexionGitHub se encarga de crear la fachada necesaria para tratar con GitHub.
- La clase FachadaConexionGitHub se encarga de crear la conexión con GitHub, necesario para obtener los datos de la plataforma.
- La clase FachadaMetricasGitHub es la encargada de recibir los datos y crear las métricas que van a ser utilizadas en la aplicación.
- FabricaConexion, FachadaConexion y FachadaMetricas son las interfaces utilizadas para que las clases que utilizan cada una de las clases antes mencionadas interaccionen con ellas sin necesitar conocer su funcionamiento interno.

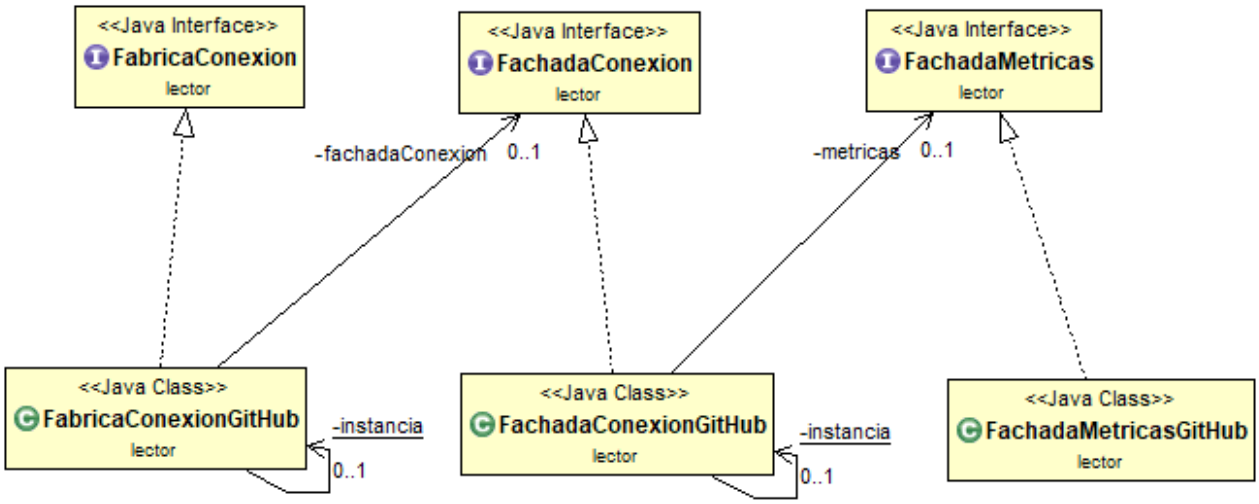


Ilustración 10: Diagrama de clases del paquete lector.

El lector contiene pequeñas refactorizaciones y cambios con respecto al utilizado en Activiti-Api.

2.2. Diagrama de clases del ManagerCSV

ManagerCSV es la clase que contiene instancias de las distintas clases necesarias para manejar la apertura, lectura y modificación de un archivo .csv. También contiene el separador que permite separar los resultados de las métricas para presentarlos de una forma más atractiva, en vez de limitarse a mostrar una cadena.

- El lector permite la apertura y tratado del archivo .csv.
- El separador realiza la separación de los resultados las métricas.
- La calculadora es la encargada de calcular los percentiles para la comparación con la base de datos (archivo .csv.)

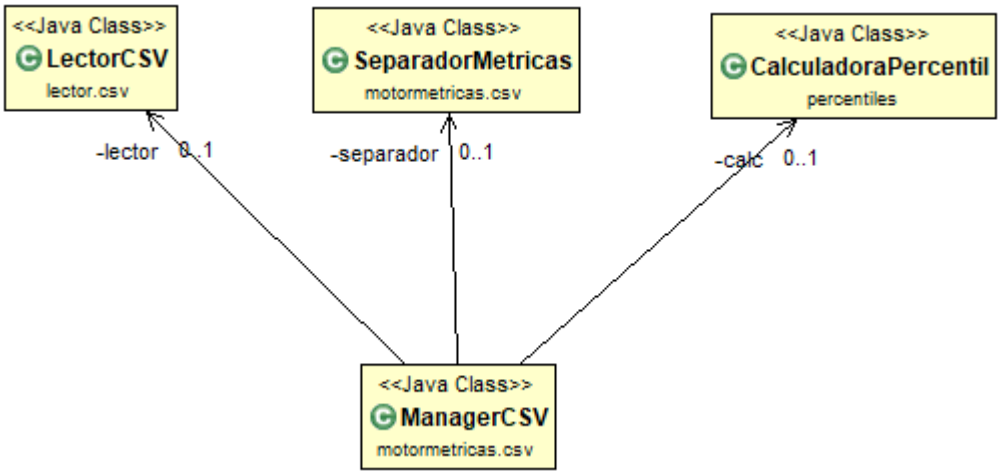


Ilustración 11: Diagrama de clases del ManagerCSV

2.3. Diagrama de clases del motor

El motor es el encargado de crear todas las métricas a partir de los datos obtenidos gracias al lector antes mostrado.

- Metrica es la calse abstracta que implementan todas las medidas del proyecto, que serán accedidas mediante la interfaz Imetric.
- Descripción contendrá toda la información perteneciente a un tipo de métrica en concreto.
- Medida contiene todos los valores calculados de las métricas que son accedidos a través de la interfaz Valor.
- ResultadoMetricas contiene un array con las medidas de cada una de las métricas usadas en el proyecto.
- Finalmente, cada uno de los valores estará formado de una manera y como he comentado, se necesitará la interfaz Valor para abstraer al resto de clases del funcionamiento de cada uno.

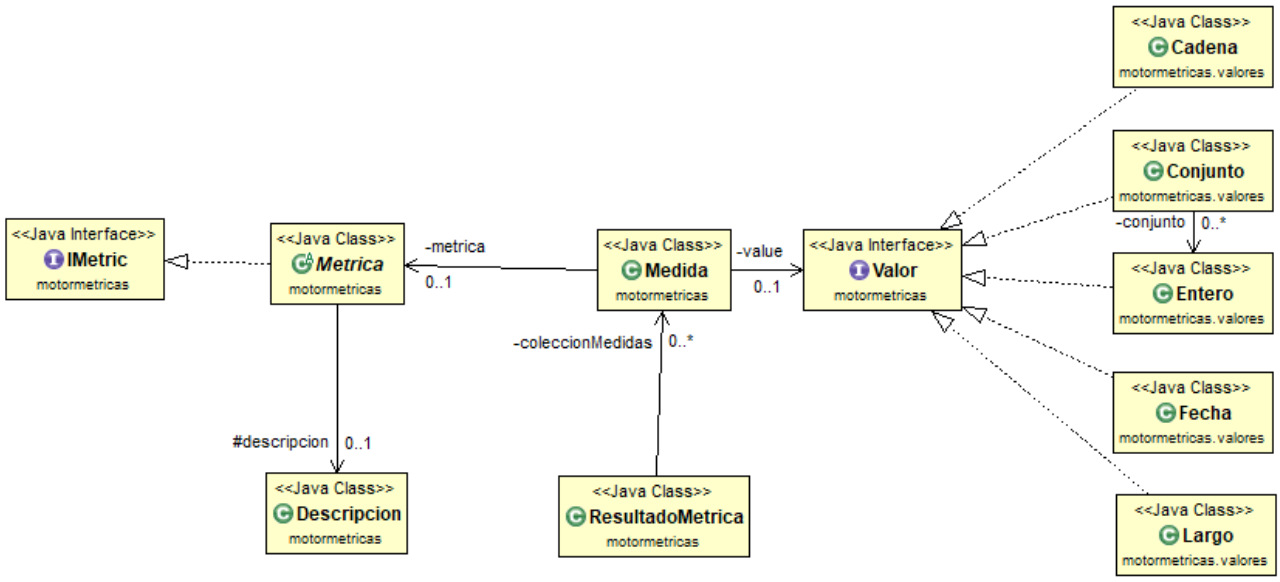


Ilustración 12: Diagrama de clases del motor de métricas.

Este motor se ha conservado intacto del utilizado por Activiti-Api, por lo que sigue estando implementado según lo aconsejado por "Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones".

3. Diseño procedimental

En este apartado voy a mostrar los diagramas de secuencia para cada operación que realiza la aplicación en las que intervengan más de una clase, puesto que las realizadas en la propia clase únicamente van a utilizar datos que contengan ellas a través de un método privado.

Puesto que la base es Activiti-Api, incluiré algun diagrama como los mostrados en ese proyecto, debido a que la realización de la operación sigue teniendo el mismo diagrama.

3.1. Log In

Para realizar la conexión, se obtienen los datos en EscenaSelConex y se llama al método createModoUsuario de PrincipalFX que contiene los métodos a utilizar durante el uso de la interfaz gráfica. Esta clase realizará una llamada al método crearFachadaConexión de la instancia de la FabricaConexionGitHub que tiene que será la que a través de la FachadaConexionGitHub creará la conexión con el usuario y contraseña obtenidos en primer lugar.

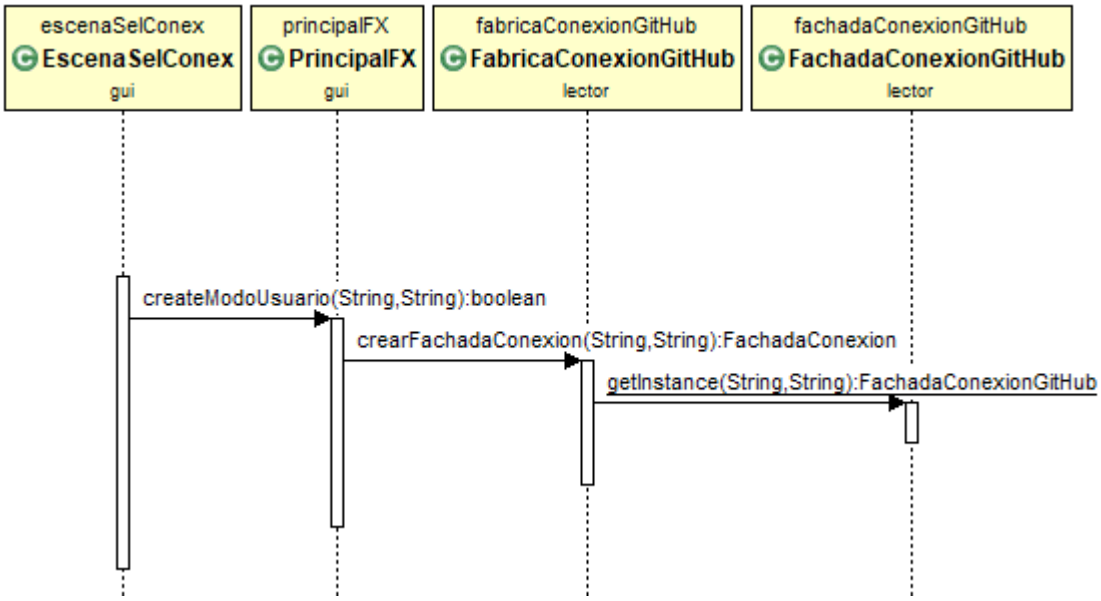


Ilustración 13: Diagrama de secuencia Log In

3.2. Buscar repositorios de un usuario

Para poder buscar un usuario, se ha de haber creado un modo de conexión anteriormente como se indica en las precondiciones.

Una vez se está en la EscenaUsuarioRep, se pasa el nombre de usuario que escribe el cliente y este es utilizado para buscar los repositorios de los que dispone. Para ello se realiza primero el método buscaRepositorios y seguidamente el buscaForks de la clase PrincipalFX que a su vez utiliza el método getNombresRepositorio y getNombresForks de la clase FachadaConexionGitHub, que será la encargada de obtener el listado de nombres de repositorios y forks a través de su método privado obtenerRepositorios.

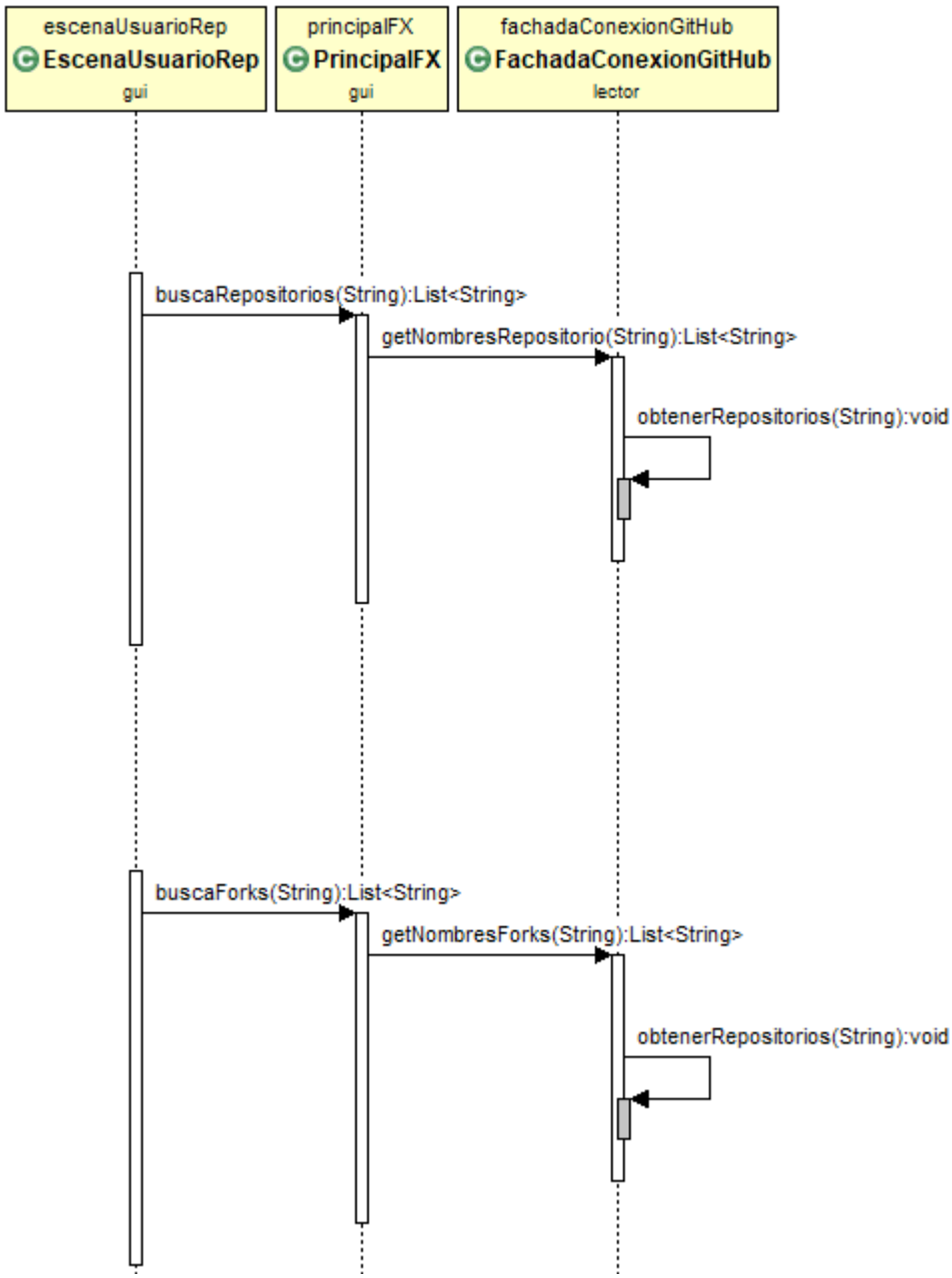


Ilustración 14: Diagrama de secuencia de Buscar repositorios de un usuario.

3.3. Guardar informe

Para guardar un informe, deberemos estar previamente en EscenaResultados, lo que quiere decir que las métricas ya han sido calculadas. Se llamará entonces al método generarArchivo de PrincipalFX que guardará los resultados obtenidos del método generarArchivo de FachadaConexionGitHub, que será la que obtenga el resultado de FachadaMetricasGitHub, donde se guardan las métricas calculadas del repositorio.

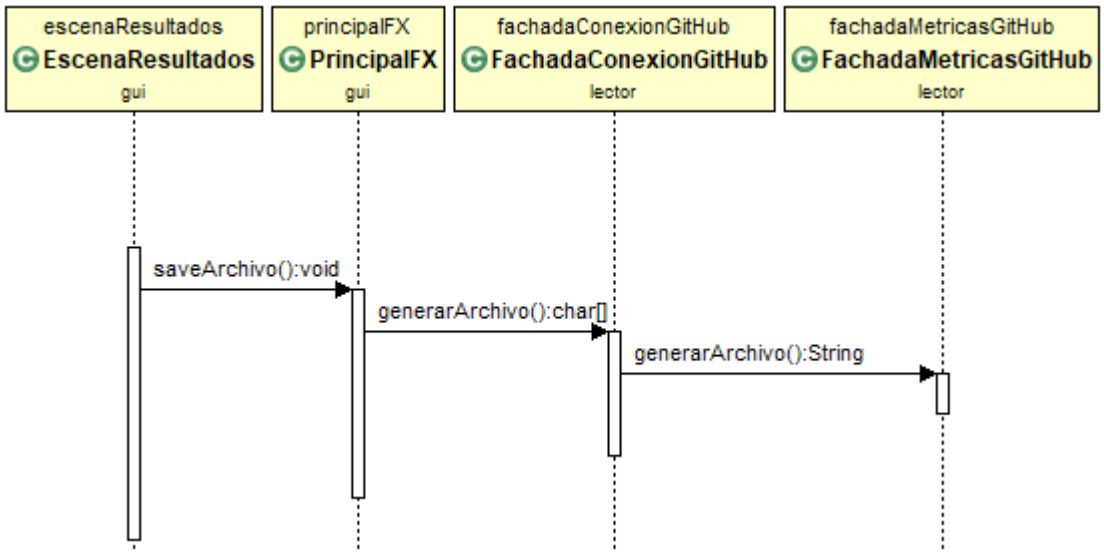


Ilustración 15: Diagrama de secuencia de Guardar Informe

3.4. Evaluar con base de datos

Para evaluar con la base de datos, hemos de encontrarnos en EscenaResultados. Esta clase utilizará el método startManager para instanciar el ManagerCSV en PrincipalFX. Tras ello, se utiliza el método creaTablaCSV desde EscenaResultados a través de PrincipalFX, que obtendrá el nombre del repositorio de FachadaConexionGitHub y utilizará el método creaTabla del ManagerCSV que obtendrá los datos del lector, los enviará al separador, obtendrá cada métrica del mismo (en el diagrama únicamente se muestran dos obtenciones de métricas porque poner todas sería demasiado) y a través de varios métodos privados crea la tabla con el resultado de la comparación.

Tras ello EscenaResultados utiliza el método getNota de PrincipalFX y éste la obtiene del ManagerCSV.

Cuando todo está creado, PrincipalFX establecerá los resultados en EscenaResultadoComparacion y cambiará a la misma.

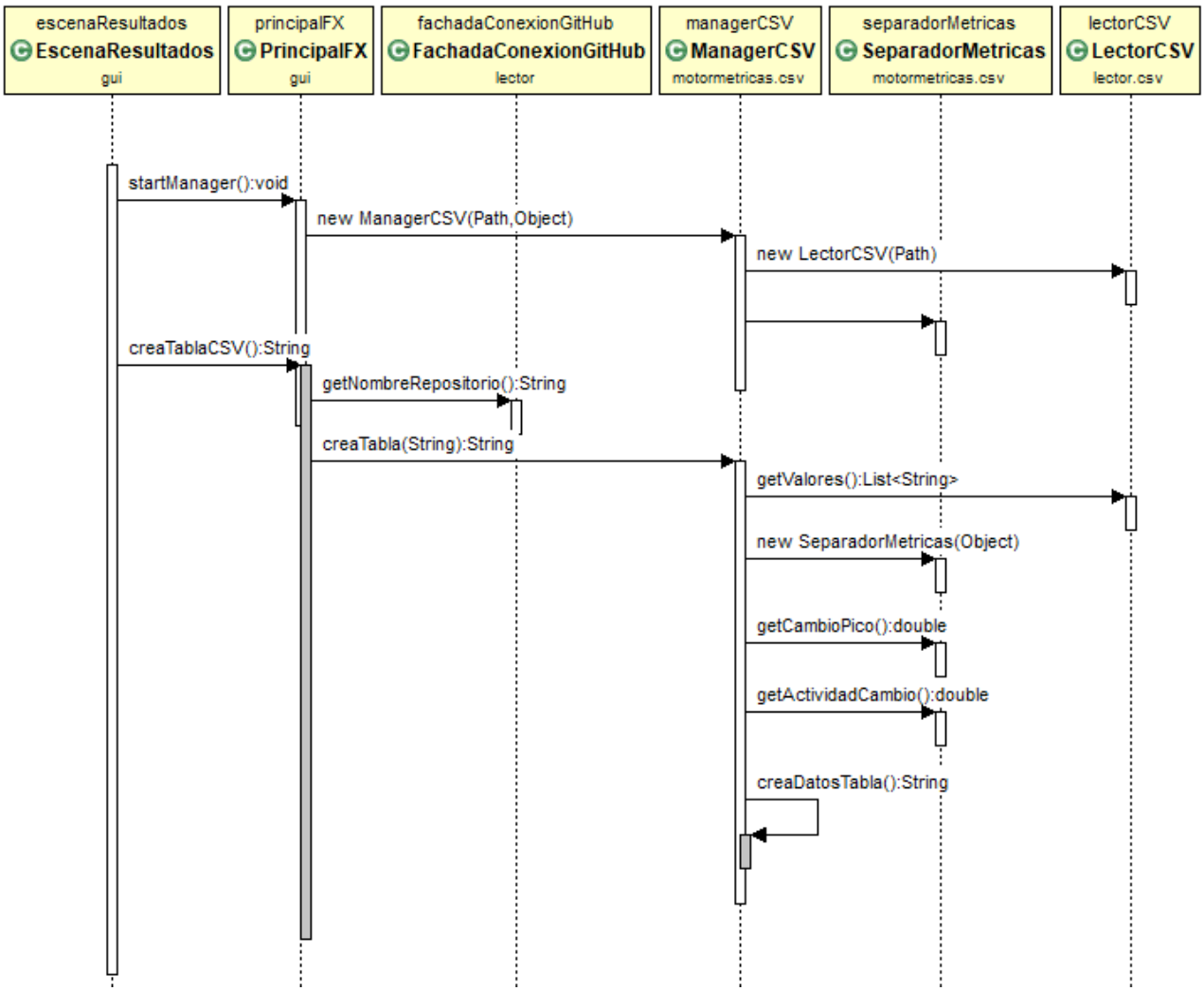


Ilustración 16: Diagrama de secuencia Evaluar con base de datos.

3.5. Guardar en la base de datos

Para guardar las métricas en la base de datos, EscenaResultados utiliza el método guardarEnCSV de PrincipalFX, que primero obtendrá el nombre del repositorio de FachadaConexionGitHub y luego pedirá el guardado a través del método addMetricasProyecto del ManagerCSV. Éste obtiene cada una de las métricas del separados (únicamente se muestran dos) y se las envía al LectorCSV a través de addMetricasProyecto que terminará guardando los datos en el .csv.

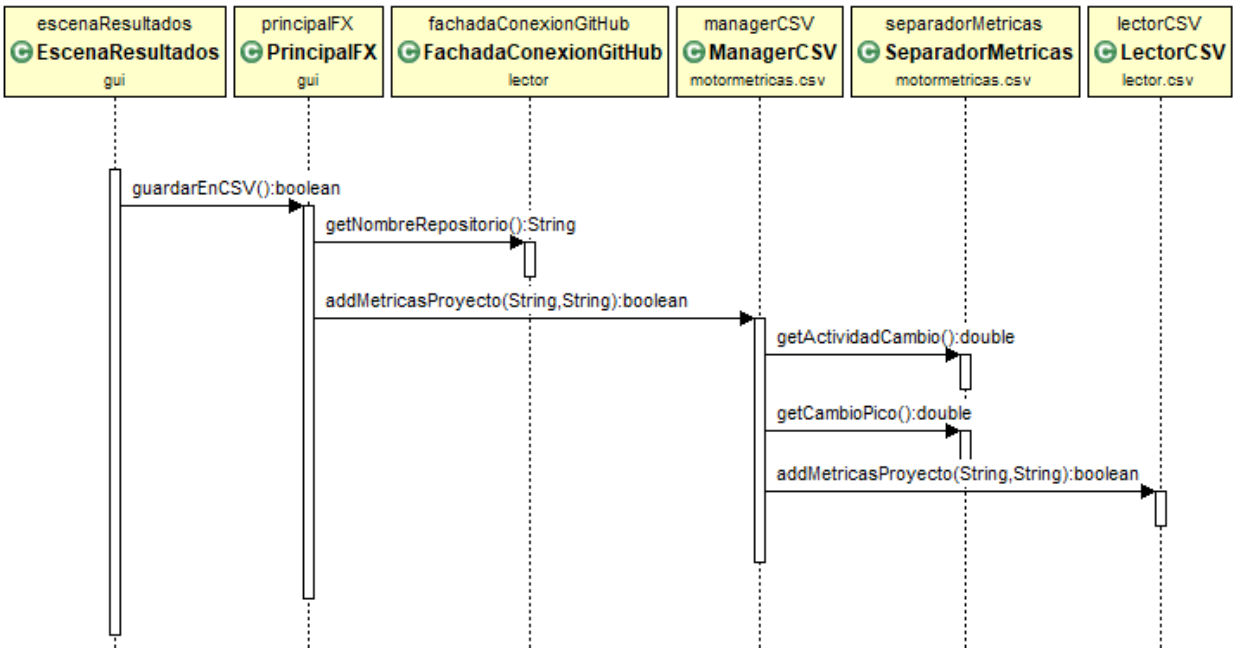


Ilustración 17: Diagrama de secuencia de Guardar en la base de datos.

3.6. Importar informe

Para importar un informe, debemos encontrarnos en EscenaInicio, puesto que a través del menú ofrece la opción. Tras la selección de la misma, se utilizará el método loadArchivo de PrincipalFX que a su vez utilizará el método leerArchivo de la clase FachadaConexionGitHub. Ésta última clase instanciará un objeto con las métricas a través de la clase FachadaMetricasGitHub que utilizando las clases Medida y ResultadoMetrica obtendrá cada métrica con su resultado. Tras ello, PrincipalFX establecerá los resultados en EscenaResultados y cambiará a la misma.

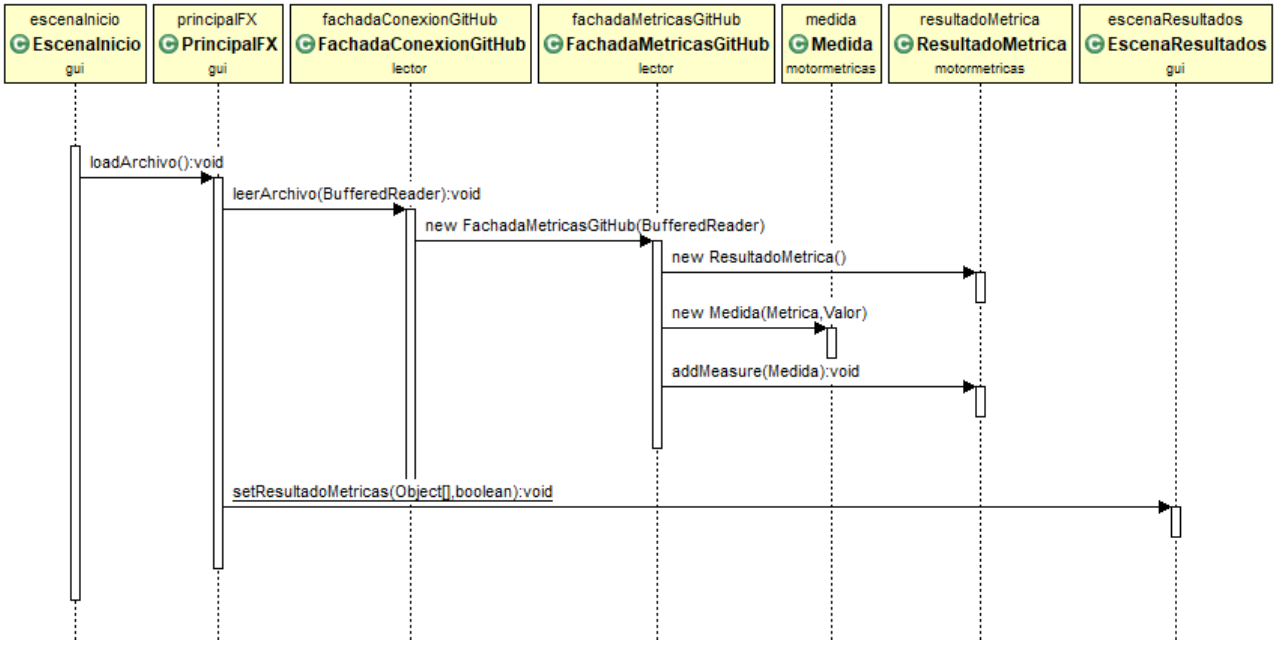


Ilustración 18: Diagrama de secuencia de Importar informe.

3.7. Comparar informes

La última funcionalidad que implica varias clases es comparar dos informes.

Para ello nos hemos de encontrar en la pantalla EscenaComparacion, que obtendrá la ventana principal para buscar 2 informes mediante el método seleccionarFichero. Una vez se tienen ambos, se utiliza el método leerFciheros que obtendra dos instancias de FabricaConexionGitHub y éstas a su vez dos de FachadaConexionGitHub.

Una vez se tienen todas las instancias, se utiliza el método leerArchivo de FachadaConexionGitHub que instanciará un objeto FachadaMetricasGitHub para obtener los valores de las métricas.

Cuando se tienen los valores de las métricas, se utiliza el método comparar de FachadaConexionGitHub que a su vez llamará al de FachadaMetricasGitHub para comparar los dos ficheros. Este resultado lo establecerá PrincipalFX en EscenaResultadoComparacion.

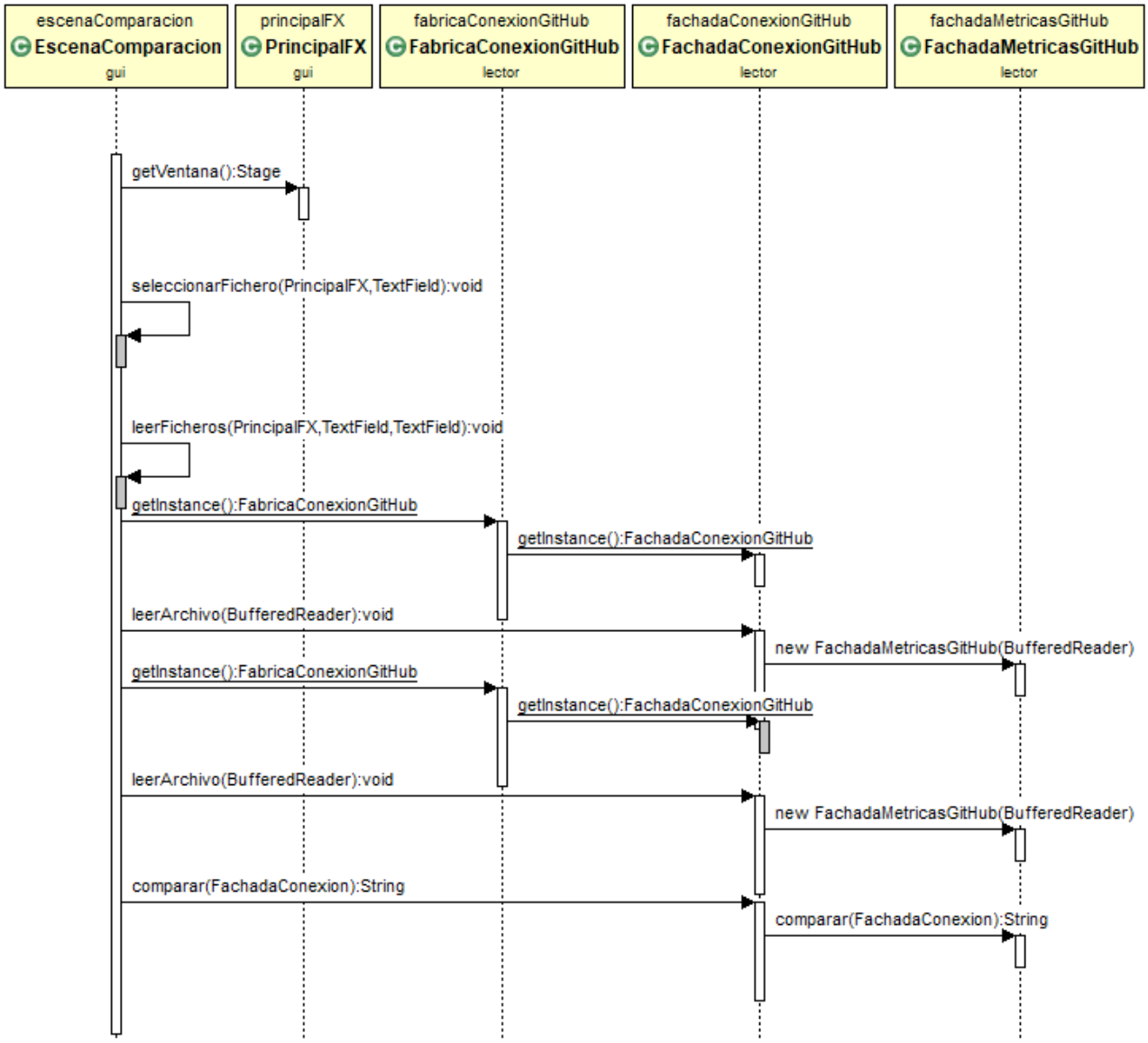


Ilustración 19: Diagrama de secuencia de Comparar informes.

4. Diseño de interfaz

La interfaz ha sido una de las partes que han variado mucho durante este proyecto, por lo que definiré el patrón de asistente utilizado y mostraré el ejemplo de aplicación en una escena.

4.1. Patrón de diseño asistente (wizard)

El patrón wizard suele utilizarse para que cualquier tipo de usuario pueda realizar una tarea medianamente compleja sin necesitar experiencia previa.

Consisten en dividir la tarea en pasos pequeños que simplifiquen la experiencia frente a realizar toda la tarea de una sola vez. Estos pequeños pasos pueden ser distintos dependiendo de la información introducida en los pasos anteriores. [22]

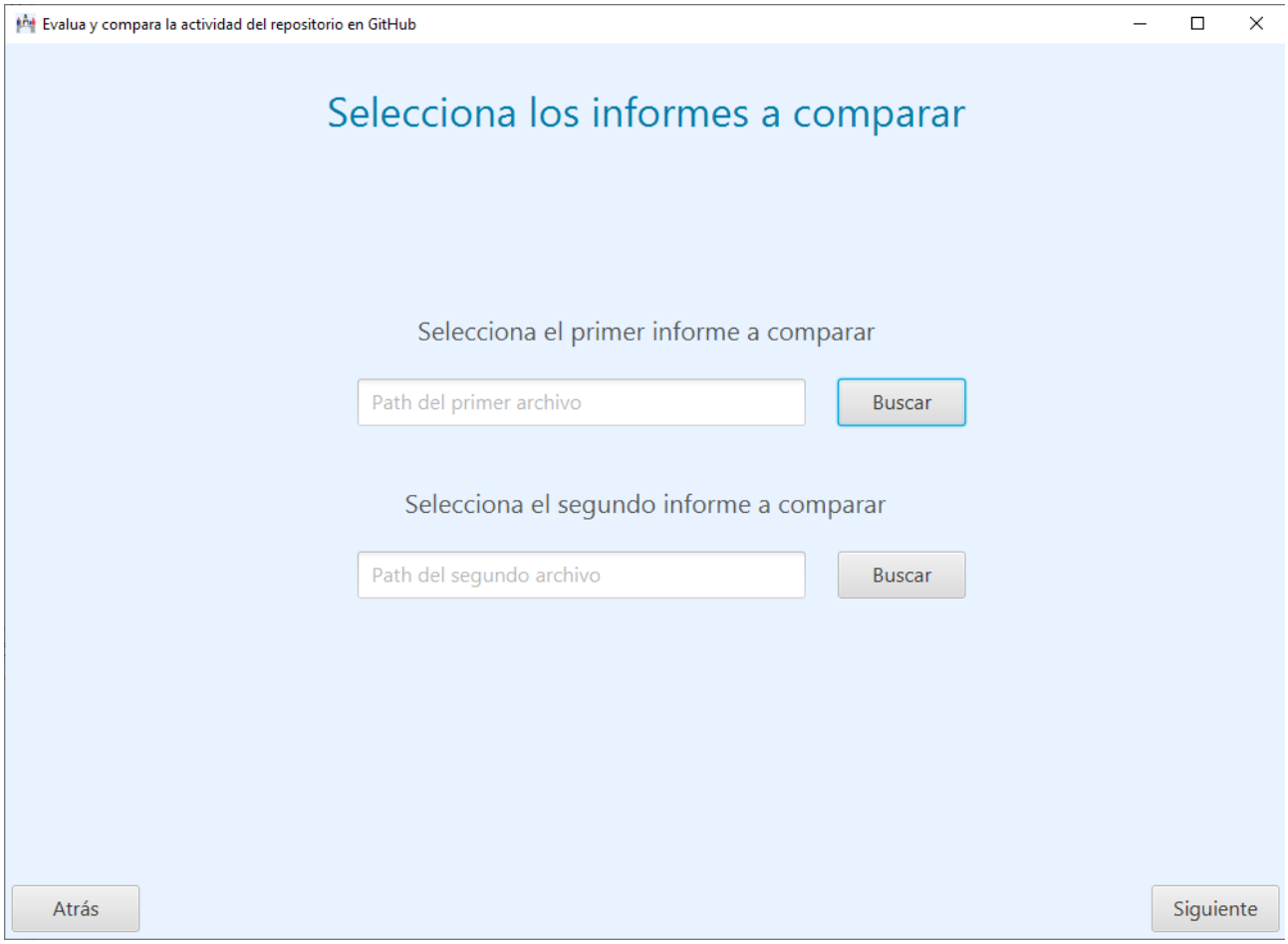


Ilustración 20: Ejemplo de pantalla con el patrón asistente aplicado.

Como se puede comprobar en la ilustración anterior, los requisitos funcionales se dividen en pequeñas tareas con un único proceso mental (en el ejemplo la selección de dos informes), y mediante los botones atrás y siguiente se vuelve a la pantalla anterior o se continúa a la siguiente tarea siempre y cuando la acción realizada sea correcta.

IV - Apéndice D. Documentación técnica de programación

1. Introducción

En este anexo se detallan los detalles técnicos de programación, como la instalación del proyecto, los servicios de integración continua implementados o los test realizados sobre el proyecto.

2. Estructura de directorios

Los directorios de la aplicación son los siguientes:

- /: contiene el archivo report utilizado para obtener el coverage, el archivo README para el repositorio GitHub, los archivos necesarios para integración continua y el archivo build.xml de Ant con los servicios de integración continua disponibles.
- /tests/tests: contiene los tests realizados sobre el proyecto.
- /rsc/config: contiene los ficheros con las traducciones a los distintos lenguajes.
- /rsc/css: contiene el archivo css aplicado en los ScrollPane
- /rsc/datoscsv: contiene las bases de datos en formato .csv necesarias para las pruebas o el funcionamiento de la aplicación.
- /rsc/imagenes: contiene las imágenes utilizadas por la aplicación.
- /rsc/informes: contiene los informes necesarios para los test.
- /rsc/lib contiene las librerías necesarias para el funcionamiento de la aplicación, los tests o los servicios de integración continua.
- /src/gui/ayuda: contiene las páginas html, los archivos y las imágenes necesarias para crear la ayuda mediante JavaHelp.
- /src/gui/herramientas: contiene el creador de elementos JavaFX utilizado en las distintas pantallas.
- /src/gui: contiene las clases de las que se compone la interfaz gráfica.
- /src/lector/csv: contiene el lector de archivos .csv.
- /src/lector: contiene las clases necesarias para crear la conexión con GitHub y obtener las métricas de los repositorios.
- /src/metricas: contiene las clases que definen cada tipo de métrica.
- /src/motormetricas/csv: contiene el ManagerCSV que es el encargado de realizar las funciones de los archivos .csv y el separador de métricas.
- /src/motormetricas/valores: contiene los distintos valores que pueden tomar las métricas.
- /src/motormetricas: contiene las clases que hacen posible la creación de métricas a partir de los valores obtenidos del repositorio.
- /src/percentiles: contiene la clase encargada de calcular los percentiles a partir de los datos de un .csv.

- /doc: contiene la documentación javadoc del proyecto.
- /memo: contiene la memoria y los anexos de la realización del proyecto, así como todas las imágenes utilizadas en ellos.

3. Manual del programador

En este apartado voy a comentar las herramientas de las que dispone el programador que quiera trabajar sobre el proyecto.

3.1. Apache Ant

Apache Ant es una librería de Java y una herramienta de línea de comandos que suministra una serie de tareas integradas que permiten compilar, ensamblar, probar y ejecutar aplicaciones Java. [23]

Apache permite programar tareas complejas o tediosas para poder ejecutarlas fácilmente y sin tener que realizar muchos pasos. Para poder crear nuevas tareas además de los que ya existen, se deberá abrir el archivo build.xml.

- Cada property indica la definición de la dirección de un directorio o fichero.
- Pathelement indica donde encontrar determinados ficheros o librerias necesarios para la compilación definida en path.
- Taskdef define una nueva tarea.
- Target crea una nueva funcionalidad, a crear por el programador.

Las funcionalidades ya creadas son las siguientes:

- Inicio: crea el sistema de carpetas necesarias y el timestamp para la distribución.
- Compilar: compila el código referente a la aplicación.
- Compilar-tests: compila el código referente a los tests de la aplicación.
- Tests: ejecuta los tests.
- Coverage: envía el fichero report.xml con el coverage a Codacy.
- Javadoc: genera el javadoc de la aplicación.
- Distr: genera una nueva distribución de la aplicación.
- Limpiar: elimina los archivos indicados en su definición.

3.2. Codacy

Codacy es una herramienta que permite la revisión automática de código que permite a los desarrolladores ubicar errores o código problemático para poder solventarlo. [24]


Tras cada commit, Travis CI mandará los resultados a Codacy que se encargará de buscar defectos en el código. También deberemos enviar el coverage resultante de la realización de los

test tras cada commit mediante la ejecución de la función coverage del archivo build.xml.

3.3. Travis CI

Travis es un servicio de integración continua utilizado para realizar las build y los test de proyectos ubicados en GitHub. [25]

Travis se ejecuta automáticamente con cada commit realizado sobre el proyecto gracias al archivo .travis.yml que por defecto está programado para ejecutar la función distr del fichero build.xml. Si se desea modificar, habrá que modificar la línea de código 5: "script: ant distr" por la función que se desee realizar.

4. Compilación, instalación y ejecución del proyecto.

4.1. Instalación

Para instalar el proyecto habrá que disponer de Java SDK 1.8 o superiores, para que JavaFX se encuentre disponible en el path y no genere ningún conflicto. Además se necesitará un entorno de desarrollo integrado (preferiblemente eclipse, ya que es el utilizado en el desarrollo) junto con EclipseEGit, que se puede encontrar en el marketplace de Eclipse (Help -> Eclipse Marketplace...)

Tras tener todo instalado, accedemos al repositorio en GitHub y hacemos click sobre el botón Clone or download, que nos mostrará una url para copiar.

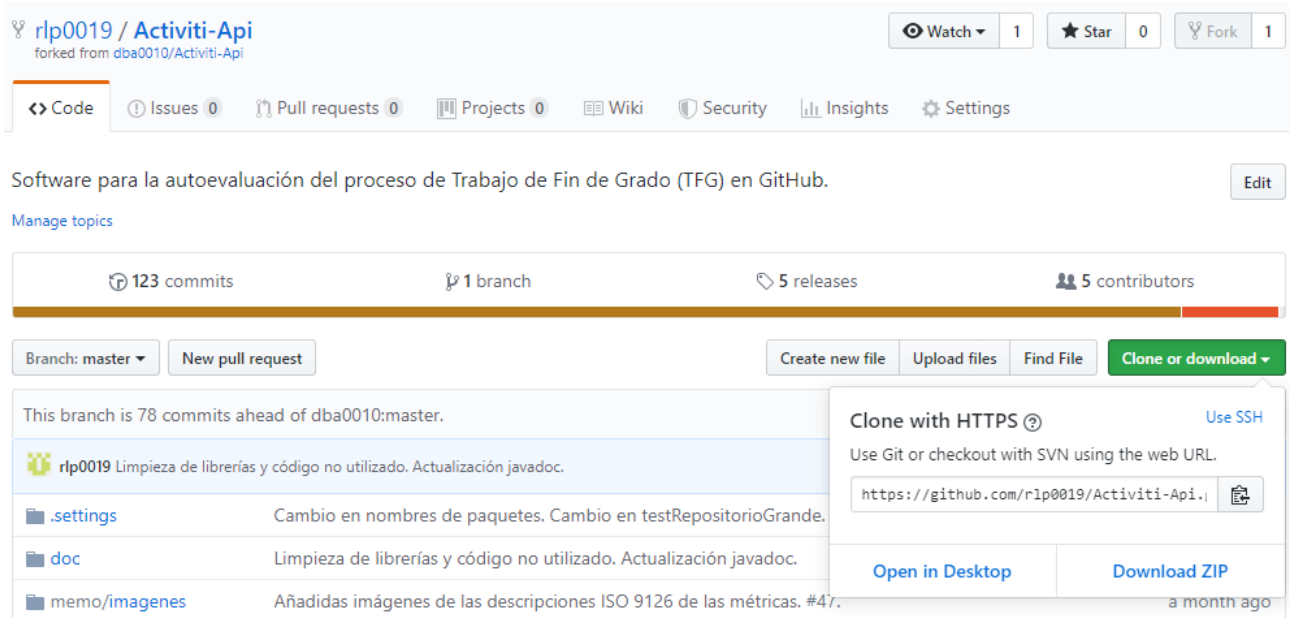


Ilustración 21: Ejemplo de la url del proyecto para la instalación.

Una vez copiada, se hace click derecho sobre la ventana Git Repositories de Eclipse (si la ventana no aparece, ir a Window -> Show view -> Other y en la carpeta Git, Git Repositories) y se selecciona la única opción que aparece.

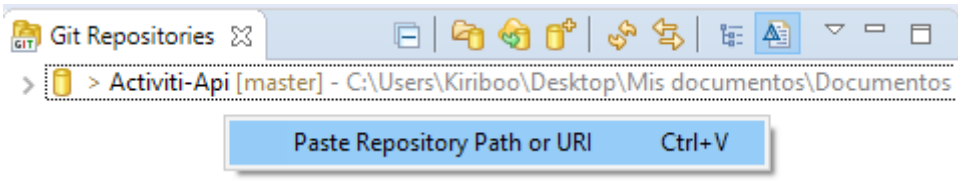


Ilustración 22: Importar repositorio GitHub en Eclipse.

4.2. Compilación

La compilación del proyecto se hace muy sencilla gracias a Apache Ant. Únicamente hay que abrir el archivo build.xml y ejecutar la funcionalidad de compilar.

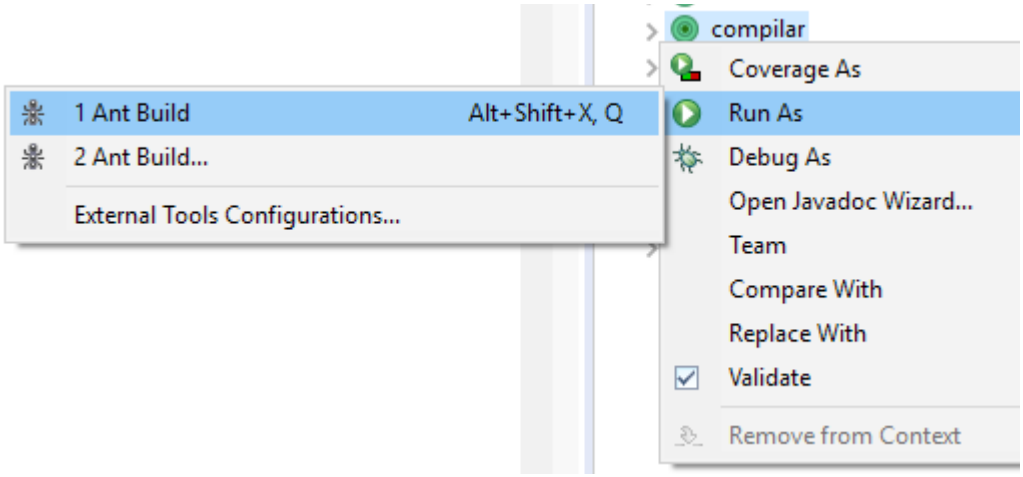


Ilustración 23: Ejecutar compilación mediante build.xml.

4.3. Ejecución

Para ejecutar el proyecto basta con ejecutar la clase PrincipalFX, que es la que contiene el método main.

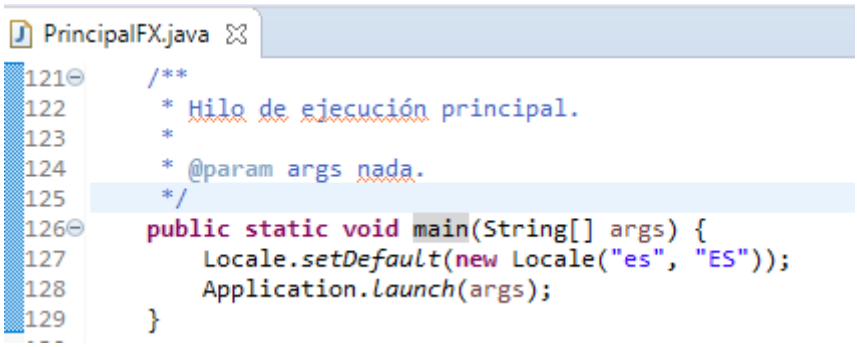


Ilustración 24: Método main el PrincipalFX.

4.4. Distribución

Si lo que se desea es generar una nueva distribución del proyecto, hay que ejecutar la funcionalidad distr del fichero build.xml.

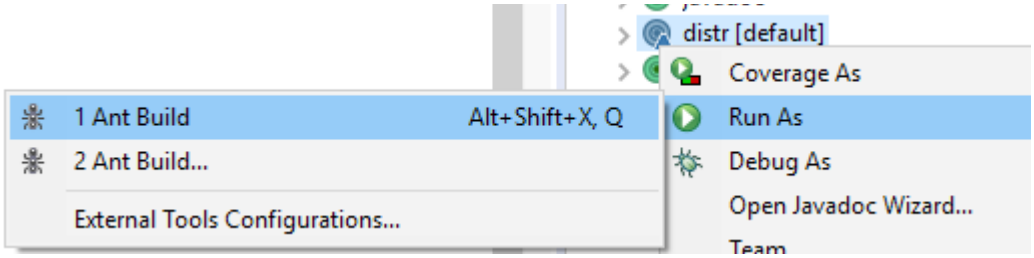
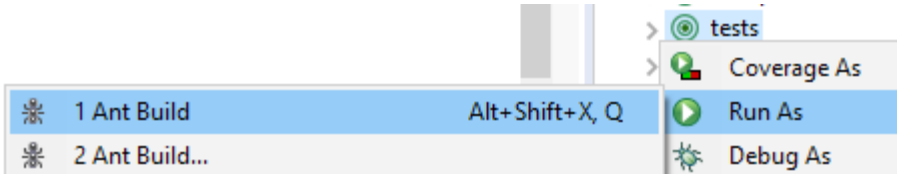


Ilustración 25: Ejecutar la generación de distribución mediante build.xml.

5. Pruebas del sistema

Para la comprobación del correcto funcionamiento de la aplicación y la posible detección de defectos, se han generado varios tests incluidos en la clase PrincipalTest del paquete /test/test.

Para su ejecución, basta con, de nuevo desde el fichero build.xml, ejecutar la funcionalidad tests.



Hay que recordar eliminar la última fila del archivo PruebaGuardar.csv disponible en /rsc/datoscsv, para que el test vuelva a funcionar correctamente tras su ejecución una segunda vez (la aplicación no dispone de método para eliminar un repositorio).

V - Apéndice E. Documentación de usuario.

1. Introducción

En este anexo aparece toda la información que necesita un usuario para la correcta instalación y uso de la aplicación.

2. Requisitos de usuarios

Los requisitos necesarios para poder ejecutar correctamente la aplicación son:

- Tener instalada la versión 1.8 de Java o superiores.
- Disponer de conexión a internet.

3. Instalación

Para instalar la aplicación hay que acudir al repositorio GitHub del proyecto y hacer click en la opción de las releases.

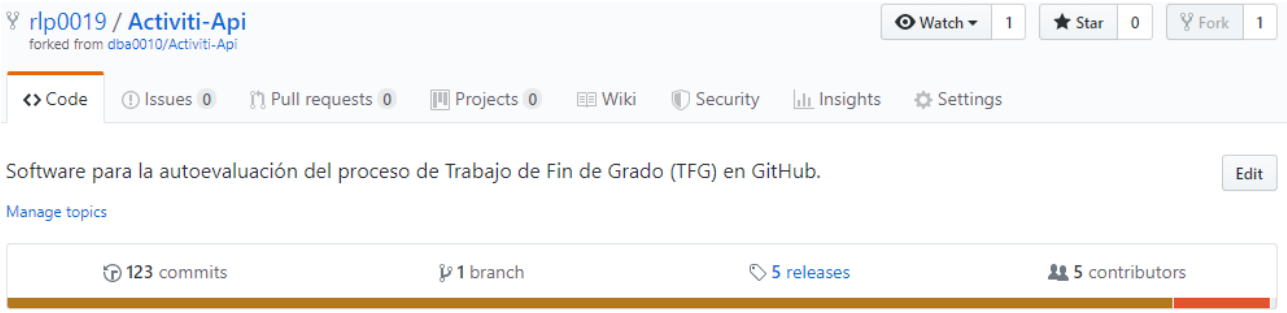
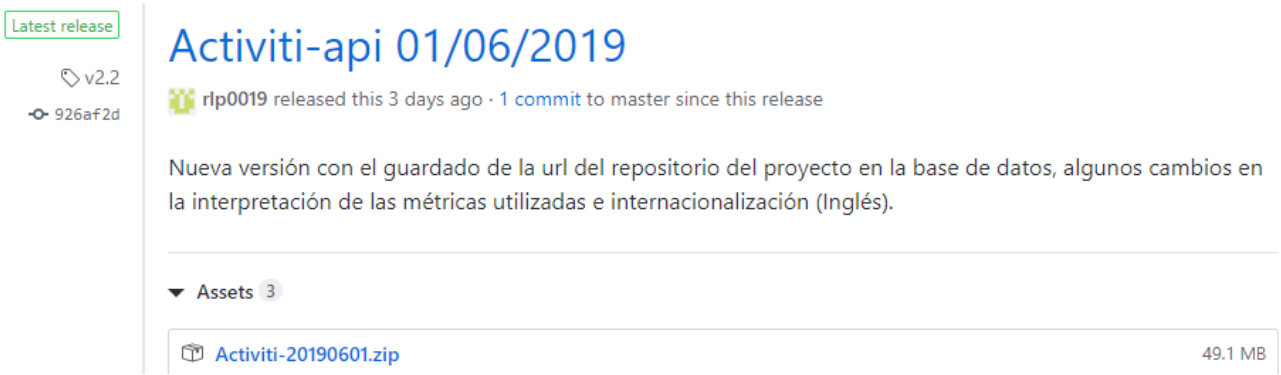


Ilustración 26: Ejemplo del repositorio en GitHub.

Una vez en la pantalla de las releases, seleccionar el archivo Activiti-xxxxxxxx.zip de aquella que contenga la etiqueta Lastest release, puesto que será la última versión lanzada y la más nueva.



Finalmente, se ha de extraer el contenido del archivo comprimido y ejecutar el archivo .jar que

contiene.

4. Manual del usuario

En este apartado se explica como realizar cada una de las funcionalidad que ofrece la aplicación.

4.1. Eliminar conexión

Para eliminar la conexión existente se ha de realizar click izquierdo en la opción Conexion del menú superior disponible en la pantalla de inicio de la aplicación y de nuevo click izquierdo en la opción Eliminar modo de conexion. La conexión se eliminará automáticamente.

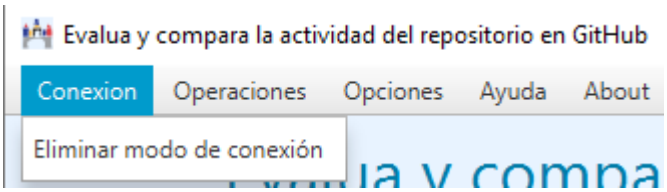


Ilustración 27: Ejemplo de opción Eliminar conexion.

4.2. Cambiar idioma

Para seleccionar el idioma entre los disponibles hay que realizar click izquierdo en la opción Opciones del menú superior disponible en la pantalla de inicio de la aplicación y en la opción Idioma seleccionar el que se desee. El idioma se cargará dinámicamente.

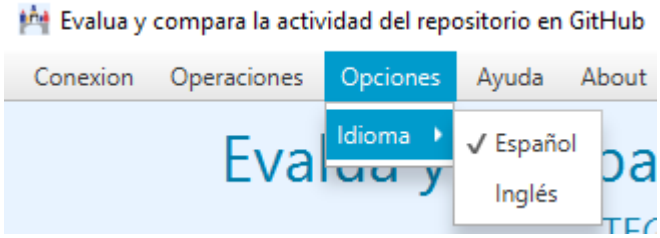


Ilustración 28: Ejemplo del cambio de idioma.

4.3. Mostrar ayuda

La ayuda actualmente solo está disponible en español.

Para consultar la ayuda hay que realizar click izquierdo en la opción Ayuda del menú superior disponible en la pantalla de inicio de la aplicación y de nuevo hay que realizar click izquierdo en la opción Mostrar ayuda.

Aparecerá una nueva ventana con la explicación de los elementos de cada pantalla y tablas con las definiciones de las métricas.

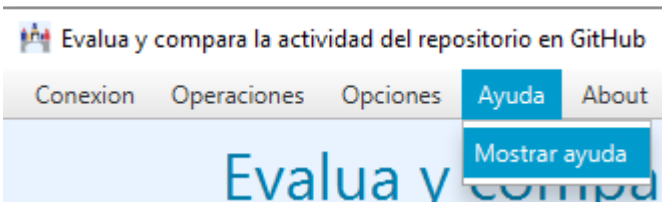


Ilustración 29: Ejemplo de Mostrar ayuda.

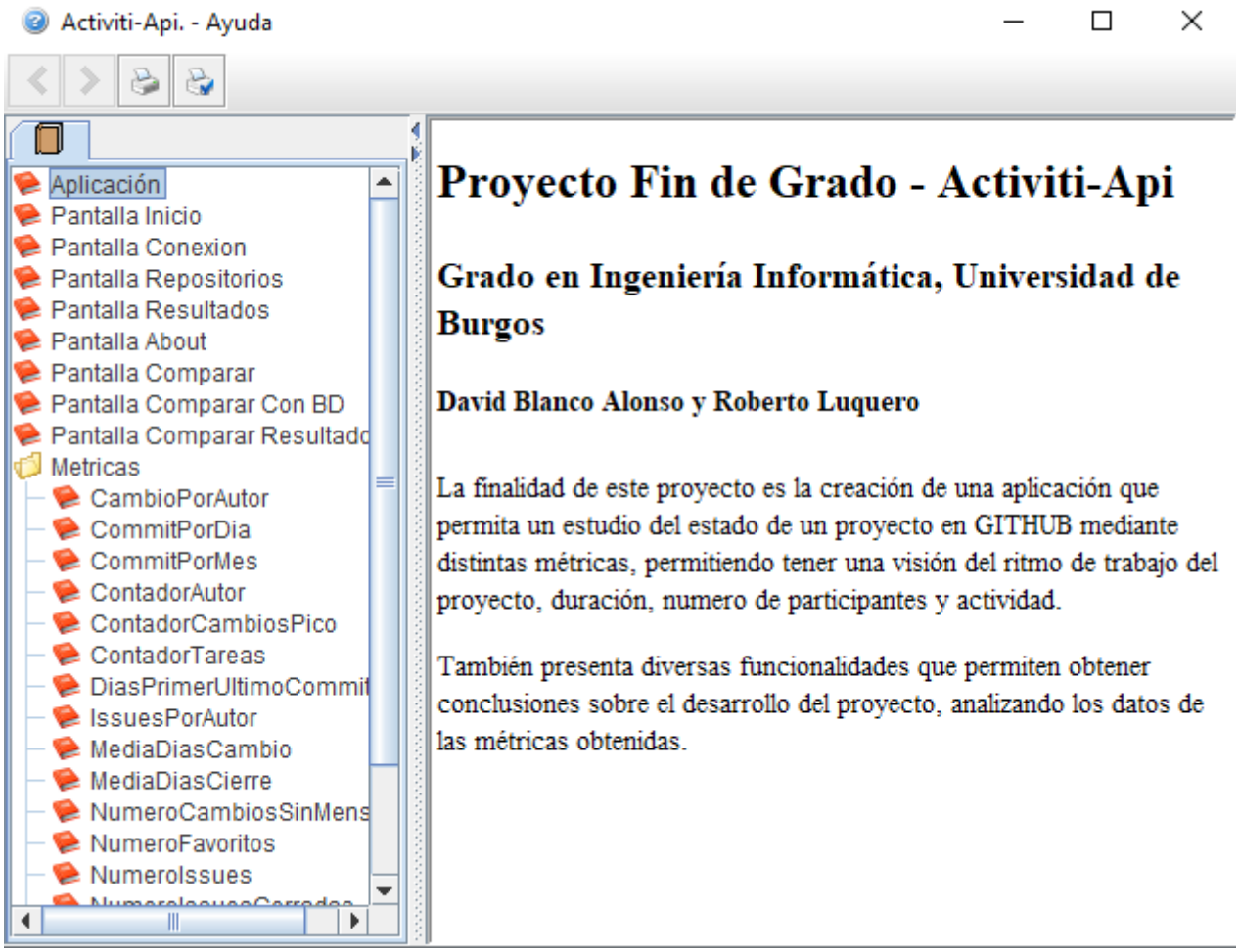


Ilustración 30: Ejemplo de la ventana de ayuda que aparecerá.

4.4. Mostrar about us

Para obtener más imformación sobre la universidad, los autores, el tutor o los repositorios del proyecto actual y el utilizado como base, basta con hacer click izquierdo en la opción About del menu superior disponible en la pantalla de inicio de la aplicación y realizar de nuevo click izquierdo en la opción Mostrar about us.

Se cambiará a la pantalla About us con la información disponible.

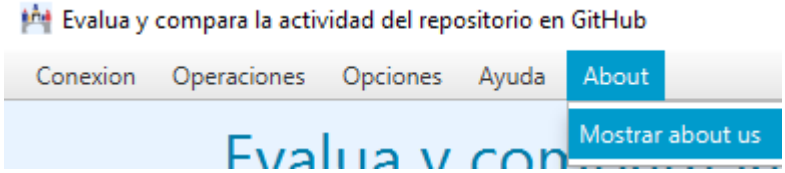


Ilustración 31: Ejemplo de Mostrar about us.

Ilustración 32: Ejemplo de pantalla about us.

4.5. Importar imforme

Para importar un informe previamente guardado hay que realizar click izquierdo en la opción Operaciones del menú superior disponible en la pantalla de inicio. A continuación hay que realizar click izquierdo en la opción Importar.

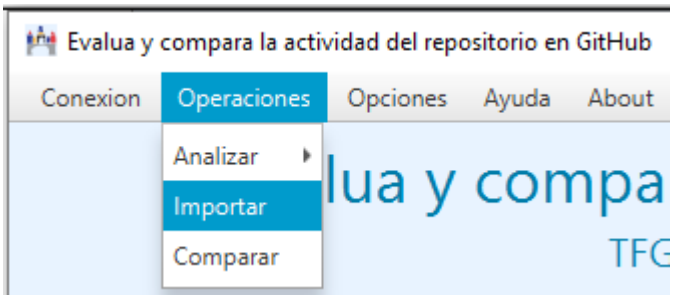


Ilustración 33: Ejemplo de Importar.

Se abrirá una nueva ventana donde seleccionar un informe guardado anteriormente.

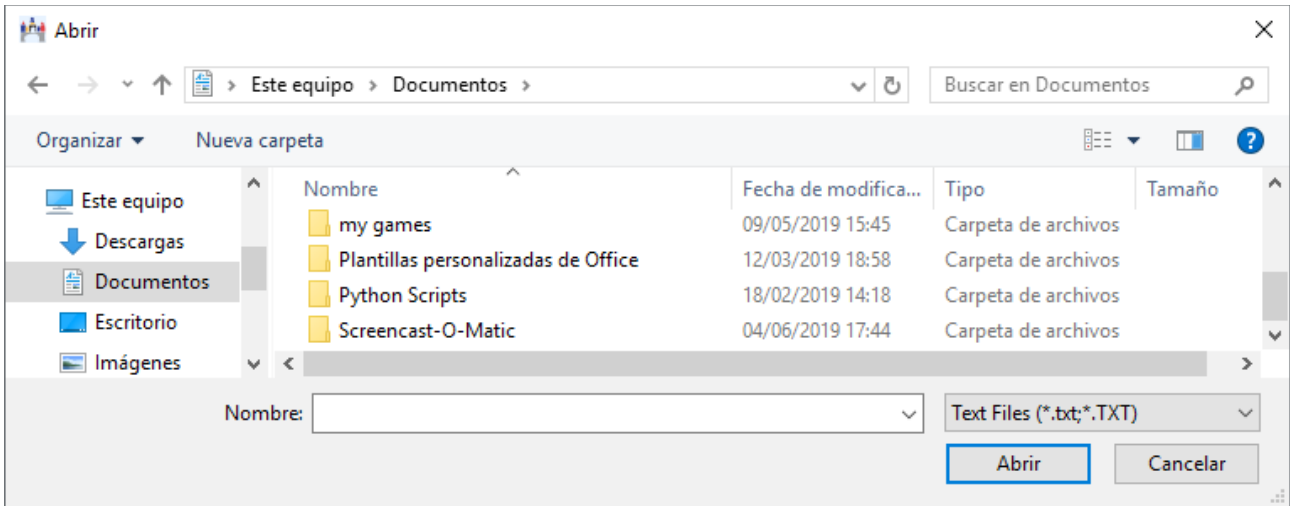


Ilustración 34: Ejemplo de selección de archivo.

Una vez se ha seleccionado un archivo, hacer click izquierdo en Abrir y el archivo se cargará, se calcularán las métricas y se pasará a la pantalla de resultados.

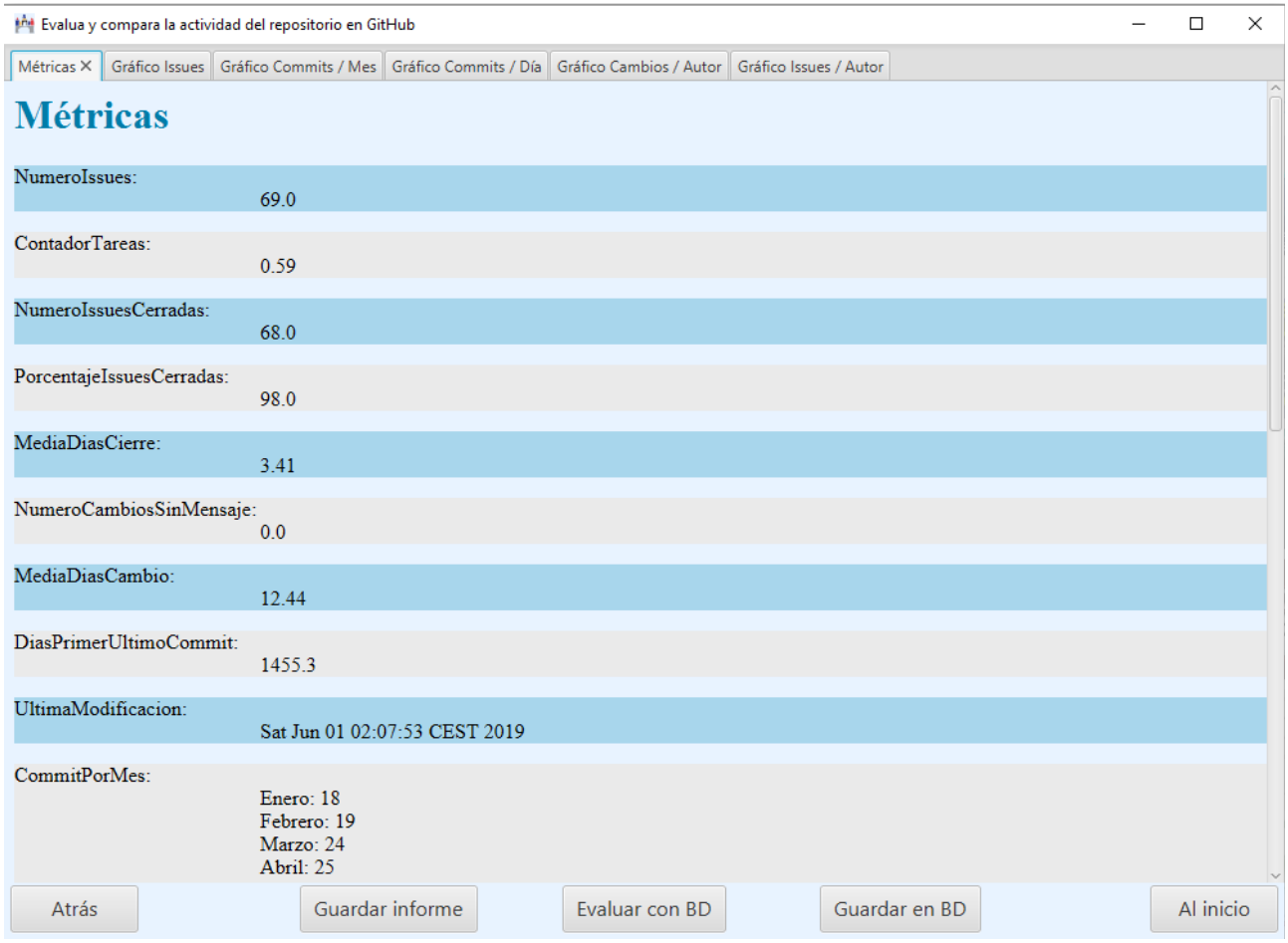


Ilustración 35: Ejemplo de la pantalla de resultados.

4.6. Comparar informes

Para comparar dos informes previamente guardados hay que realizar click izquierdo en la opción Operaciones del menú superior izquierdo disponible en la pantalla de inicio. Tras ello hay que realizar click izquierdo en la opción Comparar.

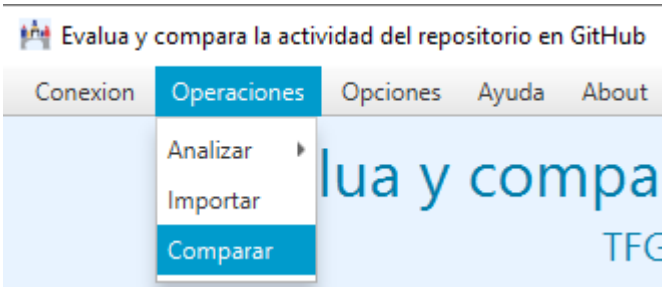


Ilustración 36: Ejemplo de Comparar informes.

Nos llevará a la pantalla de comparación.

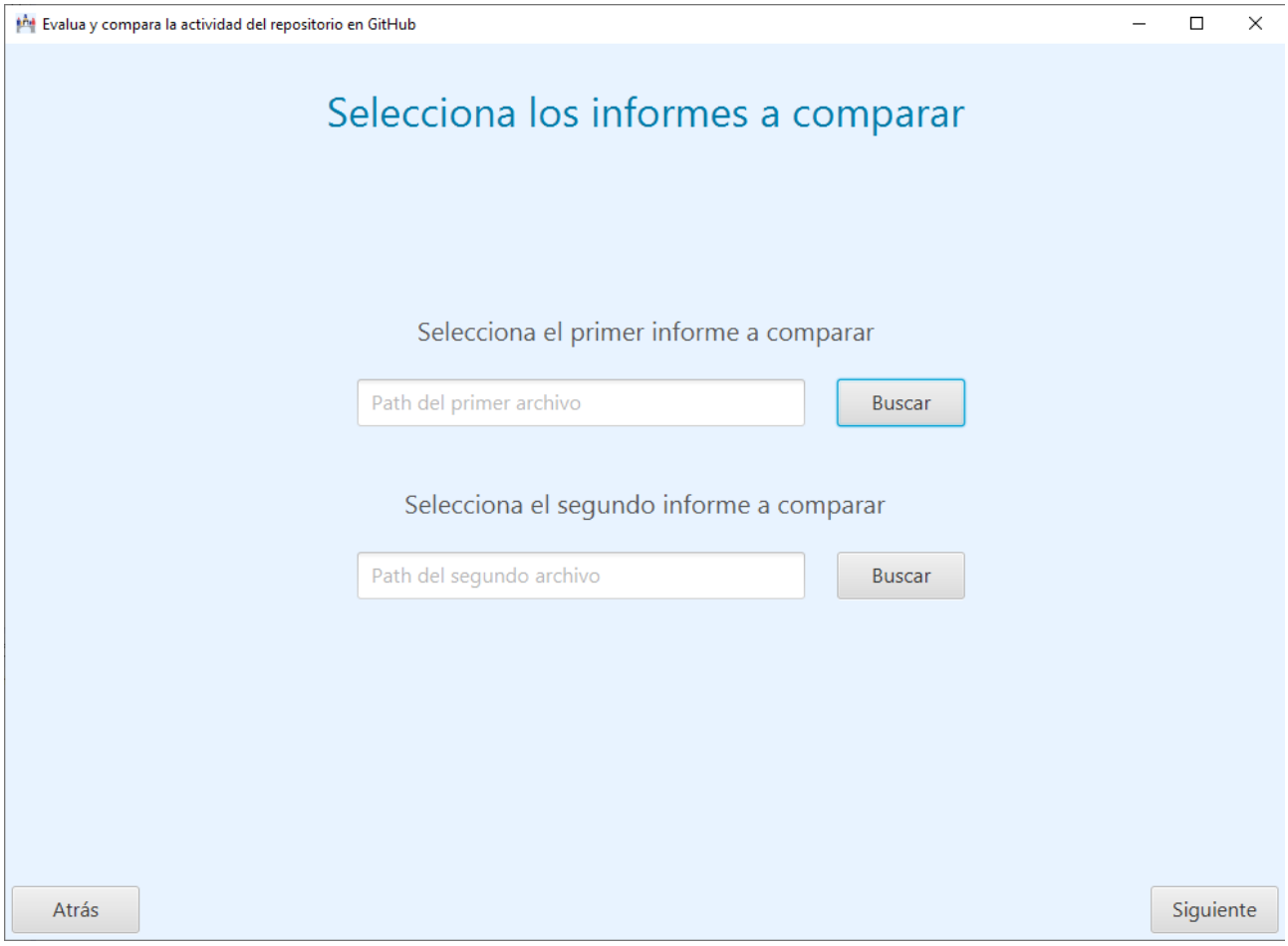


Ilustración 37: Ejemplo de la pantalla de comparación.

Una vez estemos en esta pantalla, se ha de realizar click en cada uno de los botones Buscar para seleccionar los informes a comparar.

Ambos informes han de ser distintos.

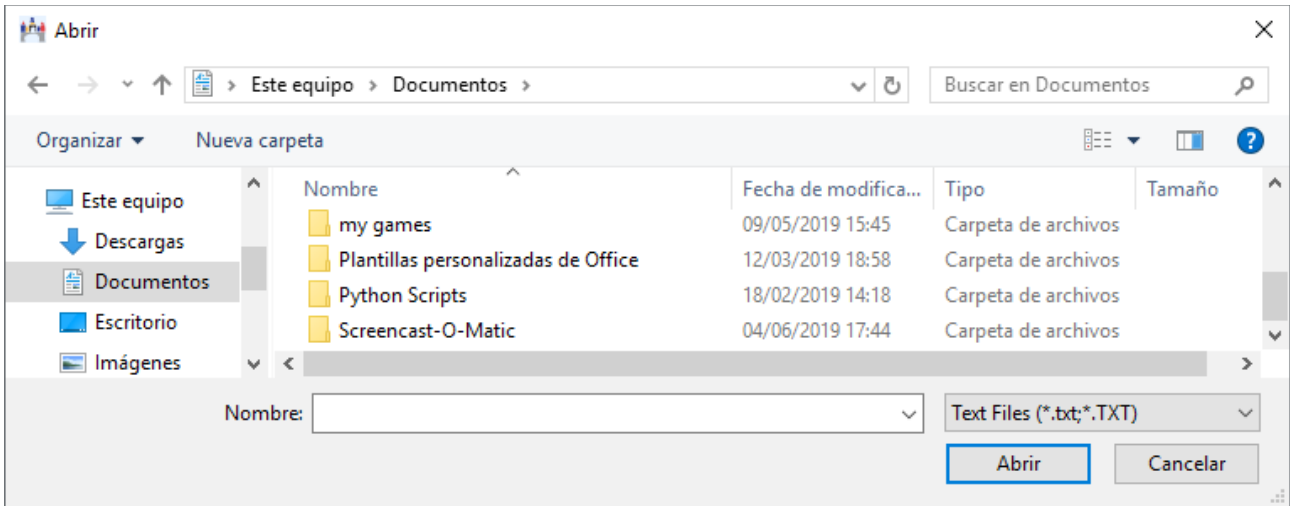
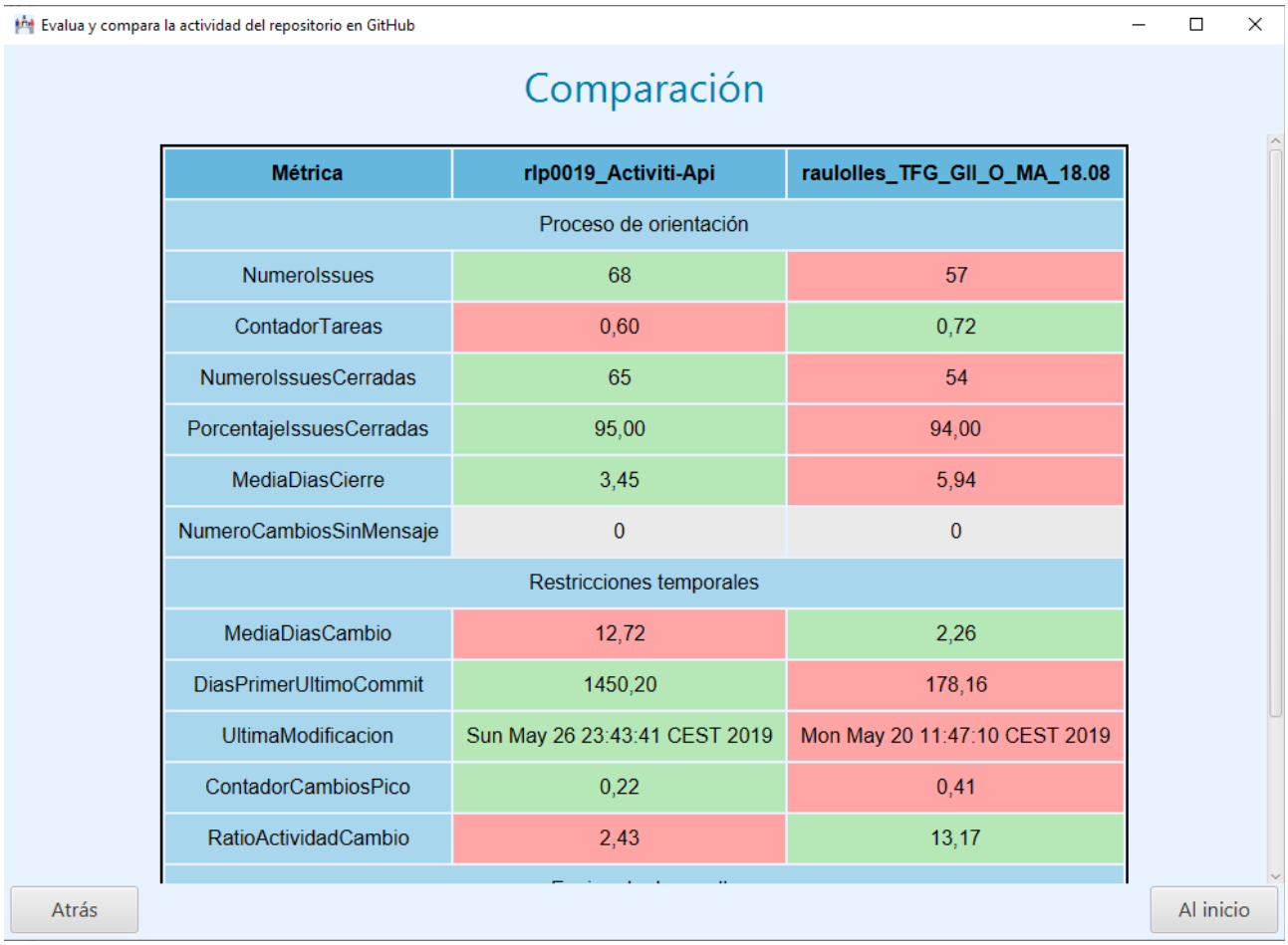


Ilustración 38: Ejemplo de selección de archivo.

Una vez se han escogido dos archivos distintos (su path aparece en los campos blancos), se pulsa en el botón Siguiente que nos llevará a la pantalla con la tabla resultado de la comparación.



Métrica	rlp0019_Activiti-Api	raulolles_TFG_GII_O_MA_18.08
Proceso de orientación		
NumeroIssues	68	57
ContadorTareas	0,60	0,72
NumeroIssuesCerradas	65	54
PorcentajeIssuesCerradas	95,00	94,00
MediaDiasCierre	3,45	5,94
NumeroCambiosSinMensaje	0	0
Restricciones temporales		
MediaDiasCambio	12,72	2,26
DiasPrimerUltimoCommit	1450,20	178,16
UltimaModificacion	Sun May 26 23:43:41 CEST 2019	Mon May 20 11:47:10 CEST 2019
ContadorCambiosPico	0,22	0,41
RatioActividadCambio	2,43	13,17

Ilustración 39: Ejemplo de la pantalla con la tabla resultado de la comparación.

4.7. Analizar repositorio

Para analizar un repositorio GitHub (la única plataforma disponible), hay que realizar click izquierdo en la opción Operaciones del menú superior disponible en la pantalla de inicio y a continuación, click izquierdo en la opción GitHub del submenú Analizar.

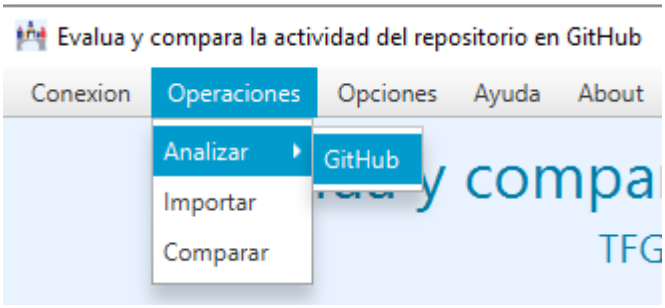


Ilustración 40: Ejemplo de Analizar repositorio GitHub.

Esto nos llevará a la pantalla de selección de conexión.

Ilustración 41: Ejemplo de pantalla de selección de conexión.

Seleccionamos el modo de conexión que queremos:

- Si es modo desconectado habrá que realizar click izquierdo en el botón Siguiente y aceptar la advertencia (que indica que disponemos de un menor número de peticiones).
- Si es modo conectado, hay que introducir los datos de nuestra cuenta GitHub, es decir, su usuario y contraseña, y realizar click izquierdo en el botón Siguiente.

En ambos casos nos llevará a la pantalla de selección de repositorio.

Ilustración 42: Ejemplo de la pantalla de selección de repositorio.

En esta pantalla hay que introducir el nombre del usuario a buscar en GitHub y hacer click izquierdo en el botón Buscar.

Tras ello, las listas desplegables de Repositorios y Forks se llenarán con los repositorios y forks del usuario en cuestión y habrá que seleccionar uno para proceder a obtener sus métricas.

Si se desea eliminar una selección, hay que realizar click izquierdo en el botón con un aspa roja del campo que se desea limpiar.

Una vez se ha seleccionado el repositorio o fork que se desee, se hace click izquierdo en el botón Siguiente, que ahora estará habilitado. Ello nos llevará a la pantalla de resultados.

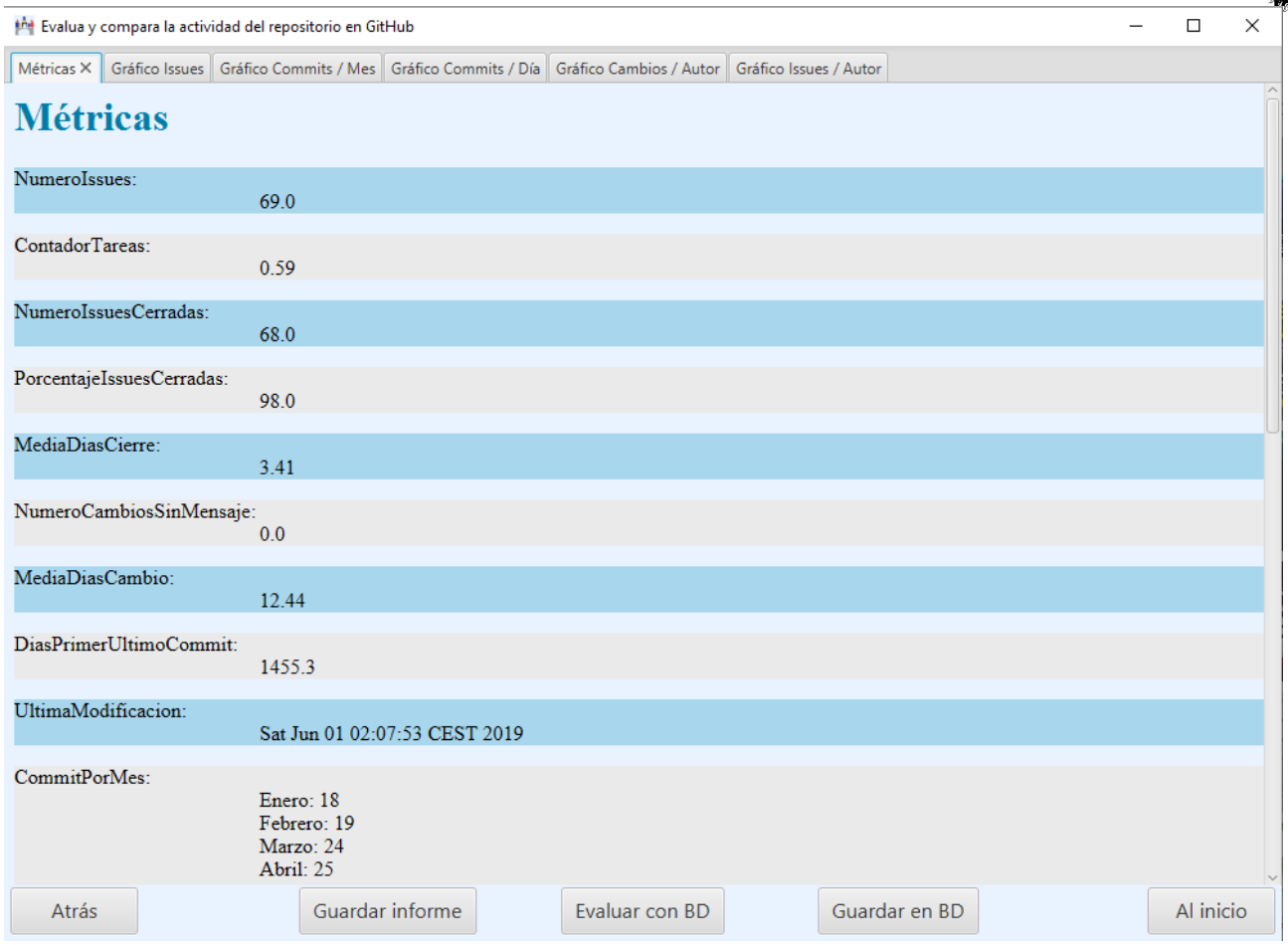


Ilustración 43: Ejemplo de la pantalla de resultados.

4.8. Guardar informe

Para guardar un informe hay que estar en la pantalla de resultados.

Se tiene que hacer click izquierdo en el botón Guardar informe, que abrirá el selector de archivo donde escribir el nombre del archivo a guardar y realizar click izquierdo en el botón Guardar para poder guardar el informe en formato .txt.

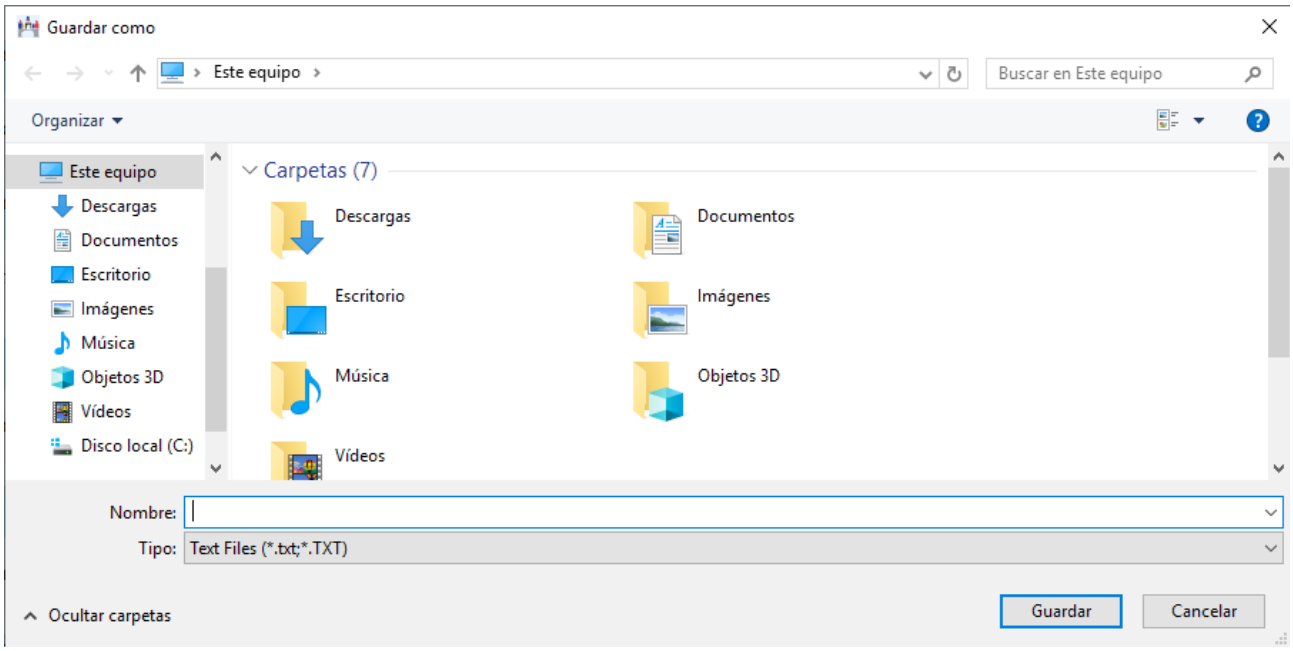


Ilustración 44: Ejemplo de selector para guardar archivo.

4.9. Evaluar con base de datos

Para evaluar un repositorio con la base de datos hay que estar en la pantalla de resultados.

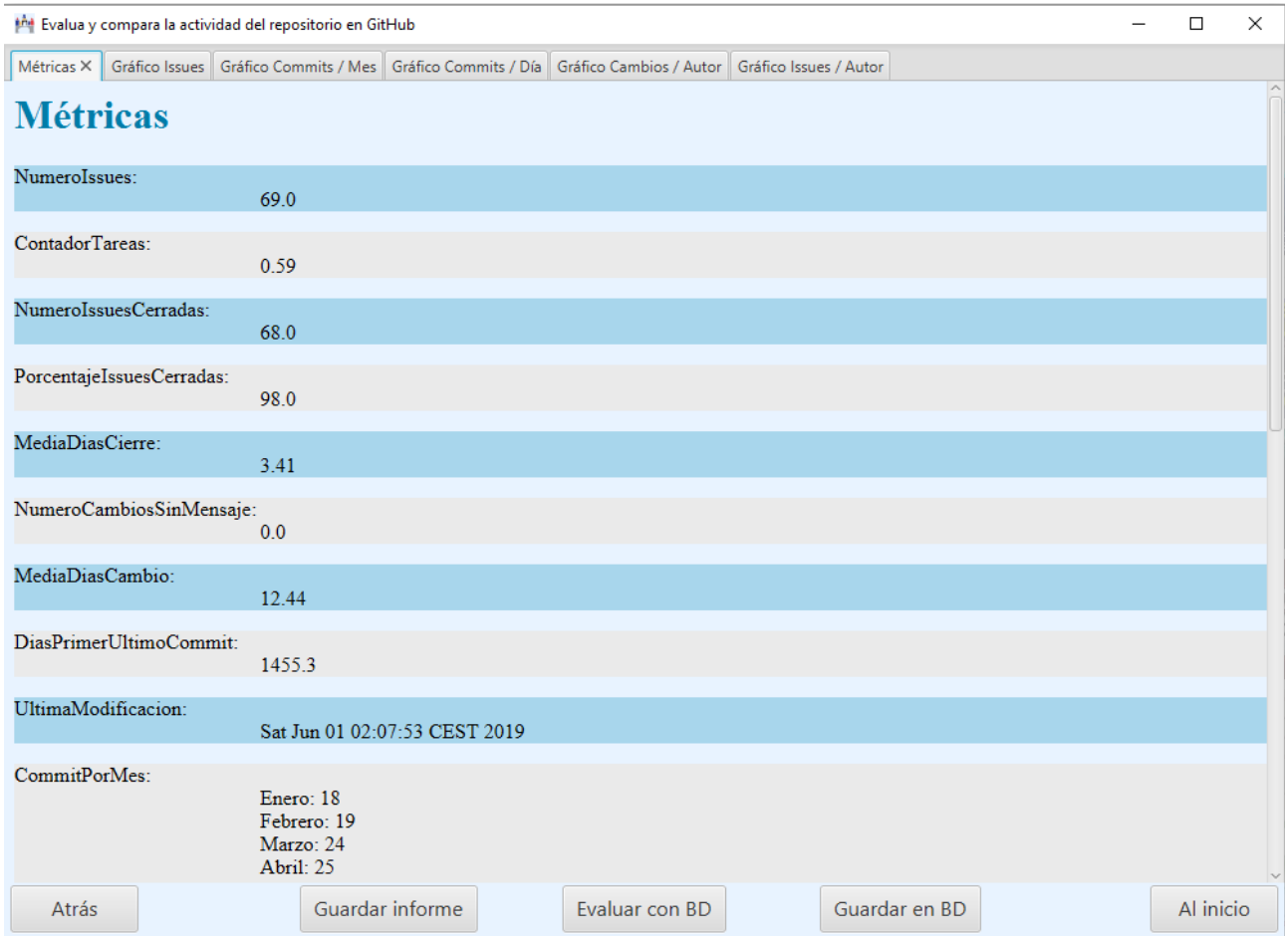


Ilustración 45: Ejemplo de la pantalla resultados.

Hay que realizar click en el botón Evaluar con BD y seleccionar la base de datos que trae la propia aplicación (tras extraerla del .zip) en el selector y volver a hacer click izquierdo en el botón Abrir.

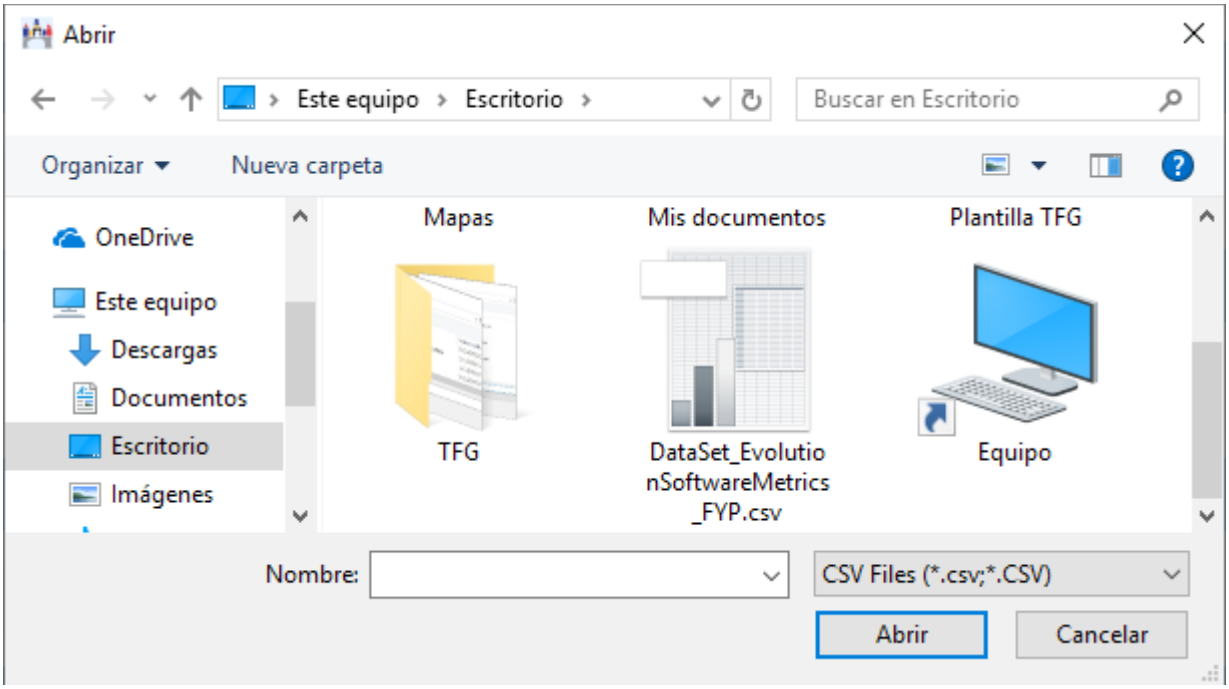


Ilustración 46: Ejemplo del selector de archivos .csv.

Se pasará entonces a la pantalla que contiene la tabla con la evaluación con la base de datos.

Métrica	Q1	davidmigloz_go-bees	Q3	Recomendado
Proceso de orientación				
NumeroIssues	6.0	207.0	53.0	Mayor que Q1
ContadorTareas	0.08	0.49	0.54	Entre Q1 y Q3
PorcentajeIssuesCerradas	87.0	90.0	100.0	Mayor que Q1
MediaDiasCierre	2.22	10.6	20.66	Entre Q1 y Q3
Restricciones temporales				
MediaDiasCambio	1.08	1.01	5.14	Entre Q1 y Q3
DiasPrimerUltimoCommit	81.58	424.22	206.68	Entre Q1 y Q3
ContadorCambiosPico	0.38	0.29	0.62	Entre Q1 y Q3
RatioActividadCambio	5.62	30.14	26.4	Entre Q1 y Q3

Ilustración 47: Ejemplo de la pantalla de evaluación con la base de datos.

Dentro de esta pantalla aparecen dos botones para poder cambiar la nota obtenida de la evaluación. La nota estricta evalúa los valores iguales que el primer y tercer cuartil (Q1 y Q3) como no válidos y la nota no estricta los evalua como medios.

4.10. Guardar en base de datos

Para guardar los resultados de un repositorio en la base de datos hay que estar en la pantalla de resultados.

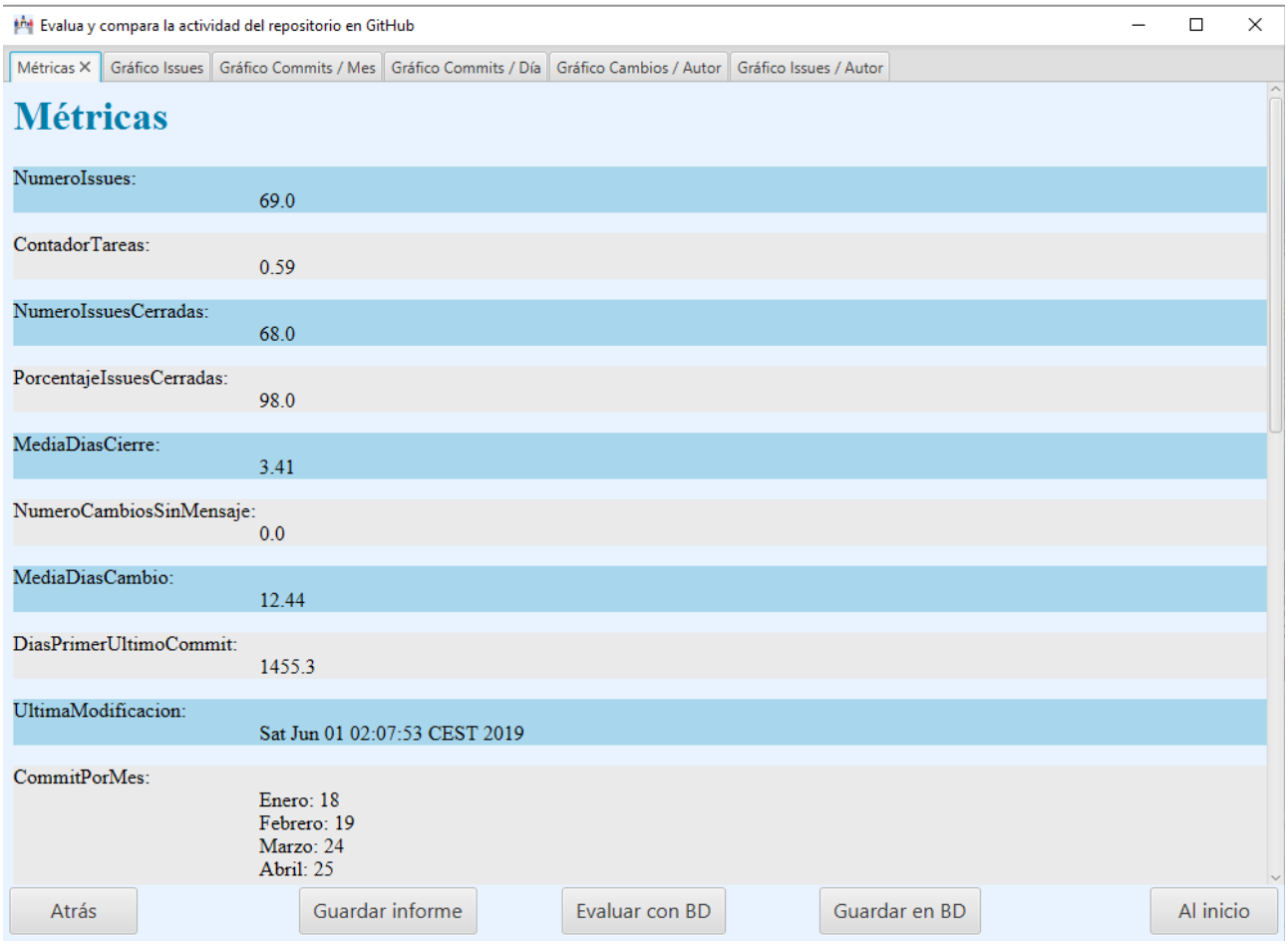


Ilustración 48: Ejemplo de la pantalla de resultados.

Hay que realizar click izquierdo en el botón Guardar en BD y aparecerá un selector para abrir el archivo .csv que se distribuye con la propia aplicación (o uno propio que tenga una estructura similar).

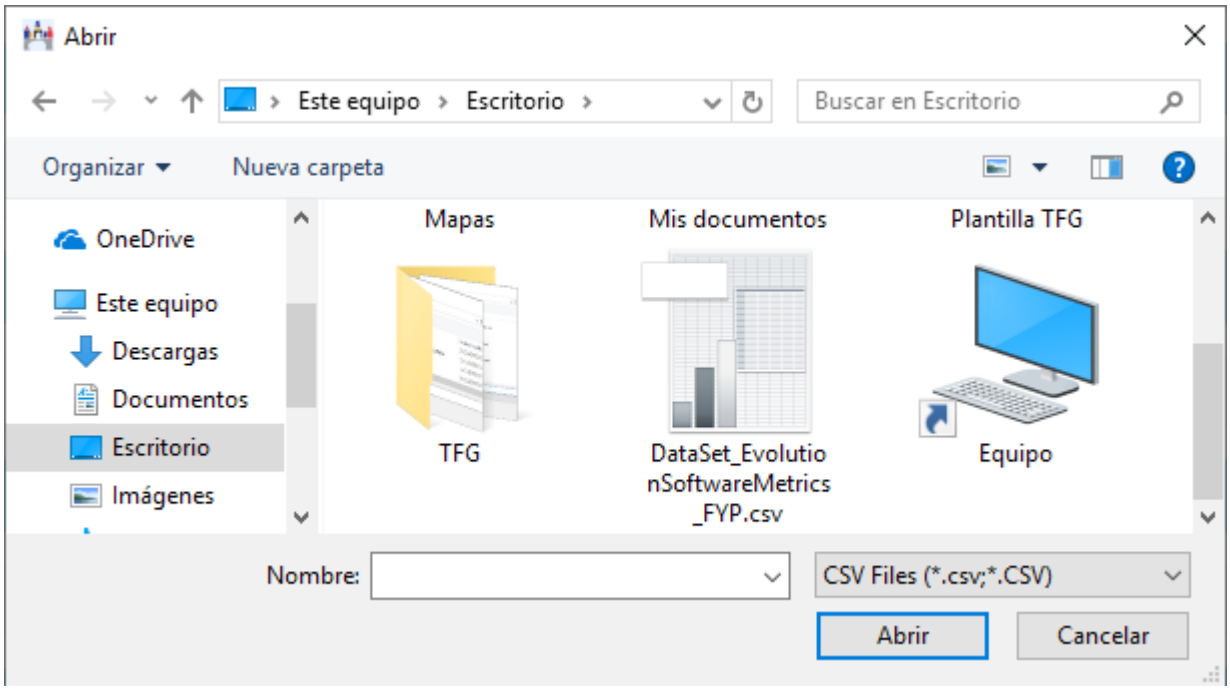


Ilustración 49: Ejemplo del selector de archivos .csv.

Una vez se ha seleccinado el archivo .csv, se hace click izquierdo en el botón Abrir y se guardarán las métricas del proyecto.

VI - BIBLIOGRAFÍA

[1]“Sprint 1 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/1?closed=1. [Accessed: 05-Jun-2019].

[2]“Sprint 2 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/2?closed=1. [Accessed: 05-Jun-2019].

[3] “Sprint 3 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/3?closed=1. [Accessed: 05-Jun-2019].

[4] “Sprint 4 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/4?closed=1. [Accessed: 05-Jun-2019].

[5] “Sprint 5 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/5?closed=1. [Accessed: 05-Jun-2019].

[6] “Sprint 6 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/6?closed=1. [Accessed: 05-Jun-2019].

[7] “Sprint 7 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/7?closed=1. [Accessed: 05-Jun-2019].

[8] “Sprint 8 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/8?closed=1. [Accessed: 05-Jun-2019].

[9] “Sprint 9 Milestone.” [Online]. Available: https://github.com/rlp0019/Activiti-Api/milestone/9?closed=1. [Accessed: 05-Jun-2019].

[10] “Tabla de coeficientes de amortización lineal. - Agencia Tributaria.” [Online]. Available: https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml. [Accessed: 05-Jun-2019].

[11] “PcCom Basic Elite AMD Ryzen 5 2600/8GB/1TB+240GB SSD/GT730.” [Online]. Available: https://www.pccomponentes.com/pccom-basic-elite-amd-ryzen-5-2600-8gb-1tb-240gb-ssd-gt730. [Accessed: 05-Jun-2019].

[12] “Comprar Windows 10 Pro - Microsoft Store es-ES.” [Online]. Available: https://www.microsoft.com/es-es/p/windows-10-pro/df77x4d43rkt/48DN. [Accessed: 05-Jun-

2019].

[13] “Ofertas de trabajo de Ingeniero informatico.” [Online]. Available: https://www.infojobs.net/ofertas-trabajo/ingeniero-informatico. [Accessed: 05-Jun-2019].

[14] “Alquiler de Oficina en Santa Lucia, Centro, Aranda de Duero.” [Online]. Available: https://www.idealista.com/inmueble/85574026/. [Accessed: 05-Jun-2019].

[15] “Math – User Guide - Overview.” [Online]. Available: https://commons.apache.org/proper/commons-math/userguide/overview.html. [Accessed: 04-Jun-2019].

[16] “gson/LICENSE at master · google/gson.” [Online]. Available: https://github.com/google/gson/blob/master/LICENSE. [Accessed: 04-Jun-2019].

[17] “Hamcrest.” [Online]. Available: http://hamcrest.org/. [Accessed: 04-Jun-2019].

[18] “JaCoCo - License.” [Online]. Available: https://www.jacoco.org/jacoco/trunk/doc/license.html. [Accessed: 04-Jun-2019].

[19] “javahelp/LICENSE at master · javaee/javahelp.” [Online]. Available: https://github.com/javaee/javahelp/blob/master/LICENSE. [Accessed: 04-Jun-2019].

[20] “JUnit – Project License.” [Online]. Available: https://junit.org/junit4/license.html. [Accessed: 04-Jun-2019].

[21] “Requisito (sistemas) - Wikipedia, la enciclopedia libre.” [Online]. Available: https://es.wikipedia.org/wiki/Requisito_(sistemas). [Accessed: 04-Jun-2019].

[22] E. Cerrillos, “Patrones de diseño de interfaz,” 19-Oct-2017. [Online]. Available: https://usersmatter.co/2017/10/19/patrones-de-diseno-de-interfaz/. [Accessed: 02-Jun-2019].

[23] The Apache Software Foundation, “Apache Ant.” [Online]. Available: https://ant.apache.org/. [Accessed: 15-May-2019].

[24] XebiaLabs, “Codacy,” *XebiaLabs*. [Online]. Available: https://xebialabs.com/technology/codacy/. [Accessed: 30-Mar-2019].

[25] “Travis CI - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Travis_CI. [Accessed: 05-Jun-2019].

Impreso en Burgos el miércoles, 5 de junio de 2019