

UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería



Informática

Monitor multiplataforma de la actividad de un proyecto

Trabajo final del GºIng.Informática

Alumnos David Blanco Alonso

Tutor Carlos López Nozal

DEPARTAMENTO DE INGENIERÍA CIVIL

Área de Lenguajes y Sistemas Informáticos

Burgos, 4 de febrero de 2016



Índice de contenido

1	Introducción.....	1
2	Planificación temporal.....	1
2.1	Iteraciones.....	1
2.1.1	1ª Iteración.....	1
2.1.2	2ª Iteración.....	2
2.1.3	3ª Iteración.....	2
2.1.4	4ª Iteración.....	3
2.1.5	5ª Iteración.....	3
3	Estudio de viabilidad.....	4
3.1	Viabilidad económica.....	4
3.1.1	Coste de personal.....	4
3.1.2	Amortización hardware y software.....	5
3.1.3	Gastos generales.....	5
3.1.4	Coste total.....	5
3.2	Viabilidad legal.....	5
4	Referencias.....	6

Índice de ilustraciones

Índice de tablas

Tabla 3.1:	Gastos generales.....	5
Tabla 3.2:	Coste total del desarrollo.....	5



I - INTRODUCCIÓN

Dado el entorno del proyecto que se va a desarrollar no se ha realizado una planificación propiamente dicha, ya que el desarrollo se ha llevado a cabo utilizando metodologías ágiles [1], esto quiere decir que el desarrollo del proyecto se realiza mediante iteraciones. Por lo que la planificación se realiza en cada iteración para organizar los objetivos y requisitos que se pretenden cumplir en esa iteración de cara a crear un prototipo.

Para mostrar el proceso de planificación se van a describir las iteraciones, los objetivos y requisitos marcados en cada una y los resultados obtenidos.

También se muestra un estudio de viabilidad tanto económica como legal del proyecto.

II - PLANIFICACIÓN TEMPORAL

En este apartado se van a presentar las distintas iteraciones realizadas durante el desarrollo del proyecto, indicando los objetivos o requisitos que se marcaron para cada una, los objetivos y requisitos que se cumplieron durante la iteración, problemas que se presentaron y resultados obtenidos.

Los roles participantes en el desarrollo del proyecto son los siguientes:

- Product owner: Es el rol que representa al cliente, en este caso el cliente es el tribunal de proyectos.
- ScrumMaster: Es el rol que se asegura del correcto uso del scrum, el cual corresponde al tutor.
- Equipo de desarrollo: Es el rol encargado realizar el producto, en este caso el alumno.

El proyecto ha sido realizado durante dos convocatorias, pero debido a cambios realizados al inicio de la segunda convocatoria se descarto casi todo el trabajo previo por lo que se va a obviar en la exposición de las iteraciones.

1. Iteraciones

1.1. 1ª Iteración

La reunión de planificación se realizó el 03 de diciembre del 2015.

En dicha reunión se puso de manifiesto la necesidad de realizar la autenticación de la conexión para mejorar la tasa de peticiones a las plataformas. En este momento el proyecto estaba enfocado a realizar peticiones a las plataformas GitHub [2] y Bitbucket [3]. Otro de los objetivos era controlar la paginación de las respuestas rest porque se detecto que se estaba perdiendo información cuando las respuestas eran demasiado largas. También se encontraron varias duplicidades en el código que representaban un

problema de cara a futuras ampliaciones de código por aumentar su complejidad de forma innecesaria, por lo que se propuso realizar una revisión del código para eliminarlas.

Durante esta iteración encontré la biblioteca GitHub Java API [4] que cumplía los objetivos marcados para la plataforma GitHub, y realicé una búsqueda de una biblioteca complementaria para cumplir los requisitos de cara a Bitbucket, pero sin resultados. Realice una revisión del código existente hasta el momento y elimine varias duplicidades que dificultaban la comprensión del código y aumentaban su complejidad.

El resultado de esta iteración era una aplicación que permitía seleccionar un repositorio y mostraba el número total de issues, issues cerradas, porcentaje y última modificación.

1.2. 2ª Iteración

La reunión de planificación se realizó el 17 de diciembre del 2015.

Ante la proximidad de la entrega y las limitaciones del actual prototipo se decidió descartar la plataforma Bitbucket y centrarse en GitHub. Se marco como requisito la creación de un test mediante JUnit [5] para controlar el correcto funcionamiento de la aplicación utilizando repositorios conocidos que no se volverían a modificar. Se plantearon problemas con la visibilidad de algunos métodos que podían suponer problemas de seguridad y funcionamiento. Se introdujo la implementación de un motor de métricas para mejorar el control de las mismas. Se marco como objetivo principal de la aplicación la creación de un conjunto de métricas que mostraran información de los repositorios. Como objetivo secundario se declaró la necesidad de mejorar la presentación de los resultados en la aplicación por medio de creación de informes o gráficos.

Durante esta iteración realice un test para controlar la correcta recepción y cálculo de la información, revisé varias clases para comprobar la visibilidad de sus métodos, implemente un motor de métricas siguiendo el framework propuesto en “Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones” [6] y añadí un conjunto de métricas extraídas de la tesis sPACE: Software Project Assessment in the Course of Evolution, desarrollada por Jacek Ratzinger [7]. Realice pruebas para realizar reportes con los resultados obtenidos con el conjunto de métricas sin obtener un resultado satisfactorio.

La aplicación en resultante de esta iteración permitía realizar una conexión a la plataforma GitHub [2] para aumentar la tasa de peticiones permitidas y seleccionar un repositorio existente en la plataforma para realizar el cálculo de las métricas declaradas y mostrándolas en una lista.

1.3. 3ª Iteración

La reunión de planificación se realizó el 13 de enero del 2016.

En esta reunión el objetivo principal era la creación de una interfaz de usuario más completa y la mejora de la muestra de los resultados. Se encontraron algunos errores matemáticos ante falta de datos al calcular las métricas. Se propuso la funcionalidad de guardar y cargar informes en archivos. Se planteó una mejora del funcionamiento del

motor de métricas a la hora de guardar los datos calculados.

En esta iteración revise los cálculos de las métricas para protegerlos ante posibles incongruencias en los datos que producían errores matemáticos. Realice la mejora planteada para el funcionamiento del motor de métricas y realice una nueva revisión del código para eliminar duplicidades del mismo. Cree una nueva interfaz dividida en distintas pantallas que permitía realizar las mismas operaciones que la interfaz anterior pero con un aspecto mucho más intuitivo y con una pantalla de ayuda para resolver posibles dudas, También se crearon varios gráficos para mostrar algunas métricas de forma más clara.

El resultado de esta iteración fue un prototipo que contaba con una interfaz gráfica más completa que permite un control más sencillo al usuario mostrando los resultados de manera más clara por medio de gráficos.

1.4. 4ª Iteración

La reunión de planificación se realizó el 22 de enero del 2016.

En dicha reunión se marco como objetivo la mejora de las pantallas de ayuda, añadir una funcionalidad para comparar informes de repositorios y la creación de una release para añadirla al repositorio alojado en GitHub [2].

En esta iteración mejore las pantallas de ayuda añadiendo imágenes de las pantallas explicadas, añadí la funcionalidad que permitía comparar dos informes ya guardados en archivos y cree una release para añadirla al repositorio para permitir una prueba más sencilla de la aplicación. También realice algunas pruebas y mejoras en el manejo de los archivos de informes a la hora de guardarlos y cargarlos.

El resultado fue la aplicación que se encuentra alojada en el repositorio pero con algunos fallos que se resolvieron en la iteración posterior.

1.5. 5ª Iteración

La reunión de planificación se realizó el 29 de enero del 2016.

En esta reunión se plantearon dudas sobre la realización de la documentación o memoria del proyecto y se encontraron algunos fallos en el funcionamiento de la aplicación.

Durante esta iteración comencé a completar la memoria con los apartados que faltaban y resolví algunos errores de funcionamiento de la aplicación resultante del proyecto.

El resultado es la release alojada en el repositorio.

Al finalizar esta iteración se realizó una reunión para revisar la calidad de la documentación de cara a realizar la entrega al tribunal.

III - ESTUDIO DE VIABILIDAD

Normalmente antes de comenzar el desarrollo de un proyecto es recomendable realizar un estudio de viabilidad tanto desde el punto de vista económico como legal, para saber si nos podemos embarcar en el desarrollo o debemos descartarlo antes de perder dinero o entrar en una batalla legal.

1. Viabilidad económica

Para realizar el estudio económico primero debemos saber cuanto supone su desarrollo, coste de personal, amortización hardware y software y gastos generales, para poder enfrentarlos a los beneficios esperados y saber si es viable el desarrollo desde el punto de vista económico.

1.1. Coste de personal

Voy a proceder a calcular el sueldo del desarrollador en primer lugar. El proyecto ha tenido una duración de 5 meses (desde septiembre hasta febrero) durante los cuales se supone una jornada de 4 horas diarias con un coste de 10€ hora.

- 5 meses x 5 días semanales x 4 semanas = 100 días
- 100 días x 4 horas diarias = 400 horas
- 400 horas x 10€/hora = 4000€ sueldo

La estimación da un sueldo bruto de 4000€ para el desarrollador por todo el proyecto.

La seguridad social [8] supone un 29,9%, de los cuales:

- Un 23,6 de contingencias comunes.
- Un 5,5 por desempleo.
- Un 0,2 para FOGASA.
- Un 0,6 por formación.

Lo que supone que el sueldo final del desarrollador es:

- $4000€ \times 29,9\% = 1196€$

Ahora calculo el sueldo del tutor. Se han realizado 6 reuniones durante el desarrollo del proyecto con una duración media de 2 horas, con un coste por hora de 15€.

- $6 \text{ reuniones} \times 2 \text{ horas/reunión} \times 15€/hora = 180€$

Aplicando la seguridad social:

- $180€ \times 29,9\% = 53,82€$

Los costes de personal totales son:

- $4000€ + 1196€ + 180€ + 53,82€ = 5429,82€$

1.2. Amortización hardware y software

La amortización software se realiza sobre el coste de las licencias de los programas y aplicaciones necesarios para realizar el desarrollo, en el caso de este proyecto todo el software utilizado es de licencia gratuita por lo que no existe ningún coste añadido.

Para la realización de este proyecto se ha utilizado un ordenador presupuestado en 300€. La amortización de equipos de computación está estipulada en 3 años, por lo que hay que calcular el coste del amortizado en el periodo de desarrollo del proyecto, esto es 5 meses.

- 3 años = 36 meses
- $300\text{€} / 36 \text{ meses} = 8,33\text{€/mes}$
- $5 \text{ meses} \times 8,33\text{€/mes} = 41,65\text{€}$

1.3. Gastos generales

En los gastos generales se introducen el gasto de luz, encuadernación y material de oficina.

Gasto de luz	50,00 €
Encuadernación	20,00 €
Material de oficina	30,00 €
Total	100,00 €

Tabla 1: Gastos generales.

1.4. Coste total

El coste total del desarrollo es la suma de todos los gastos calculados anteriormente:

Coste de personal	5.429,82 €
Amortización hardware	41,65 €
Gastos generales	100,00 €
Total	5.571,47 €

Tabla 2: Coste total del desarrollo.

Según el cálculo realizado el coste total del proyecto serían 5.571,47 € del que la mayor parte corresponde al sueldo del desarrollador.

2. Viabilidad legal

Dado que todas las licencias del software utilizado son de carácter gratuito, no existe ningún problema legal a la hora de realizar el desarrollo y de su posterior distribución o comercialización.

IV - REFERENCIAS

- [1] «Scrum - Wikipedia, la enciclopedia libre». [En línea]. Disponible en: <https://es.wikipedia.org/wiki/Scrum>. [Accedido: 29-ene-2016].
- [2] «GitHub · Where software is built». [En línea]. Disponible en: <https://github.com/>. [Accedido: 01-feb-2016].
- [3] «Bitbucket — The Git solution for professional teams». [En línea]. Disponible en: <https://bitbucket.org/>. [Accedido: 01-feb-2016].
- [4] «egit-github/org.eclipse.egit.github.core at master · eclipse/egit-github». [En línea]. Disponible en: <https://github.com/eclipse/egit-github/tree/master/org.eclipse.egit.github.core>. [Accedido: 29-ene-2016].
- [5] «JUnit - About». [En línea]. Disponible en: <http://junit.org/>. [Accedido: 30-ene-2016].
- [6] R. M. Sánchez, Y. C. González-Carvajal, y C. L. Nozal, «Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones», en *Actas de las X Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2005), September 14-16, 2005, Granada, Spain, 2005*, pp. 59–66.
- [7] Jacek Ratzinger, «sPACE Software Project Assessment in the Course of Evolution». .
- [8] «Seguridad Social:Trabajadores». [En línea]. Disponible en: http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm. [Accedido: 03-feb-2016].

UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería



Informática

Monitor multiplataforma de la actividad de un proyecto

Trabajo final del GºIng.Informática

Alumnos David Blanco Alonso

Tutor Carlos López Nozal

DEPARTAMENTO DE INGENIERÍA CIVIL

Área de Lenguajes y Sistemas Informáticos

Burgos, 4 de febrero de 2016



Índice de contenido

1	Introducción.....	1
2	Objetivos.....	1
3	Especificación de requisitos.....	1
4	Referencias.....	10

Índice de ilustraciones

Índice de tablas

Tabla 3.1:	Historia de usuario 1.....	2
Tabla 3.2:	Historia de usuario 2.....	2
Tabla 3.3:	Historia de usuario 3.....	2
Tabla 3.4:	Historia de usuario 4.....	3
Tabla 3.5:	Historia de usuario 5.....	3
Tabla 3.6:	Historia de usuario 6.....	3
Tabla 3.7:	Historia de usuario 7.....	4
Tabla 3.8:	Historia de usuario 8.....	4
Tabla 3.9:	Historia de usuario 9.....	4
Tabla 3.10:	Historia de usuario 10.....	5
Tabla 3.11:	Historia de usuario 11.....	5
Tabla 3.12:	Historia de usuario 12.....	5
Tabla 3.13:	Historia de usuario 13.....	6
Tabla 3.14:	Historia de usuario 14.....	6
Tabla 3.15:	Historia de usuario 15.....	6
Tabla 3.16:	Historia de usuario 16.....	7
Tabla 3.17:	Historia de usuario 17.....	7
Tabla 3.18:	Historia de usuario 18.....	7
Tabla 3.19:	Historia de usuario 19.....	8
Tabla 3.20:	Historia de usuario 20.....	8
Tabla 3.21:	Historia de usuario 21.....	8
Tabla 3.22:	Historia de usuario 22.....	9
Tabla 3.23:	Historia de usuario 23.....	9
Tabla 3.24:	Historia de usuario 24.....	9



V - INTRODUCCIÓN

Este anexo recoge los requisitos y objetivos del software que se pretende desarrollar durante el proyecto.

Como en el anexo anterior la especificación de requisitos se va a plantear según las metodologías ágiles, esto es según las historias de usuario [1]. Una historia de usuario es una indicación de una funcionalidad buscada por el cliente, constan de enunciado y criterios de aceptación, el enunciado es la especificación de la funcionalidad y los criterios de aceptación son las pruebas que se deben cumplir para dar por cumplida la funcionalidad.

VI - OBJETIVOS

Estos son las funcionalidades que se pretende cumplir con el proyecto de forma más abstracta que la especificada por las historias de usuario.

- Creación de un conjunto de métricas: Extraer un conjunto de métricas de las expuestas en la tesis sPACE: Software Project Assessment in the Course of Evolution, desarrollada por Jacek Ratzinger [2], de las cuales sea posible realizar el cálculo de su valor con los datos que se pueden obtener de un repositorio alojado en una plataforma.
- Crear una aplicación escalable: La aplicación resultante debe ser fácilmente escalable para añadir soporte a otras plataformas o para añadir nuevas métricas.
- Mostrar resultados de forma gráfica: Los resultados obtenidos del conjunto de métricas deben mostrarse por medio de gráficos y texto para que sea comprendida de forma sencilla.

VII - ESPECIFICACIÓN DE REQUISITOS

En este apartado se van a especificar las historias de usuario que especifican el funcionamiento buscado para esta aplicación.

Número: 1	Usuario: cliente
Nombre historia: Seleccionar un repositorio.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero seleccionar un repositorio para conocer su evolución del proyecto en el tiempo.	
Validación: <ul style="list-style-type: none"> • Se puede trabajar sin conexión con repositorios pequeños. • Se puede realizar una conexión para trabajar con repositorios grandes. • Se puede introducir un usuario propietario. • Se muestran los repositorios pertenecientes al usuario introducido. • Se puede seleccionar un repositorio de los mostrados. 	

Tabla 3: Historia de usuario 1.

Número: 2	Usuario: cliente
Nombre historia: Realizar conexión.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero poder autenticar en la plataforma para aumentar la tasa de peticiones.	
Validación: <ul style="list-style-type: none"> • Se puede indicar un usuario y una contraseña para realizar una conexión. • Al realizar una conexión se aumenta la tasa de peticiones. 	

Tabla 4: Historia de usuario 2.

Número: 3	Usuario: cliente
Nombre historia: Mostrar repositorios.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyectos quiero mostrar los repositorios pertenecientes a un usuario para poder seleccionar uno.	
Validación: <ul style="list-style-type: none"> • Se muestran todos los repositorios pertenecientes a un usuario. • Se puede seleccionar un repositorio de los mostrados para calcular las métricas de el. 	

Tabla 5: Historia de usuario 3.

Número: 4	Usuario: cliente
Nombre historia: Calcular métricas.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero obtener los valores de todas las métricas para conocer la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • Los datos recibidos del repositorio son correctos. • El cálculo de las métricas no produce errores. • Los resultados se muestran correctamente. • Se muestran gráficos de algunas métricas. 	

Tabla 6: Historia de usuario 4.

Número: 5	Usuario: cliente
Nombre historia: Calcular métrica CambioPorAutor.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica CambioPorAutor para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 7: Historia de usuario 5.

Número: 6	Usuario: cliente
Nombre historia: Calcular métrica IssuesPorAutor.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica IssuesPorAutor para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 8: Historia de usuario 6.

Número: 7	Usuario: cliente
Nombre historia: Calcular métrica ContadorAutor.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica ContadorAutor para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 9: Historia de usuario 7.

Número: 8	Usuario: cliente
Nombre historia: Calcular métrica NumeroIssues.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica NumeroIssues para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 10: Historia de usuario 8.

Número: 9	Usuario: cliente
Nombre historia: Calcular métrica NumeroIssuesCerradas.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica NumeroIssuesCerradas para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 11: Historia de usuario 9.

Número: 10	Usuario: cliente
Nombre historia: Calcular métrica MediaDiasCierre.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica MediaDiasCierre para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 12: Historia de usuario 10.

Número: 11	Usuario: cliente
Nombre historia: Calcular métrica PorcentajeIssuesCerradas.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica PorcentajeIssuesCerradas para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 13: Historia de usuario 11.

Número: 12	Usuario: cliente
Nombre historia: Calcular métrica NumeroCambiosSinMensaje.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica NumeroCambiosSinMensaje para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 14: Historia de usuario 12.

Número: 13	Usuario: cliente
Nombre historia: Calcular métrica ContadorTareas.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica ContadorTareas para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 15: Historia de usuario 13.

Número: 14	Usuario: cliente
Nombre historia: Calcular métrica MediaDiasCambio.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica MediaDiasCambio para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 16: Historia de usuario 14.

Número: 15	Usuario: cliente
Nombre historia: Calcular métrica DiasPrimerUltimoCommit.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica DiasPrimerUltimoCommit para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 17: Historia de usuario 15.

Número: 16	Usuario: cliente
Nombre historia: Calcular métrica UltimaModificacion.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica UltimaModificacion para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 18: Historia de usuario 16.

Número: 17	Usuario: cliente
Nombre historia: Calcular métrica CommitPorDia.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica CommitPorDia para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 19: Historia de usuario 17.

Número: 18	Usuario: cliente
Nombre historia: Calcular métrica CommitPorMes.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica CommitPorMes para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 20: Historia de usuario 18.

Número: 19	Usuario: cliente
Nombre historia: Calcular métrica RelacionMesPico.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica RelacionMesPico para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 21: Historia de usuario 19.

Número: 20	Usuario: cliente
Nombre historia: Calcular métrica ContadorCambiosPico.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica ContadorCambiosPico para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 22: Historia de usuario 20.

Número: 21	Usuario: cliente
Nombre historia: Calcular métrica RatioActividadCambio.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica RatioActividadCambio para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 23: Historia de usuario 21.

Número: 22	Usuario: cliente
Nombre historia: Calcular métrica NumeroFavoritos.	
Prioridad: alta	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero calcular la métrica NumeroFavoritos para mejorar la comprensión del estado y la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • El cálculo de la métrica se realiza de forma correcta. • El cálculo de la métrica no genera errores. • El resultado se guarda correctamente. 	

Tabla 24: Historia de usuario 22.

Número: 23	Usuario: cliente
Nombre historia: Funcionamiento de informes.	
Prioridad: media	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero gestionar informes para evitar tener que realizar peticiones innecesarias a la plataforma.	
Validación: <ul style="list-style-type: none"> • Se pueden guardar los resultados en un archivo. • Se puede leer un informe anteriormente guardado. 	

Tabla 25: Historia de usuario 23.

Número: 24	Usuario: cliente
Nombre historia: Comparación de informes.	
Prioridad: baja	
Programador responsable: David Blanco	
Descripción: Como gestor de proyecto quiero comparar informes para conocer mejor la evolución de un proyecto.	
Validación: <ul style="list-style-type: none"> • Se pueden seleccionar dos informes para compararlos. • Se muestran los resultados de la comparación. 	

Tabla 26: Historia de usuario 24.

VIII - REFERENCIAS

- [1] Mike Cohn, *User Stories Applied*. 2004.
- [2] Jacek Ratzinger, «SPACE Software Project Assessment in the Course of Evolution». .

UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería



Informática

Monitor multiplataforma de la actividad de un proyecto

Trabajo final del GºIng.Informática

Alumnos David Blanco Alonso

Tutor Carlos López Nozal

DEPARTAMENTO DE INGENIERÍA CIVIL

Área de Lenguajes y Sistemas Informáticos

Burgos, 4 de febrero de 2016



Índice de contenido

1	Introducción.....	1
2	Diseño arquitectónico.....	1
2.1	Paquete lector.....	1
2.2	Paquete motorMetricas.....	2
2.3	Paquete gui.....	3
2.4	Paquete charts.....	4
3	Diseño de la interfaz.....	4
4	Diseño procedimental.....	5
4.1	Conexión a la plataforma.....	6
4.2	Obtener datos del repositorio.....	6
4.3	Calcular una métrica.....	7
4.4	Leer informe.....	8
4.5	Comparar informes.....	9
5	Referencias.....	9

Índice de ilustraciones

Ilustración 1:	Paquete lector.....	1
Ilustración 2:	Paquete motorMetricas.....	2
Ilustración 3:	Paquete Gui.....	3
Ilustración 4:	Pantalla inicial de la aplicación.....	4
Ilustración 5:	Detalle del botón atrás.....	5
Ilustración 6:	Detalle del botón Guardar resultados.....	5
Ilustración 7:	Procedimiento conexión.....	6
Ilustración 8:	Procedimiento obtener datos del repositorio.....	6
Ilustración 9:	Procedimiento calcular métrica.....	7
Ilustración 10:	Procedimiento leer informe.....	8
Ilustración 11:	Procedimiento comparar informes.....	9

Índice de tablas



IX - INTRODUCCIÓN

En este anexo se describen los aspectos del diseño seguidos durante el desarrollo del proyecto para obtener una aplicación de calidad.

Los aspectos del diseño que se van a presentar son:

- Diseño arquitectónico: Diagramas de clases y exposiciones de los paquetes que integran la aplicación.
- Diseño de la interfaz: explicación detallada de las pantallas de la interfaz.
- Diseño procedimental: Diagramas de secuencias explicando su funcionamiento.

X - DISEÑO ARQUITECTÓNICO

En este apartado se describen las clases y la estructura de los paquetes de los que consta el proyecto software.

1. Paquete lector

Este paquete es el encargado de realizar la conexión a la plataforma, solicitar y recibir los datos de un repositorio y crear las métricas a calcular.

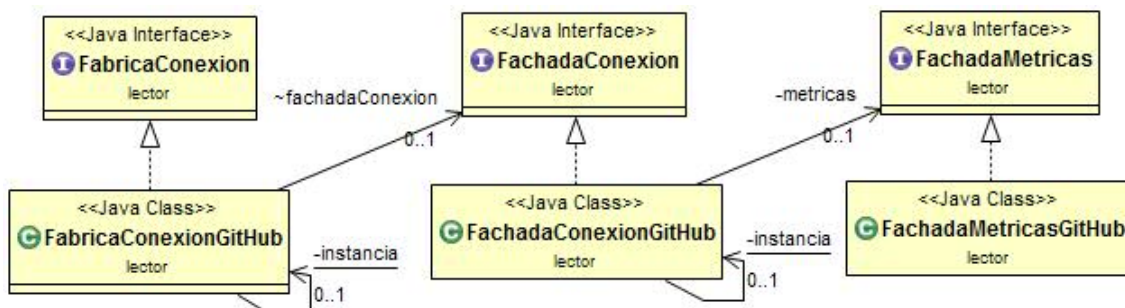


Ilustración 1: Paquete lector

- FabricaConexion: Es una clase interface para las fabricas abstractas encargadas de la creación de las fachadas.
- FabricaConexionGitHub: Es la fabrica encargada de crear la fachada para trabajar con la plataforma GitHub.
- FachadaConexion: Es la interface de las fachadas que realizan la comunicación con una plataforma.
- FachadaConexionGitHub: Es la clase encargada de todas las operaciones relacionadas con la plataforma.
- FachadaMetricas: Es la interface para los conjuntos de métricas asignados a los datos recibidos de una plataforma.

- **FachadaMetricasGitHub:** Es la clase que se encarga de realizar la comunicación con el motor de métricas, creando las métricas y trabajando con los resultados obtenidos.

2. Paquete motorMetricas

Este paquete contiene la versión implementada del framework sugerido en “Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones” [1] que maneja las métricas y los resultados obtenidos.

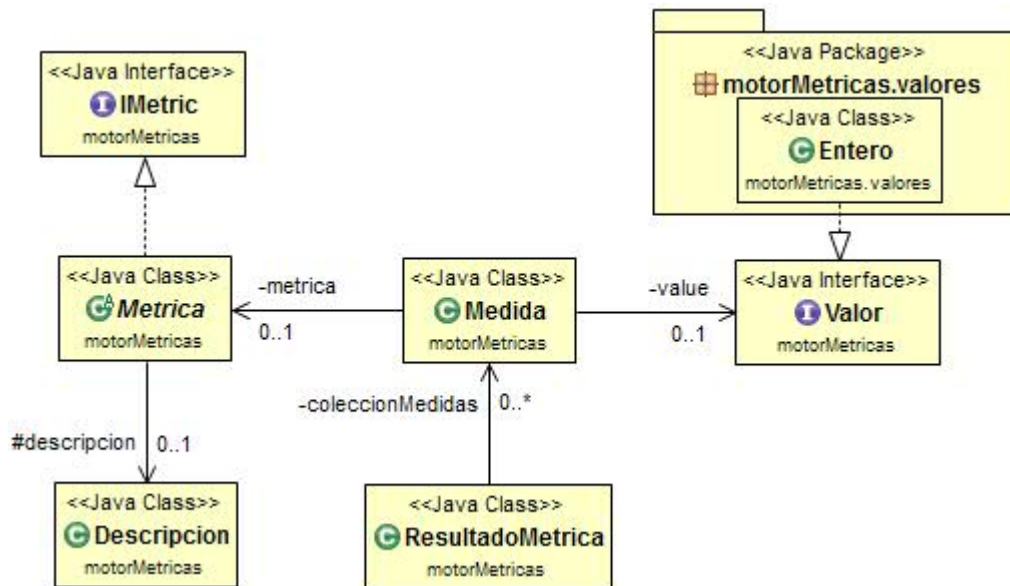


Ilustración 2: Paquete motorMetricas

- **IMetric:** Clase interface de las métricas.
- **Medica:** Clase de la que heredan todas las clases de las métricas que se añaden al proyecto.
- **Descripcion:** Clase que contiene la descripción de una métrica y los métodos para trabajar con ella.
- **Medida:** Es la clase que combina una métrica con el valor obtenido al realizar el cálculo.
- **ResultadoMetrica:** Clase que contiene el conjunto de las métricas calculadas asociadas con sus valores.
- **Valor:** Es una clase interface para los valores que pueden tomar las métricas.
- **motorMetricas.valores:** Paquete que contiene todas las clases de los posibles datos que pueden obtenerse al calcular una métrica.

3. Paquete gui

En este paquete se encuentran las clases que forman la interfaz. La mayoría de ellas son clases que heredan de la clase javax.swing.JPanel.

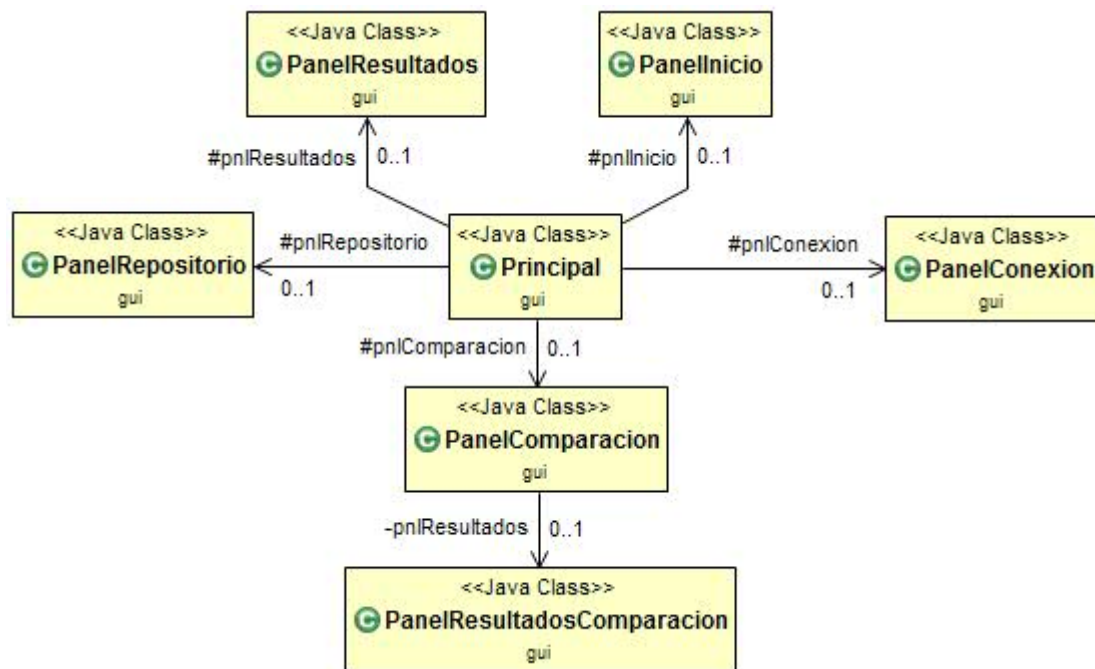


Ilustración 3: Paquete Gui.

- **Principal:** Esta clase crea un JFrame que contiene un menú para seleccionar la función que queremos realizar, un panel que contiene los botones y en la que se van cargando las distintas clases.
- **PanelInicio:** Es el panel que contiene los botones para seleccionar la plataforma a utilizar.
- **PanelConexion:** Es el panel que contiene los componentes para realizar la conexión o seleccionar el modo desconectado.
- **PanelRepositorio:** Es el panel que controla la selección de un repositorio.
- **PanelResultados:** Es el panel en el que se muestran los resultados del cálculo de métricas o de la lectura de informes.
- **PanelComparacion:** Es el panel en el que se seleccionan los informes a comparar.
- **PanelResultadosComparacion:** Es el panel en el que se muestran los resultados de comparar dos repositorios.

En este paquete se incluyen también los archivos necesarios para generar la ayuda de la aplicación.

4. Paquete charts

Este paquete contiene la clase con los métodos para crear los gráficos con los que mostrar las métricas.



- Graficos: Clase con los métodos para crear gráficos de barras o gráficos de tarta.

XI - DISEÑO DE LA INTERFAZ

El diseño de la interfaz es una de las partes más importantes, ya que es la parte de la aplicación con la que va a interactuar el usuario, por lo que es necesario que esta sea sencilla de usar.

El diseño se ha creado utilizando paneles que se van intercambiando según la acción a realizar o el paso en el que nos encontremos de dicha acción.

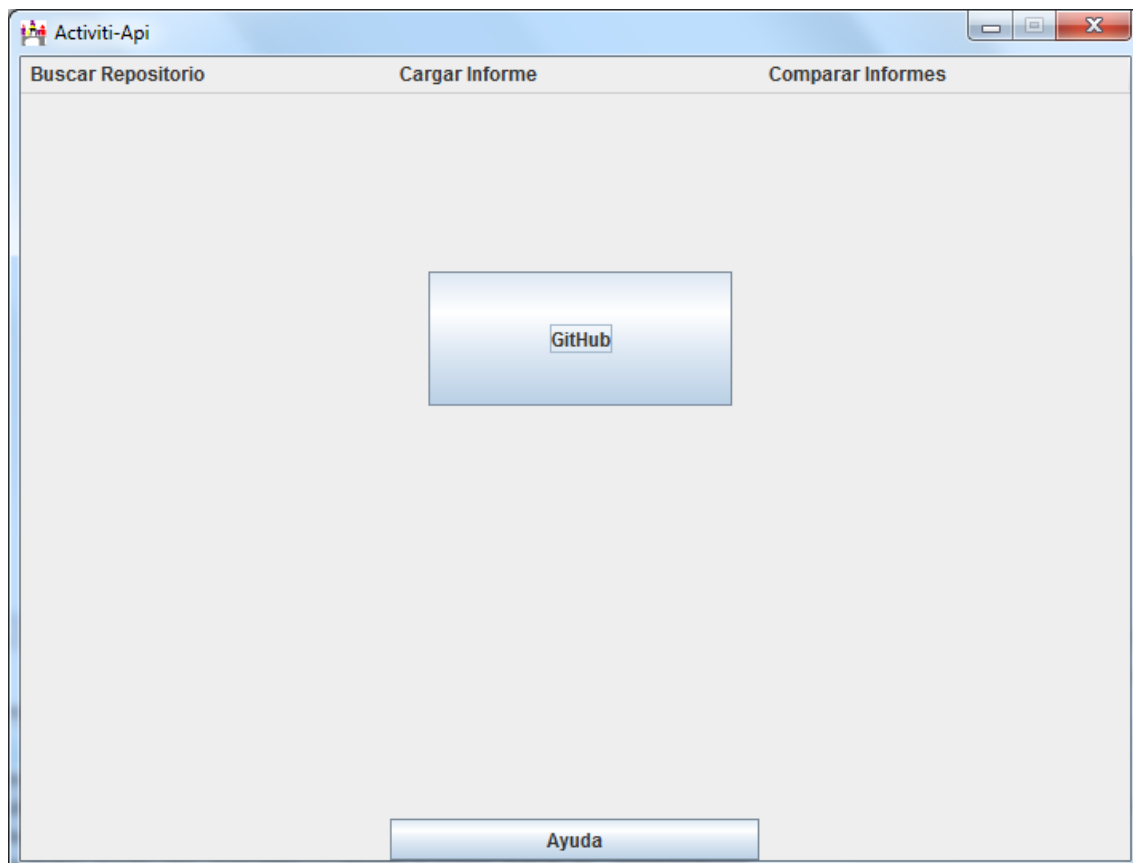


Ilustración 4: Pantalla inicial de la aplicación.

Esta es la pantalla inicial de la aplicación, lo primero que ve el usuario cuando comienza la ejecución, en ella se puede observar una barra de menú superior en la que se muestran las acciones implementadas en la aplicación.

- **Buscar repositorio:** Esta es la acción por defecto, permite al usuario buscar un repositorio en la plataforma y obtener su estado y evolución.
- **Cargar informe:** Esta opción permite seleccionar un informe anteriormente generado por la aplicación y mostrarlo en la aplicación.
- **Comparar informes:** Esta opción permite seleccionar dos informes previamente guardados y compararlos entre si.

En esta ventana inicial podemos observar también el botón ayuda que se mantendrá en esa misma posición en todas las pantallas.

En el resto de pantallas existe un botón atrás ubicado a la derecha del botón ayuda para permitir al usuario retroceder a la pantalla anterior.



Ilustración 5: Detalle del botón atrás.

En la pantalla resultados durante una búsqueda de repositorio aparece otro botón a la izquierda del botón ayuda que permite guardar un informe con los resultados del repositorio buscado.



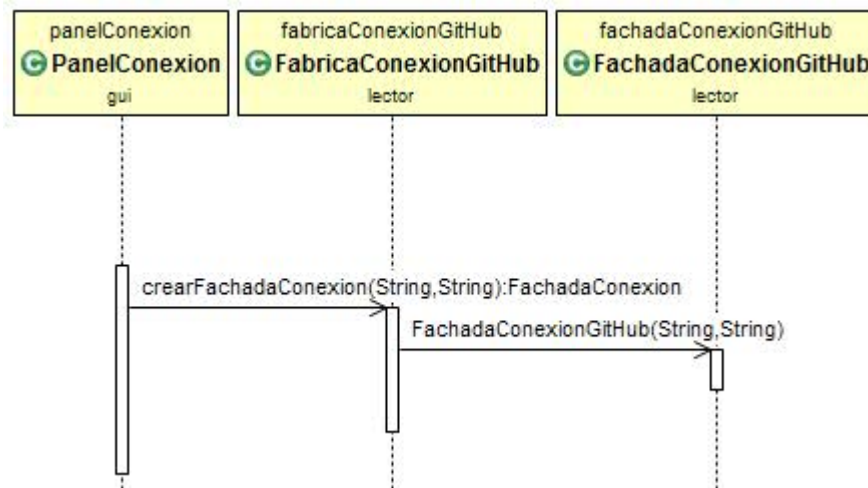
Ilustración 6: Detalle del botón Guardar resultados.

XII - DISEÑO PROCEDIMENTAL

En este apartado se muestra la secuencia que sigue el algoritmo al realizar cada operación por medio de diagramas de secuencia.

1. Conexión a la plataforma

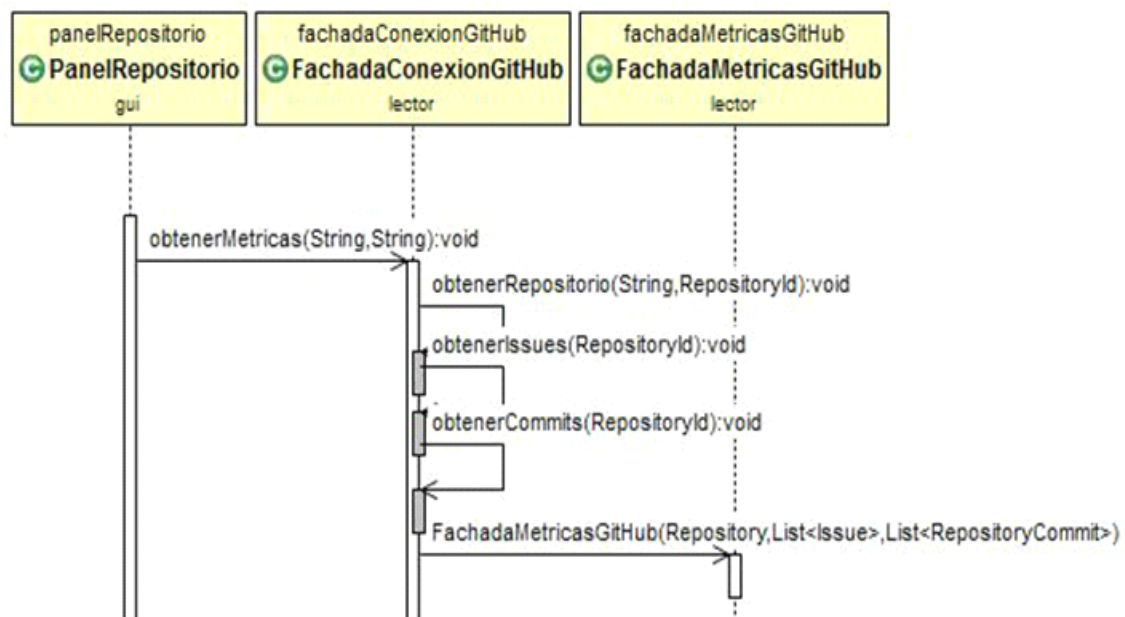
En este procedimiento se crea un cliente autenticado para solicitar los datos a la plataforma.

**Ilustración 7: Procedimiento conexión.**

La clase **PanelConexion** envía el nombre de usuario y la contraseña introducidos a la **fabrica** existente y esta se encarga de crear la **fachada** con un cliente valido para realizar peticiones a la plataforma de forma autenticada.

2. Obtener datos del repositorio

Este es el procedimiento por el cual se obtienen los datos del repositorio introducido y se crean las métricas.

**Ilustración 8: Procedimiento obtener datos del repositorio.**

La clase **PanelRepositorio** solicita a la **conexión** que obtenga los datos y las métricas, La **conexión** pide los datos y posteriormente crea las métricas enviando los datos recibidos.

3. Calcular una métrica

Vamos a poner como ejemplo del cálculo la métrica NumeroFavoritos, para ilustrar este procedimiento. Todas las métricas reciben un tratamiento similar, la única diferencia es que algunas llaman a métodos generales para realizar operaciones mas o menos repetitivas.

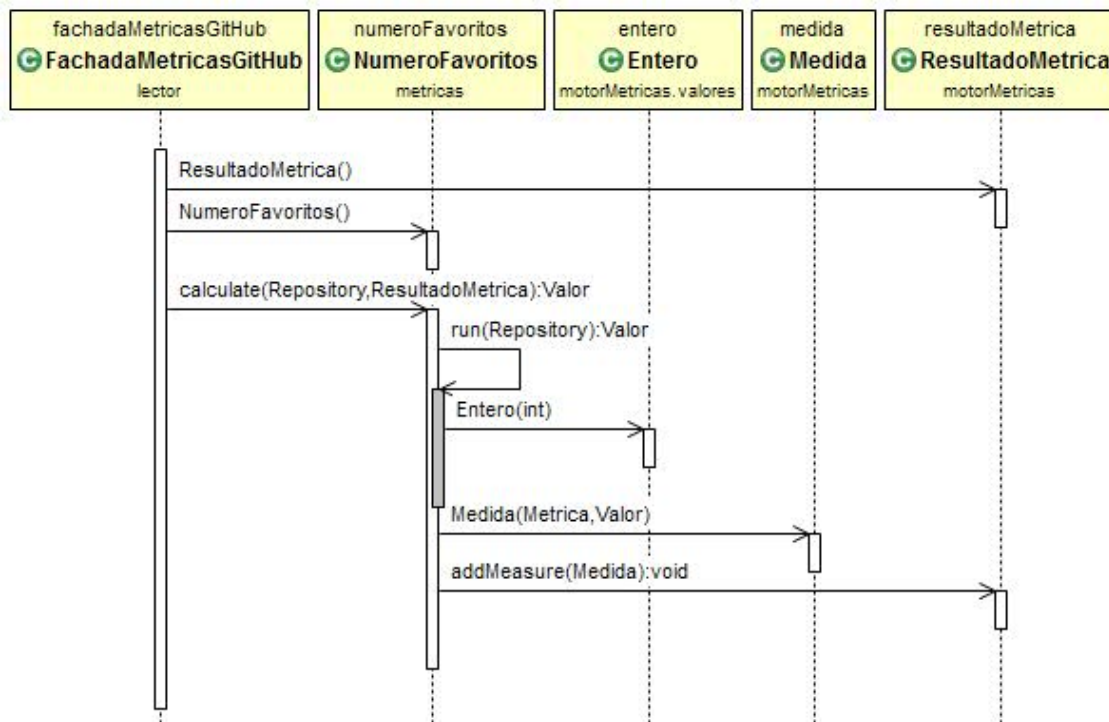


Ilustración 9: Procedimiento calcular métrica.

La clase FachadaMetricas crea un objeto ResultadoMetrica que será el encargado de recoger todas las métricas calculadas. Crea la métrica a calcular y la calcula, esta ejecuta su cálculo, creando un objeto con el tipo de valor necesario, creando una medida con ese valor y consigo misma, guarda la medida creada en el objeto ResultadoMetrica recibido.

4. Leer informe

Este procedimiento es el encargado de leer un informe y mostrar los resultados.



Ilustración 10: Procedimiento leer informe.

En este procedimiento Principal envía un `BufferedReader` a la conexión, que se le a su le envía a la fachada de métricas, que vuelve a crear un objeto `ResultadoMetrica` y va creando las medidas con los valores leídos y guardándolas en el. Posteriormente el panel resultados solicita el texto con los resultados y los gráficos a la conexión.

5. Comparar informes

Este procedimiento es el encargado de leer los dos informes y posteriormente compararlos.



Ilustración 11: Procedimiento comparar informes.

En este caso el PanelComparacion crea dos conexiones sin autenticar, no se van a realizar peticiones, y le pide una que se compare con la otra recibiendo un String con los resultados de la comparación que envía al PanelResultadosComparacion.

XIII - REFERENCIAS

- [1] R. M. Sánchez, Y. C. González-Carvajal, y C. L. Nozal, «Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones», en *Actas de las X Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2005)*, September 14-16, 2005, Granada, Spain, 2005, pp. 59–66.

UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería



Informática

Monitor multiplataforma de la actividad de un proyecto

Trabajo final del GºIng.Informática

Alumnos David Blanco Alonso

Tutor Carlos López Nozal

DEPARTAMENTO DE INGENIERÍA CIVIL

Área de Lenguajes y Sistemas Informáticos

Burgos, 4 de febrero de 2016



Índice de contenido

1	Introducción.....	1
2	Estructura de directorios.....	1
2.1	Aplicación.....	1
2.2	Documentación.....	1
2.3	Repositorio.....	2
2.4	Software.....	2
3	Manual del programador.....	2
3.1	Compilación.....	2
3.2	Importación del proyecto.....	4
4	Referencias.....	4

Índice de ilustraciones

Ilustración 1:	Estructura formato electrónico.....	1
Ilustración 2:	Detalle de la opción exportar.....	2
Ilustración 3:	Detalle de la pantalla de selección del tipo de exportación.....	2
Ilustración 4:	Pantalla de opciones de exportación del Runnable JAR File.....	3
Ilustración 5:	Detalle de la opción importar.....	4
Ilustración 6:	Detalle de la selección del tipo de exportación.....	4
Ilustración 7:	Detalle la opción a seleccionar.....	4

Índice de tablas



XIV - INTRODUCCIÓN

En este anexo se explicara todo lo referente a la estructura de los datos guardados en el soporte electrónico entregado. También se explicara como poder instalar el repositorio y como se realiza su ejecución.

XV - ESTRUCTURA DE DIRECTORIOS

La estructura de los ficheros existente en el soporte electrónico entregado consta de 4 carpetas:

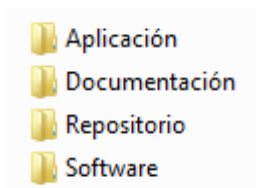


Ilustración 12:
Estructura formato
electrónico.

- **Aplicación:** Contiene el paquete JAR resultante de la aplicación.
- **Documentación:** Contiene la documentación generada por el proyecto en formato ODT y en formato PDF.
- **Repositorio:** Contiene una copia del repositorio existente en la plataforma GitHub [1].
- **Software:** Contiene el software y las aplicaciones que han sido necesarias para el desarrollo.

1. **Aplicación**

Esta carpeta contiene exclusivamente el archivo JAR resultante que permite ejecutar la aplicación sin necesidad de instalación, solamente es necesario contar con Java instalado en el equipo en el que se pretenda realizar la ejecución. Este archivo JAR ha sido generado mediante la exportación desde Eclipse. Es el mismo archivo asignado a la release del repositorio alojado en GitHub.

2. **Documentación**

Esta carpeta contiene la documentación generada para ilustrar el desarrollo del proyecto, tanto la memoria como los anexos. Consta a su vez de dos carpetas ODT y PDF, la primera contiene los archivos en formato ODT que permiten la edición de todos los archivos que forman la memoria, documentos maestros y contenido. La segunda contiene los archivos PDF generados desde los archivos ODT.

3. Repositorio

Esta carpeta contiene una copia del repositorio alojado en GitHub [1], con la misma estructura. Puede utilizarse para exportar el proyecto como proyecto java en Eclipse, realizando una copia al workspace para poder realizar modificaciones, sin necesidad de conectarse a la plataforma GitHub.

4. Software

Esta carpeta contiene las aplicaciones software que han sido utilizadas para el desarrollo del proyecto, incluyendo los plugin y las bibliotecas utilizadas.

XVI - MANUAL DEL PROGRAMADOR

En este apartado se explica cómo generar el archivo JAR resultante del desarrollo. Y como importar el proyecto al IDE de Eclipse.

1. Compilación

Seleccionamos la opción exportar en File > Export...

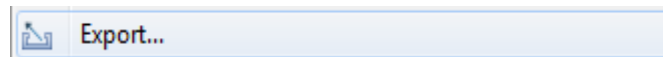


Ilustración 13: Detalle de la opción exportar.

Seleccionamos el tipo Runnable JAR file.

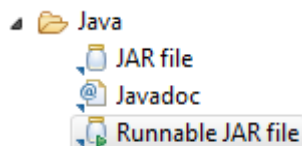


Ilustración 14: Detalle de la pantalla de selección del tipo de exportación.

En la pantalla de opciones del archivo JAR, seleccionamos Principal-Activiti-Api como configuración de launch. Seleccionamos el destino donde queremos generar el archivo JAR y marcamos la opción extraer bibliotecas en el archivo JAR par que el archivo JAR contenga las bibliotecas necesarias y no surjan problemas.

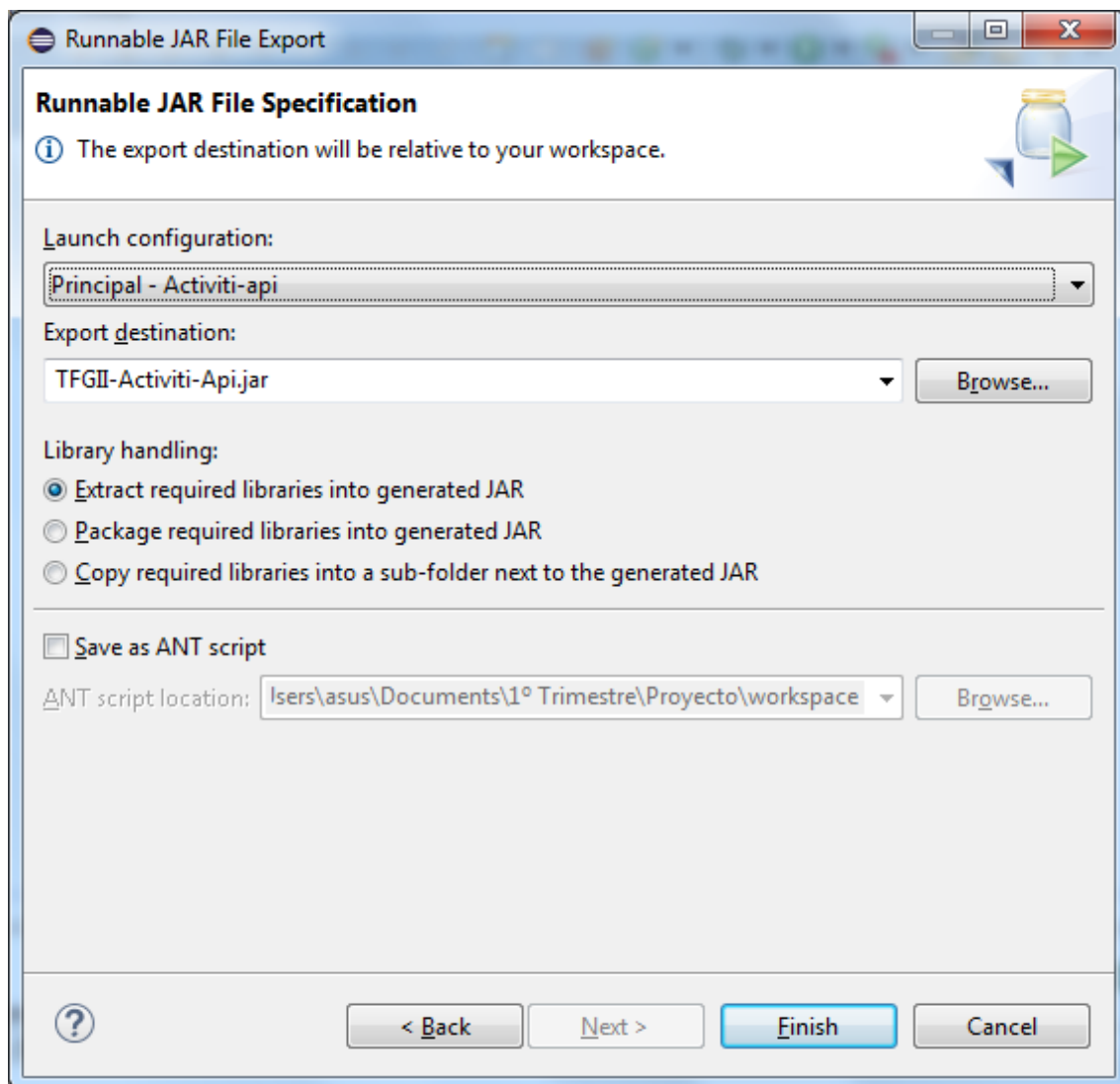


Ilustración 15: Pantalla de opciones de exportación del Runnable JAR File.

2. Importación del proyecto

Seleccionamos la opción File > Import...

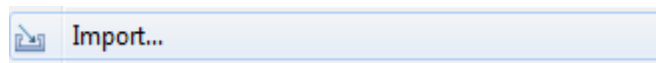


Ilustración 16: Detalle de la opción importar.

Seleccionamos Existing Projects into Workspace.

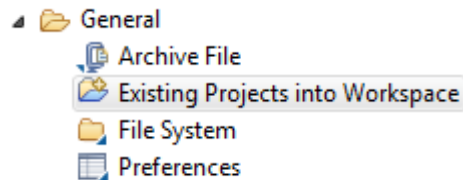


Ilustración 17: Detalle de la selección del tipo de exportación.

Seleccionamos la ubicación del proyecto a importar y marcamos la opción de copiar en el workspace, ya que si realizamos la importación desde el CD no será modificable.

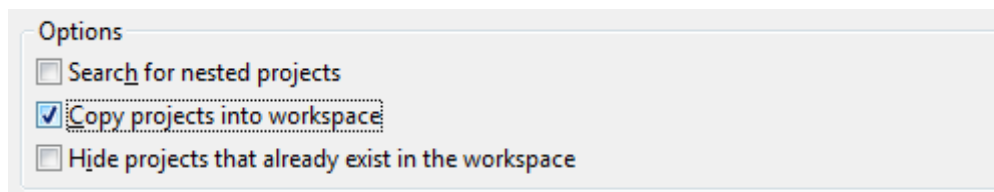


Ilustración 18: Detalle la opción a seleccionar.

XVII -REFERENCIAS

- [1] «dba0010/Activiti-API - Java». [En línea]. Disponible en: <https://github.com/dba0010/Activiti-API>. [Accedido: 04-feb-2016].

UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería



Informática

Monitor multiplataforma de la actividad de un proyecto

Trabajo final del GºIng.Informática

Alumnos David Blanco Alonso

Tutor Carlos López Nozal

DEPARTAMENTO DE INGENIERÍA CIVIL

Área de Lenguajes y Sistemas Informáticos

Burgos, 4 de febrero de 2016



Índice de contenido

1	Introducción.....	1
2	Instalación.....	1
3	Manual de usuario.....	1
3.1	Buscar repositorio.....	1
3.2	Cargar informe.....	3
3.3	Comparar informes.....	4
4	Referencias.....	5

Índice de ilustraciones

Ilustración 1:	Detalle de la ejecución del comando "java -version".....	1
Ilustración 2:	Pantalla inicio.....	2
Ilustración 3:	Pantalla conexión.....	2
Ilustración 4:	Pantalla seleccionar repositorio.....	3
Ilustración 5:	Pantalla resultados.....	3
Ilustración 6:	Cuadro de selección de informe.....	4
Ilustración 7:	Pantalla resultados para la carga de un informe.....	4
Ilustración 8:	Pantalla selección de informes a comparar.....	5
Ilustración 9:	Pantalla resultados comparación.....	5

Índice de tablas



XVIII - INTRODUCCIÓN

En este anexo se explica la instalación necesaria para ejecutar la aplicación y se explica cómo utilizar la aplicación.

XIX - INSTALACIÓN

Para ejecutar la aplicación solo es necesario tener la menos la versión 1.7 de Java [1] instalada en el ordenador donde se quiera ejecutar la aplicación.

Para comprobar si se tiene Java instalado y su versión se puede ejecutar el comando “java -version” en la consola de comandos.

```
C:\Users\asus>java -version
java version "1.8.0_71"
Java(TM) SE Runtime Environment (build 1.8.0_71-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.71-b15, mixed mode)
```

Ilustración 19: Detalle de la ejecución del comando "java -version".

XX - MANUAL DE USUARIO

En este apartado se explica cómo realizar cada una de las características de la aplicación.

1. *Buscar repositorio*

El primer paso es pulsar el botón de la plataforma con la que se quiere trabajar.

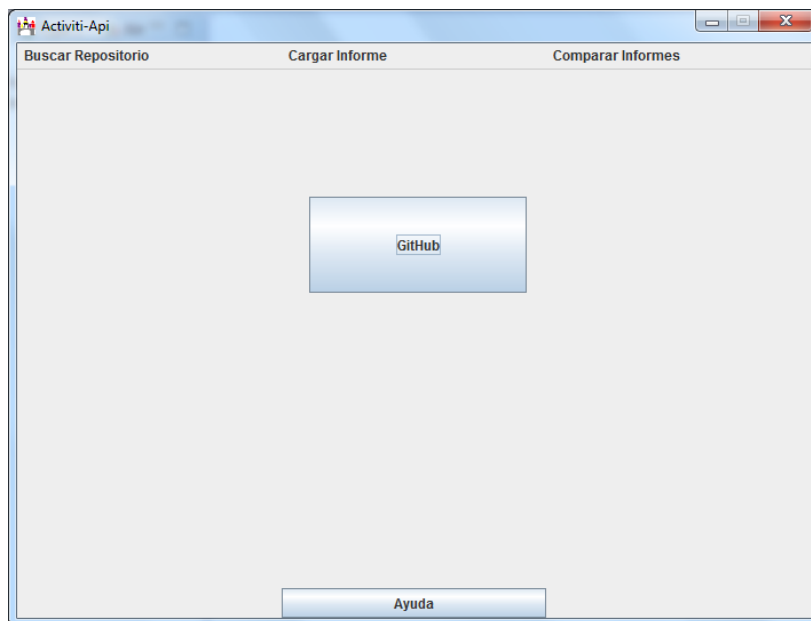


Ilustración 20: Pantalla inicio.

Dependiendo de cómo se pretenda trabajar se puede seleccionar el modo conectado o el modo desconectado, aunque se aconseja que si se pretende trabajar con repositorios grandes es necesario hacerlo en modo conectado. Para realizar la conexión hay que introducir un usuario activado en la plataforma y la contraseña para validarlo.

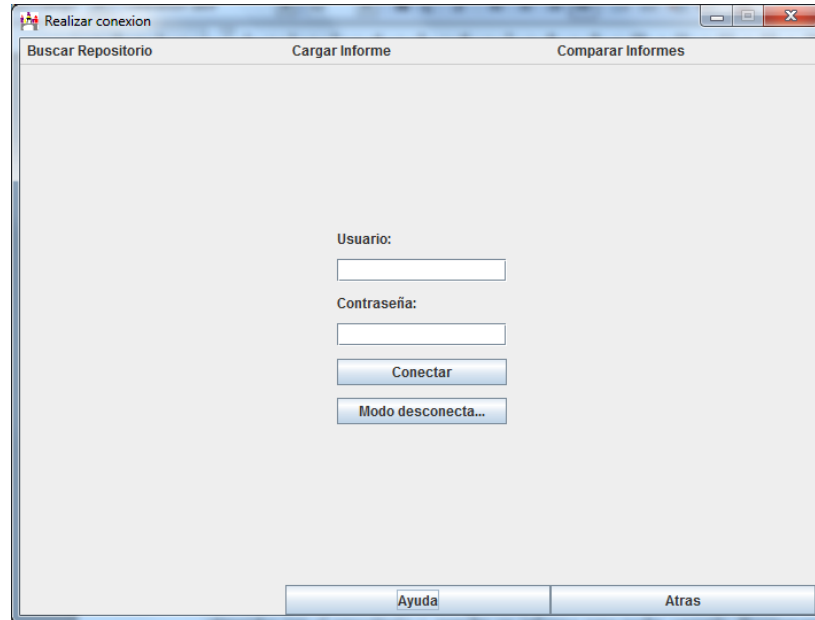


Ilustración 21: Pantalla conexión.

Una vez seleccionado el modo de trabajo, llegamos a la pantalla de selección de repositorio. En esta pantalla se introduce un usuario del que buscar los repositorios que posee y se presiona “Enter” para que realice la búsqueda y los muestre, permitiendo seleccionar de cual se quieren conocer las métricas.

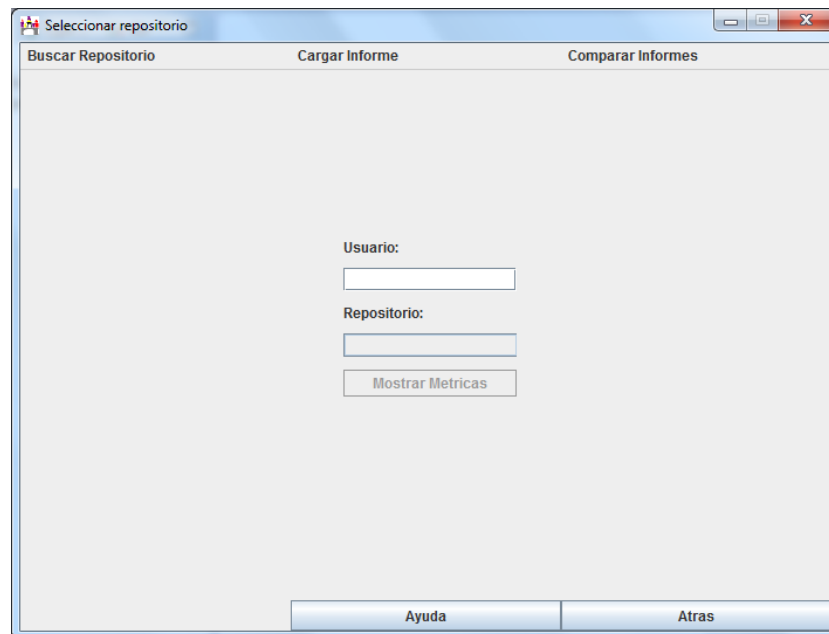


Ilustración 22: Pantalla seleccionar repositorio.

En la pantalla de resultados podemos observar los resultados de las métricas

obtenidos por el repositorio y guardar un informe para poder cargarlo directamente sin necesidad de una conexión.

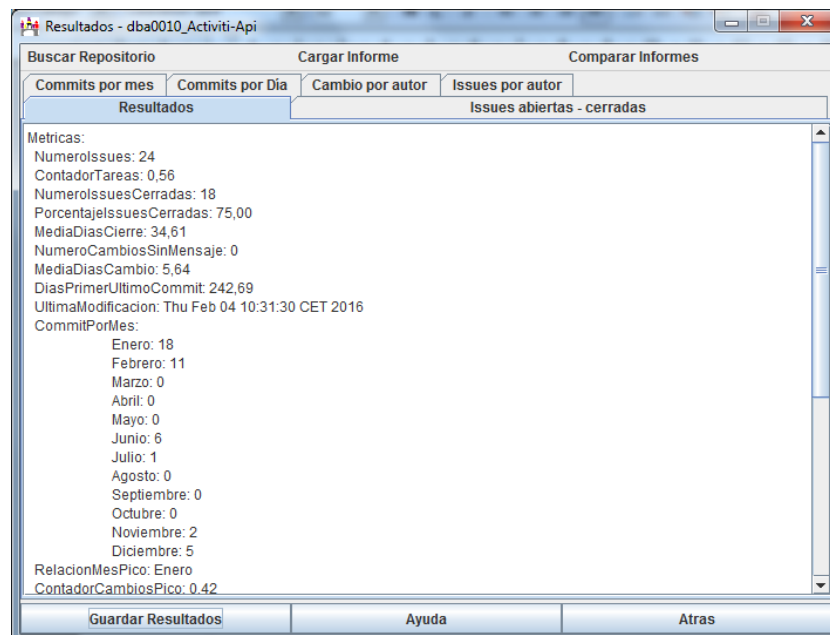


Ilustración 23: Pantalla resultados.

2. Cargar informe

Esta acción permite leer un informe guardado y mostrar sus resultados.

Para acceder a ella solo hay que pulsar en Cargar Informe en el menú superior. Esto abre un cuadro de dialogo para seleccionar el informe que se quiere guardar.

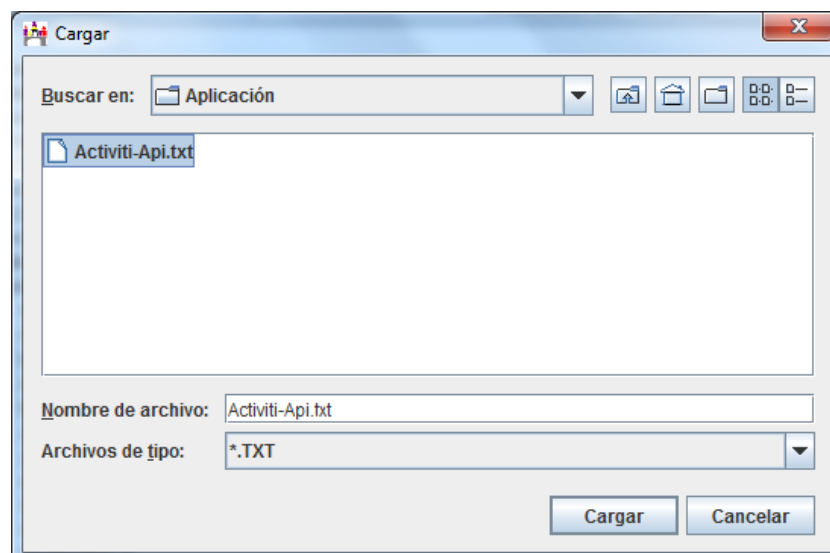


Ilustración 24: Cuadro de selección de informe.

Una vez seleccionado el informe se muestra directamente en la misma pantalla de

resultados que para la búsqueda de repositorio, pero sin el botón de guardar.

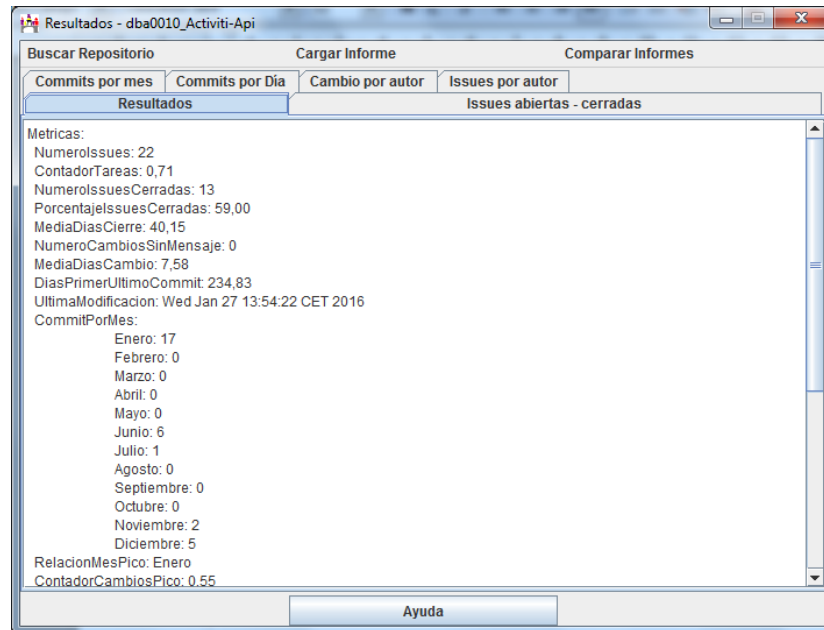


Ilustración 25: Pantalla resultados para la carga de un informe.

3. Comparar informes

Esta característica permite comparar dos informes. Para comenzar a realizar solo se tiene que pulsar en Comparar Informes para que muestre la pantalla de selección. En esta pantalla se puede introducir a mano la ruta de cada informe que se pretende comparar o utilizar el cuadro de dialogo para seleccionar un informe. Una vez seleccionados los dos informes se puede realizar la comparación.

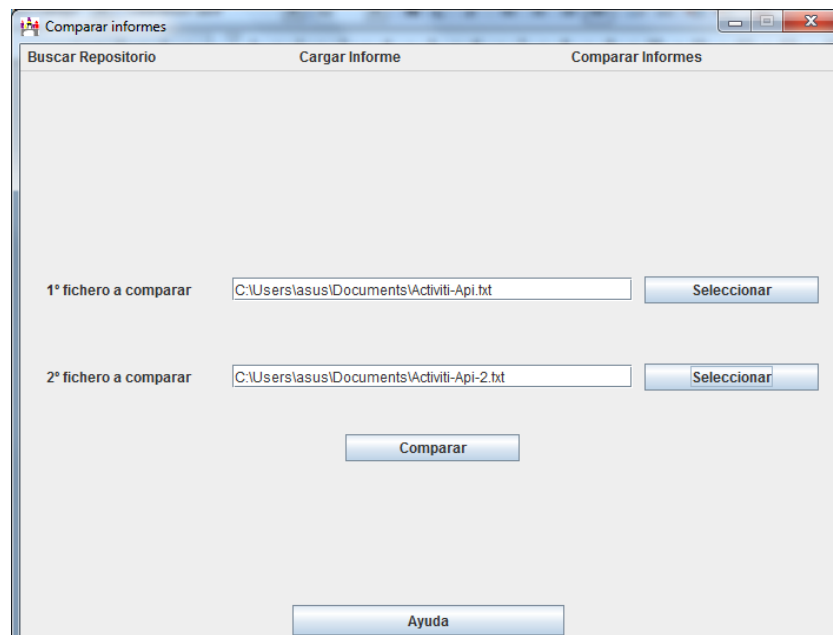
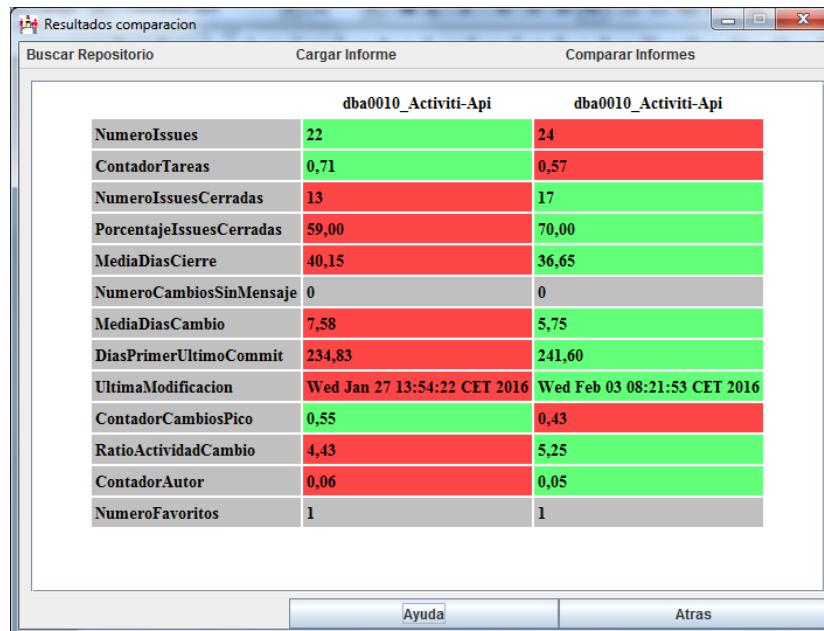


Ilustración 26: Pantalla selección de informes a comparar.

Los resultados de las comparaciones se realizan solo de las métricas que se pueden comparar, mostrándolas en una tabla que cambia el fondo dependiendo de si es mejor uno u otro.



The screenshot shows a window titled 'Resultados comparacion' with three tabs: 'Buscar Repositorio', 'Cargar Informe', and 'Comparar Informes'. The 'Comparar Informes' tab is active, displaying a table comparing two API versions: 'dba0010_Activiti-API' and 'dba0010_Activiti-API'. The table has 15 rows of metrics. The background color of each cell indicates the comparison result: green for 'better', red for 'worse', and grey for 'equal'.

	dba0010_Activiti-API	dba0010_Activiti-API
NumeroIssues	22	24
ContadorTareas	0,71	0,57
NumeroIssuesCerradas	13	17
PorcentajeIssuesCerradas	59,00	70,00
MediaDiasCierre	40,15	36,65
NumeroCambiosSinMensaje	0	0
MediaDiasCambio	7,58	5,75
DiasPrimerUltimoCommit	234,83	241,60
UltimaModificacion	Wed Jan 27 13:54:22 CET 2016	Wed Feb 03 08:21:53 CET 2016
ContadorCambiosPico	0,55	0,43
RatioActividadCambio	4,43	5,25
ContadorAutor	0,06	0,05
NumeroFavoritos	1	1

At the bottom of the window are two buttons: 'Ayuda' and 'Atras'.

Ilustración 27: Pantalla resultados comparación.

XXI - REFERENCIAS

- [1] «Download Free Java Software». [En línea]. Disponible en: <https://www.java.com/en/download/>. [Accedido: 04-feb-2016].