



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Comparador de métricas de  
evolución en repositorios  
software  
Documentación Técnica**



Presentado por Miguel Ángel León Bardavío  
en Universidad de Burgos — 10 de septiembre  
de 2019

Tutor: Carlos López Nozal



---

# Índice general

---

<b>Índice general</b>	<b>I</b>
<b>Índice de figuras</b>	<b>III</b>
<b>Índice de tablas</b>	<b>IV</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	6
<b>Apéndice B Especificación de Requisitos</b>	<b>7</b>
B.1. Introducción . . . . .	7
B.2. Objetivos generales . . . . .	7
B.3. Actores . . . . .	9
B.4. Catalogo de requisitos . . . . .	9
B.5. Especificación de requisitos . . . . .	11
<b>Apéndice C Especificación de diseño</b>	<b>13</b>
C.1. Introducción . . . . .	13
C.2. Diseño de datos . . . . .	13
C.3. Diseño procedimental . . . . .	13
C.4. Diseño arquitectónico . . . . .	13
C.5. Diseño de la interfaz de usuario . . . . .	13
<b>Apéndice D Documentación técnica de programación</b>	<b>15</b>
D.1. Introducción . . . . .	15

D.2. Estructura de directorios . . . . .	15
D.3. Manual del programador . . . . .	16
D.4. Compilación, instalación y ejecución del proyecto . . . . .	18
D.5. Pruebas del sistema . . . . .	18
<b>Apéndice E Documentación de usuario</b>	<b>21</b>
E.1. Introducción . . . . .	21
E.2. Requisitos de usuarios . . . . .	21
E.3. Instalación . . . . .	21
E.4. Manual del usuario . . . . .	22
<b>Bibliografía</b>	<b>35</b>

---

## Índice de figuras

---

A.1. Sprints iniciales del proyecto en GitLab. . . . .	3
A.2. Primeras issues del proyecto, la mayoría etiquetadas como ‘Research’ o ‘Project Configuration’ . . . . .	3
A.3. Gráfico burndown del segundo sprint Acceso a repositorio público, documentar y comparar las dos APIS . . . . .	4
D.1. Diagrama del Framework para el cálculo de métricas con perfiles. . . . .	18
E.1. Diálogo de conexión . . . . .	26
E.2. Crear un <i>Personal Access Token</i> desde GitLab . . . . .	27
E.3. Página principal . . . . .	27
E.4. Tabla que muestra los valores medidos en los proyectos para las métricas . . . . .	30

---

## Índice de tablas

---

## Apéndice A

---

# Plan de Proyecto Software

---

### A.1. Introducción

Una de las actividades de un proyecto software es la fase de planificación. En esta fase se estima el esfuerzo, el tiempo y el dinero que se espera invertir en el proyecto. El objetivo principal de la fase de planificación es estimar si se puede realizar el proyecto con éxito y, en ese caso, tener una guía de referencia para el proceso de desarrollo. Este anexo recoge los documentos generados en este proceso y se divide en dos partes:

**Planificación temporal.** En esta parte se estima el tiempo y el esfuerzo que se requieren para la realización del proyecto.

**Estudio de la viabilidad.** Esta parte se estima si el proyecto es viable tanto económica como legalmente, por tanto se dividirá en dos secciones:

**Viabilidad económica.** Análisis del coste y del beneficio que supondría la realización del proyecto.

**Viabilidad legal.** Análisis de las leyes que se aplicarían desde el comienzo del proyecto. En un proyecto software tienen especial importancia las licencias y la Ley de Protección de Datos.

### A.2. Planificación temporal

No se ha realizado una planificación temporal del proyecto. Se han seguido los 12 principios del manifiesto ágil y el modelo SCRUM [1]:

- Se ha aplicado un desarrollo incremental y evolutivo.
- Se han realizado iteraciones (*sprints*) de dos semanas. Al finalizar un sprint se realizaba una reunión entre el tutor y el alumno que daba comienzo al siguiente sprint y que consta de dos partes:
  - Una parte de revisión del sprint en la que se exponía una parte operativa del producto realizada durante el sprint.
  - Otra de planificación del siguiente sprint en la que se determinaba el trabajo y los objetivos a alcanzar durante el siguiente sprint. Esto quedaba reflejado como una pila de tareas que se debían completar durante el sprint y que han sido registradas en el sistema de gestión de incidencias de GitLab.

## Sprints

A continuación se definen los sprints y sus respectivas pilas de tareas que se llevaron a cabo durante la realización del proyecto.

Según la naturaleza de los sprints, en el proceso del desarrollo se pueden diferenciar varias etapas:

- Una primera etapa de **investigación** de las herramientas que se utilizarán durante el proceso y de **configuración** del entorno de desarrollo.
- En la segunda etapa se aprecian tareas de **diseño** de la parte lógica de la aplicación. Se diseña el framework de conexión a forjas de repositorios, se implementa el framework descrito en *Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones* [4] para el cálculo de métricas y se diseñan los modelos de datos que serán utilizados por la aplicación.
- Durante la segunda etapa se apreció que se debía facilitar el **flujo de trabajo**, la comunicación entre tutor y alumno y facilitar las reuniones de revisión y planificación del sprint.
- Etapa de diseño e implementación de la **interfaz gráfica y mejora de características** que precisaban de cambios en la parte lógica de la aplicación que se diseñó en la segunda etapa.
- Etapa de **documentación** en la que se escribe la memoria y los anexos. También se preparan videotutoriales y manuales de usuario.



## Investigación y configuración

Los dos primeros sprints estuvieron dedicados a la investigación de herramientas y configuración del entorno de desarrollo. Por cada sprint se observa en la Fig. A.1 su descripción, la fecha de apertura y cierre, y el número de issues asociadas. Y en la Fig. A.2 se muestran las descripciones de las primeras tareas que se definieron, la mayoría etiquetadas como ‘*Research*’ (Estudio o investigación) y ‘*Project Configuration*’ (Configuración del proyecto). En la Fig. A.3 se muestra un gráfico *burndown*<sup>1</sup> del segundo sprint, ya que del primero no hay sido posible obtenerlo debido a que GitLab no lo ha podido generar seguramente por una mala gestión de las issues.

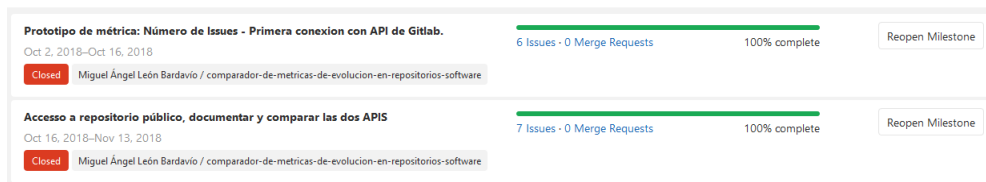


Figura A.1: Sprints iniciales del proyecto en GitLab.



Figura A.2: Primeras issues del proyecto, la mayoría etiquetadas como ‘*Research*’ o ‘*Project Configuration*’

<sup>1</sup>Gráfico de trabajo pendiente. En él se muestran las issues abiertas en el ciclo de vida del sprint

### Acceso a repositorio público, documentar y comparar las dos APIS

- Acceso a repositorio publico mediante nombre repositorio
- Acceso a repositorios publicos de un usuario
- Comparar dos APIs: Funcional y no funcional
- Pruebas con JUnit
- Documentación en LATEX

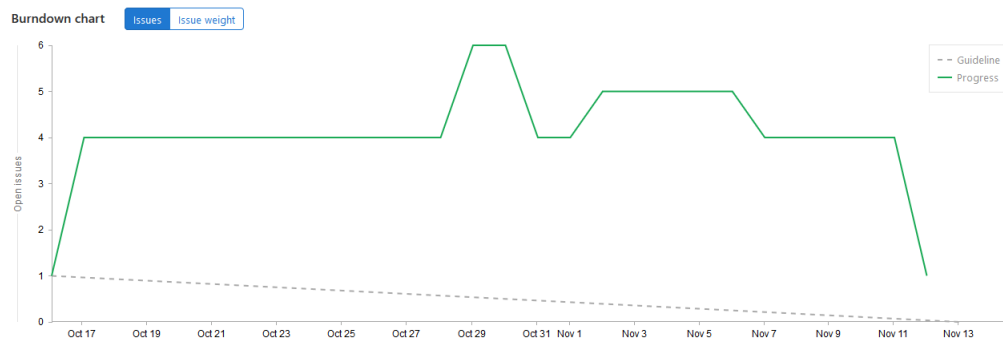


Figura A.3: Gráfico burndown del segundo sprint Acceso a repositorio público, documentar y comparar las dos APIS

Esta etapa de investigación y configuración duró un mes (dos sprints) y sus decisiones afectaron a todo el proyecto. Se recopiló información sobre trabajos relacionados como *Activiti-API* [6], *Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones* [4] y *sPACE: Software Project Assessment in the Course of Evolution* [5] y se estudió los entornos y herramientas que se utilizarían para el desarrollo del proyecto.

## Diseño e implementación

Después de esta etapa de investigación, comenzó una etapa de diseño. En esta etapa, las tareas que se definen se relacionan con el diseño de la parte lógica del sistema software a construir, se comienza con la implementación y se empiezan a utilizar las herramientas investigadas anteriormente para construir el sistema.

Esta etapa de diseño tuvo duración de 3 sprints y uno extra para la configuración del flujo de trabajo que se explica en el siguiente apartado. Esto se puede apreciar en la Fig. que muestra los sprints relacionados a esta etapa y en la Fig. que muestra las issues relacionadas con la etapa de diseño. La mayoría de estas issues cuentan con la etiqueta ‘*Design*’.

### Configuración del flujo de trabajo

Dentro de la fase de diseño se decidió dedicar un tiempo a un tipo de configuración del proyecto que facilitaría el flujo de trabajo y las reuniones de revisión/planificación del sprint. Las principales tareas para mejorar el flujo de trabajo fueron:

- Configurar la gestión del proyecto con Maven
- Configurar los procesos de integración y despliegue continuo con GitLab (CI/CD)
- Realizar pruebas unitarias con JUnit y automatizar su ejecución gracias a Maven y los *pipelines* (CI/CD) de GitLab .
- Configurar revisiones automáticas de calidad y de cobertura de las pruebas gracias a Codacy, Jacoco y GitLab.
- Configurar un entorno en Heroku donde poder desplegar la aplicación y así poder ser revisada por el tutor fácilmente.
- Configurar badges <sup>2</sup> para representar los el estado de la calidad de código, cobertura, despliegue y los trabajos de CI/CD.

Esta fase duro un sprint y comenzó antes de terminar la fase de diseño, como se observa en la Fig. en un milestone con título “Excepciones, log, test, pom y CI”. Las issues que se definieron se caracterizan por tener las etiquetas ‘*Test*’, ‘*Project Configuration*’ o ‘*Question*’, como se puede observar en la Fig. . También se muestra en la Fig. un gráfico burndown del sprint.

### Implementación de la interfaz gráfica y mejora de las catacterísticas

Es la etapa más extensa, duró 8 sprints que se muestran en la Fig. . Esta etapa comienza con un estudio de la herramienta Vaadin para crear la interfaz gráfica y la viabilidad de su uso en el proyecto. Para ello, se realizaron varias pruebas de integración con el entorno de desarrollo (Versión de Java, Eclipse, Maven, etc), se realizó un primer prototipo sencillo y se comprobó que se podría hacer una interfaz gráfica con la licencia gratuita. Una vez terminado con éxito estas pruebas de viabilidad, comenzó el desarrollo de la interfaz gráfica del proyecto.

---

<sup>2</sup>Son placas que aportan información rápida sobre el estado del proyecto en ciertos aspectos como la cobertura, la calidad de código o el estado del proceso de CI/CD

La interfaz gráfica da un aspecto más visible de lo que el proyecto podría llegar a ser. Los requisitos funcionales evolucionaban conforme evolucionaba la interfaz. Por ejemplo, se decidió agregar varias formas de añadir un repositorio o añadir más formas de conectarse a GitLab.

En la Fig. se muestran las primeras tareas de esta etapa. Destacan por ser etiquetadas con ‘GUI’ y ‘Feature’.

Se muestra en la Fig. los gráficos burndown de esta fase.

### **Documentación**

Esta etapa comenzó al terminar el desarrollo del proyecto. En esta se realiza la memoria y los anexos del proyecto, así como videotutoriales u otros manuales de usuario. Esta fase duró 2 sprints (ver Fig. ) y sus tareas son etiquetadas todas con ‘Project Memory’ (ver Fig. ). En la Fig. se muestran los gráficos burndown del sprint.

## **A.3. Estudio de viabilidad**

### **Viabilidad económica**

En esta sección se realiza un análisis coste-beneficio del proyecto.

### **Viabilidad legal**

En esta sección se realiza un estudio de las leyes que se aplican a este proyecto para asegurar que la realización del proyecto no supondrá ninguna violación de estas leyes.

## *Apéndice B*

---

# Especificación de Requisitos

---

## B.1. Introducción

Este anexo tiene como objetivo analizar y documentar las necesidades funcionales y no funcionales que deberán ser soportadas por el software que se va a construir.

Para representar estas necesidades se utilizarán historias de usuario, que describen lo que el usuario debería poder hacer, en lugar de los tradicionales requisitos funcionales, que describen características más específicas del desarrollo del sistema de un modo más técnico [7].

También se recurrirá a diagramas de casos de uso y a diagramas de secuencia. Los casos de uso describen todas las interacciones que tendrán los usuarios con el software; mientras que los diagramas de secuencia muestran la interacción de un conjunto de entidades del sistema software a través del tiempo y se modelan, normalmente, para cada caso de uso, por tanto incluirán detalles de la implementación como operaciones, clases, mensajes, etc.

## B.2. Objetivos generales

El objetivo general de este TFG es diseñar una aplicación web en Java que permita obtener un conjunto de métricas de evolución del proceso software [5] a partir de repositorios de GitLab, esto permitirá comparar los distintos procesos de desarrollo software entre varios repositorios.

La aplicación será probada con datos reales de repositorios de Trabajos Fin de Grado del Grado de Ingeniería Informática presentados en GitLab.

A continuación se desglosa el objetivo general en objetivos más detallados.

- Se obtendrán medidas de métricas de evolución de uno o varios proyectos alojados en repositorios de GitLab.
- Las métricas que se desean calcular de un repositorio son algunas de las especificadas en *sPACE: Software Project Assessment in the Course of Evolution* [5] y adaptadas a los repositorios software:
  - Número total de incidencias (*issues*)
  - Cambios(*commits*) por incidencia
  - Porcentaje de incidencias cerrados
  - Media de días en cerrar una incidencia
  - Media de días entre cambios
  - Días entre primer y último cambio
  - Rango de actividad de cambios por mes
  - Porcentaje de pico de cambios
- El objetivo de obtener las métricas es poder evaluar el estado de un proyecto comparándolo con otros proyectos de la misma naturaleza. Para ello se deberán establecer unos valores umbrales por cada métrica basados en el cálculo de los cuartiles Q1 y Q3. Además, estos valores se calcularán dinámicamente y se almacenarán en perfiles de configuración de métricas.
- Se dará la posibilidad de almacenar de manera persistente estos perfiles de métricas para permitir comparaciones futuras. Un ejemplo de utilidad es guardar los valores umbrales de repositorios por lenguaje de programación, o en el caso de repositorios de TFG de la UBU por curso académico.
- También se permitirá almacenar de forma persistente las métricas obtenidas de los repositorios para su posterior consulta o tratamiento. Esto permitiría comparar nuevos proyectos con proyectos de los que ya se han calculado sus métricas.

## Objetivos técnicos

Este apartado recoge los requisitos más técnicos del proyecto.

- Diseñar la aplicación de manera que se puedan extender con nuevas métricas con el menor coste de mantenimiento posible. Se aplicará un diseño basado en frameworks y en patrones de diseño [2].

- El diseño de la aplicación debe facilitar la extensión a otras plataformas de desarrollo colaborativo como GitHub o Bitbucket. Se aplicará un diseño basado en frameworks y en patrones de diseño [2].
- Aplicar el *frameworks* modelo vista controlador para separar la lógica de la aplicación para el calculo de métricas y la interfaz de usuario. Se aplicará un diseño basado en frameworks y en patrones de diseño [2].
- Crear una batería de pruebas automáticas con cobertura del por encima del 90 % en los subsistemas de lógica de la aplicación .
- Utilizar una plataforma de desarrollo colaborativo que incluya un sistema de control de versiones, un sistema de seguimiento de incidencias y que permita una comunicación fluida entre el tutor y el alumno.
- Utilizar un sistema de integración y despliegue continuo.
- Diseñar una correcta gestión de errores definiendo excepciones de biblioteca y registrando eventos de error e información en ficheros de *log*.
- Aplicar nuevas estructuras del lenguaje Java para el desarrollo, como son expresiones lambda.
- Utilizar sistemas que aseguren la calidad continua del código que permitan evaluar la deuda técnica del proyecto.
- Probar la aplicación con ejemplos reales y utilizando técnicas avanzadas, como entrada de datos de test en ficheros con formato tabulado tipo CSV (*comma separated values*).

### B.3. Actores

Se consideran dos actores: El usuario de la aplicación y el desarrollador. Además de poder ser utilizado por un usuario, la aplicación deberá estar preparada para que en un futuro se pueda ampliar

### B.4. Catalogo de requisitos

Este apartado enumera los diferentes requisitos que el sistema software deberá satisfacer. Se divide en dos apartados. El primero detalla los servicios que prestará el sistema al usuario final o a otros sistemas; el segundo detalla funciones más técnicas de cómo se desarrollará el proceso software y otras propiedades del sistema como eficiencia o mantenibilidad.

## Requisitos funcionales

### Conexión

**RF 1. Establecer conexión con GitLab.** El usuario podrá establecer una conexión con GitLab de varias formas para poder acceder a los repositorios almacenados en el host <https://gitlab.com>. De esta forma puede añadir nuevos repositorios y volver a obtener métricas de repositorios ya existentes.

**RF 1.1. Establecer conexión iniciando sesión.** El usuario podrá iniciar sesión en GitLab de distintas formas para poder acceder a sus repositorios privados.

**RF 1.1.1 Iniciar sesión mediante usuario y contraseña**

**RF 1.1.2 Iniciar sesión mediante personal access token**

**RF 1.3. Establecer conexión pública sin inicio de sesión.** De esta forma, el usuario solo puede acceder a los repositorios con visibilidad pública.

**RF 2. Utilizar la aplicación sin conexión.** El usuario podrá utilizar la aplicación sin establecer una conexión a GitLab. De esta forma podrá consultar los datos que haya obtenido previamente, pero no añadir nuevos repositorios ni volver a obtener datos de los repositorios ya existentes.

### Gestión de repositorios

**RF 3. Añadir un repositorio publico.** El usuario podrá añadir un repositorio público de GitLab siempre que haya establecido una conexión pública o con sesión GitLab.

**RF 4. Añadir un repositorio privado.** El usuario podrá añadir un repositorio privado de GitLab siempre que haya establecido una conexión con una sesión a GitLab iniciada y que el repositorio sea accesible desde el usuario que haya iniciado sesión.

**RF 5. Distintas formas de añadir un repositorio tanto público como privado.** El usuario tendrá distintas formas de acceder a un repositorio.

**RF 5.1. Seleccionar desde los repositorios de un usuario** El usuario podrá añadir un repositorio seleccionándolo de un listado de



repositorios de un usuario de GitLab que el usuario indique mediante el nombre o el ID de usuario. Para ello se requiere una conexión a GitLab. Para listar los repositorios privados, además de los públicos, de ese usuario la conexión deberá tener una sesión iniciada con acceso a los repositorios privados de ese usuario.

**RF 5.2. Seleccionar desde los repositorios de un grupo** El usuario podrá añadir un repositorio seleccionándolo de un listado de repositorios de un grupo de GitLab, que el usuario indique mediante el nombre o el ID de grupo. Para ello se requiere que el grupo tenga visibilidad pública o sea privada y la conexión tenga una sesión abierta a un usuario de GitLab que tenga acceso a ese grupo.

**RF 5.3. Añadir repositorio mediante su URL web.** El usuario podrá añadir un repositorio mediante su URL web (la que se muestra en el navegador).

**RF 6. Evitar repositorios duplicados.** El usuario no podrá añadir el mismo repositorio más de una vez.

. Cada vez que se añada un repositorio, se calcularán las métricas y se evaluará con el perfil activo en ese momento.

**RF .** El usuario podrá volver a obtener la información de cualquier repositorio siempre que la conexión activa en ese momento tenga acceso a ese repositorio.

**RF 7. Eliminar un repositorio.** El usuario podrá eliminar cualquier repositorio de los que haya añadido previamente.

**RF 7. Mostrar los repositorios que el usuario ha añadido.** El usuario podrá ver todos.

## Requisitos no funcionales

## B.5. Especificación de requisitos



## *Apéndice C*

---

# **Especificación de diseño**

---

### **C.1. Introducción**

Este apartado recoge el porqué de las decisiones finales que se han tomado acerca del diseño del software desarrollado, dividido en el diseño de datos, diseño procedimental, diseño arquitectónico y diseño de la interfaz de usuario.

### **C.2. Diseño de datos**

El diseño de los datos es el diseño que tienen las entidades.

### **C.3. Diseño procedimental**

### **C.4. Diseño arquitectónico**

### **C.5. Diseño de la interfaz de usuario**



## Apéndice *D*

---

# Documentación técnica de programación

---

## D.1. Introducción

Este documento detalla asuntos técnicos de programación.

## D.2. Estructura de directorios

El código fuente presenta la siguiente estructura:

- /.gitignore*** Contiene los ficheros y directorios que el repositorio git no tendrá en cuenta
- /.gitlab-ci.yml*** Contiene las etapas y trabajos que se han definido para que se ejecuten en una máquina virtual proporcionada por GitLab (*runner*) tras hacer un commit. Permite la integración y el despliegue continuo.
- /README.md*** Fichero con información relevante sobre el proyecto.
- /codacy-coverage-reporter-4.0.5-assembly.jar*** Ejecutable Java que necesario para una de las tareas de la integración continua: El informe de cobertura de las pruebas. Se ejecutará en una de las tareas definidas en el fichero */.gitlab-ci.yml*.
- /pom.xml*** Fichero de configuración del proyecto maven.
- /system.properties*** Fichero con propiedades del proyecto. Ha sido necesario su uso para el despliegue en Heroku.
- /MemoriaProyecto*** Memoria del proyecto según la plantilla definida en <https://github.com/ubutfgm/plantillaLatex>.

## APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

- /src/test/resources* Datos almacenados en ficheros CSV para proporcionar datos a test parametrizados.
- /src/test/java* Casos de prueba JUnit para la realización de pruebas. Se organiza de la misma forma que */src/main/java*
- /src/main/webapp/VAADIN/themes/MyTheme* Tema principal utilizado por la aplicación. Generado por Vaadin.
- /src/main/webapp/frontend* Ficheros *.css* utilizados por la interfaz gráfica.
- /src/main/webapp/images* Imágenes que se muestran en la interfaz gráfica.
- /src/main/resources* Ficheros de configuración de la aplicación. En este caso el fichero *log4j2.properties* para configurar el log.
- /src/main/java* Contiene todo el código fuente
- /src/main/java/app/* Contiene fachadas que conectan la interfaz de usuario con el resto de componentes que componen la lógica de la aplicación.
- /src/main/java/app/listeners* Contiene observadores y eventos utilizados por la aplicación
- /src/main/java/datamodel* Contiene el modelo de datos de la aplicación.
- /src/main/java/exceptions* Contiene excepciones definidas en la aplicación.
- /src/main/java/gui* Contiene la interfaz de usuario.
- /src/main/java/gui/views* Contiene páginas y componentes de Vaadin que componen la interfaz gráfica de la aplicación.
- /src/main/java/metricsengine* Define el motor de métricas.
- /src/main/java/metricsengine/numeric\_value\_metrics* Métricas definidas por el programador y sus respectivas fábricas (Patrón de diseño método fábrica<sup>1</sup>). Todas las métricas tienen resultados numéricos.
- /src/main/java/metricsengine/values* Valores que devuelven las métricas.
- /src/main/java/repositorydatasource* Framework de conexión a una forja de repositorios como GitLab.

### D.3. Manual del programador

Se explica en este apartado algunas bases para entender como continuar la programación de la aplicación y los puntos de extensión que se han definido.

---

<sup>1</sup><https://refactoring.guru/design-patterns/factory-method>

Cada apartado de esta sección se centra en cada uno de los módulos que contiene la aplicación.

## Framework de conexión

El framework de conexión a una plataforma de desarrollo colaborativo está definido en el paquete *repositorydatasource*. Consta de dos interfaces, la más importante es la interfaz *RepositoryDataSource*.

## Motor de métricas

El motor de métricas se ha desarrollado con una base inicial a una solución propuesta en *Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones*[4]. El diseño se puede observar en la figura [D.1](#).





carpeta *src/test/resources*.



## *Apéndice E*

---

# Documentación de usuario

---

## E.1. Introducción

Este documento detalla cómo un usuario, puede utilizar la aplicación una vez desplegada en un servidor.

## E.2. Requisitos de usuarios

Los requisitos para poder utilizar la aplicación son:

- Tener la aplicación desplegada en algún servidor.
- Disponer de conexión al servidor y tener instalado un navegador web con el que poder acceder a la aplicación. Se recomienda:
  - Google Chrome Versión 75.0.3770.100 o superior
  - Firefox Quantum Versión 67.0.4 o superior
  - IE11 Versión 11.829.17134.0 o superior
  - Opera Versión:62.0.3331.18 o superior

## E.3. Instalación

Al ser una aplicación web no requiere instalación. Solo es necesario desplegar la aplicación en un servidor.

## E.4. Manual del usuario

La aplicación permite añadir proyectos de GitLab y evaluarlos mediante métricas de evolución.

### Conceptos

Para utilizar la aplicación es importante entender los siguientes conceptos:

#### ***Medición***

El proceso de medición es un proceso en el que se asignan números o símbolos a atributos de entidades del mundo real, de tal forma que los caracteriza a través de reglas.

#### ***Métrica***

Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado (IEEE, 1993).

#### ***Indicador***

Métrica o combinación de métricas que proporcionan una visión profunda del proceso, del proyecto o del producto.

#### ***Métrica de evolución***

Es una métrica que mide un atributo del proceso de desarrollo de un producto software.

#### ***Evaluación***

Es uno de los objetivos del proceso de medición. Consiste en determinar el estado de un proyecto en relación con otros proyectos de la misma naturaleza.

#### ***Proyecto (software)***

Proyecto en el cual se desarrolla un producto software.

#### ***Repositorio de código***

Lugar dónde se almacena el código de un proyecto software. A menudo cuentan con un sistema de control de versiones.

#### ***Sistema de control de versiones (VCS - Version Control System)***

Sistema que registra los cambios que se producen sobre los ficheros de un proyecto software almacenados en un repositorio de código.

#### ***Sistema de seguimiento de incidencias (Issue tracking system)***

Sistema que gestiona las diferentes tareas o incidentes que se definen en un proyecto software y que pueden ser asignadas a colaboradores del proyecto.

### Las métricas que se gestionan en la aplicación

Las métricas que se calculan de los proyectos son un conjunto de métricas que proceden de la Master Tesis titulada *sPACE: Software Project*

*Assessment in the Course of Evolution* [5].

### I1 - Número total de issues

- **Categoría:** Proceso de Orientación
- **Descripción:** Número total de issues creadas en el repositorio
- **Propósito:** ¿Cuántas issues se han definido en el repositorio?
- **Fórmula:** NTI.  $NTI = \text{número total de issues}$
- **Fuente de medición:** Repositorio de un gestor de repositorios
- **Interpretación:**  $NTI \geq 0$ . Valores bajos indican que no se utiliza un Sistema de seguimiento de incidencias, podría ser porque el proyecto acaba de comenzar
- **Tipo de escala:** Absoluta
- **Tipo de medida:**  $NTI = \text{Contador}$

### I2 - Commits por issue

- **Categoría:** Proceso de Orientación
- **Descripción:** Número de commits por issue
- **Propósito:** ¿Cuántos commits realizados por cada issue?
- **Fórmula:**  $CI = NTC/NTI$ .  $NTI = \text{Numero total de issues}$ ,  $NTC = \text{Número total de commits}$
- **Fuente de medición:** Repositorio de un gestor de repositorios
- **Interpretación:**  $CI \geq 0$ , Si se acerca a 1 se definen bien las issues, si alto: no se definen bien las issues, si bajo: desarrollo del proyecto lento
- **Tipo de escala:** Ratio
- **Tipo de medida:**  $NTI, NTC = \text{Contador}$

### I3 - Porcentaje de issues cerradas

- **Categoría:** Proceso de Orientación
- **Descripción:** Porcentaje de issues cerradas
- **Propósito:** ¿Qué porcentaje de issues definidas en el repositorio se han cerrado?
- **Fórmula:**  $PIC = NTIC*100/NTI$ .  $NTIC = \text{Número total de issues cerradas}$ ,  $NTI = \text{Numero total de issues}$
- **Fuente de medición:** Repositorio de un gestor de repositorios
- **Interpretación:**  $0 \leq PIC \leq 100$ . Cuanto más alto mejor
- **Tipo de escala:** Ratio

- **Tipo de medida:** NTI, NTIC = Contador

#### TI1 - Media de días en cerrar una issue

- **Categoría:** Constantes de tiempo
- **Descripción:** Media de días en cerrar una issue
- **Propósito:** ¿Cuánto se suele tardar en cerrar una issue?
- **Fórmula:**  $MDCI = \text{SUM}(DCI) / NTIC$  . NTIC = Número total de issues cerradas, DCI = Días en cerrar la issue
- **Fuente de medición:** Repositorio de un gestor de repositorios
- **Interpretación:**  $MDCI \geq 0$ . Cuanto más pequeño mejor.
- **Tipo de escala:** Ratio
- **Tipo de medida:** NTI, NTIC = Contador

#### TC1 - Media de días entre commits

- **Categoría:** Constantes de tiempo
- **Descripción:** Media de días que pasan entre dos commits consecutivos
- **Propósito:** ¿Cuánto tiempo suele pasar desde un commit hasta el siguiente?
- **Fórmula:**  $MDEC = [\text{Sumatorio de } (TC_i - TC_j) \text{ desde } i=1, j=0 \text{ hasta } i=NTC] / NTC$ . NTC = Número total de commits, TC = Tiempo de Commit
- **Fuente de medición:** Repositorio de un gestor de repositorios
- **Interpretación:**  $MDEC \geq 0$ . Cuanto más pequeño mejor.
- **Tipo de escala:** Ratio
- **Tipo de medida:** NTC = Contador; TC = Tiempo

#### TC2 - Días entre primer y último commit

- **Categoría:** Constantes de tiempo
- **Descripción:** Días transcurridos entre el primer y el ultimo commit
- **Propósito:** ¿Cuántos días han pasado entre el primer y el último commit?
- **Fórmula:**  $DEPUC = TC2 - TC1$ . TC2 = Tiempo de último commit, TC1 = Tiempo de primer commit.
- **Fuente de medición:** Repositorio de un gestor de repositorios
- **Interpretación:**  $DEPUC \geq 0$
- **Tipo de escala:** Absoluta
- **Tipo de medida:** TC = Tiempo

### TC3 - Ratio de actividad de commits por mes

- **Categoría:** Constantes de tiempo
- **Descripción:** Muestra el número de commits relativos al número de meses
- **Propósito:** ¿Cuál es el número medio de cambios por mes?
- **Fórmula:**  $RCM = NTC / 12$
- **Fuente de medición:** Repositorio de un gestor de repositorios
- **Interpretación:**  $RCM > 0$ . Cuanto más alto mejor
- **Tipo de escala:** Ratio
- **Tipo de medida:**  $NTC = \text{Contador}$

### C1 - Número de commits en el mes pico

- **Categoría:** Constantes de tiempo
- **Descripción:** Número de commits en el mes que más commits se han realizado en relación con el número total de commits
- **Propósito:** ¿Cuál es la proporción de trabajo realizado en el mes con mayor número de cambios?
- **Fórmula:**  $CCP = NCMP / NTC$ .  $NCMP = \text{Número de commits en el mes pico}$ ,  $NTC = \text{Número total de commits}$
- **Fuente de medición:** Repositorio de un gestor de repositorios
- **Interpretación:**  $0 \leq CCP \leq 1$ . Mejor valores intermedios
- **Tipo de escala:** Ratio
- **Tipo de medida:**  $NCMP, NTC = \text{Contador}$

## Establecer una conexión a GitLab

Una vez arrancada la aplicación se mostrará un diálogo que le pedirá elegir un tipo de conexión, tal y como se muestra en la figura [E.1](#).

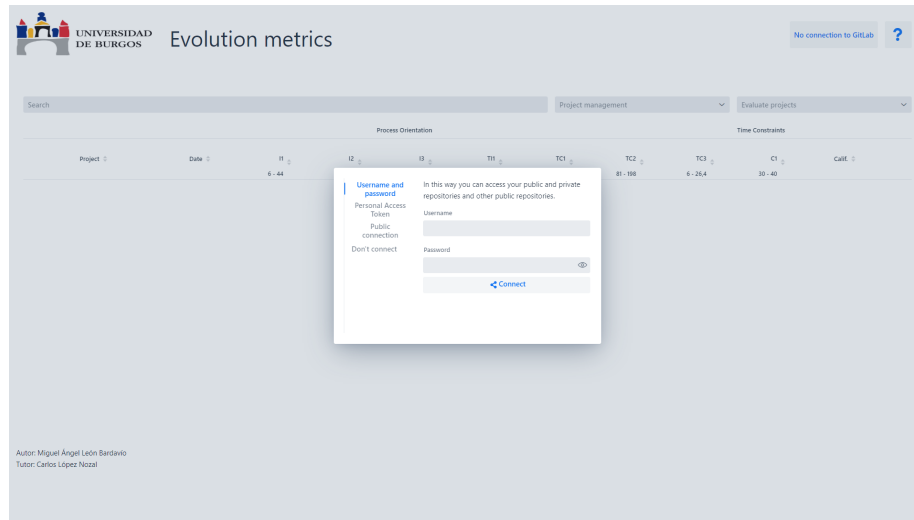


Figura E.1: Diálogo de conexión

Hay 4 posibilidades, que permiten establecer una conexión con la sesión iniciada (mediante usuario y contraseña o mediante *PA Token*), una conexión pública o utilizar la aplicación sin conexión:

**Iniciar sesión en GitLab mediante usuario y contraseña.** Se establece una conexión a GitLab iniciando sesión mediante un nombre de usuario y una contraseña. De esta forma se puede acceder a todos los repositorios públicos y privados accesibles por el usuario. Ver figura ??.

**Iniciar sesión en GitLab mediante *Personal Access Token*.** Se establece una conexión a GitLab iniciando sesión mediante un *Personal Access Token*. De esta forma se puede acceder a todos los repositorios públicos y privados accesibles por usuario, como ocurría en el caso anterior. Si se accede a GitLab desde una cuenta externa a GitLab como Google o GitHub, esta opción es la única manera de iniciar sesión con su cuenta de GitLab. Ver figura ??.

Para generar un *Personal Access Token* desde GitLab hay que iniciar sesión desde la web y entrar en la configuración del usuario. En el apartado de *Access Token* se debe dar un nombre, opcionalmente una fecha de expiración y los permisos. Para utilizar la aplicación se necesitan estos permisos: *api*, *read\_user*, *read\_repository*, *read\_registry*. Una vez finalizado, pulsar sobre el botón "*Create personal access token*", copiar el token y utilizarlo. Una vez se salga de la ventana en la que



se muestra el token, no volverá a aparecer, por lo que se recomienda copiarlo en algún lado. Ver figura E.2.

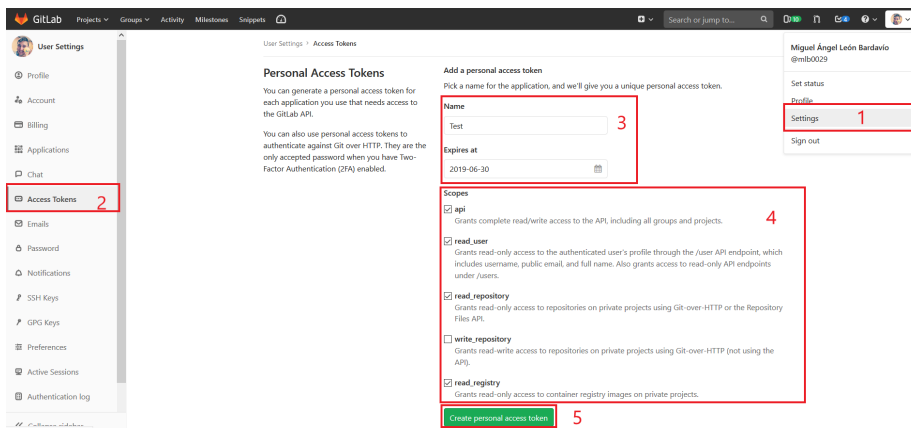


Figura E.2: Crear un *Personal Access Token* desde GitLab

**Usar una conexión pública hacia GitLab.** Se establece una conexión pública a GitLab sin iniciar sesión, por lo que solo se podrá acceder a repositorios públicos. Ver figura ??.

**No utilizar ninguna conexión.** No se realizará ninguna conexión a GitLab. Solo se podrá trabajar con proyectos y perfil de métricas importados y con el perfil de métricas por defecto. Ver figura ??.

## Página principal

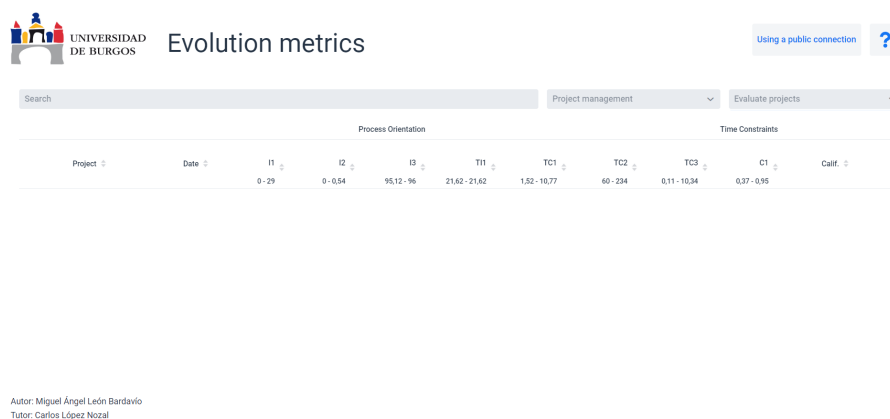


Figura E.3: Página principal

Una vez elegida por primera vez el tipo de conexión deseado se accede a la página principal, como se observa en la figura E.3.

### Cambiar el tipo de conexión

En la parte superior se puede observar el botón de conexión, que indica el tipo de conexión actual.

- Si se ha iniciado sesión mediante usuario y contraseña o mediante un *personal access token*, se mostrará la imagen del usuario y el texto: “Connected as: <nombre de usuario>”
- Si se ha establecido una conexión pública, se mostrara el texto: “Using a public connection”
- Y si no se ha establecido ninguna conexión, el texto mostrado será: “No connection to GitLab”.

Para cambiar el tipo de conexión es obligatorio cerrar, si existe, la conexión actual. Por ello, al pulsar sobre el botón de conexión, se muestra el diálogo de la figura ?? si existe una conexión y el diálogo de la figura ?? si no existe conexión. Al pulsar sobre “*Connect*” o sobre “*Close connection*” se abrirá el diálogo de conexión de las figuras E.1 y ??.

### Botón de ayuda

A la derecha del botón de conexión se encuentra un botón que da acceso a este manual en la Wiki del proyecto.

### Listado de proyectos

En el centro de la página principal se pueden gestionar los proyectos. Consta de una barra de búsqueda, dos menús, y una tabla que visualiza las métricas de los proyectos que se añadan.

En el cuadro de búsqueda se filtrarán los repositorios por su nombre mientras se vaya escribiendo.

En el menú de “*Project management*” existen estas opciones, como se muestra en la figura ??:

- **Add new.** Permite añadir uno o varios proyectos.
- **Import.** Permite importar proyectos a partir de un fichero previamente exportado.

- ***Export***. Permite exportar todos los proyectos existentes a un fichero, lo que permitirá su posterior importación. Se almacena en un fichero con formato “.emr”.
- ***Export to CSV***. Permite generar un fichero CSV que contenga toda la información de la tabla de proyectos. Este fichero no servirá para importar los proyectos posteriormente.

En el menú de “*Evaluate projects*” existen estas opciones, como se muestra en la figura ??:

- ***Evaluate with new profile***. Permite evaluar los proyectos calculando los valores mínimos y máximos de cada métrica a partir de los repositorios actuales.
- ***Evaluate with default profile***. Permite evaluar los proyectos con un perfil por defecto creado a a partir de un conjunto de datos<sup>1</sup> de un estudio empírico de las métricas de evolución del software en trabajos finales de grado[3].
- ***Evaluate with imported profile***. Permite evaluar los proyectos a partir de un perfil de métricas previamente exportado.
- ***Export actual profile***. Permite exportar el perfil de métricas actual para su posterior importación. Se almacena en un fichero con formato “.emmp”.

La tabla muestra los valores medidos de las métricas para cada proyecto, ver figura E.4.

---

<sup>1</sup>[https://github.com/clopezno/clopezno.github.io/blob/master/agile\\_practices\\_experiment/DataSet\\_EvolutionSoftwareMetrics\\_FYP.csv](https://github.com/clopezno/clopezno.github.io/blob/master/agile_practices_experiment/DataSet_EvolutionSoftwareMetrics_FYP.csv)


		Process Orientation					Time Constraints			
Project	Date	I1	I2	I3	TI1	TC1	TC2	TC3	C1	Calif.
		0 - 41	0 - 0,52	95,12 - 97,75	12,51 - 21,62	1,08 - 8,84	110 - 252	0,14 - 11	0,28 - 0,93	
 UBU-EnergySaver	2/7/19	0	0	NC	NC	1,1	11	11	0,91	25%
 UBU-ReactVR	2/7/19	25	0,57	96%	21,62	5	229	5,5	0,27	62,5%
 UBU-MultiAPI	2/7/19	0	0	NC	NC	12,69	208	2,12	0,47	37,5%
 UBU-SmartFiller	2/7/19	41	0,51	95,12%	21,62	2,77	235	10,12	0,64	87,5%
 comparador-de-métricas	2/7/19	89	0,44	97,75%	12,51	1,07	268	22,67	0,28	37,5%
 UBU-BC4Distribution	2/7/19	0	0	NC	NC	NC	NC	0,14	1	0%
 UBU-TestinoAWE	2/7/19	0	NC	NC	NC	NC	NC	0	NC	0%

Figura E.4: Tabla que muestra los valores medidos en los proyectos para las métricas

La tabla presenta las siguientes columnas:

- **Botón de eliminar.** Permite eliminar de la tabla el proyecto seleccionado.
- **Project.** Nombre del proyecto con enlace al repositorio de GitLab. Si el nombre es demasiado largo y se corta, se puede utilizar el tooltip que aparece al pasar el ratón por encima del nombre del proyecto.
- **Date.** Fecha de la última vez que se obtuvieron las métricas del proyecto.
- **Métricas.** Valor medido de las métricas y un color que evalúa la medida en relación a un perfil de métricas.  
Las métricas están clasificadas por categoría: Proceso de orientación (*Process Orientation*) y Constantes de tiempo (*Time Constraints*). En la cabecera se muestra el nombre de la métrica pero aparecerá la descripción al pasar el puntero del ratón por encima del nombre en forma de tooltip. Un **perfil de métricas** es un conjunto de valores mínimo y máximo definidos para cada métrica. Los valores que se encuentran debajo del nombre de las métricas en la cabecera son esos valores mínimo y máximo separados por un guión para su respectiva métrica. En el tooltip se muestra el valor mínimo como Q1 y el valor máximo como Q3.
- **Botón de recalcular métricas.** Permite volver a obtener las métricas del proyecto (si es posible según la conexión actual a GitLab) y evaluar las nuevas métricas de acuerdo al perfil actual. Se mostrará un mensaje de aviso si se han recalculado correctamente y un mensaje de error en caso contrario.

### Añadir un proyecto

Para añadir un nuevo proyecto, el tipo de conexión deberá ser distinto de “Sin conexión” (*No connection to GitLab*), es decir que debe haber una conexión. Seleccionar la opción “*Add new*” del menú “*Project management*”, ver figura ???. Se abrirá un diálogo como el de la figura ???. Para cancelar se puede pulsar *Esc* o hacer click fuera del diálogo. Existen tres posibilidades para añadir un proyecto:

**Añadir por pertenencia a un usuario.** Se solicita en el campo izquierdo del formulario el nombre de usuario o ID del usuario del cual se desean cargar los proyectos en campo desplegable de la derecha. Se mostrará un mensaje en rojo si el usuario no se existe “*User not found*” y un mensaje si el usuario existe “*User found*”, en ese caso se cargarán todos los proyectos del usuario en el desplegable de la derecha. Seleccionar uno de sus proyectos y pulsar sobre el botón “*Add*”. Se mostrarán solo los repositorios públicos (incluyendo *forks*) del usuario. No se mostrarán proyectos privados a menos que se haya establecido una conexión con sesión y el usuario especificado en el campo de la izquierda coincida con el usuario que haya iniciado sesión.

**Añadir por pertenencia a un grupo.** El funcionamiento es el mismo que en el caso anterior, con la diferencia de que se solicita el nombre de grupo o ID del grupo en el campo izquierdo. Si el grupo es privado y el tipo de conexión es pública o el usuario que haya iniciado sesión no tiene acceso al grupo, se mostrará un mensaje en rojo de la misma forma que si el grupo no existiera “*Group not found*”. Si se encuentra el grupo, se mostrará el mensaje “*Group found*” y se cargarán todos los proyectos del usuario en el desplegable de la derecha. Seleccionar uno de sus proyectos y pulsar sobre el botón “*Add*”.

**Añadir por URL Web.** Se solicita la URL Web del proyecto de GitLab. Si no se encuentra (porque no existe o porque con la conexión actual no se tiene acceso al proyecto) se mostrará un mensaje en rojo al pulsar sobre “**Add**”: “*Project not found. It doesn't exists or may be inaccessible due to your connection level.*”

Al añadir un nuevo proyecto, se calcularán por primera vez sus métricas y se evaluarán de acuerdo al perfil de métricas actual. Si no se ha creado o importado ningún perfil, se evaluará según el perfil por defecto.

### Importar proyectos

Los proyectos se pueden importar independientemente del tipo de conexión que exista. Se mostrará un diálogo que permite seleccionar o arrastrar al cuadro un fichero con formato *.emr*, ver figura ???. Se puede seleccionar otro fichero en caso de haber escogido un fichero no deseado. Una vez se cargue el fichero, pulsar sobre “Import”. Se mostrará un mensaje de error en caso de que el fichero este corrompido (ha sido modificado por una herramienta externa a la aplicación) y no se podrá utilizar ese fichero.

### Exportar proyectos

Para exportar proyectos debe haber al menos un proyecto en la tabla. Se puede exportar a un fichero *.emr* para su posterior importación o en un fichero *.csv*. Para exportar hay que seleccionar la opción correspondiente en el menú “*Project management*”, ver figura ???. El dialogo para la exportación se muestra en la figura ???. Basta con pulsar sobre “*Download*” para poder descargar el fichero.

### Evaluar los proyectos

Evaluar un proyecto es evaluar y todas las medidas de las métricas que se realizaron sobre el proyecto en relación a un perfil de métricas en el que se definen los valores mínimos y máximos.

El resultado de la evaluación puede ser bueno (la medida se pinta en verde en la tabla), malo (la medida se pinta en rojo) o “advertencia” (la medida equivale al valor mínimo o al valor máximo).

Para evaluar un proyecto hay que elegir el perfil de métricas con el que se va a evaluar. Por defecto se coge un perfil de métricas en el que los valores mínimos se corresponden con los cuartiles Q1 y los valores máximos con cuartiles Q3 de un conjunto de medidas tomadas sobre TFGs<sup>2</sup>[3].

Al evaluar se evalúan todos los proyectos. Se puede elegir el perfil de métricas según la opción elegida en el menú “*Evaluate projects*” de la figura ???:

- ***Evaluate with new profile.*** Coge como entrada todas las medidas de la tabla y calcula, por cada métrica, los cuartiles Q1 y Q3 para definirlos como valor mínimo y valor máximo de la métrica, respectivamente.

---

<sup>2</sup>[https://github.com/clopezno/clopezno.github.io/blob/master/agile\\_practices\\_experiment/DataSet\\_EvolutionSoftwareMetrics\\_FYP.csv](https://github.com/clopezno/clopezno.github.io/blob/master/agile_practices_experiment/DataSet_EvolutionSoftwareMetrics_FYP.csv)

- ***Evaluate with default profile.*** Permite evaluar los proyectos con el perfil por defecto mencionado anteriormente.
- ***Evaluate with imported profile.*** Permite importar el perfil de métricas de un fichero *.emmp*. El perfil se debe haber creado y exportado anteriormente.

### Exportar perfil de métricas

Se puede exportar a un fichero *.emmp* para su posterior importación. Para exportar el perfil de métricas, seleccionar la opción correspondiente del menú “*Evaluate projects*” de la figura ??: “*Export actual profile*”. El diálogo para la exportación es similar al de la figura ?. Basta con pulsar sobre “*Download*” para poder descargar el fichero que contendrá el perfil de métricas actual.





---

## Bibliografía

---

- [1] Iubaris Info 4 Media SL. . Scrum Manager: Temario Troncal I. Versión 2.6.1:94, January 2019.
- [2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Patrones De Diseño: Elementos De Software Orientado a Objetos Reutilizable*. Addison-Wesley, Madrid, 1 ed. en es edition, 2002.
- [3] Carlos López. Portal web para la gestión de la asignatura de trabajos fin de Grado de Ingeniería : clopezno/clopezno.github.io, June 2019. original-date: 2016-09-27T09:36:06Z.
- [4] Raúl Marticorena, Yania Crespo, and Carlos López. Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones. In *X Jornadas Ingeniería del Software y Bases de Datos (JISBD 2005)*, Granada, Spain ISBN: 84-9732-434-X, pages 59–66, September 2005.
- [5] Jacek Ratzinger. *sPACE: Software Project Assessment in the Course of Evolution*. PhD Thesis, 2007.
- [6] rlp0019. Software para la autoevaluación del proceso de Trabajo de Fin de Grado (TFG) en GitHub.: rlp0019/Activiti-API, June 2019. original-date: 2019-02-17T19:30:04Z.
- [7] Author Angel Luis Lozano Sánchez. Requisitos vs Casos de uso vs Historias de Usuario, February 2016.