



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Evaluación de simulaciones de  
proyectos en GitHub  
orientadas a docencia.  
Documentación Técnica**



Presentado por Licinio Fernández Maurelo  
en Universidad de Burgos — 11 de junio  
de 2023

Tutor: Dr. Carlos López Nozal



---

# Índice general

---

Índice general	i
Índice de figuras	iii
Índice de tablas	iv
<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>9</b>
B.1. Introducción . . . . .	9
B.2. Objetivos generales . . . . .	9
B.3. Catalogo de requisitos . . . . .	10
B.4. Especificación de requisitos . . . . .	10
<b>Apéndice C Especificación de diseño</b>	<b>13</b>
C.1. Introducción . . . . .	13
C.2. Diseño de modelo de dominio . . . . .	13
C.3. Diseño procedimental . . . . .	15
C.4. Diseño arquitectónico . . . . .	17
<b>Apéndice D Documentación técnica de programación</b>	<b>19</b>
D.1. Introducción . . . . .	19
D.2. Estructura de directorios . . . . .	19
D.3. Manual del programador . . . . .	21

D.4. Compilación, instalación y ejecución del proyecto . . . . .	23
D.5. Pruebas del sistema . . . . .	24
<b>Apéndice E Documentación de usuario</b>	<b>27</b>
E.1. Introducción . . . . .	27
E.2. Requisitos de usuarios . . . . .	27
E.3. Instalación . . . . .	27
E.4. Manual del usuario . . . . .	27
<b>Bibliografía</b>	<b>29</b>

---

# Índice de figuras

---

A.1. Detalle parcial de sprints cerrados en este trabajo en GitHub . .	2
A.2. Tareas sprint 6 en GitHub . . . . .	3
A.3. Java Developer Hourly Rate Guide . . . . .	7
A.4. Inclusión de licencia MIT del trabajo en GitHub . . . . .	8
B.1. Diagrama de casos de uso . . . . .	10
C.1. Modelo de datos para la gestión de proyectos ágiles . . . . .	14
C.2. Modelo de datos para el control de versiones . . . . .	15
C.3. Diseño procedimental de la evaluación del laboratorio virtual de gestión de proyectos ágiles . . . . .	16
C.4. Diseño procedimental de la evaluación del laboratorio virtual de control de versiones . . . . .	17
C.5. Arquitectura hexagonal simplificada del proyecto . . . . .	18
D.1. Estructura de directorios del proyecto . . . . .	20
D.2. Estado del proyecto en SonarCloud . . . . .	22
D.3. Captura definición de la integración continua en GitHub. Las referencias a los tokens de GitHub y SonarCloud se recuadran en color rojo . . . . .	23
D.4. Resultados de la ejecución de los tests. En el recuadro rojo aparece una url temporal con el enlace específico al informe de tests realizados con Cucumber . . . . .	24
D.5. Resumen de la ejecución de tests con Cucumber . . . . .	25
D.6. Detalle de la ejecución del escenario de importación de una tarea	25

---

# Índice de tablas

---

B.1. UC1 APM Case Study Simulation Evaluation. . . . .	11
B.2. UC2 SCM Case Study Simulation Evaluation. . . . .	12

## *Apéndice A*

---

# **Plan de Proyecto Software**

---

### **A.1. Introducción**

En todo proyecto software es fundamental considerar los aspectos relativos a:

- Planificación temporal
- Estudio de viabilidad económica y legal

### **A.2. Planificación temporal**

La planificación temporal es un componente vital de la gestión de proyectos, que contribuye a la finalización exitosa del mismo. La gestión temporal eficaz de un proyecto implica:

- Creación de un cronograma de proyecto realista
- Identificación de hitos clave
- Definición de entregables

En este trabajo se ha hecho uso de metodología scrum, dividiendo las tareas del proyecto en sprints de duración de dos semanas como norma general si bien en algunos casos estos sprints se han ampliado en duración.

El detalle de los sprints del proyecto puede consultarse en **GitHub**<sup>1</sup> como puede apreciarse en A.1:

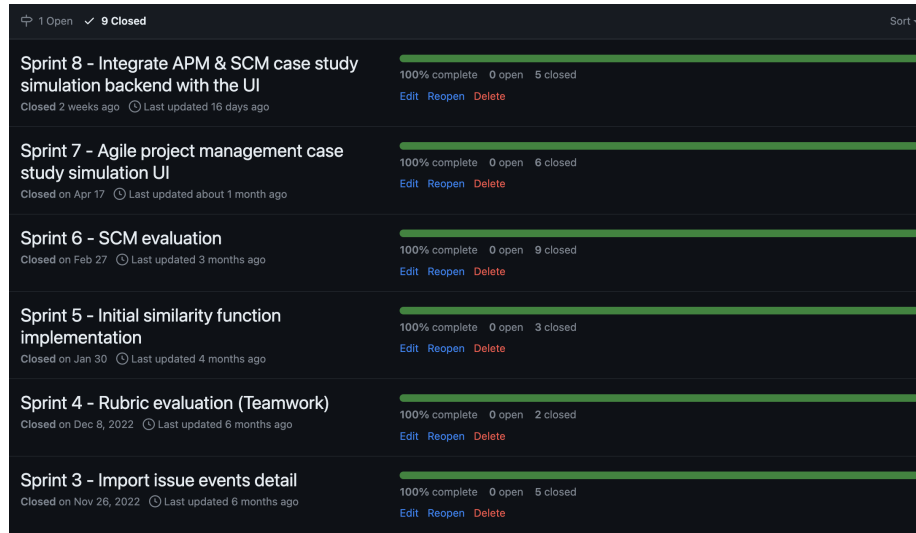


Figura A.1: Detalle parcial de sprints cerrados en este trabajo en GitHub

Cada sprint contiene las tareas que lo componen como puede apreciarse en la captura del sprint 6 A.2.

<sup>1</sup>GitHub, sprints finalizados: <https://github.com/lfx1001/avrela/milestones?state=closed>



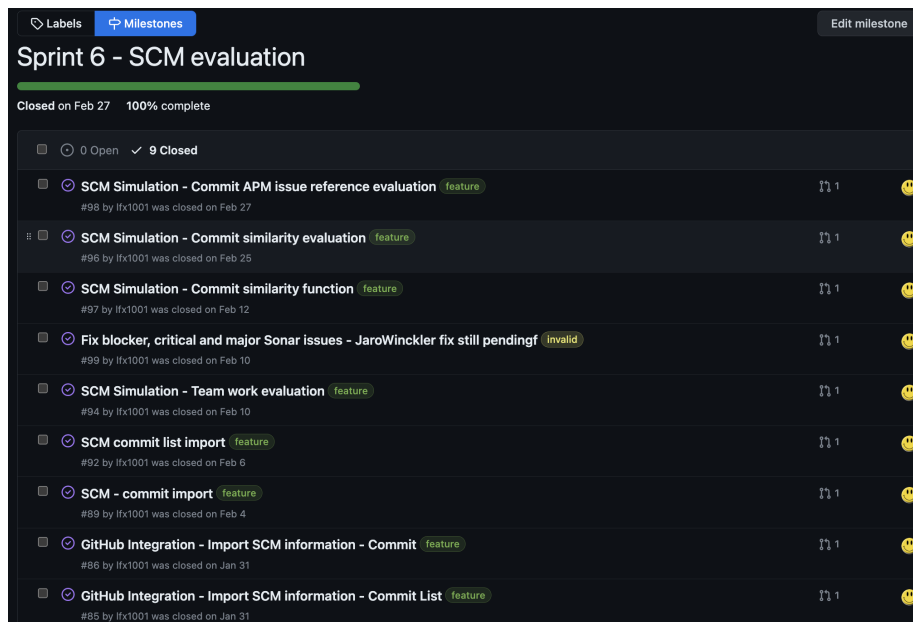


Figura A.2: Tareas sprint 6 en GitHub

Se enumeran a continuación los sprints que se han realizado en el proyecto junto con una descripción breve de los mismos.

**Sprint 0 - Kick-off initial project documentation.** En este primer sprint

- revisamos la temática de este trabajo y comenzamos la redacción de los apartados de introducción y objetivos.
- se creó el repositorio GitHub como soporte de los activos documentales.
- se realizaron unas primeras pruebas de trabajo con las plantillas  $\text{\LaTeX}$  correspondiente a la memoria y anexos de este trabajo.

**Sprint 1 - Initial domain model, GitHub integration .** En este segundo sprint:

- se creó el proyecto Java asociado.
- se codificaron los primeros elementos del modelo.
- experimentamos con el API REST de GitHub y realizamos pruebas creando un cliente del API REST personalizado utilizando Open Feign.

- se configuró la integración continua para poder tener un feedback actualizado de métricas fundamentales en la calidad de software (cobertura de tests, duplicados, code smells).

**Sprint 2 - Import agile project management data.** En este sprint:

- se implementa la ingesta de información de la gestión ágil de proyectos.
- se incluyen tests funcionales utilizando Cucumber.
- se analiza la problemática de la integración de la información de los puntos de historia de usuario desde ZenHub.

**Sprint 3 - Import issue events detail.** En este sprint:

- se complementó la información relativa a la gestión de proyectos ágiles añadiendo e integrando la parte de eventos de las tareas.
- se llevaron a cabo tareas de documentación de algunas de las herramientas y técnicas utilizadas hasta el momento.

**Sprint 4 - Rubric evaluation (Teamwork).** En este sprint:

- se trabaja el aspecto de evaluación de las simulaciones.
- se implementa la evaluación de trabajo el equipo en la gestión de proyectos ágiles utilizando Cucumber.

**Sprint 5 - Initial similarity function implementation.** En este sprint:

- se implementa la función de semejanza para la comparación de tareas utilizando el índice de Jaccard.
- se evalúa el criterio de semejanza de descripción de tareas.

**Sprint 6 - SCM evaluation.** En este sprint:

- se implementa la importación de commits dado un intervalo de tiempo.

- se implementa la función de semejanza de commits. Se utiliza la distancia de Jaro-Wincker junto con el índice de Jaccard.
- se evalúa el criterio de trazabilidad de la simulación de control de versiones con la herramienta de gestión de proyectos.
- se evalúa el criterio de trabajo en equipo de la simulación de control de versiones.
- se evalúa el criterio de semejanza de la simulación de control de versiones.

**Sprint 7 - Agile project management case study simulation UI.**

En este sprint:

- se realizan pruebas de implementación de interfaces de usuario con Thymeleaf.
- se implementa una versión inicial de pantallas para la evaluación de proyectos ágiles.

**Sprint 8 - Integrate APM SCM case study simulation backend with the UI.** En este sprint:

- se implementa una versión inicial de pantallas para la evaluación de control de versiones.
- se integran las pantallas de evaluación de gestión ágil de proyectos con la lógica de negocio.
- se integran las pantallas de evaluación de control de versiones con la lógica de negocio.

**Sprint 9 - Documentation Web application improvements.** En este sprint:

- se corrigen incidencias de las evaluación.
- se realizan mejoras de la interfaz visual de la evaluación de simulaciones de control de versiones.
- se trabaja en la documentación de anexos y memoria.

**Sprint 10 - Documentation Web application improvements.** En este sprint:

- se corrigen incidencias de las evaluación.
- se realizan mejoras de la interfaz visual de la evaluación de simulaciones de gestión de proyectos ágiles.
- se trabaja en la documentación de anexos y memoria.

## A.3. Estudio de viabilidad

### Viabilidad económica

Los costes de este proyecto software se basan en las siguientes líneas:

- Coste de desarrollo.
- Coste de recursos asociados.

En este caso vamos a optar por un cálculo simplificado que incluya los propios recursos asociados en el coste de desarrollo bajo el supuesto de que este proyecto se gestionase "llave en mano". De esta forma, el cliente (en esta caso, la universidad) encargaría a un tercero (el alumno) la realización del proyecto como un proveedor externo.

Este proveedor externo gestionaría internamente los gastos correspondientes recursos humanos y materiales. Proporcionaría al cliente una cifra final de costes. Este coste está determinado por la expresión:

$$\text{Coste} = \text{Jornadas} \times \text{Tarifa jornada}$$

Jornadas representa el número de jornadas necesarias para acometer el proyecto. Se estima en 40 jornadas (320 horas).

Tarifa jornada representa el coste unitario de la jornada. Este coste está relacionado con el perfil del recurso que vaya a acometer el proyecto. Para el cálculo se considera un perfil de desarrollador Java Senior. La tarifa presenta una gran variabilidad en función de la localización. Aprovecharemos el estudio realizado por **flexiple**<sup>2</sup> cuya captura se recoge en A.3. Este estudio estima en 55,43 Euros (60 Dólares) la tarifa hora, lo que supone un coste de 443,7 Euros por jornada.

---

<sup>2</sup><https://flexiple.com/java/hourly-rate/>

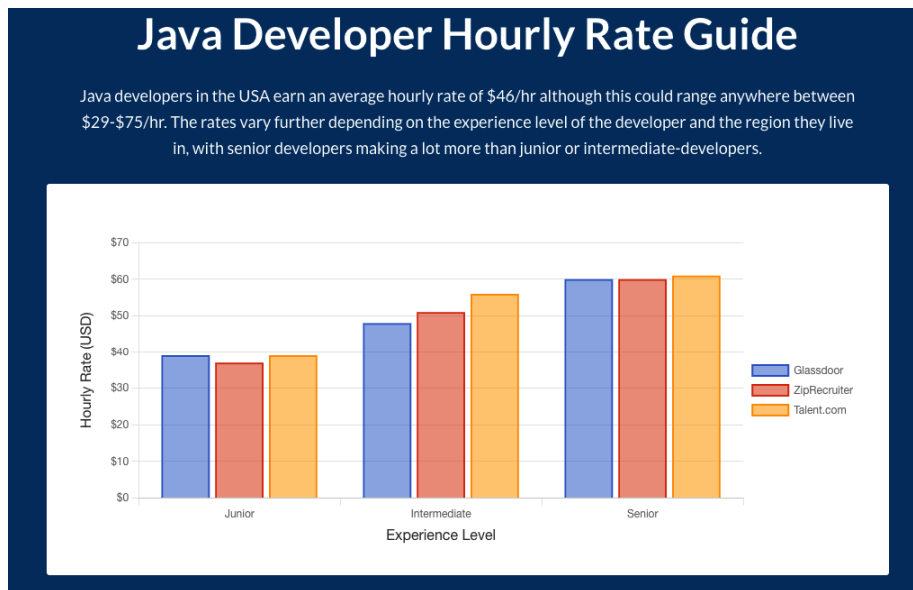


Figura A.3: Java Developer Hourly Rate Guide

En base a los supuestos anteriores, una estimación posible del coste del proyecto, impuestos no incluidos, es:

$$40 \text{ Jornadas} \times 443,7 \text{ Euros Tarifa jornada} = 17.748 \text{ Euros}$$

## Viabilidad legal

La viabilidad legal abarca el cumplimiento de las leyes, reglamentos y derechos de propiedad intelectual pertinentes. Implica consideraciones tales como licencias, privacidad de datos, seguridad y cualquier obligación legal asociada con el proyecto de software. Al abordar de manera proactiva los aspectos legales, mitigamos los riesgos potenciales y nos aseguramos de que el proyecto opere dentro de los límites de las leyes aplicables.

Se listan a continuación las librerías y agrupaciones de librerías (indicadas con la terminación `.*`) con su licencia correspondiente:

- `org.springframework.boot.*` : licencia Apache-2.0
- `org.springframework.cloud.*` : licencia Apache-2.0
- `org.webjars:bootstrap` : licencia MIT

Las licencias anteriores nos permitirían el uso de las mismas con fines comerciales. Este trabajo se distribuye bajo licencia **MIT**<sup>3</sup>. Para facilitar conocer la licencia de este trabajo se ha añadido la misma en GitHub [A.4](#).

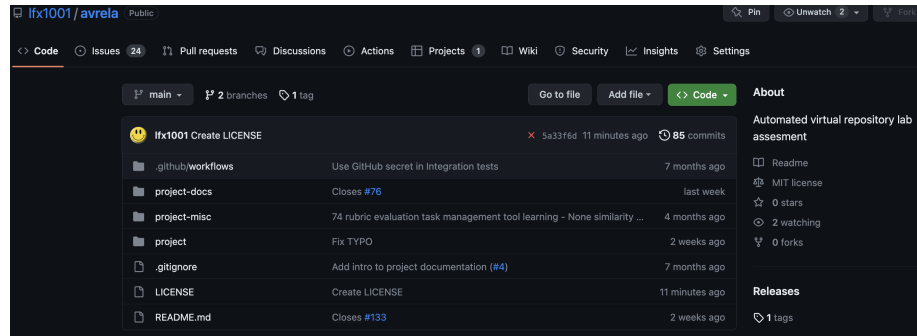


Figura A.4: Inclusión de licencia MIT del trabajo en GitHub

---

<sup>3</sup>Licencia MIT: <https://choosealicense.com/licenses/mit/>

## *Apéndice B*

---

# Especificación de Requisitos

---

### B.1. Introducción

En este apéndice se detalla lo que software debe hacer y cómo debe comportarse con un doble objetivo:

- aclarar las expectativas y necesidades de los usuarios o partes interesada.
- servir como referencia para el equipo de desarrollo.

### B.2. Objetivos generales

Se enumeran a continuación los objetivos generales del proyecto:

- Aplicación de rúbrica de evaluación sobre:
  - La información de gestión de tareas y del control de versiones de un repositorio.
  - La comparación de la información de gestión de tareas y de control de versiones de dos repositorios, el original y el resultante de la simulación.
- Proporcionar retroalimentación de la comparación de la información de gestión de tareas y de control de versiones de dos repositorios.

### B.3. Catalogo de requisitos

La lista de requisitos funcionales es:

- REQ1: Evaluación del laboratorio virtual de gestión de proyectos ágil y retroalimentación.
- REQ2: Evaluación del laboratorio virtual de control de versiones y retroalimentación.

### B.4. Especificación de requisitos

A partir de los requisitos funcionales se plantea el diagrama de casos de uso de la Figura B.1:

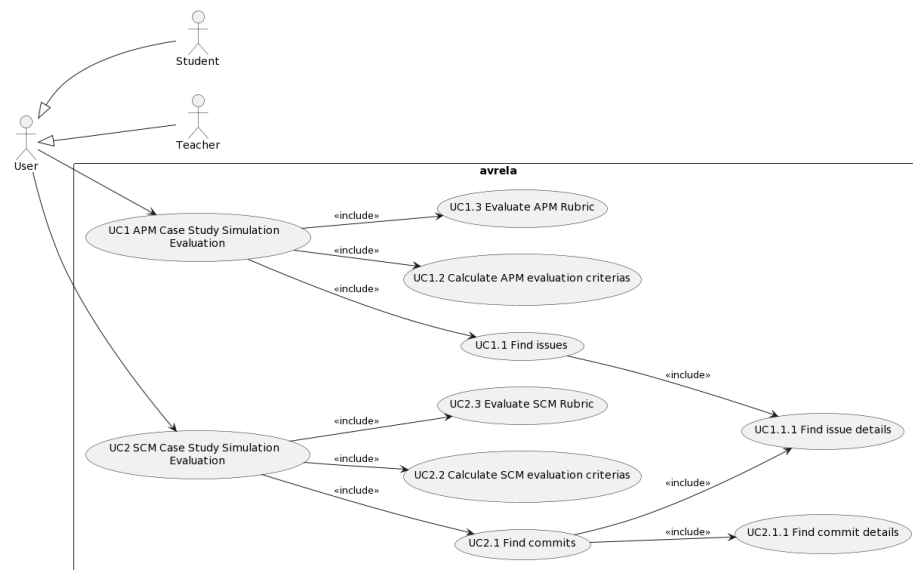


Figura B.1: Diagrama de casos de uso



UC1	APM Case Study Simulation Evaluation
<b>Versión</b>	1.0
<b>Autor</b>	Alumno
<b>Requisitos asociados</b>	REQ1
<b>Descripción</b>	Evaluación y retroalimentación del laboratorio virtual de gestión de proyectos ágil
<b>Precondiciones</b>	<ol style="list-style-type: none"> <li>1. Existe el caso de estudio</li> <li>2. Existe la simulación</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. Recuperar datos del caso de estudio</li> <li>2. Recuperar datos de la simulación</li> <li>3. Calcular criterios de evaluación</li> <li>4. Aplicar rúbrica</li> <li>5. Mostrar información recuperada del caso de estudio y de la simulación</li> <li>6. Mostrar rúbrica y nota final</li> <li>7. Mostrar retroalimentación</li> </ol>
<b>Postcondición</b>	
<b>Excepciones</b>	
<b>Importancia</b>	Alta

Tabla B.1: UC1 APM Case Study Simulation Evaluation.

UC2	SCM Case Study Simulation Evaluation
<b>Versión</b>	1.0
<b>Autor</b>	Alumno
<b>Requisitos asociados</b>	REQ2
<b>Descripción</b>	Evaluación y retroalimentación del laboratorio virtual de gestión de control de versiones
<b>Precondiciones</b>	<ol style="list-style-type: none"> <li>1. Existe el caso de estudio</li> <li>2. Existe la simulación</li> </ol>
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. Recuperar datos del caso de estudio</li> <li>2. Recuperar datos de la simulación</li> <li>3. Calcular criterios de evaluación</li> <li>4. Aplicar rúbrica</li> <li>5. Mostrar información recuperada del caso de estudio y de la simulación</li> <li>6. Mostrar rúbrica y nota final</li> <li>7. Mostrar retroalimentación</li> </ol>
<b>Postcondición</b>	
<b>Excepciones</b>	
<b>Importancia</b>	Alta

Tabla B.2: UC2 SCM Case Study Simulation Evaluation.

## *Apéndice C*

---

# Especificación de diseño

---

### C.1. Introducción

La especificación de diseño sirve de referencia durante todo el proceso de desarrollo ayudando a garantizar que el software se construya de acuerdo con el diseño deseado. También sirve como documentación para futuras modificaciones o mantenimiento.

### C.2. Diseño de modelo de dominio

En este proyecto se trabaja con varios dominios:

- Gestión de proyectos ágil.
- Control de versiones.

El dominio de la gestión de proyectos ágil se refleja en el diagrama de la Figura [C.1](#).



Figura C.1: Modelo de datos para la gestión de proyectos ágiles

El modelo de dominio del control de versiones obedece al diagrama de la Figura C.2.

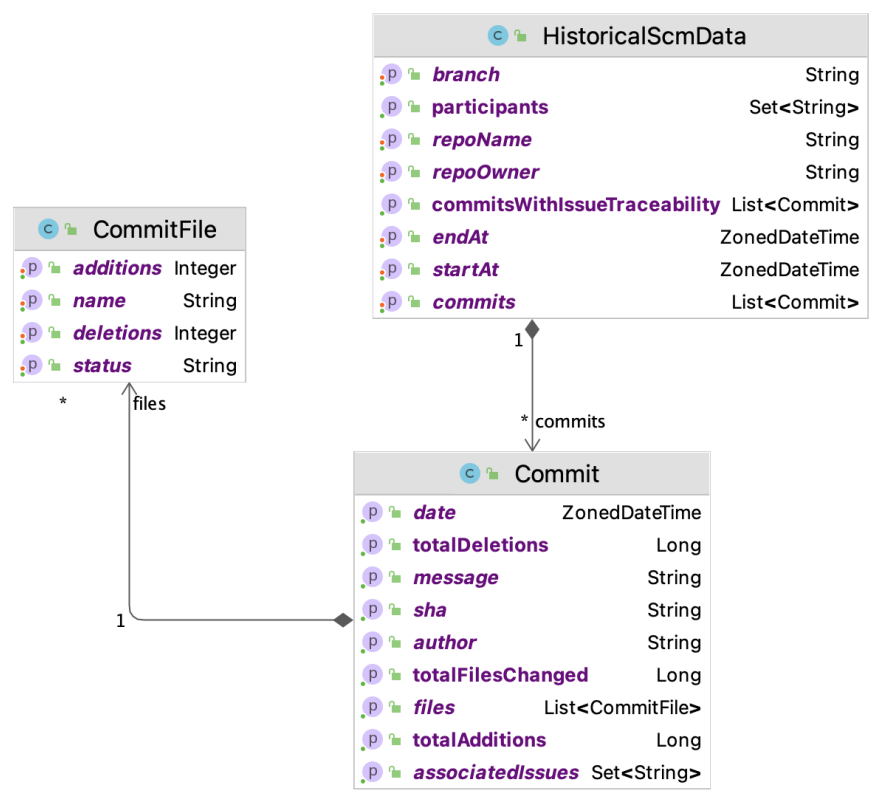


Figura C.2: Modelo de datos para el control de versiones

### C.3. Diseño procedimental

Para modelar este diseño se utilizarán diagramas UML de secuencia. En ellos se han creado grupos para facilitar la identificación de la correspondencia entre los casos de uso y el diseño procedimental.

El diseño procedimental de la evaluación del laboratorio virtual de gestión de proyectos ágil corresponde al diagrama de la Figura C.3:

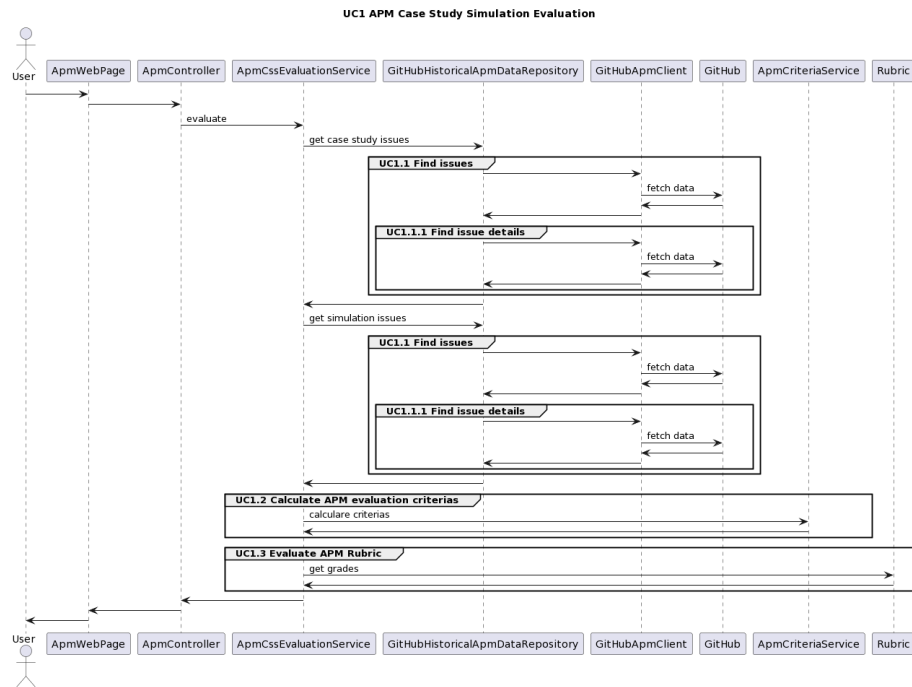


Figura C.3: Diseño procedimental de la evaluación del laboratorio virtual de gestión de proyectos ágiles

El diseño procedimental de la evaluación del laboratorio virtual de control de versiones corresponde al diagrama de la Figura C.4:

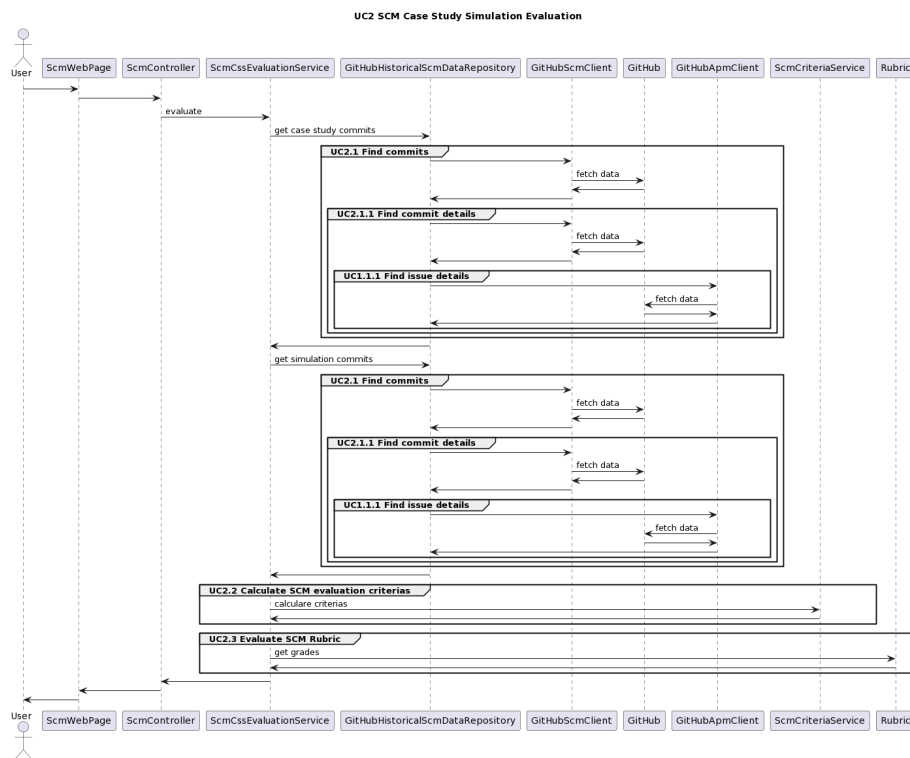


Figura C.4: Diseño procedimental de la evaluación del laboratorio virtual de control de versiones

## C.4. Diseño arquitectónico

La aplicación se ha desarrollado siguiendo una arquitectura hexagonal cuya estructura simplificada se muestra en la Figura C.5:

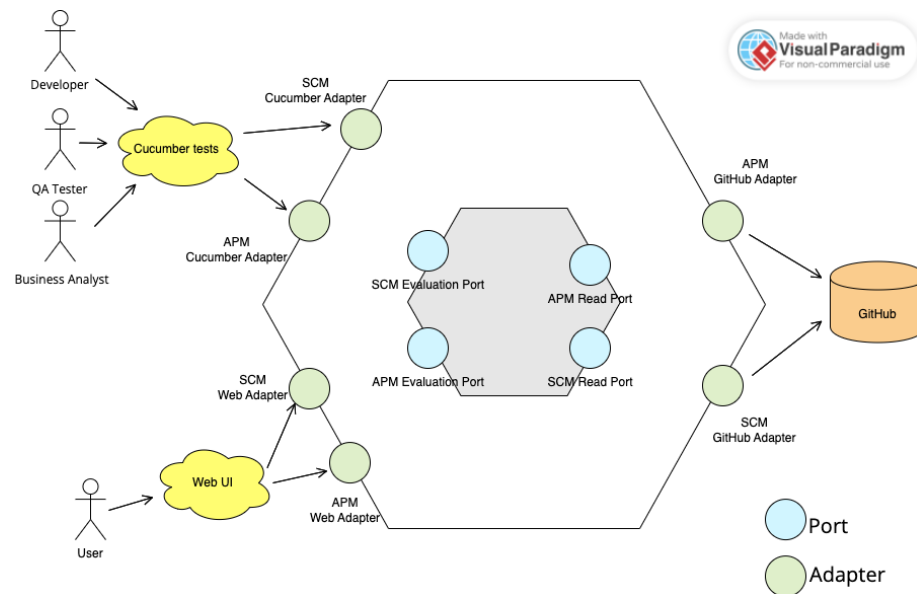


Figura C.5: Arquitectura hexagonal simplificada del proyecto



## *Apéndice D*

---

# Documentación técnica de programación

---

## D.1. Introducción

En este apéndice se incluye información de utilidad para los trabajos de mantenimiento y evolución del software.

## D.2. Estructura de directorios

La configuración de la integración continua se encuentra en la carpeta `.github/workflows`.

El proyecto software desarrollado se encuentra en la carpeta `/project`.

Para el desarrollo del proyecto software se ha utilizado la herramienta **Maven** que promueve una estructura de proyectos estándar. Esta estructura puede observarse a un primer nivel en la Figura [D.1](#).

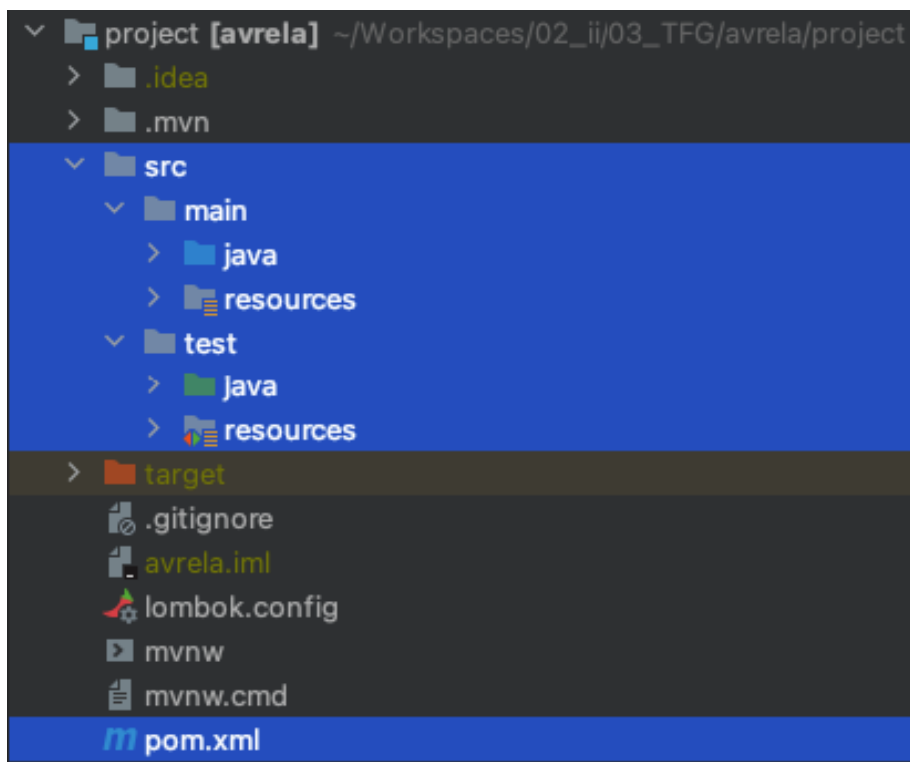


Figura D.1: Estructura de directorios del proyecto

Se enumeran a continuación los elementos principales presentes en la estructura de directorios:

- **pom.xml**: Contiene la configuración maven necesaria para el proyecto
- **src/main**: Activos de la aplicación para su ejecución
  - **src/main/java**: Activos de código Java
  - **src/main/resources**: Recursos adicionales. Contiene la definición de las plantillas para las páginas web y recursos estáticos utilizados por la misma (hojas de estilo css e imágenes).
- **src/test**: Activos de pruebas de la aplicación
  - **src/test/java**: Activos de código Java de los test
  - **src/test/resources**: Recursos para los activos de tests.

Los activos relacionados con Cucumber se encuentran en las ubicaciones siguientes:

- **src/test/resources/features:** Recursos para las definiciones de escenarios con la descripción de los escenarios Cucumber, también llamados features. Subcarpetas **./apm**, **./scm**, **./css**: directorios con fichero **.features** de Cucumber con los escenarios.
- **src/test/java/es/ubu/lsi/avrela/bdd:**
  - **CucumberTests.java:** Configuración para los tests
  - Subcarpetas **./apm**, **./scm**, **./css**: directorios con activos de código Java correspondientes a la definición de pasos de los escenarios Cucumber.

## D.3. Manual del programador

El repositorio git del proyecto se encuentra disponible en <https://github.com/lfx1001/avrela>.

Para descargar el repositorio en nuestro almacenamiento local ejecutar:

```
$ git clone https://github.com/lfx1001/avrela
```

Una vez descargado el proyecto podemos importarlo como proyecto Maven en el Integrated Development Environment (IDE) de nuestra elección. Para la realización de este proyecto se ha utilizado IntelliJ IDEA 2022.2.3 (Ultimate Edition).

Se recomienda crear un nuevo repositorio remoto para poder sincronizar los cambios locales en la plataforma de hosting git GitHub para poder adaptar la integración continua del proyecto fácilmente. Una vez creado, podemos apuntar a nuestro nuevo repo:

```
$ git remote set-url origin <url del nuevo repositorio>
```

Una vez hayamos realizado la configuración anterior, bastará subir los cambios a nuestro repositorio remoto:

```
$ git push origin
```

Una vez dispongamos de nuestro propio repositorio remoto alojado, es necesario realizar las siguientes tareas para mantener operativa la integración continua:

- Creación de un nuevo proyecto en SonarCloud.

- Configuración de la integración continua.

Para el análisis de calidad de código del proyecto se utiliza la plataforma **SonarCloud**. La url del proyecto avrela en SonarCloud es [https://sonarcloud.io/project/overview?id=lfx1001\\_avrela](https://sonarcloud.io/project/overview?id=lfx1001_avrela). El estado del proyecto se muestra en la Figura D.2

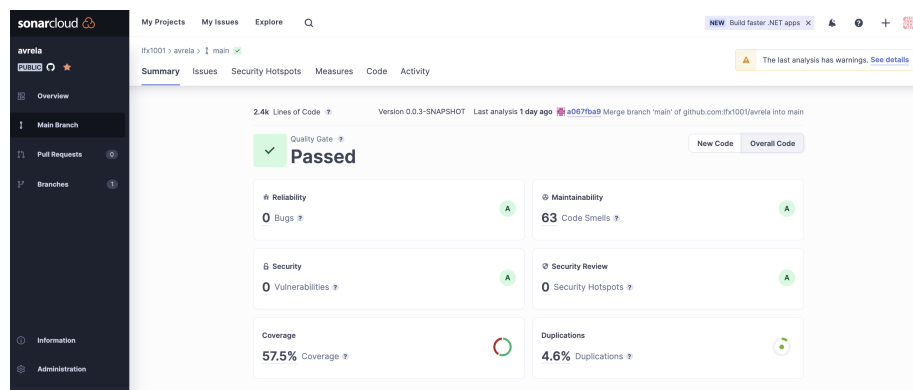
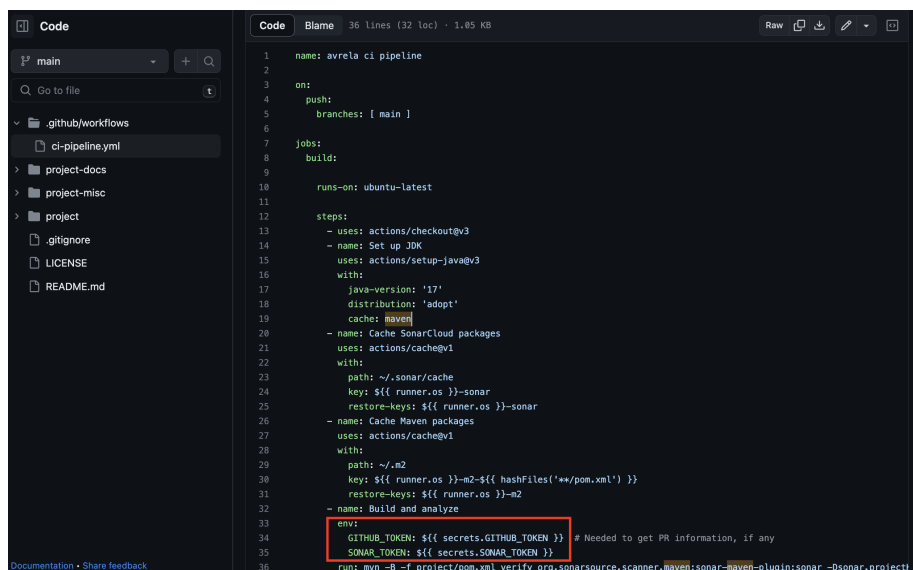


Figura D.2: Estado del proyecto en SonarCloud

Una vez dispongamos de nuestro propio proyecto SonarCloud, necesitaremos realizar las siguientes modificaciones en el fichero `.github/workflows/ci-pipeline.yml` de la Figura D.3.



```
1 name: avrela ci pipeline
2
3 on:
4   push:
5     branches: [ main ]
6
7 jobs:
8   build:
9
10    runs-on: ubuntu-latest
11
12    steps:
13      - uses: actions/checkout@v3
14      - name: Set up JDK
15        uses: actions/setup-java@v3
16        with:
17          java-version: '17'
18          distribution: 'adopt'
19      - name: Cache Maven
20        uses: actions/cache@v1
21      - name: Cache SonarCloud packages
22        uses: actions/cache@v1
23        with:
24          path: ~/.sonar/cache
25          key: ${{ runner.os }}-sonar
26          restore-keys: ${{ runner.os }}-sonar
27      - name: Cache Maven packages
28        uses: actions/cache@v1
29        with:
30          path: ~/.m2
31          key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
32          restore-keys: ${{ runner.os }}-m2
33      - name: Build and analyze
34        env:
35          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }} # Needed to get PR information, if any
36          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
37        run: mvn -B -f project/pom.xml verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar -Dsonar.projectKey=avrela
```

Figura D.3: Captura definición de la integración continua en GitHub. Las referencias a los tokens de GitHub y SonarCloud se recuadran en color rojo

- Configurar los tokens de acceso de acceso para GitHub y SonarCloud definidos por sus respectivas variables de entorno
- Referenciar el identificador de clave correcto en el parámetro `-Dsonar.projectKey` (actualmente `-Dsonar.projectKey=lfx1001_avrela`).

## D.4. Compilación, instalación y ejecución del proyecto

Los requisitos para poder compilar y ejecutar el proyecto son:

- Java 17 o superior.
- Maven 3.0 o superior.

Para ejecutar el proyecto se utiliza el comando maven siguiente desde el directorio raíz del proyecto:

```
$ mvn sprint-boot:run
```

Para la interacción con GitHub se utiliza la variable de entorno `GITHUB_TOKEN` para especificar el token a utilizar. De lo contrario la interacción con GitHub

estará limitada a las cuotas de uso para usuarios no autenticados. En un entorno Linux o Mac la variable puede definirse mediante el comando export:

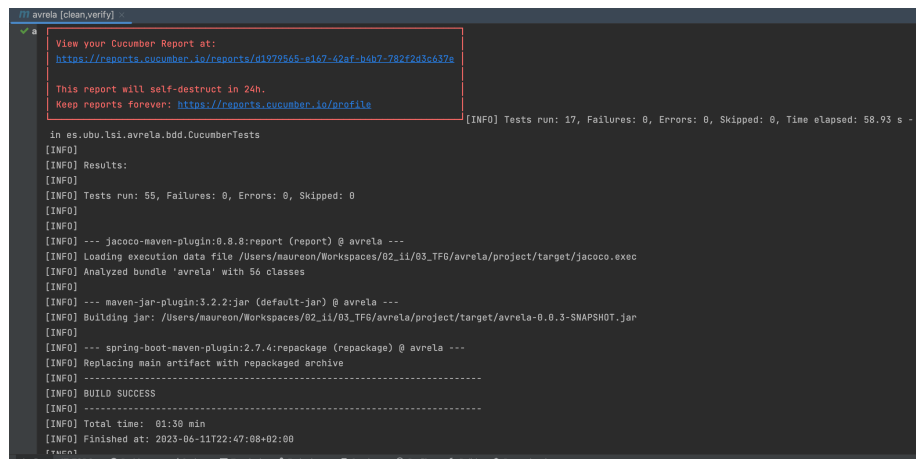
```
$ export GITHUB_TOKEN = <valor>
```

## D.5. Pruebas del sistema

Para ejecutar las pruebas del sistema se utiliza el comando maven siguiente desde el directorio raíz del proyecto:

```
$ mvn verify
```

Durante la ejecución se muestra por pantalla el resultado de los tests realizados con JUnit y Cucumber. Se muestra el resultado en la Figura D.4.



```
avrela [clean,verify]
View your Cucumber Report at:
https://reports.cucumber.io/reports/d1979565-e167-42af-b5b7-782f2d3c637a
This report will self-destruct in 24h.
Keep reports forever: https://reports.cucumber.io/profile

[INFO] Tests run: 17, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 58.93 s
in es.ubu.lsi.avrela.bdd.CucumberTests
[INFO] Results:
[INFO] Tests run: 55, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.8:report (report) @ avrela ---
[INFO] Loading execution data file /Users/maureon/Workspaces/02_11/03_TF6/avrela/project/target/jacoco.exec
[INFO] Analyzed bundle 'avrela' with 56 classes
[INFO]
[INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ avrela ---
[INFO] Building jar: /Users/maureon/Workspaces/02_11/03_TF6/avrela/project/target/avrela-0.0.3-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.7.4:repackage (repackage) @ avrela ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 01:30 min
[INFO] Finished at: 2023-06-11T22:47:08+02:00
[INFO]
```

Figura D.4: Resultados de la ejecución de los tests. En el recuadro rojo aparece una url temporal con el enlace específico al informe de tests realizados con Cucumber

En el caso de los tests realizados con Cucumber se genera una url accesible en internet donde estarán alojados los informes temporalmente. Se muestra en la Figura D.5 el resumen de la ejecución.

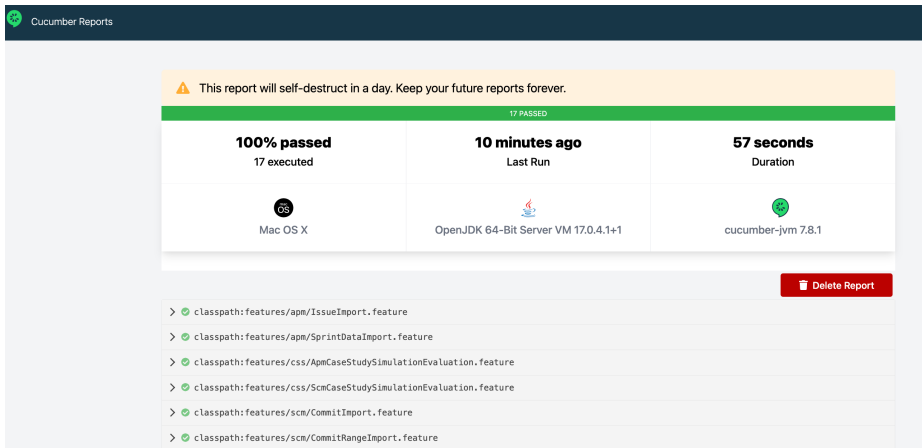


Figura D.5: Resumen de la ejecución de tests con Cucumber

También podemos visualizar el detalle de la ejecución de cada uno de los escenarios ejecutados por Cucumber. La Figura D.6 muestra el detalle de la ejecución del escenario de importación de una tarea.

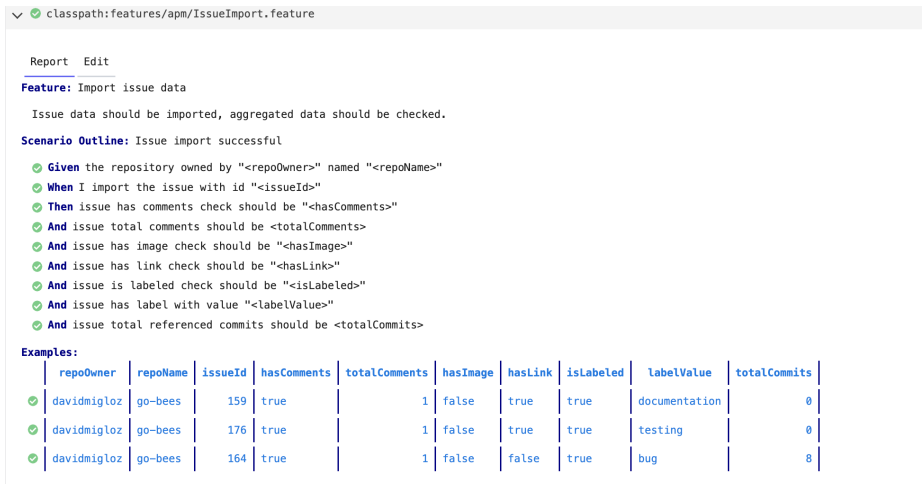


Figura D.6: Detalle de la ejecución del escenario de importación de una tarea





## *Apéndice E*

---

# **Documentación de usuario**

---

- E.1. Introducción
- E.2. Requisitos de usuarios
- E.3. Instalación
- E.4. Manual del usuario



---

## **Bibliografía**

---