



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**GII 20.09 Herramienta web
repositorios de TFGII
Documentación Técnica**



Presentado por Diana Bringas Ochoa
en Universidad de Burgos — 5 de julio
de 2021

Tutor: Álvaro Arnaiz González y Carlos López
Nozal

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	41
Apéndice B Especificación de Requisitos	43
B.1. Introducción	43
B.2. Objetivos generales	43
B.3. Catálogo de requisitos	44
B.4. Especificación de requisitos	45
Apéndice C Especificación de diseño	51
C.1. Introducción	51
C.2. Diseño de datos	51
C.3. Diseño procedimental	58
C.4. Diseño arquitectónico	59
C.5. Diseño gráfico	60
Apéndice D Documentación técnica de programación	63
D.1. Introducción	63
D.2. Estructura de directorios	63

D.3. Manual del programador	64
D.4. Compilación, instalación y ejecución del proyecto	70
Apéndice E Documentación de usuario	73
E.1. Introducción	73
E.2. Requisitos de usuarios	73
E.3. Instalación	73
E.4. Manual del usuario	73
Bibliografía	75

Índice de figuras

A.1. Gráfica Control chart- Sprint 0	2
A.2. Gráfica Control chart- Sprint 1	4
A.3. Gráfica Control chart- Sprint 2	6
A.4. Gráfica Control chart- Sprint 3	8
A.5. Gráfica Control chart- Sprint 4	10
A.6. Gráfica Control chart- Sprint 5	12
A.7. Gráfica Control chart- Sprint 6	14
A.8. Gráfica Burndown- Sprint 6	15
A.9. Gráfica Control chart- Sprint 7	19
A.10.Gráfica Burndown- Sprint 7	20
A.11.Gráfica Control chart- Sprint 8	22
A.12.Gráfica Burndown- Sprint 8	23
A.13.Gráfica Control chart- Sprint 9	25
A.14.Gráfica Burndown- Sprint 9	26
A.15.Gráfica Control chart- Sprint 10	28
A.16.Gráfica Burndown- Sprint 10	29
A.17.Gráfica Control chart- Sprint 11	31
A.18.Gráfica Burndown- Sprint 11	32
A.19.Gráfica Control chart- Sprint 12	36
A.20.Gráfica Burndown- Sprint 12	37
A.21.Gráfica Control chart- Sprint 13	39
A.22.Gráfica Burndown- Sprint 13	40
 B.1. Diagrama de casos de uso	 47
 C.1. Diagrama de clase - Persistencia	 53
C.2. Diagrama de clase - Vistas	54
C.3. Diagrama de clase - Componentes	54

C.4. Diagrama de clase - Util	55
C.5. Diagrama de clase - Entidades	56
C.6. Diagrama de clase - Autenticación con moodle	57
C.7. Diagrama de clase -Servicios web	57
C.8. Diagrama de clase - Tests	58
C.9. Diagrama uml de la estructura del Singleton	59
C.10. Diagrama uml de la estructura de la Fachada	60
C.11. Login de la aplicación	61
C.12. Gráfica de ejemplo de la aplicación	62
D.1. Descarga de JDK 11	65
D.2. Descarga JDK 11 Licencia	66
D.3. Descargar IDE Eclipse	67
D.4. Seleccionar Eclipse	68
D.5. Seleccionar JDK que usará el IDE	68
D.6. Eclipse marketplace	69
D.7. Plugin Vaadin	70
D.8. Copiar URL repositorio	71

Índice de tablas

A.1. Dependencias del proyecto	41
B.1. Actores de la aplicación	46
B.2. Caso de uso 1: Autenticar usuarios.	48
B.3. Caso de uso 2: Actualizar ficheros xls.	49
B.4. Caso de uso 3: Visualizar los rankings sobre las notas de los TFG.	49

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En esta sección se detallará la planificación que se ha realizado, el estudio de viabilidad tanto de la parte económica como de la legal.

A.2. Planificación temporal

Sprint 0 - Puesta a punto (26/10/20 - 25/11/20)

Puesta a punto del proyecto, planteamiento de las herramientas con las que trabajar, búsqueda de alternativas y toma de contacto con las herramientas nuevas que se van a emplear. Las tareas que se realizaron fueron:

- Añadir la extensión ZenHub al navegador Desde el **Chrome Web Store** de Google Chrome se añadió la extensión **ZenHub for GitHub**.
- Clonar el repositorio en local. Mediante la aplicación **GitHub Desktop**, se clonó el repositorio del Gestor de TFG mediante el enlace HTTP que proporciona GitHub.
- Investigar sobre Vaadin. A través de la página oficial de **Vaadin** se realizó la instalación y el aprendizaje acerca de Vaadin.
- Actualización del README.md Se modificó el README.md del proyecto para que refleje los cambios respecto a la versión anterior.
- Investigar LaTeX Se procedió a buscar información sobre cómo realizar el proceso de instalación y manejar **Latex** para realizar la documentación del proyecto posteriormente.

Se puede ver el trascurso de estas tareas gráficamente en la siguiente ilustración A.1.

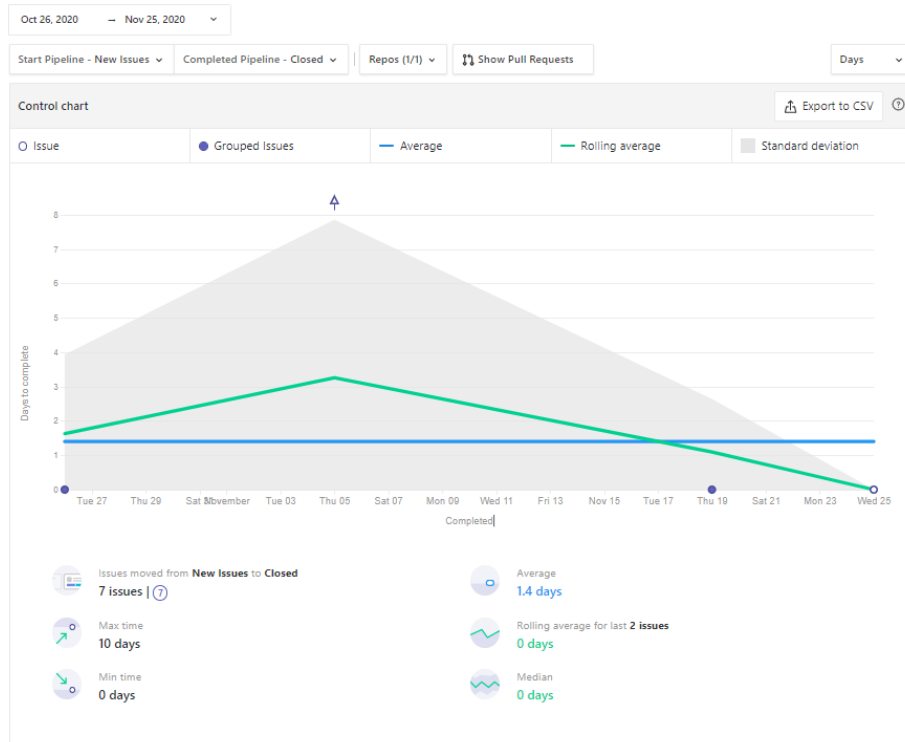


Figura A.1: Gráfica Control chart- Sprint 0

Sprint 1 - Toma de contacto y comienzo de documentación (25/11/20 - 12/01/21)

Generación de test unitarios, búsqueda de trabajos similares, cambio del driver para conectarse con el excel, información para obtener ideas de como realizar ciertas mejoras y comienzo de la documentación del proyecto. Mejora de la cobertura de la aplicación web.

Las tareas planteadas fueron:

- Instalación Miktex + TexStudio Tras la reunión del 25/11/20, se decidió que la mejor opción sería emplear un editor local, en lugar de usar **Overleaf**, un editor online. Para ello, se instaló la distribución de TeX/LaTeX, **Miktex**, junto al editor de texto llamado **TexStudio**.

- Se comienza la documentación en LaTeX - Sprint 0 Creación de la memoria y los anexos en LaTeX a partir de las plantillas.
- Generar nuevos test Para aumentar la cobertura de la aplicación se definieron nuevos test.
- Creación carpeta pruebas Se creó la carpeta pruebas, dentro del apartado de documentación, donde se irán realizando pruebas antes de introducirlas en la aplicación. Dentro de esta carpeta se encontrarán aplicaciones prototipo, gracias a las cuales se logro una mejor comprensión del funcionamiento de Vaadin.
- Búsqueda de trabajos relacionados con la gestión de TFG/TFM Se realizó una investigación con el fin de encontrar proyectos similares a la aplicación web, es decir, que consistan en la gestión de trabajos de fin de grado o similares. Los proyectos encontrados serán explicados en el apartado **Trabajos relacionados** de la memoria.
- Realización de cambios sugeridos en la memoria y los anexos Se cambiaron algunos aspectos de la memoria y los anexos comentados en la reunión realizada el 15/12/20. Como la incorporación de diagramas de Zenhub para visualizar la evolución de las issues.
- Cambiar driver JDBC Para poder usar ficheros en formato .ods se buscó una alternativa similar al driver para .csv que se emplea. Primero se optó por la opción de **Microsoft Excel JDBC Driver** con el cual se puede leer, escribir y actualizar Excel mediante JDBC. Sin embargo, esta opción es de pago, por lo que fue descartada. Los drivers gratuitos para .ods que se encontraron fueron **ODFDOM** y **JopenDocument** los cuales eran bastante viejos y en desuso.

Se puede ver el trascurso a través de la siguiente imagen [A.2](#).

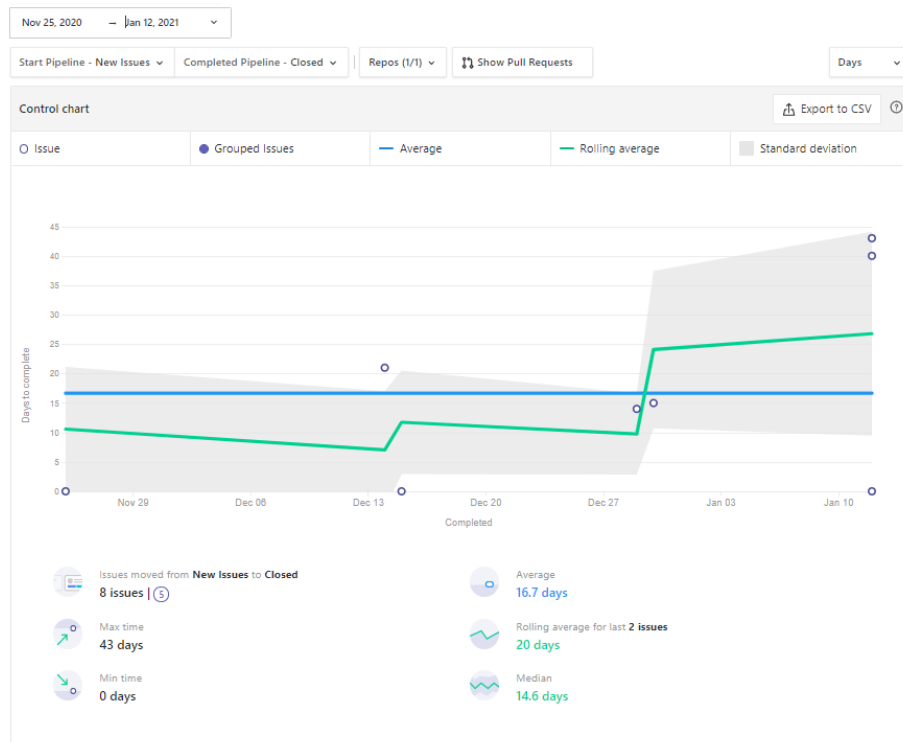


Figura A.2: Gráfica Control chart- Sprint 1

Sprint 2 - Incorporación de la API Fillo (12/01/21 - 26/01/21)

Búsqueda de nuevos drivers que implementen JDBC para ficheros .xls. Verificar el funcionamiento de las opciones encontradas para leer los .xls como [Apache POI](#), [SQLSheet](#), [API Fillo](#), [Cdata JDBC for Excel](#) y [JdbcOdbcDriver](#). Integrar la API Fillo en el proyecto.

Las tareas planteadas fueron:

- Instalación del LibreOffice Se instaló [Apache LibreOffice](#) para manipular los fichero .xls y .csv que se emplean para la parte de datos de la aplicación.
- Cambiar de formato al fichero “**BaseDeDatosTFG.ods**”. Se modificó el formato del fichero “**BaseDeDatosTFG.ods**” a “**BaseDeDatosTFG.xls**”.
- Anexos - Modificación para poder cambiar el tamaño de las imágenes. Cambios en la plantilla de **anexos.tex** para poder modificar el tamaño de las imágenes al deseado.

- Probar el Apache POI Para verificar el funcionamiento de Apache POI, se incluyó en el proyecto de prueba **HolaMundoVaadin** y se realizaron varios ejemplos de prueba tomando como referente el proyecto principal.
- Error al intentar ejecutar la imagen del proyecto gabrielstan/gestion-tfg Bug realizado al intentar desplegar el proyecto gabrielstan/gestion-tfg.
- Desplegar proyectos relacionados con la gestión de TFG Proceso por el cual se probó a desplegar los proyectos relacionados con **GII 20.09 Herramienta web repositorios de TFGII**. En el apartado de la memoria Trabajos relacionados se detallará sobre este tema.
- Memoria - Mejoras Se realizaron varios cambios en la memoria.
- Incorporación del driver para fichero Excel Prueba de la **API Fillo** en el proyecto de prueba **HolaMundoVaadin**, y tras verificar su funcionamiento se procedió a incluirlo en el proyecto principal.
- Modificación de los nombres de los fichero csv Se cambiaron los nombres de los ficheros .csv para que coincidieran con los nombres de las hojas del fichero .xls.
- Memoria - Documentación Sprint 2 Comienzo de la documentación de lo realizado en el Sprint 2. En el cual se detallan las tareas realizadas.
- Modificaciones en los test Para probar la incorporación de la **API Fillo** se modificaron el test “**SintInfDataTest**”.
- Realización de cambios para el funcionamiento de la nueva conexión para los fichero .xls Para poder usar la **API Fillo** se han tenido que realizar multitud de cambios en el proyecto. Aunque tanto el “**CsvDriver**” y “**Fillo**” emplean lenguaje SQL, “**CsvDriver**” emplea JDBC puro mientras que “**Fillo**” usa funciones y clases propias.

La siguiente imagen **A.3** muestra cómo se han desarrollado las tareas a lo largo del tiempo.

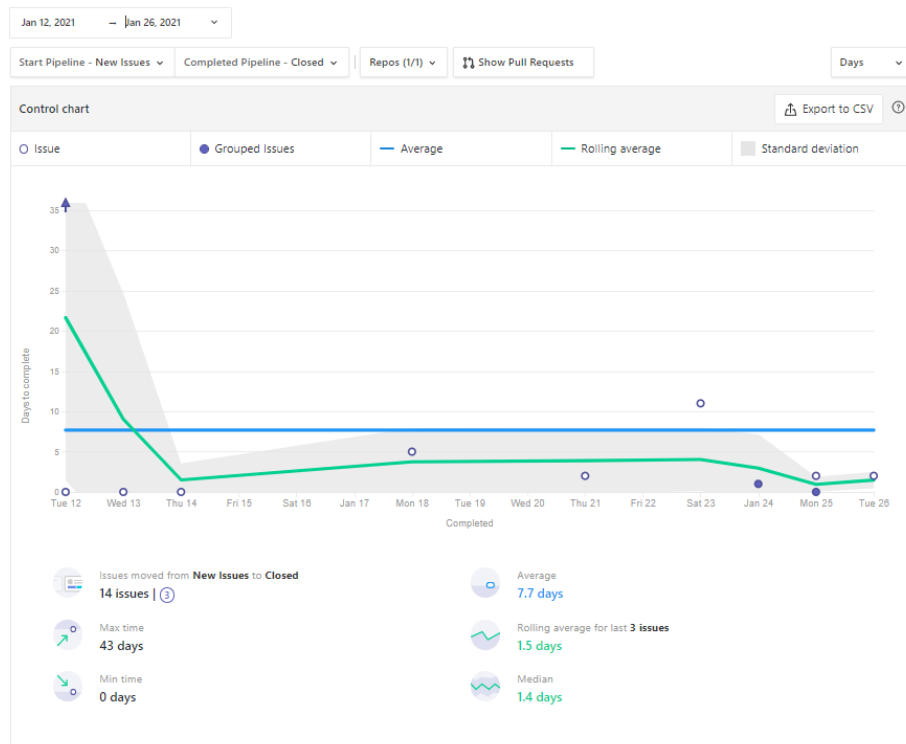


Figura A.3: Gráfica Control chart- Sprint 2

Sprint 3 - Rediseño e integración API Fillo (26/01/21 - 11/02/21)

Las tareas planteadas fueron:

- Investigar opciones de hosting para el despliegue Se comparan varias opciones gratuitas para desplegar la aplicación, las cuales son: GitHub Pages, LucusHost, Awardspace, RunHosting, FreeHostia, X10Hosting y Heroku. Eligiendo la opción de **Heroku**, entre otras razones, porque proporciona la opción de conectar el despliegue con GitHub. En la **issue de Github**, correspondiente a esta tarea, se encuentra detallado cada opción de despliegue.
- Rediseño fachada y vistas Se elimina la parte vinculada a los datos de las vistas incluyéndola en a las clases fachada. Consiguiendo que unicamente las clases fachadas comuniquen directamente con la capa de datos, y el resto de la aplicación emplee de intermediario a las clases fachada.

- Cambiar nombres a inglés. Se traducen los nombres de las variables, métodos y clases a inglés.
- Actualización de la Memoria y Anexos. Se realizan cambios y ampliaciones en el contenido de la documentación, en la Memoria y el Anexo.
- Instalación Heroku CLI Para realizar el despliegue del proyecto se instala el terminal de Heroku, denominado Heroku CLI. La instalación se llevo a cabo según el tutorial de instalación de la página oficial de [Heroku](#).
- Incorporación del patrón Factory Se crea una nueva clase con la función de seleccionar el tipo de acceso de datos, ya sea la clase fachada encargada de los ficheros .csv o, la que gestiona los ficheros .xls.
- Creación branch de prueba Rama del repositorio donde se almacena el proyecto de prueba formularioVaadin, el cual posteriormente será usado para realizar una prueba de despliegue en Heroku.
- Modificación de Test JUnit Se cambian los test para que sirvan para las funciones de la clase fachada para los ficheros .xls.
- Probar el despliegue del proyecto de prueba Se emplea el proyecto de prueba, formularioVaddin, para realizar una prueba de despliegue en Heroku. Se intentará realizarlo tanto, a través de Heroku CLI como, mediante la interfaz la página de Heroku.

En la siguiente imagen se enseña gráficamente el desarrollo de las issues [A.4](#).

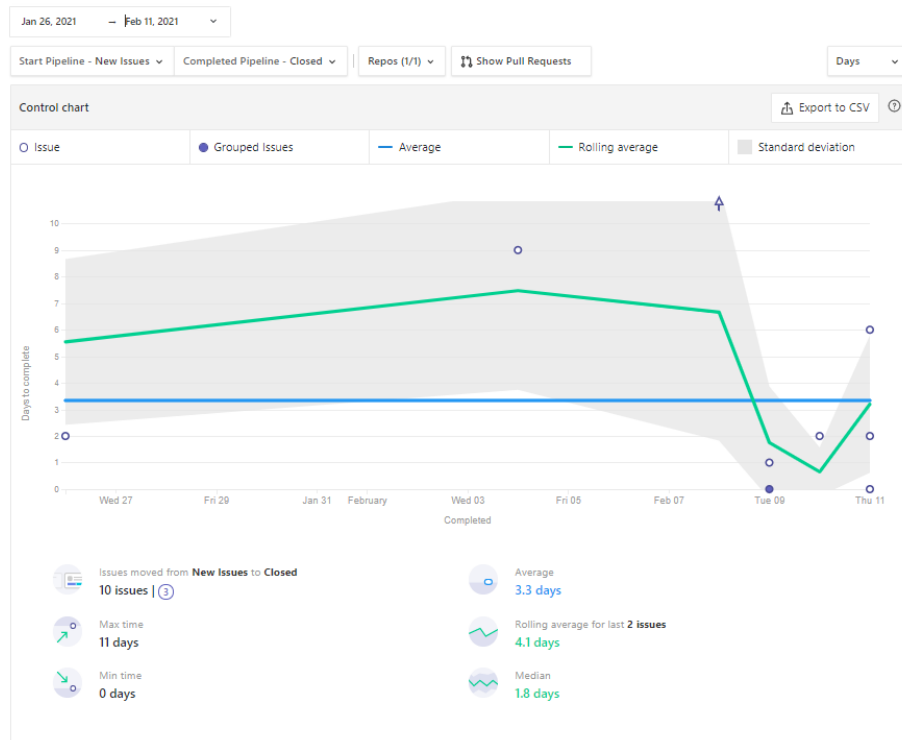


Figura A.4: Gráfica Control chart- Sprint 3

Sprint 4 - Integración Continua (11/02/2021 - 23/02/2021)

Se intentará incorporar la integración continua. Realizar el despliegue de la aplicación con Heroku. Añadir una columna de rankings en el Histórico. Cambiar las notas del histórico a privadas.

Las tareas planteadas fueron:

- Realizar la Integración Continua. A través de la opción **GitHub Actions** se incorporará la opción de compilar y ejecutar los test cuando se realiza un cambio (ya sea un push o un pull request) en el repositorio. Tiene como objetivo detectar fallos eficazmente mediante la integración automática frecuentemente de un proyecto. Se explica cómo se realizó en la [Github](#).
- Actualización bibliografía Se incorporan los enlaces de las páginas o datos bibliográficos que se han empleado hasta ahora en la realización del proyecto.

- Cambiar las notas del Histórico a privadas Se cambiará la nota que figura en la tabla "Descripción proyectos" del Histórico para que no se muestre (como privada).
- Añadir dependencia Apache Commons Math Se empleará Apache Commons Math para calcular los percentiles de los ranking de las notas.
- Crear ranking de notas en Histórico Para sustituir la columna de notas se integrarán tres nuevas columnas con rankings. Se trata de un ranking por curso, un ranking sobre todos los TFG (total) y un ranking de percentiles sobre el total de 5 niveles A,B,C,D,E.
- Actualización del despliegue del proyecto de prueba en Heroku Se desplegó el proyecto principal, sistinf, en **Heroku**. Se podrá encontrar cómo realizar este proceso tanto en la [issue en Github](#) como en el apartado del **Manual del programador**.
- Mejora de test JUnit Se modificarán los tests existentes para aumentar la zona verificada por los tests.

Se puede apreciar gráficamente el desarrollo de las issues en la siguiente ilustración [A.5](#).

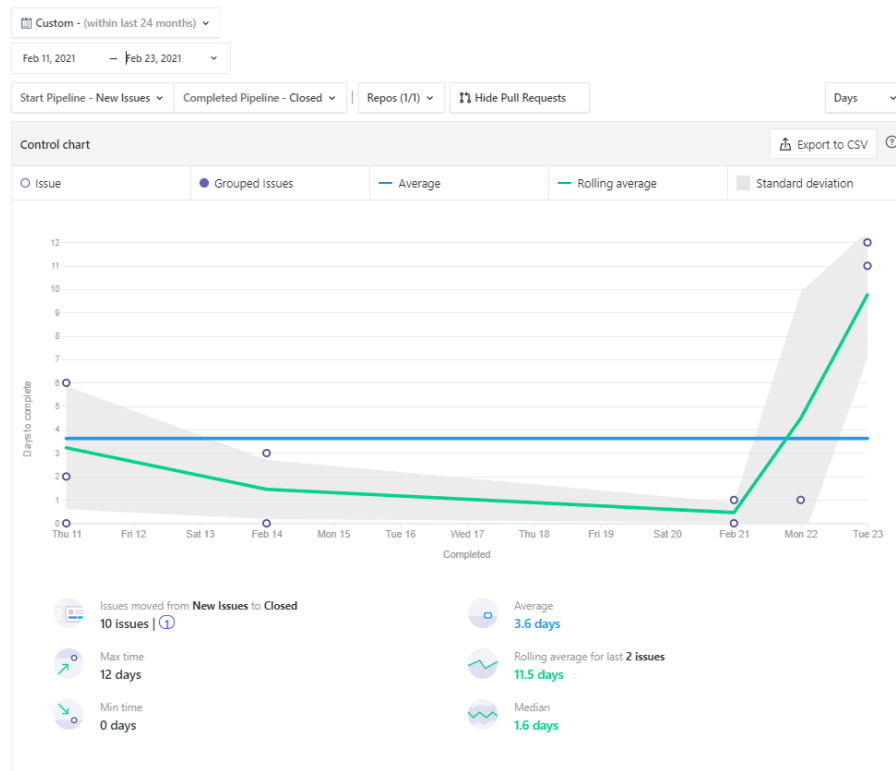


Figura A.5: Gráfica Control chart- Sprint 4

Sprint 5 - Investigación de la plataforma SonarCloud y creación de rankings (23/02/21 - 09/03/21)

Añadir de nuevas columnas en la tabla de Históricos para reemplazar la columna Notas, la cual será eliminada. Crear test para comprobar los datos de los ficheros XLS. Investigación y prueba de análisis de la calidad del código con **SonarCloud**.

Las tareas realizadas fueron:

- Eliminación de la columna Nota del Histórico Debido al carácter sensible de las notas, se eliminarán de la tabla de la vista del Histórico.
- Crear columna del ranking total en el Histórico. Se creó una nueva columna en la tabla de la vista del Histórico para representar las notas en comparación al resto de ellas.
- Investigar sobre **SonarCloud**. Para realizar las métricas que posteriormente irán en el apartado de Métricas, se analizará la calidad del código de algunos de los proyectos realizados anteriormente. Para

ello, se investigará como usar el software **SonarCloud** para realizar las mediciones de calidad. Consta de dos versiones, una de pago y otra gratuita, siendo esta última la que se empleará. Al ser una versión gratuita los proyectos serán públicos.

- Crear columna del ranking por cursos en el histórico. Al igual que con el caso del ranking total, se añadirá una nueva columna en la tabla de Históricos que contenga el ranking de una nota con respecto al resto de notas de ese mismo curso escolar (1 de septiembre a 30 de junio).
- Probar a desplegar el proyecto en Linux. Al estar el proyecto desplegado en **Heroku** se podrá acceder a él a través de la url, donde se encuentra el proyecto, desde cualquier SO.
- Parsear ficheros CSV. Se añadirán nuevos test que verifiquen que no existan errores de formato, por ejemplo que el formato de las fechas sea el indicado
- Separación de la clase SintInfDataTest en **SintInfDataTestXLS** y **SintInfDataTestCSV**. Al añadir una nueva clase fachada para el tipo de datos XLS se necesita otra clase para testarla, por lo que se dividirá la clase SintInfDataTest en dos clases, SintInfDataTestXLS, encargada de verificar los archivos y funciones correspondientes a los datos en XLS, y SintInfDataTestCSV, la cual testará los ficheros CSV.
- Añadir botón quality gate SonarCloud. Se incorporará en el README.md del proyecto un acceso directo a la página donde figurarán los análisis de los proyectos en SonarCloud.
- Añadir botón despliegue Heroku. Al igual que con SonarCloud, se incluirá un acceso a la página donde se encuentra desplegado el proyecto en Heroku. Para realizarlo se siguió los pasos de la documentación de **Heroku**.
- Analizar el proyecto sistinf con SonarCloud. Se realizará el análisis automático del proyecto principal, en el cual no se incluye el análisis del código en Java ya que este lenguaje no está soportado en **SonarCloud**.
- Incluir el análisis automático del proyecto en Sonarcloud. Se seguirá el siguiente tutorial de la página de **SonarCloud** para que se realice el análisis cada vez que se realice un push.
- Anexos actualización – B-Requisitos. Se añadirá el apartado de requisitos a la documentación de LaTeX
- Anexos actualización - A-Plan-Proyecto. Modificación de los Anexos con el apartado A-Plan-Proyecto.

En la siguiente gráfica se puede ver el desarrollo de las tareas del Sprint **A.6**.

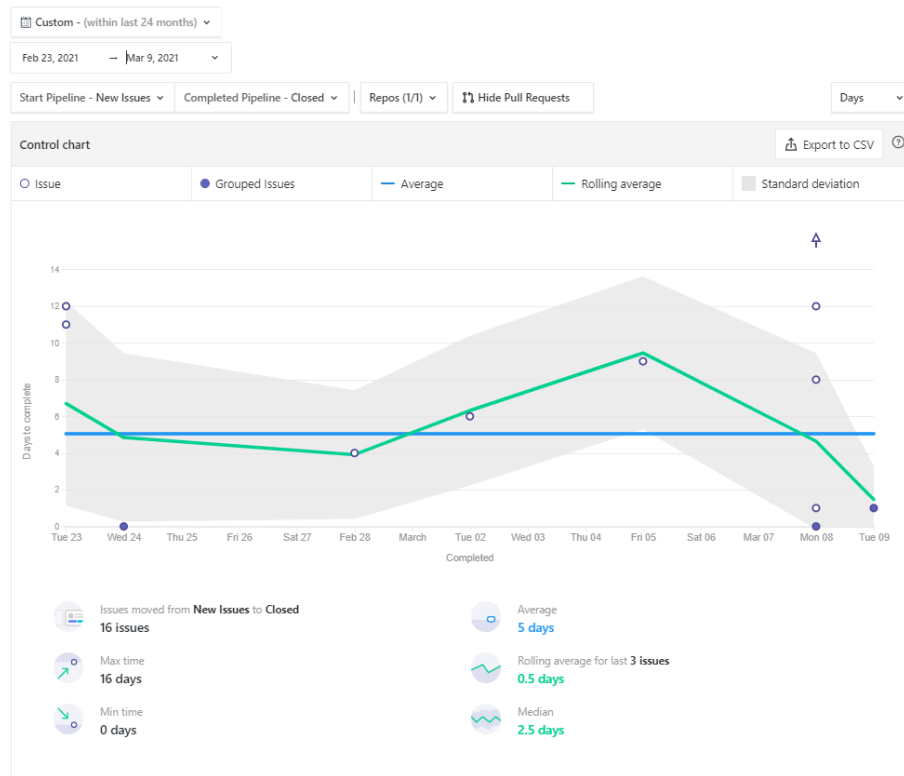


Figura A.6: Gráfica Control chart- Sprint 5

Sprint 6 - Integración y actualización de ficheros (09/03/21 - 23/03/21)

Análisis del proyecto poolobject del código en Java en SonarCloud. Se investigó y probó a realizar el login a través de distintos métodos.

Las tareas realizadas fueron:

- Análisis del proyecto de prueba poolobject con SonarCloud. Se trata de un proyecto en Java al igual que el del Gestor-TFG-2021. Para ello, se realizará un fork del proyecto en Github y se procederá a realizar el análisis especificando la ubicación del código fuente en Java y los archivos binarios. Se debe realizar de esta forma debido a que en el análisis automático de **SonarCloud** no se incluye Java.
- Pruebas de Login a través de Heroku. Se probó a acceder a la parte de actualización de ficheros a través del login y subir un fichero, pero, no se modificaron los datos de las vistas ni la fecha de actualización.

- Prueba Login con la app desplegada con Tomcat. Al ejecutar la app manualmente desde el IDE Eclipse empleando como herramienta para desplegar el proyecto Tomcat, se modificó la fecha de actualización de subida de los ficheros pero no se actualizó los datos de las vistas.
- Solucionar problema en la actualización de los ficheros csv. Examinar y arreglar la causa por la cual los ficheros no se estaban actualizando. Para ello, se llevo a cabo varias modificaciones en las clases fachada de los datos.
- Cambiar configuración SonarCloud. Se excluirá los ficheros que no se deseen examinar en el análisis de la calidad del código como los ficheros propios de Vaadin, los ficheros CSS, entre otros. Anteriormente el análisis que se había realizado sobre **SonarCloud** no estaba teniendo en cuenta los códigos en Java, esto es debido a que el análisis automático de SonarCloud no es compatible con Java. Se deberá especificar donde se encuentra el código fuente que se desea examinar y los archivos binarios de Java.
- Investigar cómo realizar el Login a través de UBUVirtual. Uno de los requisitos es realizar un Login que permita autenticarse con el correo de la UBU o similares, por lo que, se realizará una búsqueda sobre cómo realizar esta conexión. Finalmente la opción que se escogió fue **Firestore**, un servicio de backend que dispone de SDKs fáciles de usar y bibliotecas de IU ya elaboradas.
- Investigar cómo importar los ficheros CSV y XLS a Heroku. Se puede subir los ficheros mediante Amazon S3 (o otro almacenamiento en la nube que se pueda conectar con Heroku) y, a través de Skyvia importar los datos (volcarlos) en la base de datos (Heroku Postgresql). O simplemente subir los ficheros en el .war y cuando se actualicen modificarlos, esta será la opción que se empleará.
- Prueba - Login con Microsoft. Se probará el código de ejemplo para iniciar sesión mediante Microsoft en aplicaciones web en Java. Siguiendo el tutorial de la página de **Microsoft**. Se registró la app en **Azure** y se siguió el tutorial de ejemplo **Azure**. No se consiguió realizar el login ya que se necesitan permisos que no dispongo.
- Añadir más extensiones a gitignore. Para evitar que se suban ficheros no deseados se incluyeron más extensiones a ignorar en el fichero gitignore.

Se puede ver el transcurso de las tareas del Sprint en las siguientes gráficas **A.7**.

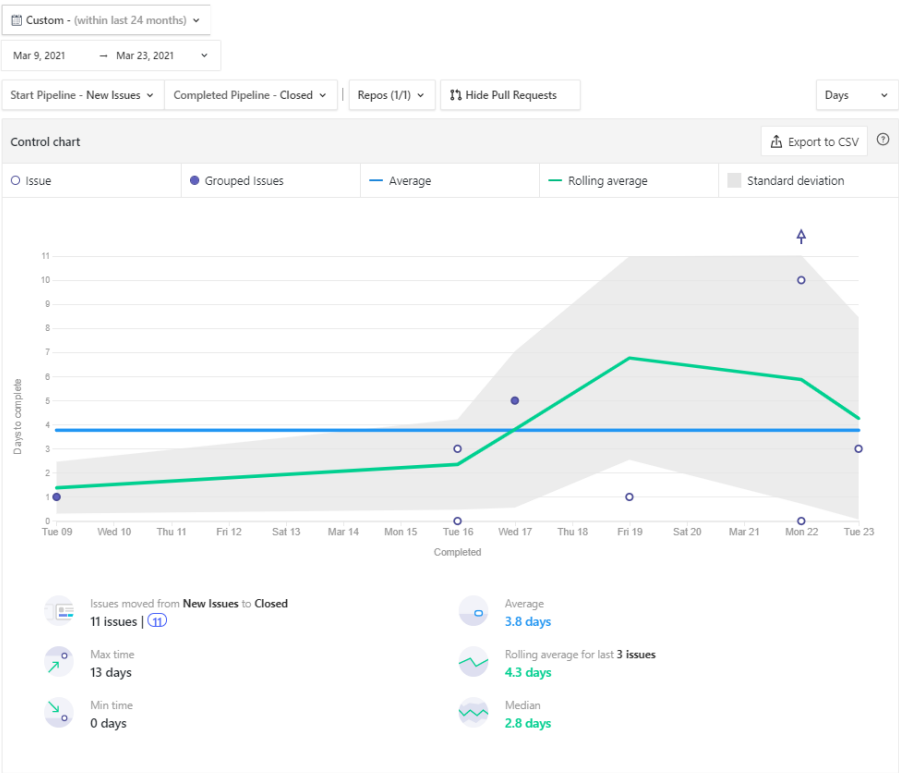


Figura A.7: Gráfica Control chart- Sprint 6

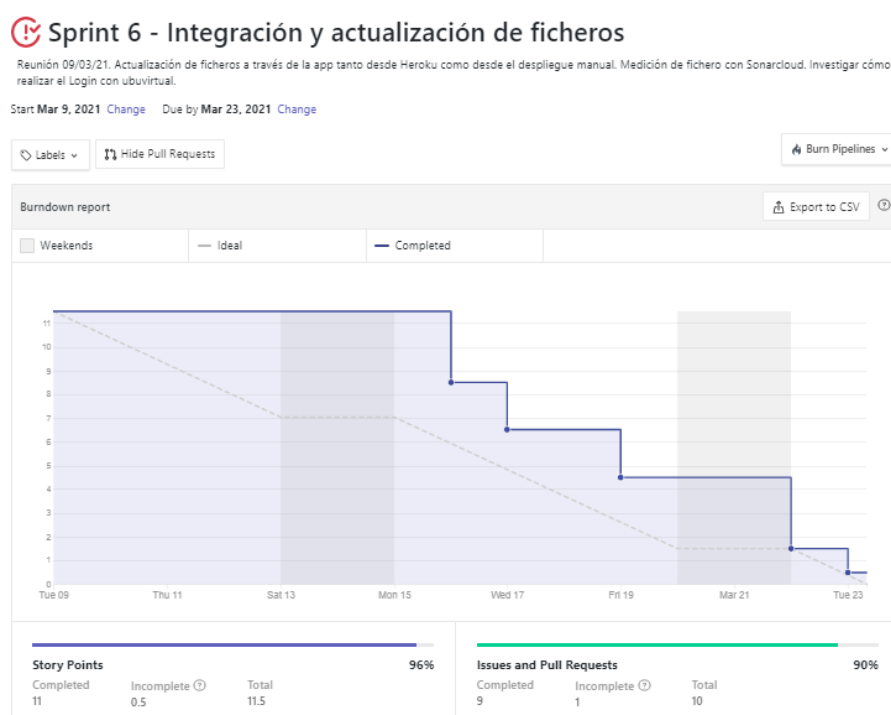


Figura A.8: Gráfica Burndown- Sprint 6

Sprint 7 - Análisis de proyectos con SonarCloud (23/03/21 - 13/04/21)

Análisis de la calidad del código de los proyectos presentados en 2020 con SonarCloud. Investigar cómo realizar el login con Firebase. Búsqueda y comienzo de proceso de migración de versión de Vaadin.

Las tareas realizadas fueron:

- Forks de todos los proyectos presentados en 2020. Se realizará un fork de todos los proyectos que se desean analizar y se añadirá al principio del nombre de los trabajos "ÜBU-TFG" para identificarlos.
- Instalar SonarCloud CLI. Se realizará la instalación de SonarCloud en local siguiendo la documentación de [SonarCloud](#)
- SonarCloud - Analizar proyecto Gestión Aulas Informática Al ser un proyecto **Maven** en Java se realizará el análisis empleando el fichero pom.xml como se especifica en [Github](#).
- SonarCloud - Analizar proyecto UBUMonitor Clustering Este proyecto no se pudo analizar, porque aparecía un error con respecto a la versión de java que empleaba. El error se puede apreciar en la [issue de Github](#) vinculada a esta tarea.
- SonarCloud - Analizar proyecto Medidor estadístico metajuego Magic The Gathering Al ser un proyecto en Java se tuvo que realizar el análisis manual cómo se explica en las [tarea del proyecto de Github](#).
- SonarCloud - Analizar proyecto TourPlanner-FrontEnd-Cliente En [Github](#) se puede ver el proceso que se llevo a cabo para realizar el análisis.
- SonarCloud - Analizar proyecto UBUVoiceAssistant. Se puede ver en detalle cómo se realizó el análisis del proyecto en la tarea de [Github](#) asociada.
- SonarCloud - Analizar proyecto LogScope. En la issue de [Github](#) se puede ver los pasos que se realizaron para analizar el trabajo.
- SonarCloud - Analizar proyecto PruebaNetExtractor. Al ser un proyecto en Python se empleo el análisis automático. En la [issue de Github](#) se explica cómo realizar el análisis automático.
- SonarCloud - Analizar proyecto Reserva aulas informática Se realizó el análisis automático como se indica en la tarea de [Github](#).
- SonarCloud - Analizar proyecto MetrominutoWeb. Al igual que en los anteriores casos, se realizó el análisis automático de la calidad del código. En [Github](#) se puede ver el proceso que se llevo a cabo.

- SonarCloud - Analizar proyecto Sentinel. Como el proyecto emplea lenguajes compatibles con el análisis automático de SonarCloud, no se requerirá de modificaciones en el repositorio. Se puede obtener más información en [Github](#).
- SonarCloud - Analizar proyecto CENIEH and Ariadne. En [Github](#) se puede ver el proceso que se llevo a cabo para realizar el análisis.
- SonarCloud - Analizar proyecto Plataforma de text mining sobre repositorios. Se realizan el análisis automático como se especifica en la [tarea de Github correspondiente](#).
- SonarCloud - Analizar proyecto UBUEstelas. En este proyecto se empleo Gradle para el análisis del código, en la siguiente [tarea de Github](#) se detalla cómo se realizó.
- Investigar cómo migrar de versión de Vaadin. En [Github](#) se puede ver el proceso que se llevo a cabo para realizar el análisis.
- SonarCloud - Analizar proyecto XRayDetector. Al ser un proyecto en Python bastó con realizar el análisis automático de SonarCloud. Se puede observar cómo se realizó en [Github](#).
- SonarCloud - Analizar proyecto Jellyfish Forecast. Al igual que el caso anterior, es un proyecto en Python. En la [tarea de Github](#) asociada se puede ver cómo se crea el análisis.
- SonarCloud - Analizar proyecto Análisis Comercial Urbano En la [issue de Github](#) relacionada con este apartado se muestra el resultado del análisis de la calidad del código con SonarCloud.
- SonarCloud - Analizar proyecto Iris classifier. En [Github](#) se explica los pasos seguidos para hacer el análisis.
- SonarCloud - Analizar proyecto Asistente de programación C. Esta tarea se encuentra explicada en la [issue de Github](#) se puede ver el proceso que se llevo a cabo para realizar el análisis.
- SonarCloud - Analizar proyecto Flutter Serpiente. Este proyecto fue eliminado del análisis de SonarCloud porque solo se consiguió que se analizará el lenguaje Kotlin y no el lenguaje principal, **Dart**. En la [tarea de Github](#) se proporcionan enlaces de fuentes donde se da información acerca de este lenguaje y cómo analizarlo.
- Instalación NodeJS. Se modificó el pom.xml según el [manual de migración con mpr a Vaadin 14](#). Según la documentación de Vaadin 14 se requiere tener instalado en el ordenador [NodeJs](#). En la [issue de Github](#) se ilustra un problema que surgió referente a las dependencias.
- SonarCloud - Analizar proyecto Blockchain en una cadena de distribución de productos. Al igual que alguno de los proyectos anteriores, se realizó un análisis automático del proyecto. El resultado se puede ver en la [tarea de Github](#) asociada a esta tarea.

- Crear proyecto en Firebase. Para crear el proyecto de Firebase se creó una cuenta en el mismo y se procedió a añadir el proyecto desde la [consola de Firebase](#) siguiendo los pasos ue se ilustran en la [issue de Github](#) asociada a esta tarea.
- Migración de versión de Vaadin Se decidió migrar la versión de Vaadin a la 14, debido a que la versión que se estaba empleando, Vaadin 7, ya tenía soporte y no era compatible con Java 11. Para realizar a migración se intentó incorporar **MPR** lo cual permite ejecutar la aplicación original en Vaadin 7 dentro de **Vaadin 14**. Se siguió la [documentación de MPR en Vaadin](#) y se tomó como ejemplo el siguiente [repositorio de demostración](#).
- SonarCloud - Analizar proyecto Estudio de herramientas para realidad aumentada. No se consiguió analizar el lenguaje principal del proyecto, por lo que fue descartado.
- SonarCloud - Analizar proyecto ARBUBU. Al intentar analizar el proyecto daba un error que se puede ver en la ilustración compartida en la [issue](#) de Github. Este proyecto se dejará pendiente de revisión.
- SonarCloud - Analizar proyecto Free Connect. No se consigue analizar el lenguaje principal del proyecto por lo que queda pendiente de eliminación o descarte.
- Memoria - Objetivos del proyecto Se desarrolla el apartado de Objetivos del proyecto de la memoria.
- Anexo - Especificación de Requisitos Se comienza a documentar en la parte de la Especificación de Requisitos de los anexos.

Se puede apreciar el progreso de las tareas del Sprint 7 en las siguientes gráficas [A.9](#).

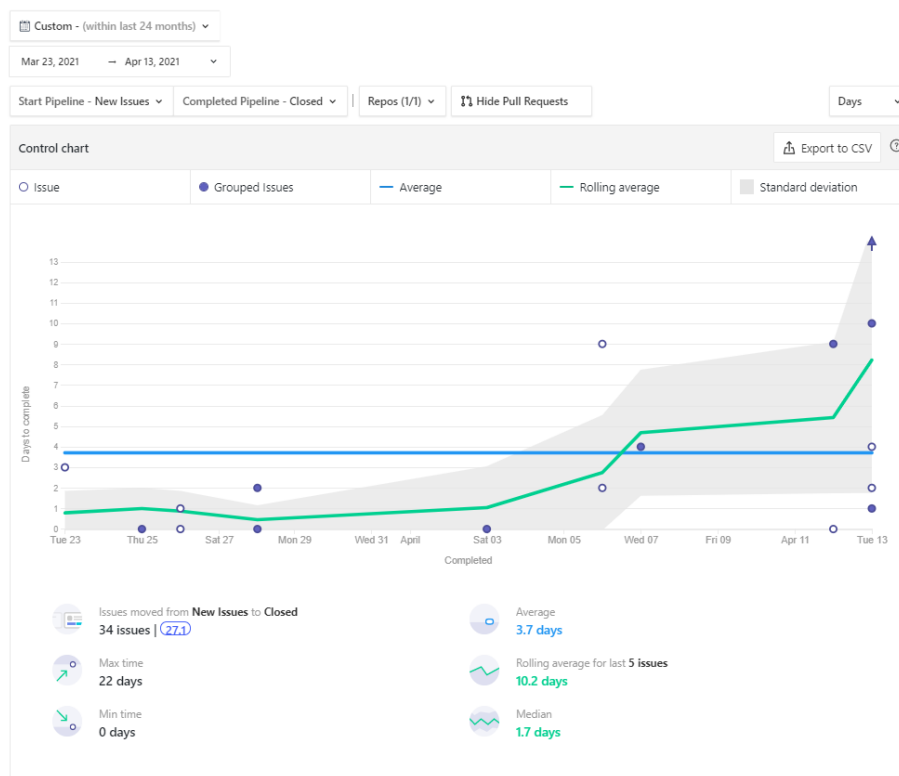


Figura A.9: Gráfica Control chart- Sprint 7

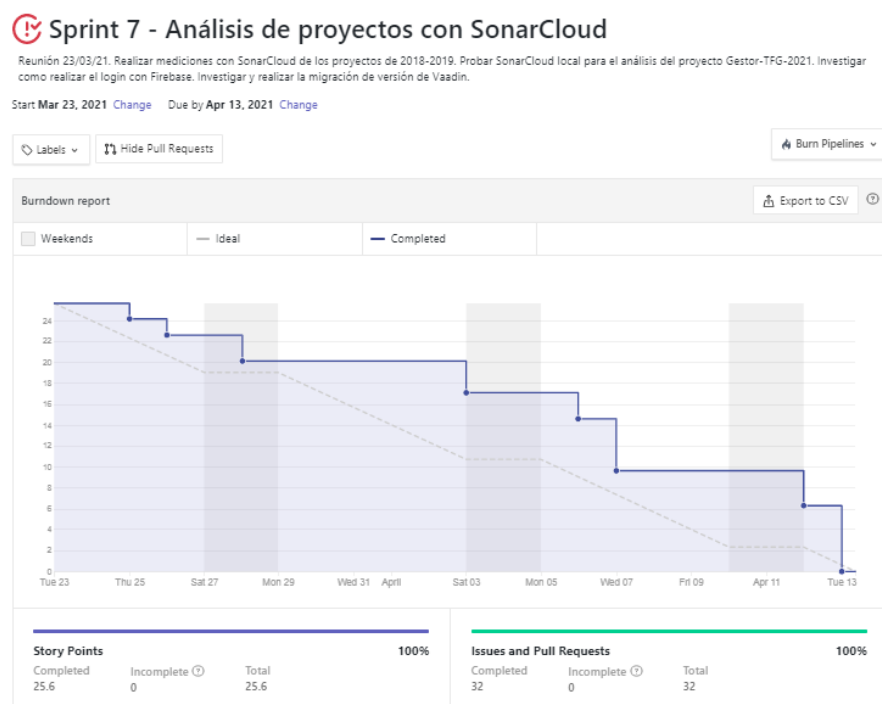


Figura A.10: Gráfica Burndown- Sprint 7

Sprint 8 - Migración a Vaadin 14 y Release 0.7 (13/04/21 - 04/05/21)

Continuación del proceso de migración el proyecto a Vaadin 14. Creación de la primera release (versión 0.6) con la app incluyendo la posibilidad de subir tanto .csv, como .xls, y su correspondiente actualización de las vistas.

Las tareas realizadas fueron:

- Revisión de memoria y anexos. Realización de correcciones recomendadas por el tutor de la documentación en LaTeX.
- Actualización subida de ficheros. Se llevaron a cabo varias modificaciones en el código para que se realizará correctamente la actualización de las vistas con los nuevos datos, tanto xls como csv.
- Incorporación Firebase para el Login. Se siguió con la integración del login con **Firebase**.
- Migración a Vaadin 14. Continuación del proceso de migración del proyecto a Vaadin 14 para lo cual se debió de investigar y sustituir múltiples elementos que ya no existían en la nueva versión.
- Creación Release 0.6. Primera versión con la aplicación en la versión con Vaadin 7 con la posibilidad de subir tanto ficheros csv como xls.

Se puede ver el transcurso de las tareas del Sprint en las gráficas [A.11](#).

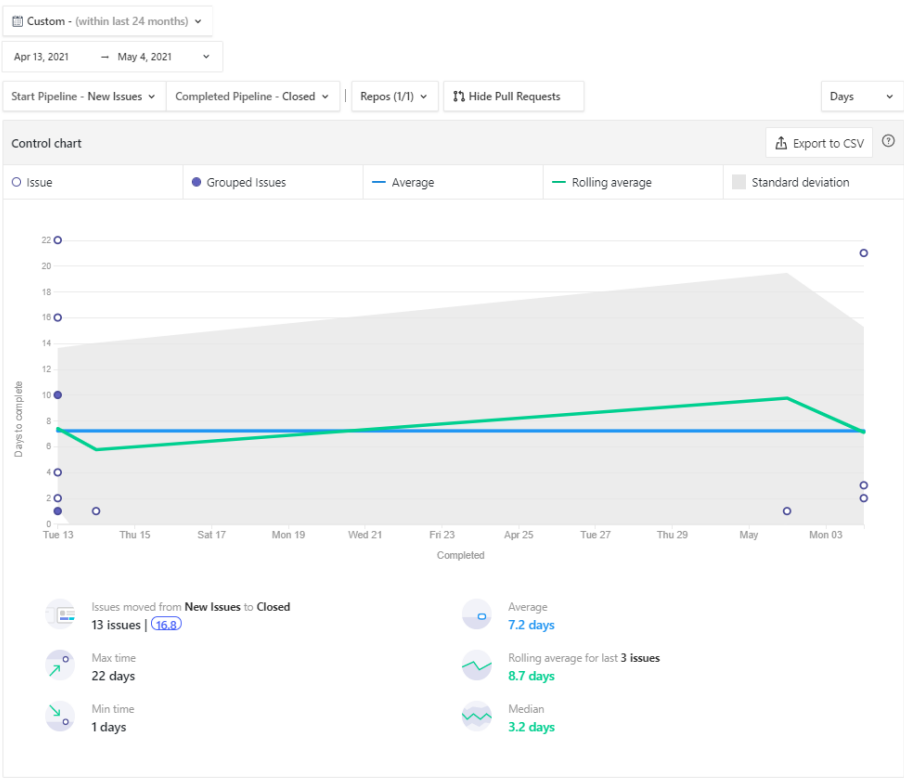


Figura A.11: Gráfica Control chart- Sprint 8

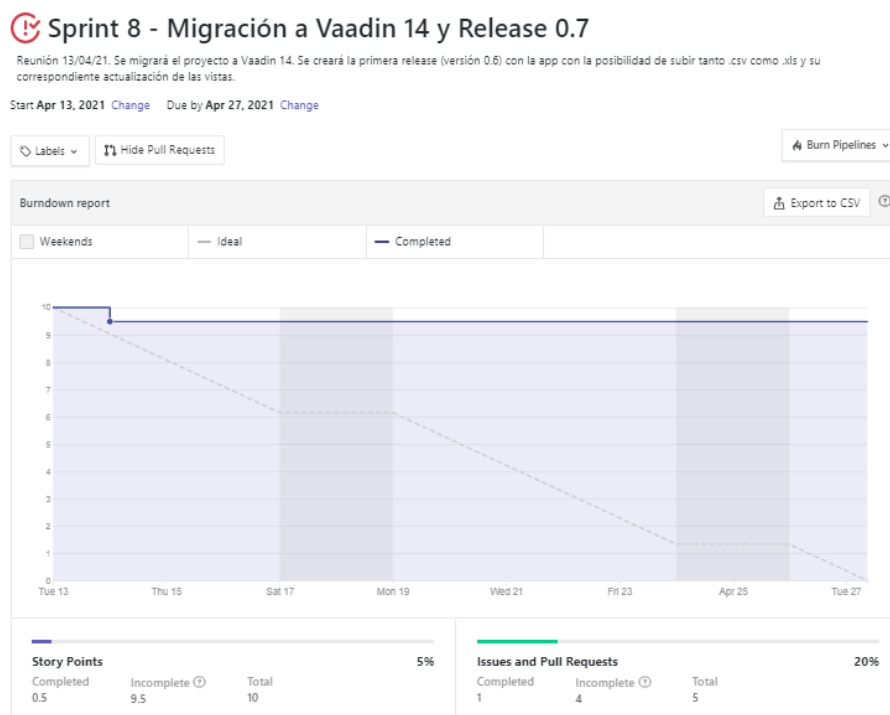


Figura A.12: Gráfica Burndown- Sprint 8

Sprint 9 - Migración Vaadin 14 (04/05/21 - 11/05/21)

Finalización del proceso de migración del proyecto a Vaadin 14 y despliegue de la aplicación en su nueva versión en **Heroku**.

Las tareas realizadas fueron:

- Investigar nueva API para realizar las gráficas del Histórico. Al cambiar de versión de Vaadin, **JFreeChart**, el componente empleado para realizar las gráficas del histórico, deja de ser compatible. Se realiza una búsqueda, en **Vaadin Directory**, de componentes que puedan sustituirlo. Las dos mejores opciones encontradas son **Vaadin Chart**, la cual es de pago por lo que es descartada y, **ApexChart**, la opción elegida.
- Continuar la migración a Vaadin 14. Se concluye la migración de la app Gestor-TFG-2021 a Vaadin 14. Se pueden encontrar los enlaces de las páginas de documentación sobre el proceso y proyectos de referencia en la **issue de Github**.
- Actualización componentes de las Vistas. Se implementaron diversas modificaciones en las vistas para conseguir que su aspecto fuese similar a su versión anterior.
- Despliegue del proyecto en Heroku. Se vuelve a desplegar el proyecto en **Heroku** con su versión en Vaadin 14 en su versión con Java 8.
- Modificación Login. Se prosigue la integración del login con Firebase.

En la siguiente ilustración se muestra la realización de las tareas del Sprint con respecto al tiempo **A.13**.

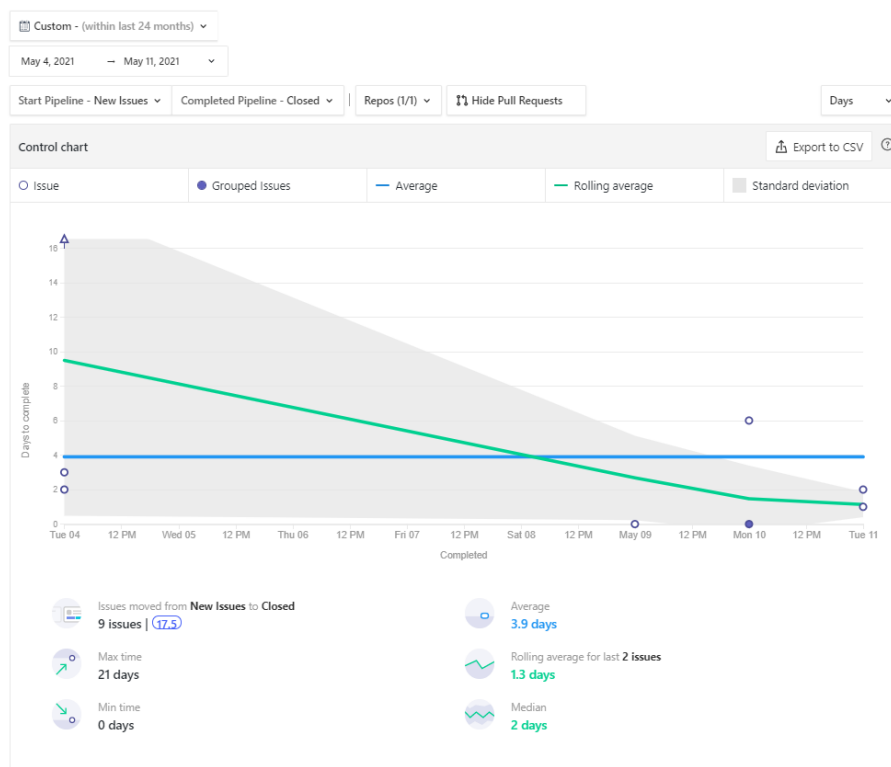


Figura A.13: Gráfica Control chart- Sprint 9

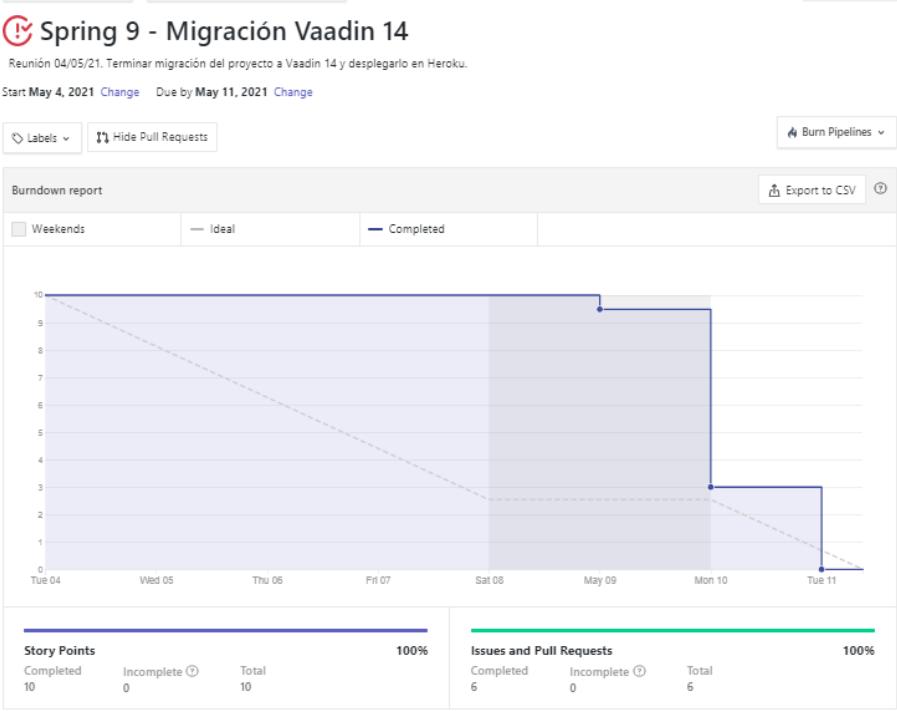


Figura A.14: Gráfica Burndown- Sprint 9

Sprint 10 - Despliegue en Heroku y Login Firebase (11/05/21 - 18/05/21)

Realizar despliegue en Java 11 con la nueva versión en Vaadin 14. Continuar con el nuevo login con Firebase. Introducir mejoras en el código.

Las tareas realizadas fueron:

- Pruebas de actualización de ficheros en el proyecto en el despliegue de heroku. Se probará a actualizar varios ficheros csv y xls para comprobar que se realiza correctamente. También se realizó una prueba con el fichero con extensión xls obtenido por los tutores en la anterior reunión para probar. Los test realizados se documentaron en la [tarea de Github](#) asociada a esta tarea.
- Despliegue del proyecto en Vaadin 14 en Heroku. Se subió el proyecto en Java 11 a [Heroku](#) según se indica en la [issue de Github](#). Donde también se explica los cambios implementados para usar Java 11.
- Actualizar obtención ranking por cursos. La obtención de los cursos para el ranking se obtenía de la vista del proyecto ($N2_{proyecto}$). Se cambiará para que obtenga el curso a partir de la fecha de presentación y asignación del Histórico ($N3_{historico}$).
- Realizar mejoras en el código. Se realizará diversos cambios para aumentar la información y calidad del código, cómo la introducción de más información en el logger, actualización de los filtros empleados en las tablas (Grid), eliminación warnings y imports no usados.
- Cambio de versión a Java 11 en el workflow Maven CI/CD. En los últimos commits no se pasaron exitosamente los test de la Integración continua, debido a que se cambió la aplicación a Java 11, en el pom.xml, pero no en el workflow. Esto se solucionará cambiando la versión de Java (java-version) en el workflow, en el fichero github-ci.yml concretamente.
- Creación release 0.8 Creación nueva release con la aplicación en **Java 11** y Vaadin 14.
- Actualizar estilos del diseño gráfico del proyecto. Modificaciones estéticas de la aplicación para conseguir un resultado más similar a la versión anterior de la aplicación, con Vaadin 7. En estos cambios destacan, por ejemplo, el cambio del estilo del texto, tablas o títulos.

Se puede ver el tiempo que se tardó en realizar las tareas y valores estadísticos del Sprint 10 [A.15](#).

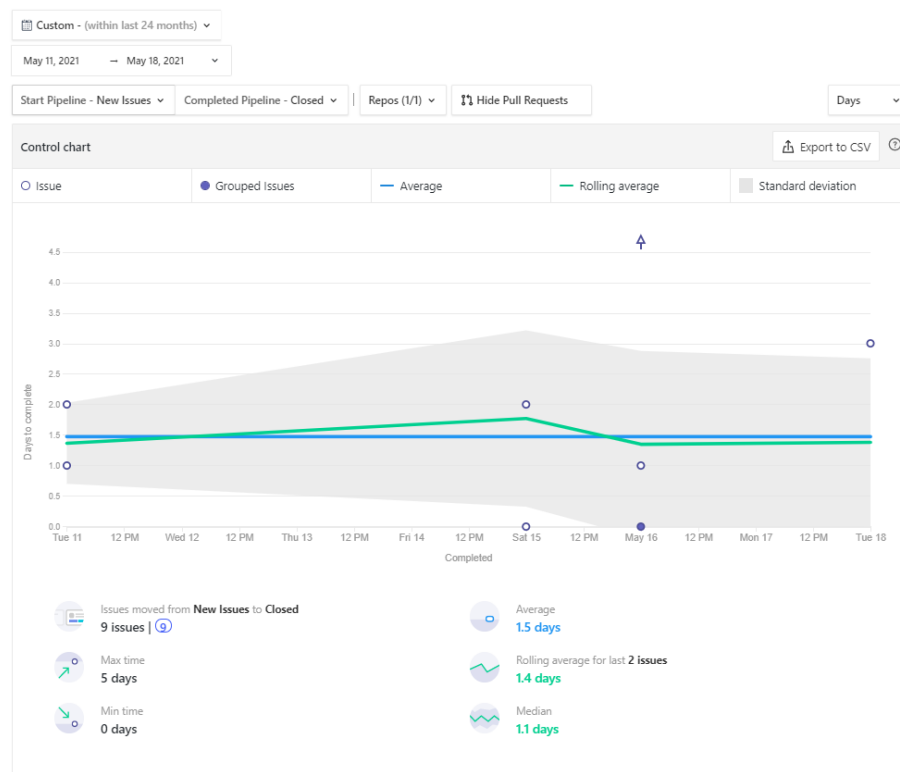


Figura A.15: Gráfica Control chart- Sprint 10

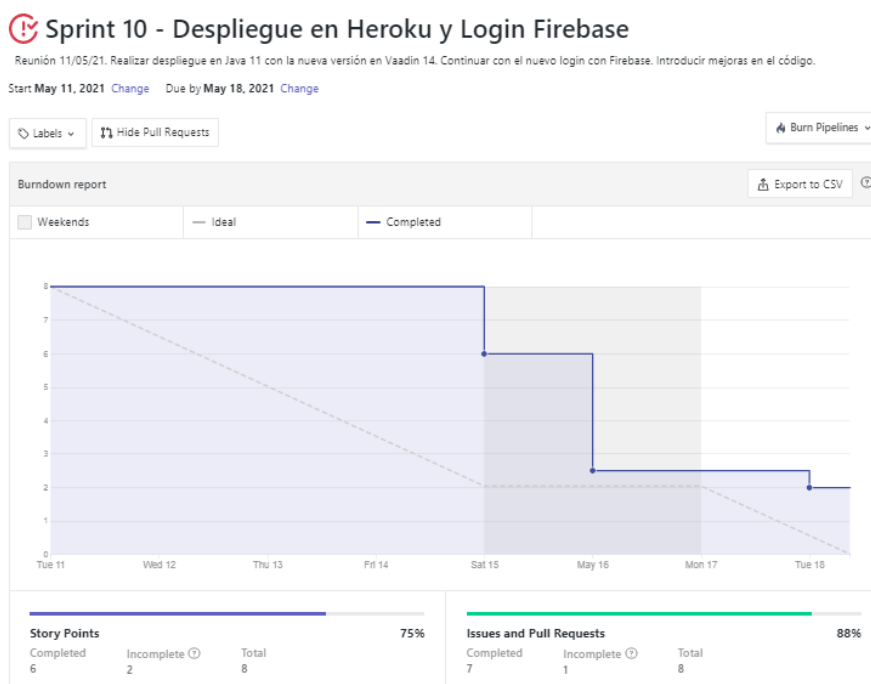


Figura A.16: Gráfica Burndown- Sprint 10

Sprint 11 - Login Firebase(18/05/21 - 01/06/21)

Realizar login con base de datos Firestore. Añadir comprobaciones antes de acceder a la vista de la actualización de ficheros para que no se pueda acceder si no hay un usuario logueado. Modificaciones estéticas de la app. Cambiar la vista al iniciar la app a la vista de InformationView. Añadir en la vista de Métricas los análisis de calidad de código de SonarCloud.

Las tareas realizadas fueron:

- Continuar con Login con Firebase. En la [tarea de Github](#) vinculada a este apartado se detalla los pasos realizados.
- Actualizar README. Actualizar enlace al despliegue de Heroku y introducir más información acerca de la app y cómo ejecutarlo.
- Añadir reglas de seguridad en Firestore. Tras crear la base de datos Firestore se añadirán nuevas reglas de seguridad para impedir su modificación a usuarios no permitidos. Para realizar las modificaciones se siguió la documentación de [Firebase](#) cómo se ve en la [issue](#).
- Añadir iconos faltantes en la app. Se buscó iconos equivalentes a los que anteriormente había en las vistas de la aplicación, ya que al actualizar de versión de Vaadin ya no existen.
- Actualizar el pie de página web (footer). Añadir iconos en el Footer, los nombres faltantes (y sus respectivos correos de la ubu) y revisar fecha actualización ficheros. Se agregó, además de la fecha de actualización de los ficheros CSV, la fecha de actualización del archivo XLS.
- Renombrar ficheros XLS al subirse. Se modificó la lógica de la vista de actualización de ficheros (UploadView) para que se pudiese subir el fichero XLS con cualquier nombre y fuese la propia app la que lo renombrará con el nombre requerido. En el caso de los ficheros csv, se indicará en la vista de UploadView cómo deben llamarse: *N1_Documento*, *N1_Norma*, *N1_Tribunal*, *N2_Alumno*, *N2_Proyecto* y *N3_Historico*.
- Añadir seguridad con *Spring boot*. Para evitar que se pueda acceder a la vista de la actualización de ficheros sin haber iniciado sesión previamente, se comenzó introduciendo Spring boot security pero, finalmente se encontró una forma más sencilla y se descartó el uso de esta opción.
- Modificar UploadView para verificar si hay un usuario que haya iniciado sesión (logueado). Se añadió un método que comprueba y controla, antes de entrar al evento de UploadView, si hay algún usuario logueado. En caso contrario, redirige al login para que el usuario inicie sesión.

En la siguiente gráfica se ilustrará el transcurso de las tareas del Sprint 11A.17.

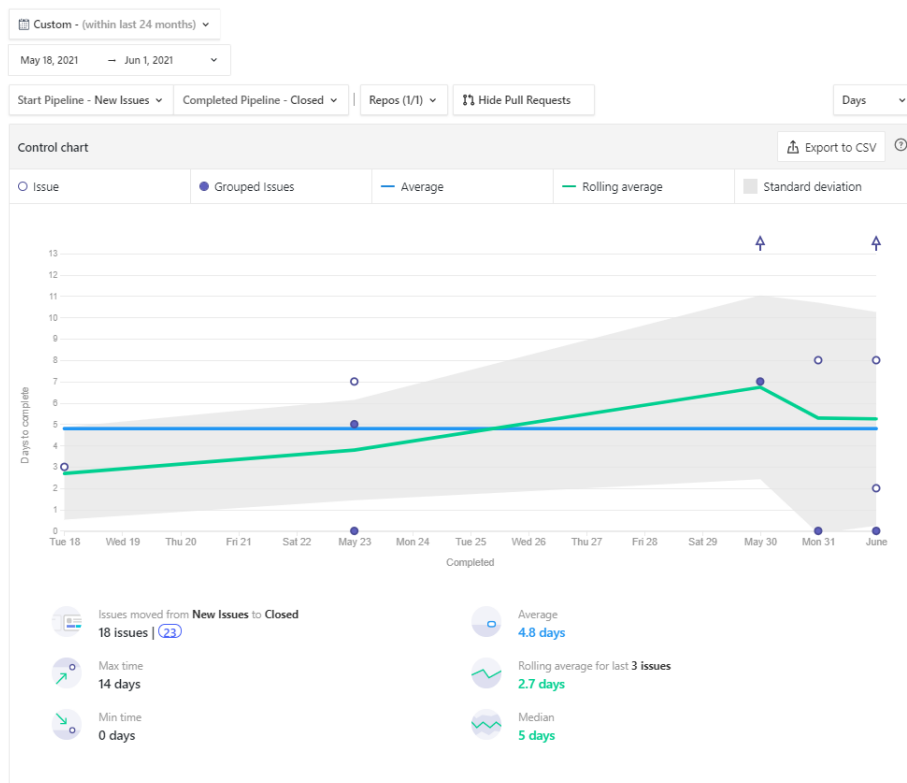


Figura A.17: Gráfica Control chart- Sprint 11

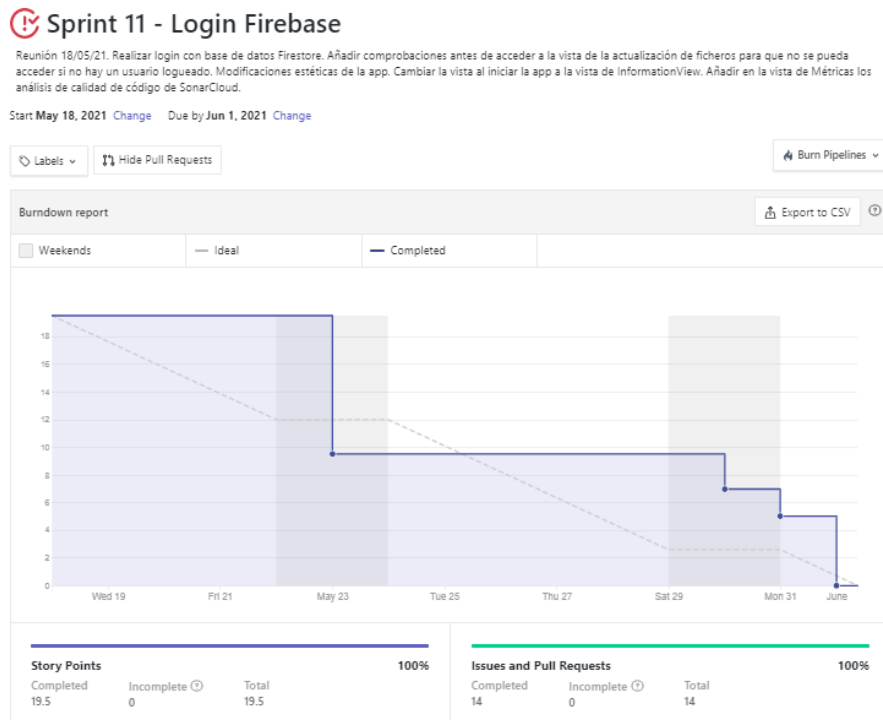


Figura A.18: Gráfica Burndown- Sprint 11

Sprint 12 - Login con UbuVirtual (01/06/2021 - 08/06/2021)

Reemplazar el login existente por un login empleando **UbuVirtual** con Moodle. Añadir enlace en el botón de métricas de la navegación a SonarCloud, a los análisis de los proyectos. Realizar mejoras de código y documentar. Revisar análisis de proyectos realizados anteriormente con SonarCloud y crear nuevos con los proyectos entregado en 2019.

Las tareas realizadas fueron:

- Crear licencia. Crear fichero con licencia (LICENSE.md).
- Anexos - Especificación Requisitos. Incorporación de mejorar y ampliación del apartado Especificación Requisitos.
- Anexos - *D_{Manual}programado* - Estructura de directorios. Añadir el apartado Estructura de directorios, donde se especifica el contenido de cada carpeta del repositorio del proyecto.
- Cambiar el color de la celda en función del Ranking. Se intentó colorear las celdas del **rankings de percentiles** en función de su valor para

dar así una mejor visibilidad. Se intentó realizar cómo se explica en diversos foros de la página de Vaadin pero, no se consiguió que se aplicarían los cambios sobre las celdas especificados en los estilos en css.

- Añadir plugin de cobertura en pom.xml. Añadir plugin que se empleaba en la aplicación anterior, con Vaadin 7, para incorporar documentos de medición de la cobertura de la app.
- Cambiar url del enlace a SonarCloud. Especificar la url en el apartado de métricas de la barra de acceso a las vistas a los [proyectos en Sonarcloud](#).
- Añadir test para lograr una mayor cobertura. Modificación de los test y incorporación de nuevos.
- Comentar y mejorar calidad código. Se revisará los comentarios y se mejorará la calidad del código, teniendo en cuenta el análisis realizado en la siguiente [issue de Analizar la app de Gestor-TFG-2021 con SonarCloud](#).
- Validar el fichero subido. Verificar el estado de la app tras subir un fichero incorrecto.
- Fork todos los proyectos entregados en 2019. Se realizará un Fork de todos los proyectos con FechaPresentacion 2019.
- SonarCloud - Analizar proyecto AVC. Se intentará analizar el proyecto correspondiente al TFG Asistente Virtual para la Comunicación, se puede ver los pasos realizados en [Github](#).
- SonarCloud - Analizar proyecto UBUMonitor Clustering . Al realizar el análisis del proyecto surgieron varios errores debido a la falta de referencias de ciertas librerías necesarias. Al ser un proyecto Maven, en el pom.xml deberían estar referenciadas. Para intentar arreglar este problema, se modifico el pom.xml pero, no se consiguió solucionar el error. Se pueden ver los errores y los apsos seguidos en [Github](#).
- SonarCloud - Analizar proyecto HealthApp. Se trata de un trabajo en Python, por tanto, se usó el análisis automático cómo puede verse en la [issue de Github](#).
- SonarCloud - Analizar proyecto SmartBeds. Al igual que el proyecto anterior, se usó el análisis automático ya que usa Python. En la [issue de Github](#) se explica cómo se realizó.
- SonarCloud - Analizar proyecto Ububooknet. Se hizo uso del análisis automático cómo se detalla en la [tarea](#) asociada de Github.
- SonarCloud - Analizar proyecto InterpretacionEscalas UBU. El proyecto Interpretación de Escalas UBU se encuentra programado en su mayoría por Java, por lo que se necesito especificar los archivos fuente y binarios. En la [issue](#) de Github viene explicado este proceso.

- SonarCloud - Analizar proyecto Monitor-en-tiempo-real-de-un-sistema-de-fabricacion-aditiva para Octoprint. En la **tarea** de Github correspondiente a este apartado viene detallado el proceso de análisis.
- SonarCloud - Analizar proyecto PCVN. Se realizó el análisis de la calidad del código de este proyecto como se describe en la **issue** de Github.
- SonarCloud - Analizar proyecto NoisyNER. Se trata de un trabajo en su mayoría en Python por lo que se realizará el análisis automático. Se puede ver cómo se realizó en la **issue** vinculada con esta tarea.
- Renombrar proyectos con UBU-TFG. Se les añadirá ÜBU TFG.^{al} principio del nombre del repositorio para facilitar su búsqueda. Para modificar los nombres se accede al apartado "Settings» .options" del repositorio y, una vez ahí, modificar el nombre.
- SonarCloud - Analizar proyecto Detección de Pallets mediante láser. El proceso de esta tarea viene detallado en **Github**.
- SonarCloud - Analizar proyecto Plataforma para el Análisis de Trayectorias Semánticas. El análisis del TFG Plataforma para el Análisis de Trayectorias Semánticas se realizó como se describe en la **issue de Github**.
- SonarCloud - Analizar proyecto Sistema Recomendación TFG. El proceso viene explicado en la **issue** de Github.
- SonarCloud - Analizar proyecto Datos públicos. Se analizó el proyecto Aplicación Web para la recopilación, tratamiento y visualización de datos públicos ² con SonarCloud como se indica en la **issue**.
- SonarCloud - Analizar proyecto Sistema Información sobre Matriculación. Para hacer una evaluación de la calidad del código se realizaron los pasos indicados en la **issue** de Github.
- SonarCloud - Analizar proyecto Amazon-Scraper. Se analizará el proyecto Extracción y procesamiento de datos de Amazon para su utilización en un estudio de marketing SonarCloud.
- SonarCloud - Analizar proyecto SmartBeds 1. Al igual que en proyectos anteriores, los procedimientos seguidos se encuentran explicados en **Github**.
- SonarCloud - Analizar proyecto Aproximación hacia la identificación automática de lesión de ictus en TC craneal. Esta tarea se encuentra detallada en la **tarea** de Github vinculada a esta.
- Revisar proyectos SonarCloud. Se repasó los proyectos que se habían analizado previamente en SonarCloud para verificar que no se habían analizando lenguajes no deseados u otros errores. En la **tarea** de Github se enumeran las mejoras realizadas.

- Analizar la app de Gestor-TFG-2021 con SonarCloud Se volvió a realizar la medición de la calidad del código de la aplicación web con SonarCloud. Al emplear **Java 11** ya no hubo problemas de compatibilidad como pasaba anteriormente. En [Github](#) se explica cómo se analizó el proyecto **sistinf**.
- Actualizar despliegue de la app en Heroku Se actualizó el despliegue en [Heroku](#) con los nuevos cambios introducidos. Una de las modificaciones más significativas fue la autenticación del login con el moodle de UbuVirtual.
- Añadir Login verificado con Moodle Se realizó modificaciones en el Login para realizar la verificación del usuario mediante el Moodle de **UbuVirtual**. Se tomará como referencia el proyecto **UBUMonitor** y la documentación de Moodle. También se contó con el asesoramiento del profesor Raúl Marticorena Sánchez.
- Cambiar el banner para que aparezca el nombre de la app Gestor-TFG-2021 Cuando se ejecutaba la app salía la cabecera de Apexchart, por lo que, se cambió para que figurará el nombre del proyecto "Gestor-TFG-2021". El banner se generó desde la página [Sprint Boot banner.txt generator](#).
- Crear enlace en el botón de navegación Métricas a SonarCloud Se reemplazó la ventana de Métricas por un enlace que redirige a los análisis de los proyectos en SonarCloud.
- Añadir una coma entre los nombres de los tutores en las tablas Se incluyó una coma entre los nombres de los tutores y alumnos en las tablas de Proyectos Activos y Histórico.
- Eliminar icono de la vista de Información Se suprimió el icono de participantes de la vista de InformationView para mejorar la estética de esa vista.

Se puede ver el desarrollo del Sprint 12 en la ilustración mostrada previamente [A.19](#).

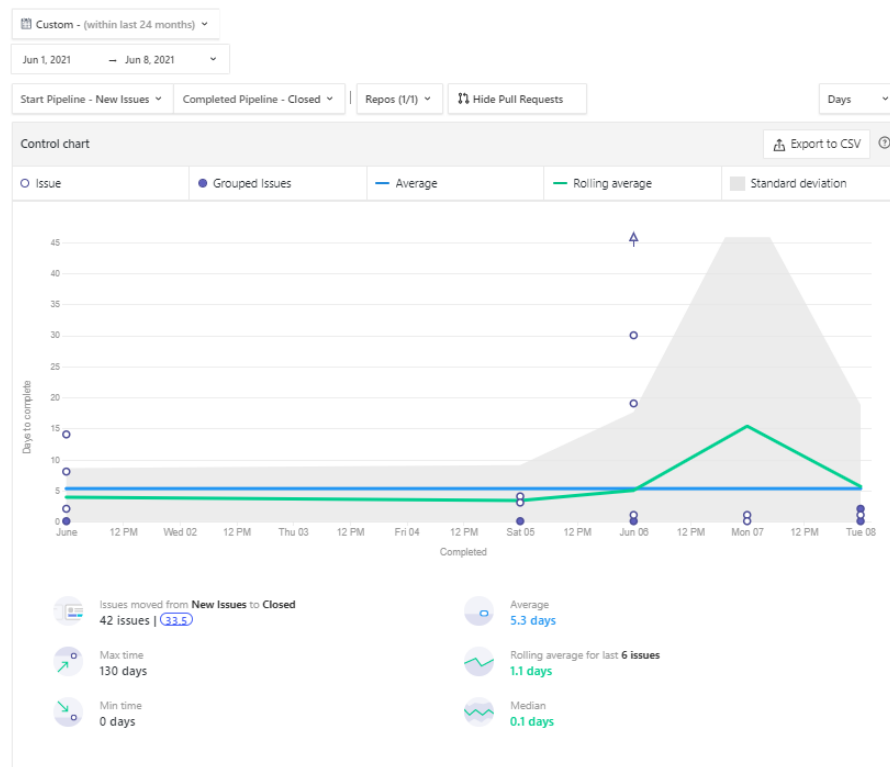


Figura A.19: Gráfica Control chart- Sprint 12

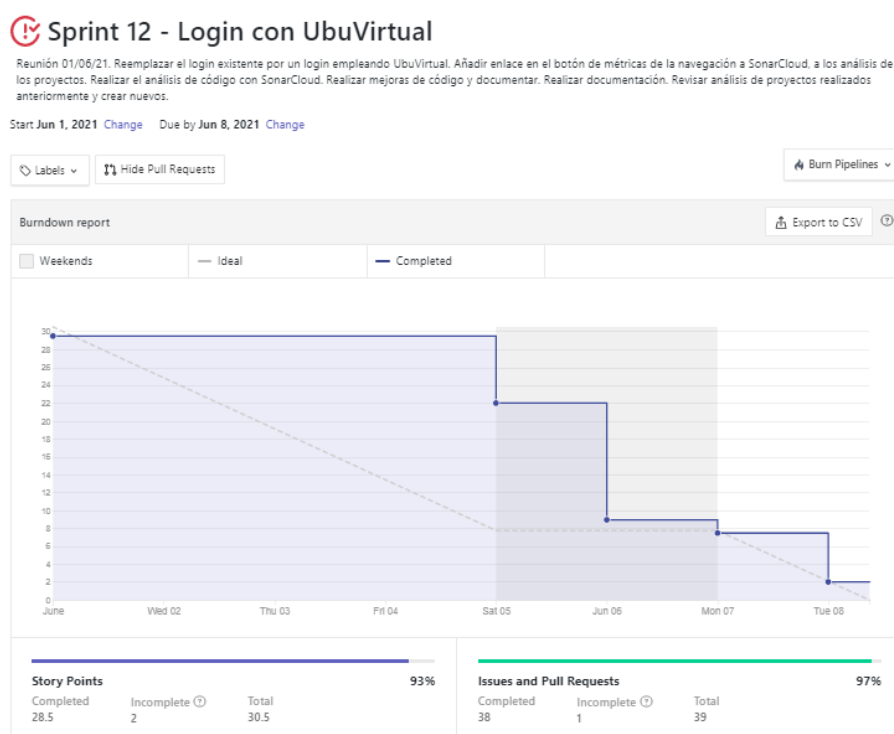


Figura A.20: Gráfica Burndown- Sprint 12

Sprint 13 - Revisión de la conexión con UbuVirtual (08/06/21 - 18/06/21)

Revisión de la autenticación con **Moodle** y añadir test para comprobar la conexión por **webServices**.

Las tareas realizadas fueron:

- Añadir test para verificar el login a un moodle de ejemplo Crear test que verifiquen el funcionamiento de las funciones de inicio de sesión, obtención de cursos y comprobación de permisos de actualización con uno de las páginas **Moodle** de ejemplo.
- Modificar Login. Modificar Login para solamente mirar los permisos de la asignatura de **Trabajos de fin de grado**, en lugar de todos los cursos.

A continuación, se expondrán dos ilustración para mostrar el trascurso del Sprint y el desarrollo de las tareas **A.21**.

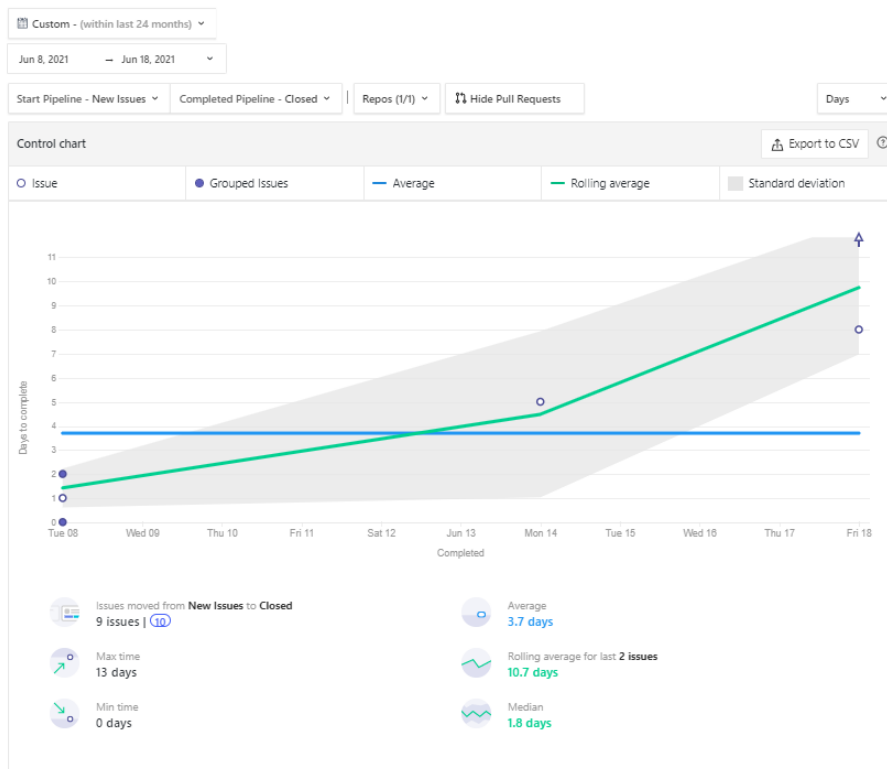


Figura A.21: Gráfica Control chart- Sprint 13

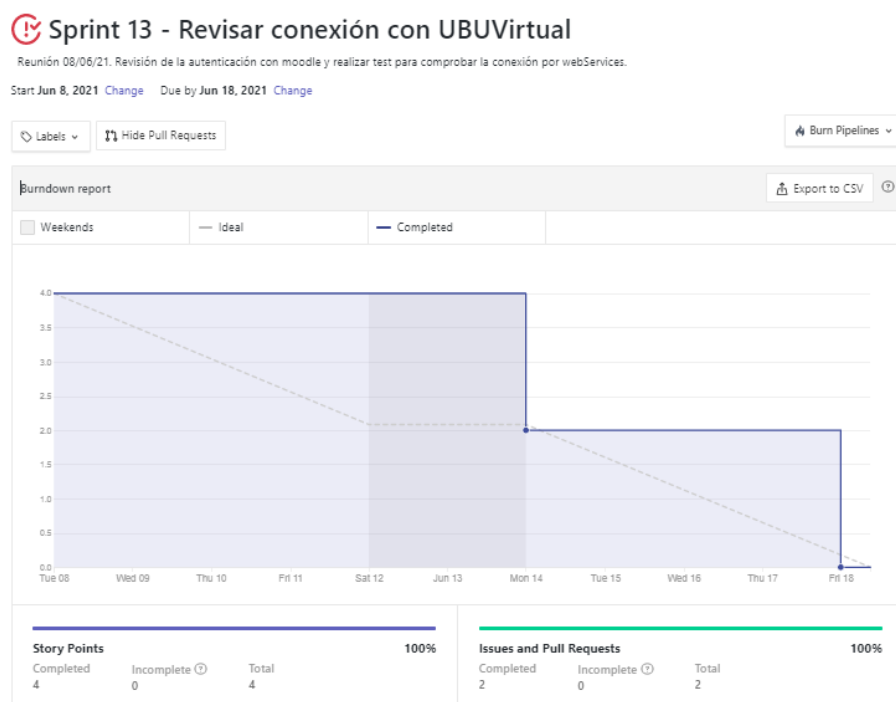


Figura A.22: Gráfica Burndown- Sprint 13

A.3. Estudio de viabilidad

Viabilidad económica

En este apartado se detallan los costes que llevaría realizar este proyecto. Se considerarán los costes de recursos humanos, el material empleado y el *Software* usado.

Coste del personal

El proyecto ha sido desarrollado por una única persona durante 9 meses a tiempo parcial. Considerando un salario bruto mensual de 2000, el coste total del personal sería el siguiente:

Coste hardware

Referente a los costes del equipo utilizado en el desarrollo del trabajo. Teniendo en cuenta el precio del ordenador empleado de aproximadamente 700 euros.

Coste *software*

Referente a los costes de las herramientas software no gratuitas empleadas en el proyecto. Como es el caso del Sistema Operativo Windows 8 Home que se emplea en el equipo usado

Viabilidad legal

Se detallaran las licencias *Software* de cada dependencia que se ha utilizado en el proyecto.

Software	Licencia
Vaadin	Apache License 2.0
Spring Boot Maven Plugin	Apache License 2.0

Tabla A.1: Dependencias del proyecto

Apéndice *B*

Especificación de Requisitos

B.1. Introducción

La especificación de requisitos hace referencia a los requerimientos que debe cumplir el software para satisfacer las necesidades del cliente. Debe incluir la suficiente cantidad de detalles para permitir a los desarrolladores software diseñar el sistema.

Solo se incluirán los requisitos realizados en esta mejora.

B.2. Objetivos generales

El objetivo general del proyecto es continuar con el desarrollo y la mejora de la aplicación web respecto a la versión anterior, centrándose en los siguientes puntos:

- Optimizar la actualización de la información pública en la página Web añadiendo la posibilidad de actualizar con una hoja de datos **Excel** con múltiples hojas. Se deberá conservar la opción de renovar la información con diversos ficheros en formato .csv.
- Evolucionar el análisis de proyectos a una plataforma pública en la nube, **SonarCloud**, en lugar de emplear un servidor local de Sonarqube. Se deberán analizar la calidad del código de los TFG de cursos anteriores que cuenten con un repositorio público en Github.
- Mejorar la *interface* gráfica de la aplicación empleando nuevos componentes gráficos de la biblioteca Vaadin.

- Incorporar un sistema de autenticación de usuarios para la actualización de la información de la página Web de los TFG.
- Añadir tres indicadores cualitativos de tipo ranking sobre las notas de los Trabajos de Fin de Grado presentados. Se trata de un ranking por curso, un ranking sobre todos los TFG (total) y un ranking de percentiles sobre el total de los TFG que consta de cinco niveles A,B,C,D,E según el percentil calculado, siendo la A el mejor resultado.

B.3. Catálogo de requisitos

Se describirán los requisitos específicos, funcionales y los no funcionales.

Requisitos funcionales

- **RF-1 Autenticar usuarios:** la aplicación debe permitir comprobar la identidad del usuario mediante UbuVirtual.
 - **RF-1.1 Verificación de la identidad del usuario:** se comprobará si las credenciales introducidas en el inicio de sesión corresponden con alguna cuenta de la UBU.
 - **RF-1.2 Obtención del curso correspondiente al Trabajo de Fin de Grado:** se buscará si el usuario tiene asignada la asignatura de Trabajo de Fin de Grado.
 - **RF-1.3 Chequeo de permisos del usuario:** la aplicación revisará si el usuario posee permisos de actualización en el curso del TFG y, por tanto, en la aplicación Web.
 - **RF-1.4 Conceder el acceso al usuario:** se aprobará la entrada del usuario autenticado a las páginas restringidas.
- **RF-2 Actualizar ficheros xls:** la aplicación permite emplear dos tipos de archivos, xls y csv, como entrada de datos.
 - **RF-2.1 Subida de datos:** el usuario autenticado podrá subir un fichero de datos, ya sea en el formato xls como en el csv.
 - **RF-2.2 Validación de los datos:** se permitirá subir únicamente los ficheros en el formato establecido.
 - **RF-2.3 Actualización de la información:** la aplicación deberá refrescar los datos con los existentes en el fichero subido.

- **RF-3 Visualizar los rankings sobre las notas de los TFG:** se incluirán tres rankings en la tabla del Histórico.
 - **RF-3.1 Mostrar los rankings de notas:** se visualizarán las clasificaciones de los notas de los proyectos realizados a modo de rankings.

Requisitos no funcionales

- **RNF-1 Seguridad:** la aplicación solamente debe permitir la subida de datos a los usuarios con permisos.
- **RNF-2 Mantenibilidad:** mejora de la aplicación para permitir la incorporación de nuevas modificaciones en el futuro de forma sencilla.
- **RNF-2 Mejora diseño:** se realizarán mejoras gráficas de la aplicación para que resulte más atractiva e informativa. Se optará siempre por opciones intuitivas y sencillas de utilizar.
- **RNF-3 Analizar la calidad de código:** cómo parte del apartado de las métricas se examinaran Trabajos de Fin de Grado de años anteriores, con el fin de añadir más información acerca de ellos.

B.4. Especificación de requisitos

Diagrama de casos de uso

En esta sección se mostrarán los diagramas de casos de uso. En la aplicación hay dos actores: el usuario con permisos de actualización de los datos de la aplicación y el usuario normal.

Usuario	
Usuario con permisos de actualización	Generalmente se tratará de un profesor, aunque también puede ser un alumno. Podrá acceder, a través del login, a la vista de actualización de los datos de la aplicación.
Responsable	El usuario podrá acceder a todas las vistas de la aplicación.

Tabla B.1: Actores de la aplicación

Se puede ver un resumen de los casos de uso descritos anteriormente en la siguiente imagen. [B.1](#).

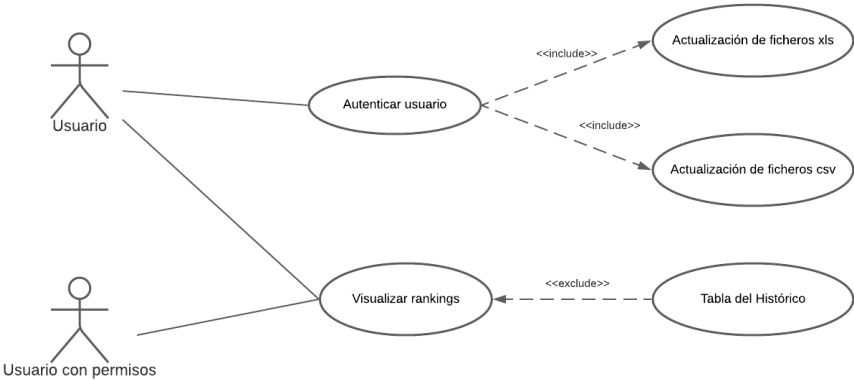


Figura B.1: Diagrama de casos de uso

Especificación de casos de uso

Caso de uso 1: Autenticar usuarios.		
Descripción	Verificación de un usuario en el login de la aplicación mediante UbuVirtual	
Precondiciones	Validar el usuario a través del login	
Requisitos	RF-2.1, RF-2.2, RF-2.3	
Secuencia normal	Paso	Acción
	1	El sistema comprobará si las credenciales introducidas corresponden con alguna cuenta de la UBU
	2	Se verificará si el usuario tiene la asignatura de Trabajo de Fin de Grado asignada.
	3	El sistema revisará si el usuario posee permisos de actualización.
	4	En caso de que tenga permisos de actualización, se permitirá el acceso a la vista de actualización.
Postcondiciones	Introducir credenciales en el login	
Excepciones	Usuario/contraseña inválidos. El usuario no tiene los permisos requeridos para acceder.	
Frecuencia	Baja	
Importancia	Alta	
Urgencia	Alta	

Tabla B.2: Caso de uso 1: Autenticar usuarios.

Caso de uso 2: Actualizar con ficheros xls.		
Descripción	Actualizar la información de la aplicación con ficheros xls	
Precondiciones	Validar el usuario a través del login	
Requisitos	RF-2.1, RF-2.2, RF-2.3	
	Paso	Acción
Secuencia normal	1	Se selecciona el fichero en formato xls que se desea subir
	2	Se presiona el símbolo de <i>play</i> para que comience el proceso de actualización
	3	Se espera a que finalice la carga del fichero
	4	Al finalizar la carga se puede subir más ficheros
Postcondiciones	Se modifican los nombres en la visualizaciones	
Excepciones	Error al cargar el fichero.	
Frecuencia	Baja	
Importancia	Alta	
Urgencia	Alta	

Tabla B.3: Caso de uso 2: Actualizar ficheros xls.

Caso de uso 3: Visualizar rankings de la tabla del Histórico.		
Descripción	Se mostrará en la tabla del histórico tres rankings sobre las notas de los proyectos	
Precondiciones	Ninguna	
Requisitos	RF-3.1	
	Paso	Acción
Secuencia normal	1	Acceder a la vista del histórico
Postcondiciones	Ninguna	
Excepciones	Niguna	
Frecuencia	Alta	
Importancia	Media	
Urgencia	Baja	

Tabla B.4: Caso de uso 3: Visualizar los rankings sobre las notas de los TFG.

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se detallarán los aspectos referentes al diseño de la aplicación en esta mejora de la aplicación.

C.2. Diseño de datos

Ficheros de datos

Uno de los requisitos del proyecto era incluir la posibilidad de subir los datos de la aplicación mediante ficheros con varios hojas de datos, en lugar de tener que subir cada hoja de datos en documentos por separado(csv). Se deben **cumplir algunas condiciones** para asegurar la **correcta obtención de los datos**:

- En el caso de las **fechas** existen dos formatos: uno de ellos en las hojas con los datos de los proyectos (N2_Proyecto) que deberán figurar los años referentes al curso de asignación separados de guiones. Por ejemplo 2016-2017. Y, en cuanto a las fechas de presentación y asignación, de las hojas con los datos de los proyecto realizados anteriormente (N3_Historico) deberán estar en europeo de DD/MM/AAAA tal como 08/07/2021.
- Respecto a las **calificaciones** de las notas del histórico (N3_Historico), los números se deberá usar el el punto como separador de decimales, como por ejemplo 5.5.

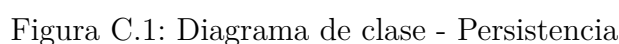
En el caso que no se cumpla estas condiciones en las vistas del histórico y los proyectos activos no se mostraran los datos.

Diagramas de clases

Con la introducción de la posibilidad de actualizar los datos de la página web con una hoja de datos Excel con múltiples hojas (.xls) se requirió una modificación en las clases fachada para la obtención de los datos.

Se dividió la clase fachada que existía anteriormente en dos, una encargada de los ficheros en formato csv y otra con los formatos xls. Se creará una clase abstracta de la que heredarán las dos clases fachada, las cuales comparten la misma estructura de funciones [C.1](#).

Se encuentra en el paquete **ubu.digit.persistence** donde se emplea el patrón de diseño **Singleton** que permite restringir la creación de ob de una clase para garantizar que unicamente existe una instancia de la clase. También se emplea el patrón **Fachada**, con una nueva clase "SistInfFactory", la cual se encarga de interactuar y intercambiar las dos fachadas de datos que existe.



- En **ubu.digit.view C.2** se encuentran las vistas de la aplicación que acceden a los datos de los ficheros a través de las dos clases fachada, "SistInfDataCsv" y "SistInfDataXls".



-
- ```
classDiagram
 class VerticalLayout {
 +serialVersionUID: long
 }
 class HorizontalLayout {
 +serialVersionUID: long
 }
 class Footer {
 +serialVersionUID: long
 +conten: HorizontalLayout
 +license: VerticalLayout
 +information: VerticalLayout
 +fileName: String
 +Footer(fileName: String)
 +addInformation(): void
 +getLastModified(fileName: String): String
 +addLicense(): void
 }
 class NavigationBar {
 +serialVersionUID: long
 +buttonInfo: Button
 +buttonActive: Button
 +buttonHistory: Button
 +buttonMetrics: Button
 +NavigationBar()
 +initComponents(): void
 }
 VerticalLayout <|-- Footer
 HorizontalLayout <|-- NavigationBar
```

Figura C.3: Diagrama de clase - Componentes

- En el paquete **ubu.digit.util** se creo una clase nueva, "UtilMethods" para albergar las funciones que obtienen la información, en formato en JSON, del moodle de UbuVirtual que se emplea en la validación de los usuario en el login.C.4. Se añadieron nuevas constantes (en "Constants") y se reemplazo la forma de obtener las rutas de los ficheros en la clase "ExternalProperties".

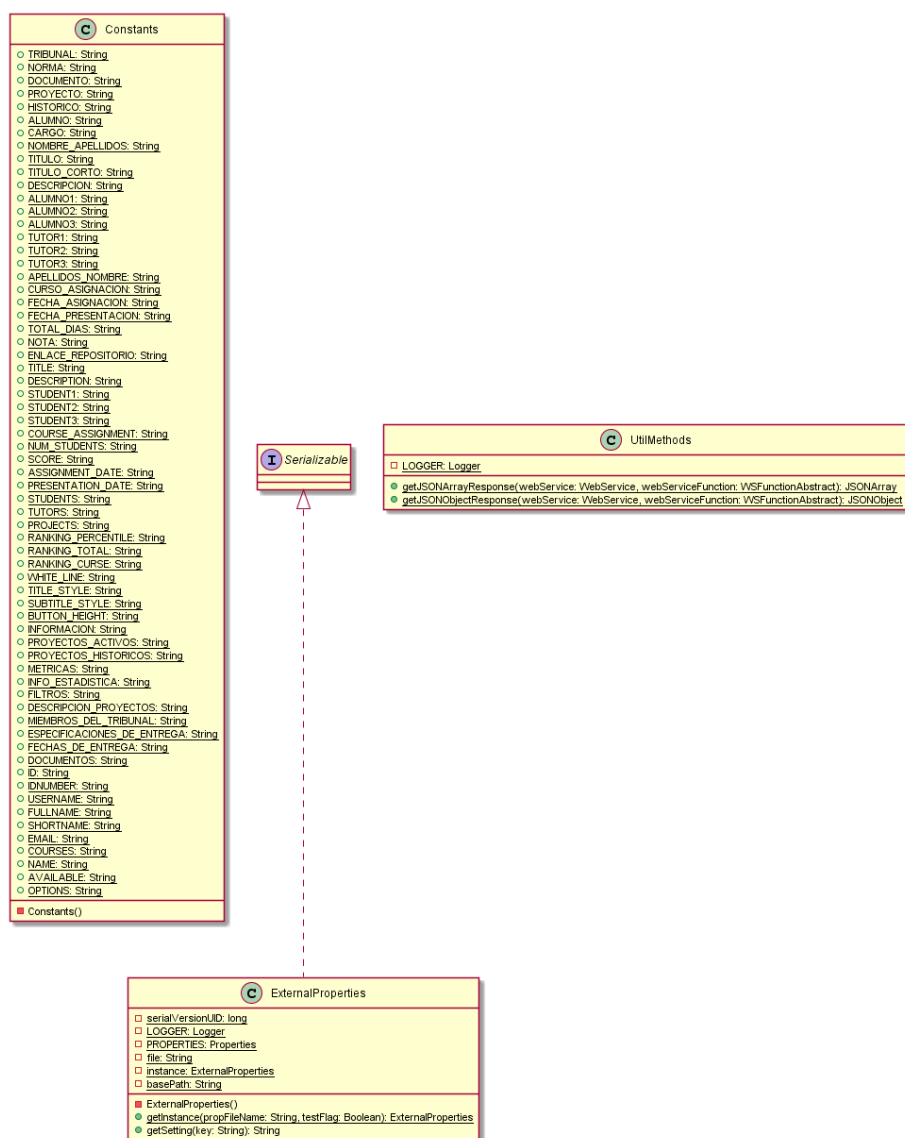


Figura C.4: Diagrama de clase - Util

- [illegible]

Figura C.5: Diagrama de clase - Entidades

- Para la realizar la conexión con el moodle de UbuVirtual se añadió un nuevo paquete, denominado **ubu.digit.security**, que alberga las clases necesarias para realizar el login del usuario en el moodle, obtener los cursos del usuario y comprobar los permisos del usuario en la asignatura de Trabajos de Fin de Grado **C.6**.

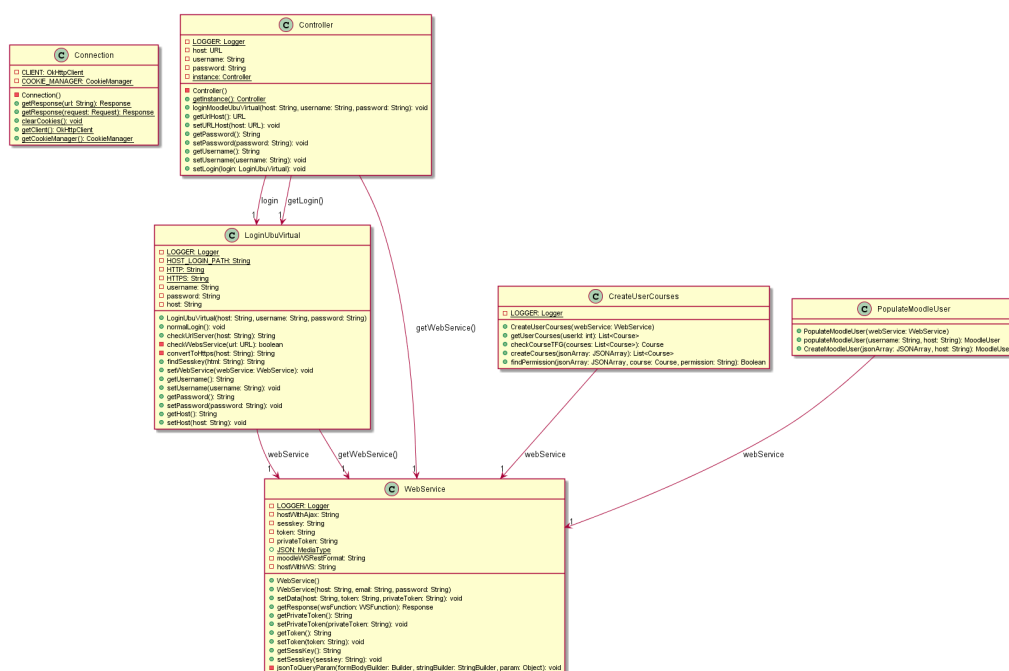


Figura C.6: Diagrama de clase - Autenticación con moodle

- Para manejar la información contenida en moodle se emplearán servicios web (*web services*). Se emplearán por ejemplo en la obtención de los cursos del usuario o de los permisos de dicho usuario en las asignaturas. Se creará una clase para cada servicio web [C.7](#).

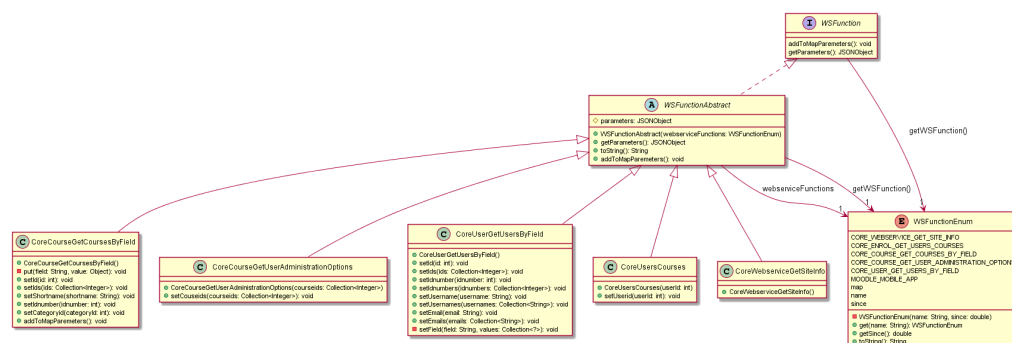


Figura C.7: Diagrama de clase -Servicios web

Con la separación de la clase fachada se requirió crear una clase test para cada fachada, "SistInfDataTestCSV" y "SistInfDataTestXLS". También se incluyó una clase, "WebServiceTest", para testear la autenticación y comprobación de permisos del usuario. Para ello se empleo uno de los [moodle de ejemplo](#) que se proporciona en su [página oficial](#). Estas clases se encuentran en el paquete **ubu.digit.persistence** [C.8](#)

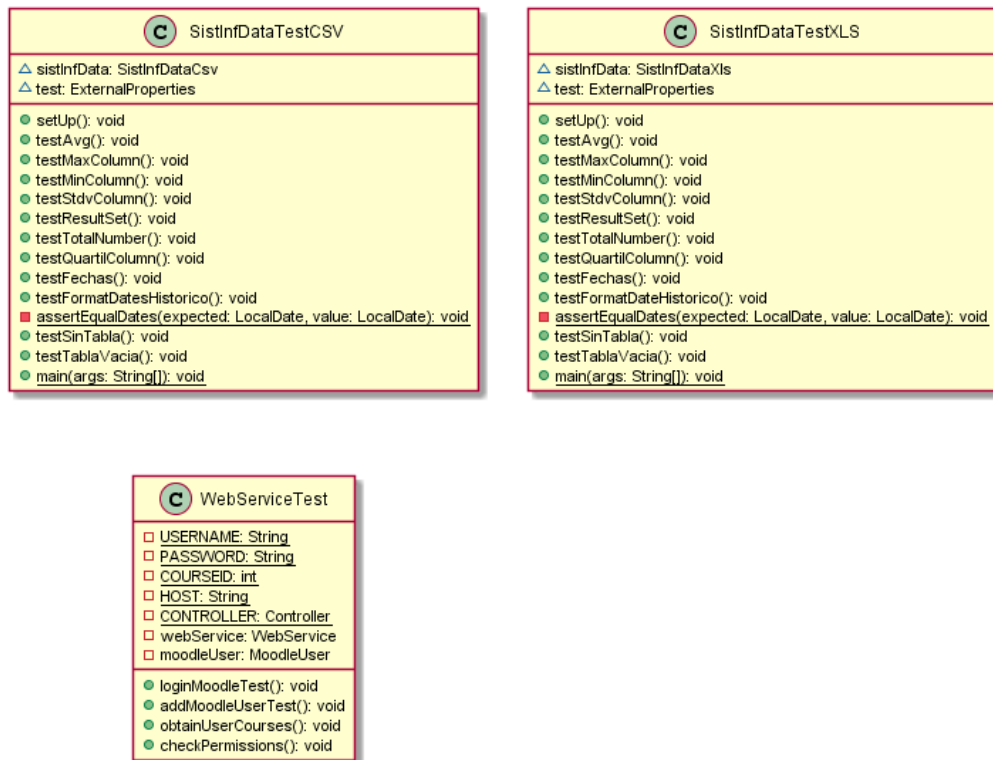


Figura C.8: Diagrama de clase - Tests

### C.3. Diseño procedimental

Para acceder a la vista de actualización de la información de la página web se debe de autenticar los permisos del usuario a través del login. El sistema sigue la siguiente lógica:



## C.4. Diseño arquitectónico

En este apartado se hablará de los patrones y estructuras que se han empleado en el proyecto.

### Singleton

Es un patrón de diseño que se basa en restringir la creación de objetos de una clase a un objeto a una única instancia. Es usado en las clases fachadas donde **solamente se tiene una instancia en todo el sistema** C.9.

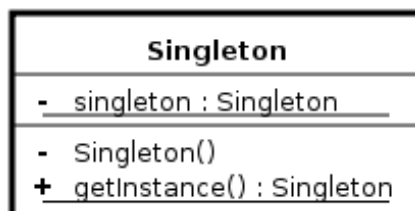


Figura C.9: Diagrama uml de la estructura del Singleton

### Fachada

Es un patrón de diseño estructural que reduce la complejidad introduciendo una división de los sistemas, consiguiendo minimizar la comunicación del sistema con los datos. Se usa en las clases de persistencia de datos, "SistInfDataCsv" y "SistInfDataXls", las cuales se relacionan directamente con los ficheros de datos y, el resto de las clases obtienen la información a través de las clases fachada C.10.

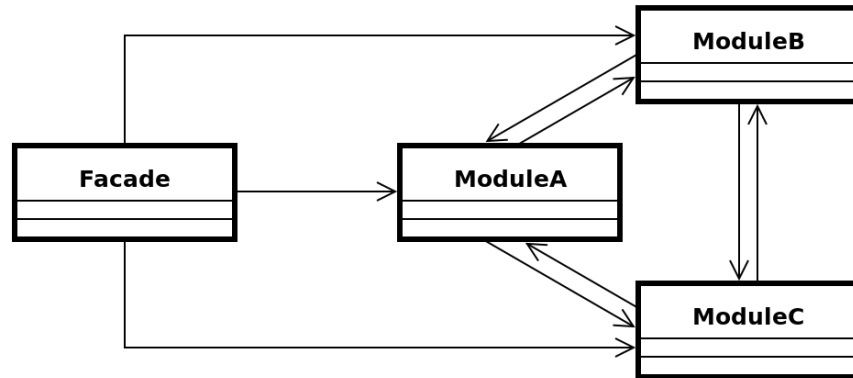


Figura C.10: Diagrama uml de la estructura de la Fachada

## C.5. Diseño gráfico

Uno de los cambios de diseño más grandes que presenta la aplicación es debido a la migración que se realizó a **Vaadin 14**. Al realizar este cambio muchos componentes pasaron a ser inválidos y se tuvieron que sustituir por nuevos. Esto supuso realizar un aprendizaje del manejo de los componentes y herramientas que Vaadin 14 proporcionaba. Los cambios a través de componentes de Vaadin 14 más significativos son:

- La introducción del **login** que permite introducir credenciales y gestionar los mensajes de errores mostrados [C.11](#).



The image shows a login interface with a white background. At the top, the title 'Iniciar Sesión' is displayed in a large, bold, black font. Below the title, there are two input fields. The first field is labeled 'Usuario' in a smaller, gray font, followed by a small blue asterisk. The field itself is a light gray rectangle. The second field is labeled 'Contraseña' in a smaller, gray font, followed by a small blue asterisk. This field is also a light gray rectangle and includes a small eye icon on the right side, indicating a toggle for password visibility. Below these two fields is a solid blue button with the text 'Iniciar Sesión' in white, bold font.

Figura C.11: Login de la aplicación

- La modificación del sistema de subida de ficheros a través del componente de **actualización** con el que se puede especificar el número máximo de ficheros a subir, el formato, entre otras muchas opciones.
- La migración de las tablas al componente **Grid**, el cual permite mostrar la información en forma de tablas e incluye diversas funcionalidades como ordenar columnas, añadir filtros y opciones para el diseño de la tabla.

Otra modificación a mencionar es el cambio de la **herramienta empleada para crear las gráficas** a **Apexcharts**. Esta herramienta proporciona una interface más atractiva debido en gran parte a la interacción con las gráficas que incluye, teniendo la posibilidad, por ejemplo, de elegir si solamente queremos que se muestre una línea de la gráfica **C.12**. Además, añade la opción de descargar la gráfica en varios formatos como por ejemplo, pdf.

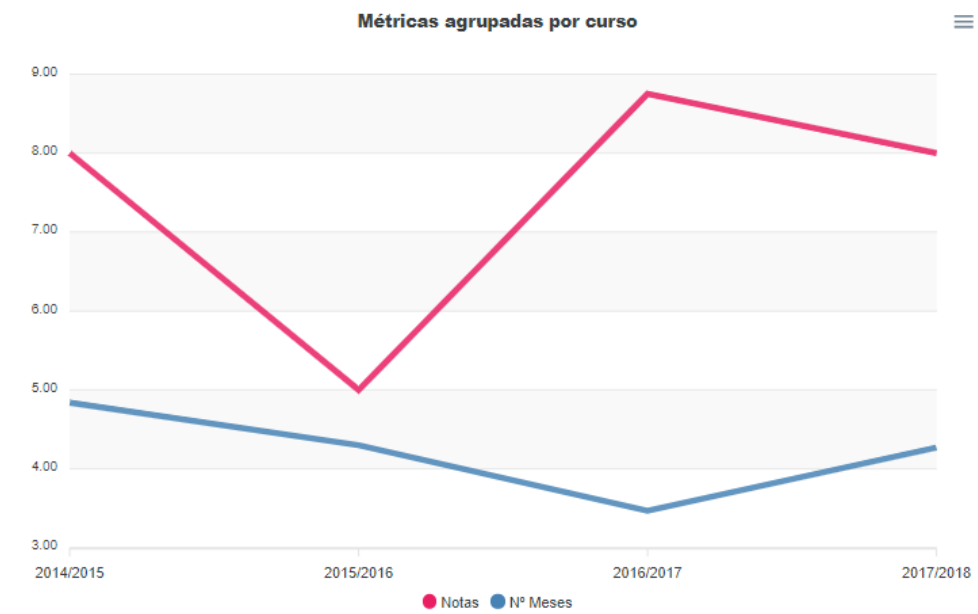


Figura C.12: Gráfica de ejemplo de la aplicación

## Apéndice *D*

---

# Documentación técnica de programación

---

### D.1. Introducción

En esta sección se describe la estructura del proyecto, el proceso de instalación del framework y herramientas necesarias para desarrollar el trabajo, cómo realizar la compilación, las pruebas realizadas, la instalación y ejecución del proyecto.

### D.2. Estructura de directorios

Se enumerarán y describirán brevemente los directorios del proyecto. Se puede encontrar el código fuente en el repositorio de Github denominado **"Gestor-TFG-2021"**.

- **/:** directorio raíz donde se ubican el README, el fichero de configuración para el despliegue de **Heroku**, los archivos de configuración de Vaadin 14, Spring Boot y Maven.
- **/.github/workflows** los archivos de *workflow* o flujo de trabajo, tanto para la Integración continua del proyecto en GitHub como para el análisis de la calidad del código en **SonarCloud**.
- **/Documentation** material de documentación del proyecto y aplicaciones de prueba empleadas.

- **/Documentacion/LaTeX** ficheros para generar la memoria y los anexos.
- **/Documentacion/Pruebas** aplicaciones prototipo para comenzar el aprendizaje con **Vaadin**.
- **/frontend** código encargado del diseño gráfico de la aplicación por el lado del cliente.
- **/src** código *backend* de la aplicación web principal, **sistinf**.
  - **/src/main/java/ubu/digit** código fuente en Java de la aplicación web.
    - **/src/main/java/ubu/digit/persistence** código fuente encargado de la conexión y lectura de los ficheros de datos (fachada de datos).
    - **/src/main/java/ubu/digit/security** código fuente de conexión y consulta con el moodle de UbuVirtual.
    - **/src/main/java/ubu/digit/ui** código en relación a las ventanas y vistas de la aplicación.
    - **/src/main/java/ubu/digit/util** incluye los métodos empleados de utilidad empleados en toda la app.
    - **/src/main/java/ubu/digit/webService** servicios web empleados para la consulta en moodle.
  - **/src/test** tests unitarios sobre las clases fachada, "SistInfDataCsv" y "SistInfDataXls", y sobre los servicios web (*web service*).

### D.3. Manual del programador

A continuación se explicará cómo realizar la instalación de los programas necesarios para el desarrollo de la aplicación.

#### Instalación de Java

Anteriormente se empleaba Vaadin 8, por lo que se debía emplear la versión Java 8, en concreto se usaba `jdk1.8.0_271`. Pero, con la migración a Vaadin 14 se cambió a Java 11.

Para ello se debe descargar la [página de descargas de Oracle Java SE 11.0](#) y descargar la versión de JDK 11, correspondiente con el sistema operativo que se posea y su arquitectura, ya sea de 64 o 32 bits. Ver imagen [D.1](#).

Tras escoger la versión según el SO, se deberán de leer y aceptar las licencias de uso de Oracle, como se muestra en la figura D.2, y dar a descargar. También se deberá cambiar la variable de entorno de java del sistema.












| Java SE Development Kit 11.0.10                                                                    |           |                                                                                                                                          |
|----------------------------------------------------------------------------------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------|
| This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE |           |                                                                                                                                          |
| Product / File Description                                                                         | File Size | Download                                                                                                                                 |
| Linux ARM 64 Debian Package                                                                        | 145.64 MB |  <a href="#">jdk-11.0.10_linux-aarch64_bin.deb</a>      |
| Linux ARM 64 RPM Package                                                                           | 152.22 MB |  <a href="#">jdk-11.0.10_linux-aarch64_bin.rpm</a>      |
| Linux ARM 64 Compressed Archive                                                                    | 169.37 MB |  <a href="#">jdk-11.0.10_linux-aarch64_bin.tar.gz</a>   |
| Linux x64 Debian Package                                                                           | 149.39 MB |  <a href="#">jdk-11.0.10_linux-x64_bin.deb</a>          |
| Linux x64 RPM Package                                                                              | 156.12 MB |  <a href="#">jdk-11.0.10_linux-x64_bin.rpm</a>          |
| Linux x64 Compressed Archive                                                                       | 173.31 MB |  <a href="#">jdk-11.0.10_linux-x64_bin.tar.gz</a>       |
| macOS Installer                                                                                    | 167.51 MB |  <a href="#">jdk-11.0.10_osx-x64_bin.dmg</a>            |
| macOS Compressed Archive                                                                           | 167.84 MB |  <a href="#">jdk-11.0.10_osx-x64_bin.tar.gz</a>         |
| Solaris SPARC Compressed Archive                                                                   | 184.82 MB |  <a href="#">jdk-11.0.10_solaris-sparcv9_bin.tar.gz</a> |
| Windows x64 Installer                                                                              | 152.32 MB |  <a href="#">jdk-11.0.10_windows-x64_bin.exe</a>      |
| Windows x64 Compressed Archive                                                                     | 171.67 MB |  <a href="#">jdk-11.0.10_windows-x64_bin.zip</a>      |

Figura D.1: Descarga de JDK 11

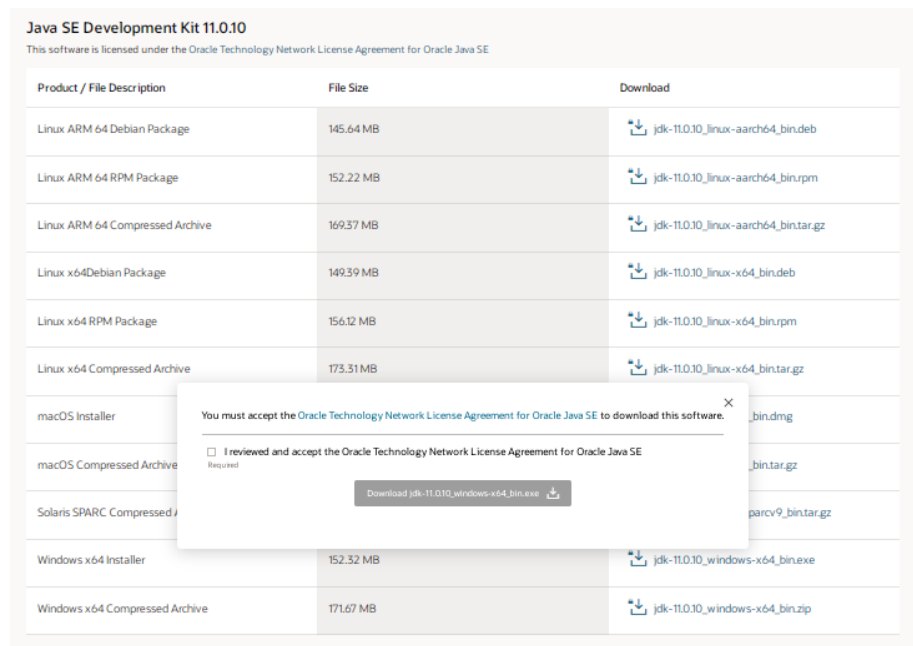


Figura D.2: Descarga JDK 11 Licencia

## Instalación de Eclipse

A continuación se instalará un entorno de desarrollo integrado(IDE) para Java, en este caso se ha utilizado **Eclipse IDE for Enterprise Java Developers** en la versión 2020-06.

Para descargar el IDE se accederá a la [página de descargas de Eclipse](#) y descargar la opción correspondiente a nuestro sistema operativo del **Eclipse Installer 2020-06 R**. Ver imagen [D.3](#).



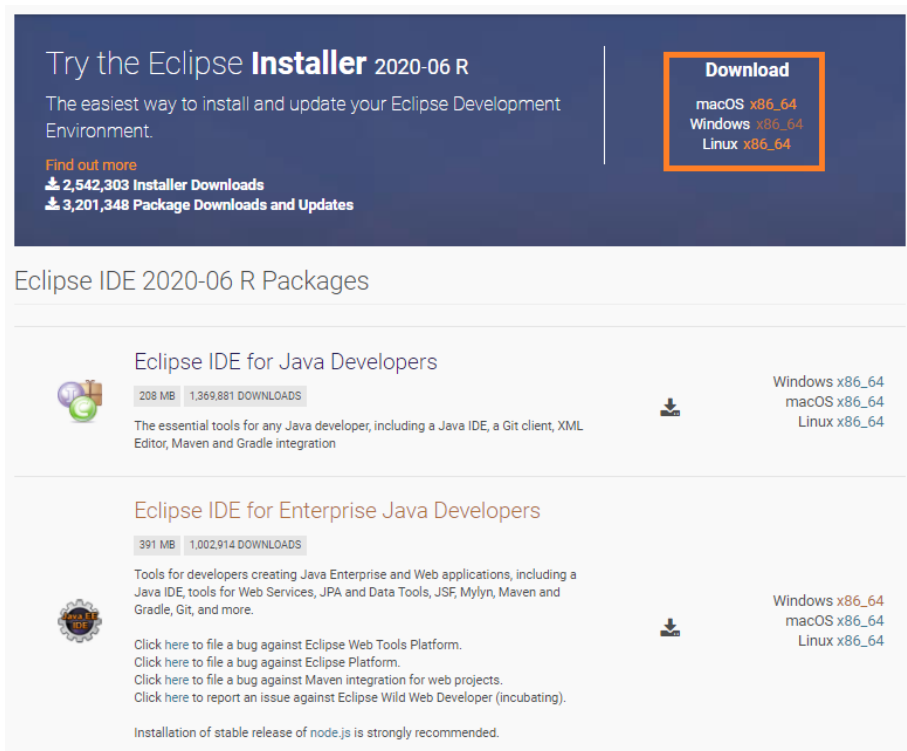


Figura D.3: Descargar IDE Eclipse

En el caso de los sistemas operativos Windows se descargará un archivo ejecutable que se deberá ejecutar como administrador. Una vez ejecutado se deberá seleccionar la opción “***Eclipse IDE for Enterprise Java Developers***”.

En el siguiente paso, en el apartado de “**Java 1.8 + VM**” se deberá seleccionar la carpeta donde se encuentra el JDK 8, instalado anteriormente.

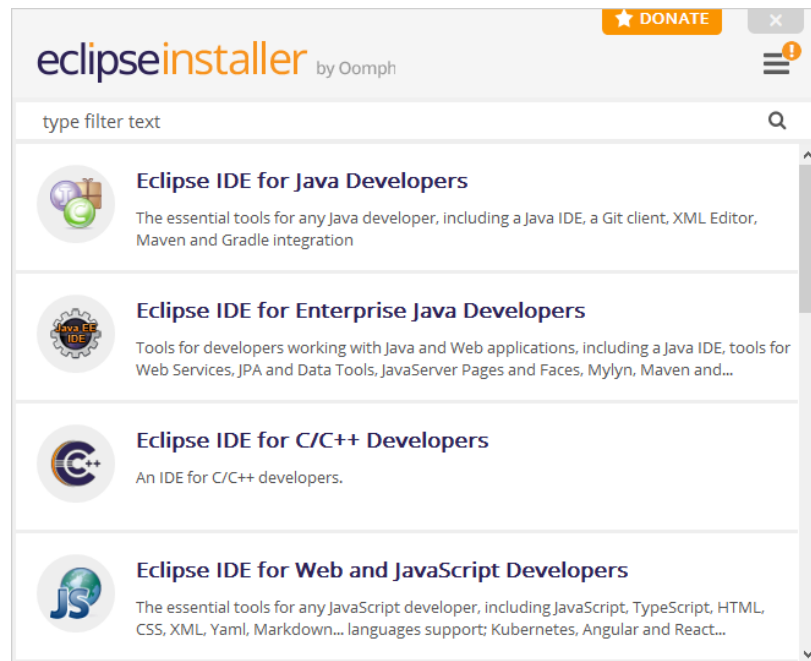


Figura D.4: Seleccionar Eclipse

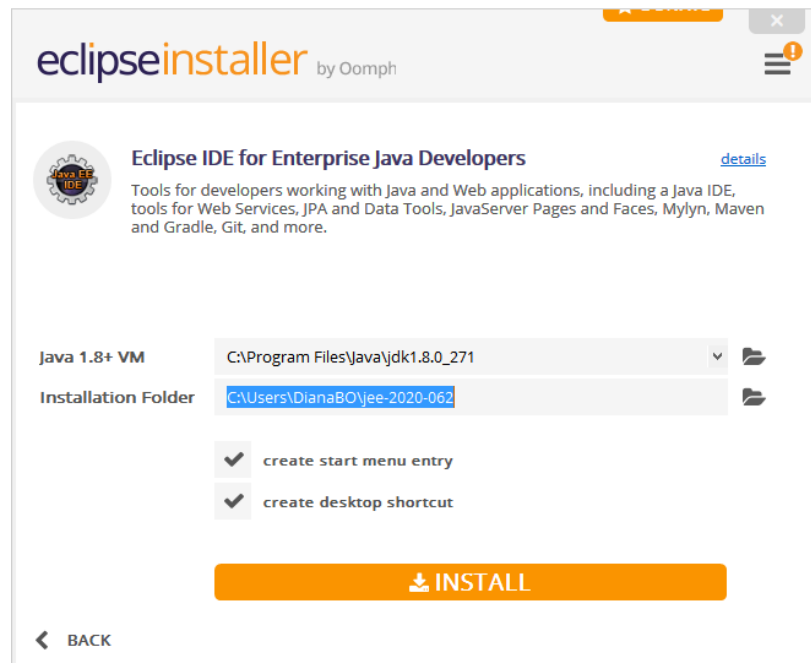


Figura D.5: Seleccionar JDK que usará el IDE

## Instalación del *plugin de Vaadin* para Eclipse

Una vez se haya instalado Eclipse, se procederá a añadir el plugin de Vaadin para Eclipse. Esto se realizará mediante el **Eclipse Marketplace de Eclipse**, el cual se encuentra en la opción de “**Help/Eclipse Marketplace...**” de la barra de herramientas.

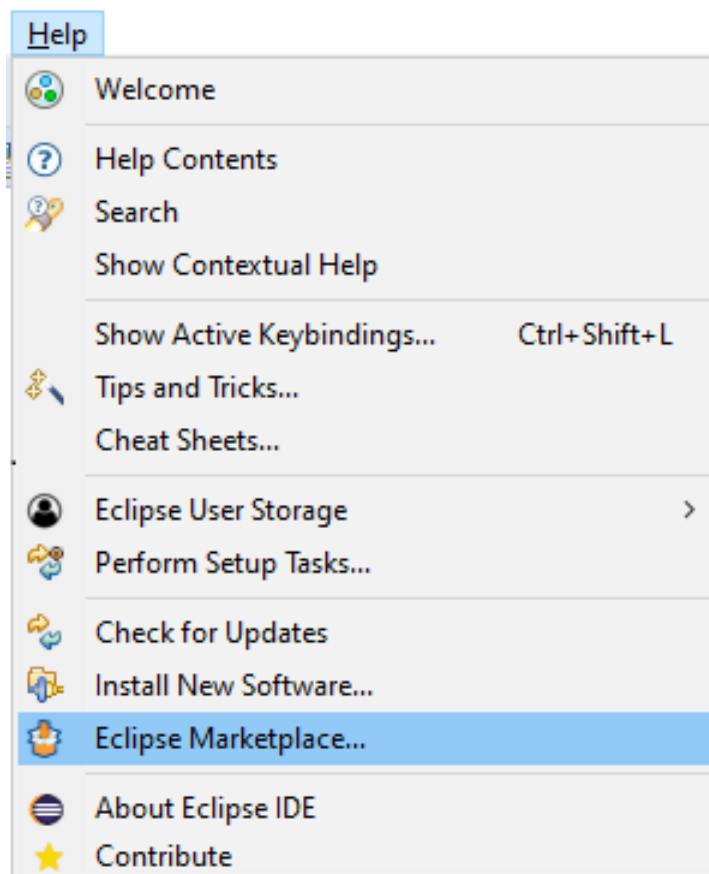


Figura D.6: Eclipse marketplace

Una vez en el Eclipse Marketplace, se buscará “**Vaadin**” y se pulsará “**Go**”. Tras salir el plugin “***Vaadin Plugin for Eclipse***”, se dará a “**Install**” y comenzará la instalación del plugin.

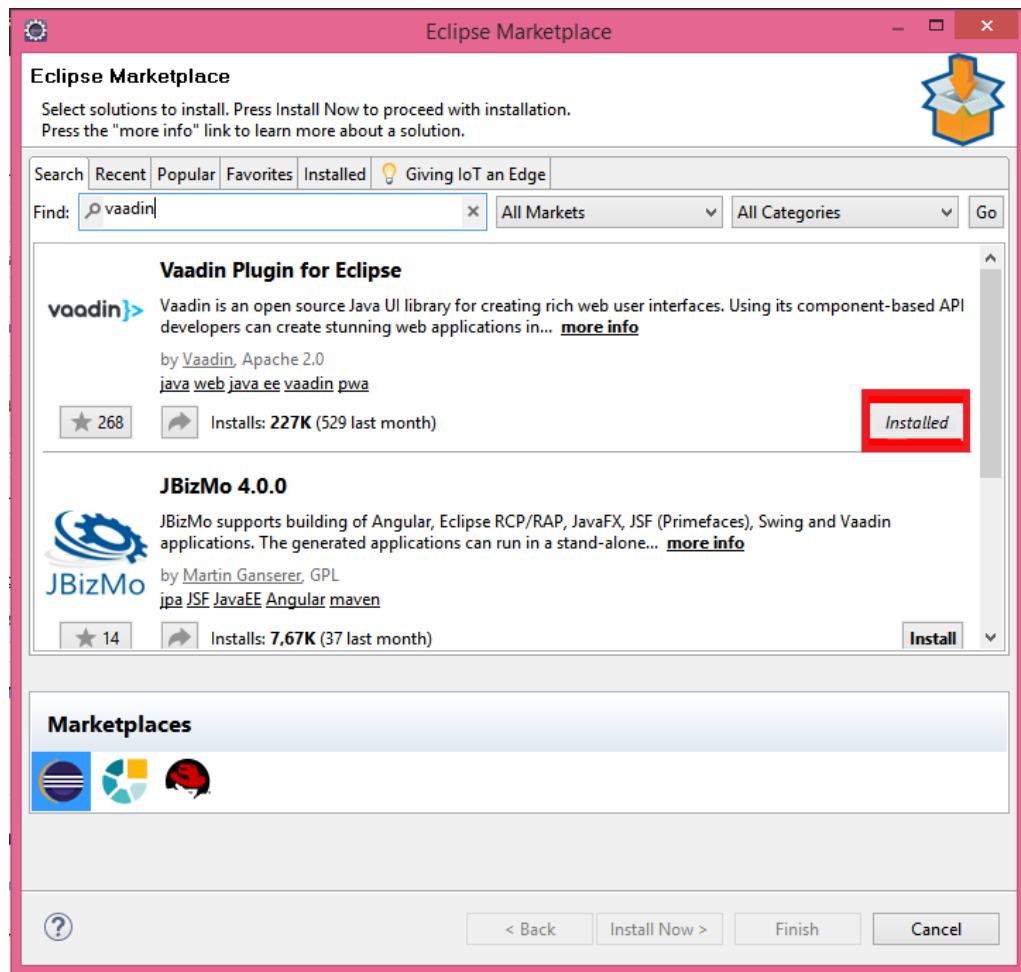


Figura D.7: Plugin Vaadin

## Instalación de Heroku CLI

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

Se explicará como compilar, instalar e ejecutar el proyecto. En el caso de la ejecución, se detallará como hacerlo con el terminal de Windows y mediante Eclipse (IDE).

## Descarga del repositorio

El código fuente se encuentra en el **repositorio del proyecto** en GitHub. Para descargarlo se deberá hacer click en “**Code**” y copiar la URL que aparece en el apartado de “**HTTP**”. Con esta URL deberemos ir al “**GitHub Desktop**” y clonar el repositorio.

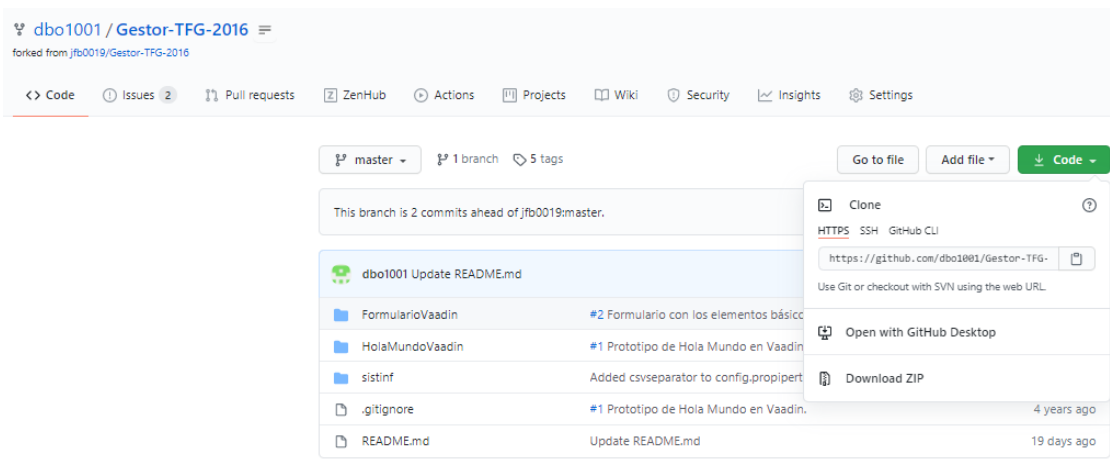


Figura D.8: Copiar URL repositorio

Si se desea tener código en local se deberá descargar el zip “**Download ZIP**” en la opción “**Code**” anteriormente mencionada. Una vez descargado el zip se descomprimirá y abrirá con Eclipse.

Para abrir el proyecto con Eclipse se seleccionará en la barra de herramientas **File/Import...** Aparecerá una ventana en la que se optará por la opción “**Projects from Folder or Archive**” y se hará click en “**Next**”.

Después se hará click en “**Directory...**” y se seleccionará la carpeta del proyecto con nombre “**sistinf**” y terminaremos la importación con “**Finish**”.

## Ejecución del proyecto

## Despliegue del proyecto

## Pruebas del sistema



## *Apéndice E*

---

# Documentación de usuario

---

### **E.1. Introducción**

### **E.2. Requisitos de usuarios**

### **E.3. Instalación**

No se requiere ninguna instalación por parte del usuario, simplemente puede acceder a la aplicación web a través del enlace .

### **E.4. Manual del usuario**





---

## **Bibliografía**

---