UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

UBULog-1.1 Documentación Técnica

Presentado por Oscar Fernández Armengol
en Universidad de Burgos — 4 de enero de 2018
Tutor: Raúl Marticorena Sánchez

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este documento hablaremos de la planificación del proyecto, que es una parte muy importante. Aquí analizaremos el tiempo de desarrollo y el presupuesto que se necesitara para llevarlo a cabo.

lo dividiremos en dos partes:

- Planificación temporal del proyecto.
- Estudio de viabilidad del proyecto.

En la planificación temporal hablaremos de cada uno de los *sprints* necesarios para su realización, el tiempo estimado y el tiempo real de realización.

En el estudio de viabilidad hablaremos de los costes y beneficios que nos aporta esta aplicación y su viabilidad legal, que se refiere a las licencias de uso del código ajeno que hemos utilizado en el desarrollo para poder comercializar nuestra aplicación.

A.2. Planificación temporal

Al inicio del proyecto se propuso utilizar una metodología ágil, en concreto Scrum, ya que se decidió tener reuniones semanales para hablar de los cambios realizados, problemas ocasionados y planificación del siguiente Sprint que se realizara. No se ha conseguido desarrollar la metodología al 100 % ya que el

el equipo de desarrollo constaba de 1 persona (Oscar Fernández Armengol), pero desechando este punto, se ha conseguido un desarrollo ágil en sus demás puntos.

- Se aplicó una estrategia de desarrollo incremental a través de iteraciones (*sprints*) y revisiones.
- La duración media de los *sprints* fue de una semana.
- Al finalizar cada *sprint* se entregaba un producto funcional con la nueva especificación en el caso de que estuviera terminada.
- Se realizaban reuniones de revisión al finalizar cada *sprint*, de resolución de dudas y al mismo tiempo de planificación del nuevo *sprint*.
- En la planificación del *sprint* se generaba una lista de tareas a realizar (nuevas funcionalidades o bugs a solucionar).
- Se estimaba el tiempo de realización de las tareas a realizar en el *canvas*.
- Para monitorizar el progreso del proyecto se utilizan los gráficos generados en github.

Sprint 1 (03/10/17 - 10/10/17)

Este *sprint* fue el comienzo del proyecto, aunque en reuniones previas se hablo con el tutor de las propuestas que tenia para la elección del proyecto, una vez el tutor (Raúl Marticorena Sanchez) acepto tutorizar al alumno (Oscar Fernández Armengol) se puedo empezar el desarrollo.

Los objetivos fueron:

- Preparación del entorno de desarrollo.
- Familiarización con la aplicación heredada UBUGrades.
- Investigación del web service de moodle
- Creación de un esqueleto del proyecto para poder empezar a trabajar.

El Sprint 1 se estimo en 6 días de trabajo y se realizo en esos 6 días.

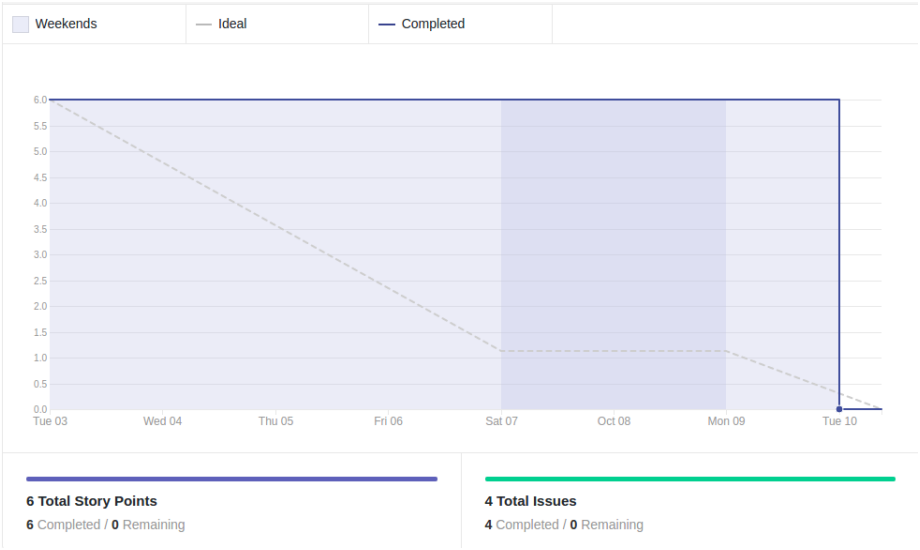

Figura A.1: Sprint 1.

Sprint 2 (10/10/17 - 17/10/17)

Los objetivos fueron:

- Análisis de modelo de datos.
- Diagrama del modelo de datos.
- Parseo de documentos csv.

El Sprint 2 se estimo en 6 días de trabajo y se realizo en esos 6 días.

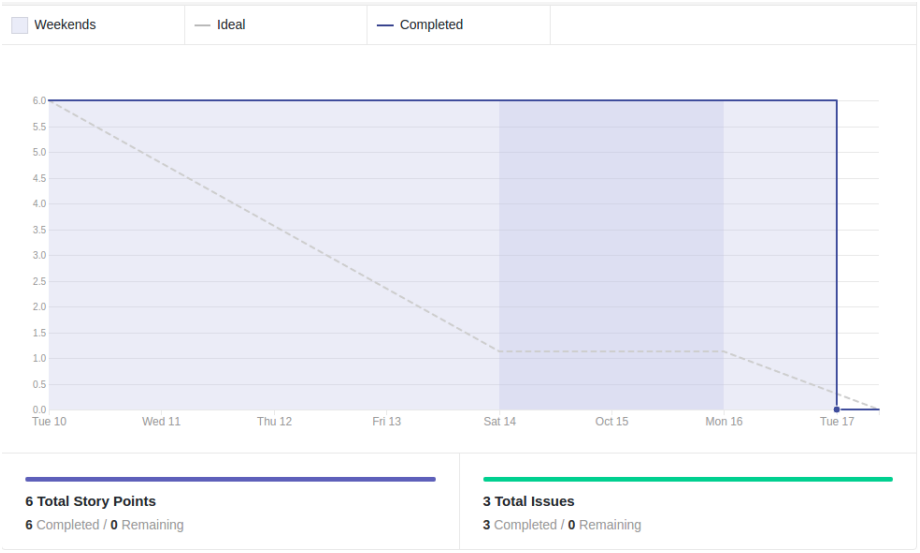

Figura A.2: Sprint 2.

Sprint 3 (17/10/17 - 24/10/17)

Los objetivos fueron:

- Integración continua con travis CI.
- Integración de sonarqube.
- Recoger token con web service.
- Recoger las asignaturas en las que imparte como profesor el usuario logeado.
- Referencia al usuario con su id en el parseo.
- Interfaz de usuario, primeros pasos.
- Limpieza del carpetas innecesarias en el repositorio.

En este sprint, cabe destacar, la integración de Travis[4] y SonarQube [2] para optimizar el tiempo de desarrollo, ya que delegamos en estas herramientas las pruebas y el análisis del código.

El Sprint 3 se estimo en 7 días de trabajo y se realizo en esos 6 días.

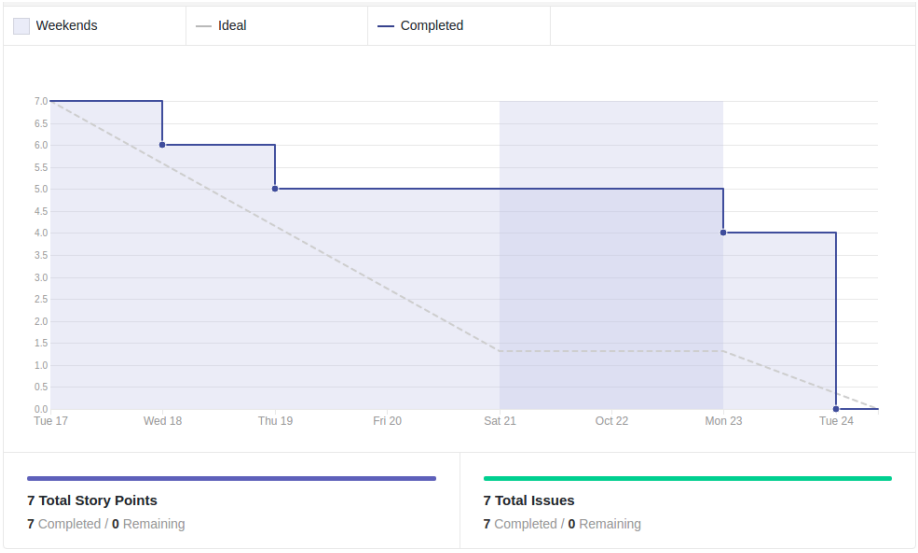

Figura A.3: Sprint 3.

Sprint 4 (24/10/17 - 7/11/17)

Los objetivos fueron:

- Creación de usuarios ficticios.
- Análisis de datos.
- Eliminación de calificaciones.
- Gestión de errores del documento.

En este sprint, cabe destacar, la creación de usuarios ficticios, ya que hay usuarios que tienen interacciones en las asignaturas y nos interesa saber que están haciendo.

El Sprint 4 se estimo en 5 días de trabajo y se realizo en esos 6 días.

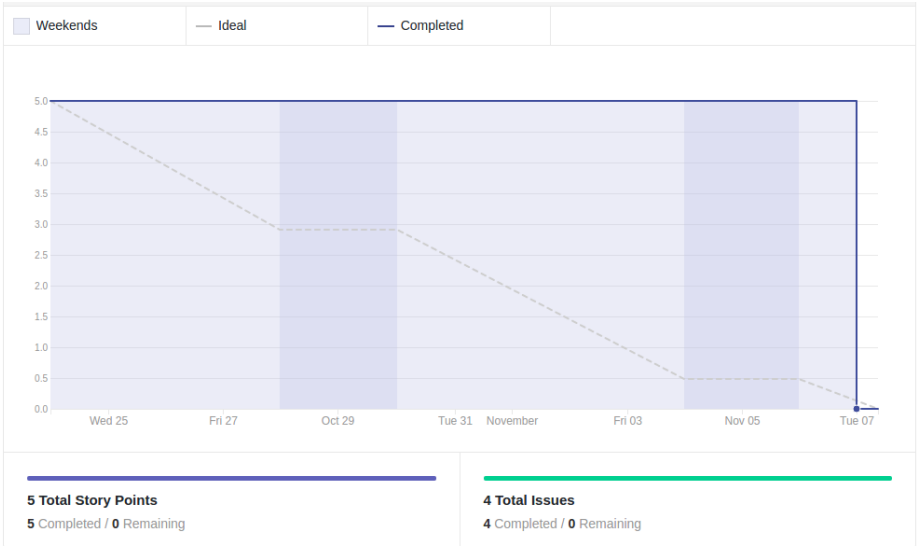

Figura A.4: Sprint 4.

Sprint 5 (7/11/17 - 14/11/17)

Los objetivos fueron:

- Cambio de árbol de actividades.
- Construir los filtros de la interfaz.
- Creación de eventos
- Interfaz nuevo diseño.

En este sprint, cabe destacar, el nuevo diseño para la ventana principal, la antigua no era re-dimensionable y como había que implementar nuevos filtros y botones mas adelante, se opto por rehacerla.

El Sprint 5 se estimo en 7 días de trabajo y se realizo en esos 6 días.

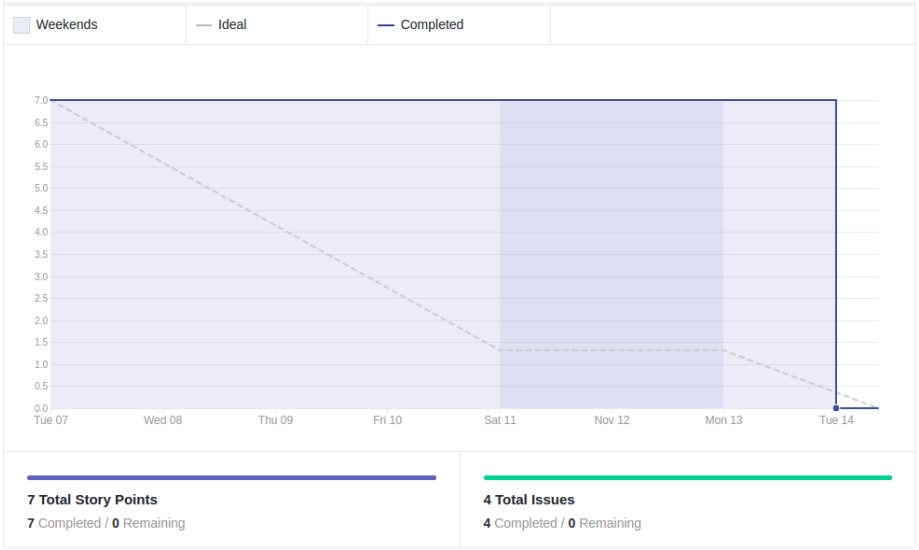

Figura A.5: Sprint 5.

Sprint 6 (14/11/17 - 22/11/17)

Los objetivos fueron:

- Generar gráficas.
- Mostrar gráficas con char.js [3]
- Creación de botón para generar las gráficas.
- Instalar sublime text para la edición de html y javascript.
- Contar los logs para las gráficas.

En este sprint, cabe destacar, la utilización de la librería chart para la generación de gráficos. Como el gráfico es cambiante dependiendo de los filtros pulsados, optamos en generar los html y javaScript en tiempo de ejecución.

El Sprint 6 se estimo en 8 días de trabajo y se realizo en esos 7 días.

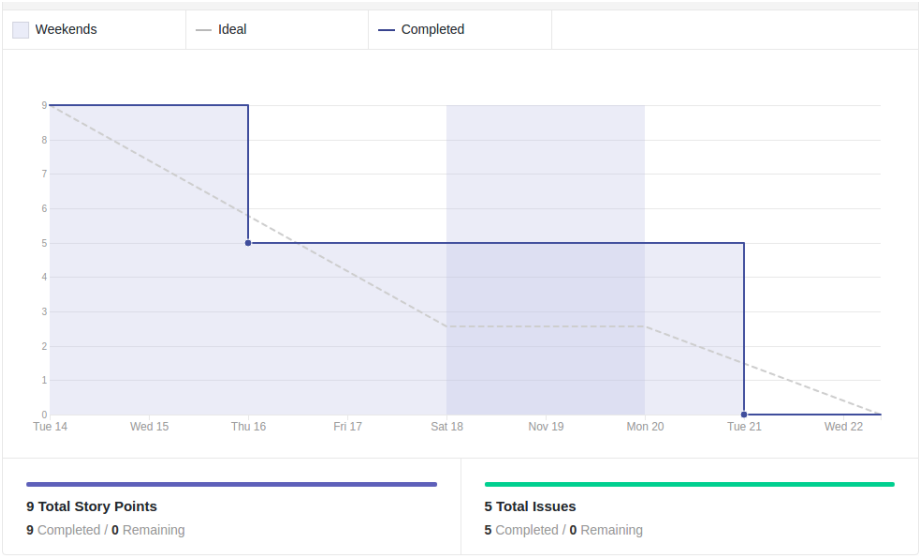

Figura A.6: Sprint 6.

Sprint 7 (22/11/17 - 29/11/17)

Los objetivos fueron:

- Integración de gráficas.
- Ordenar selección de eventos.
- imagen de usuario logeado.
- UTF-8 para el gráfico.

El Sprint 7 se estimo en 8 días de trabajo y se realizo en esos 5 días.

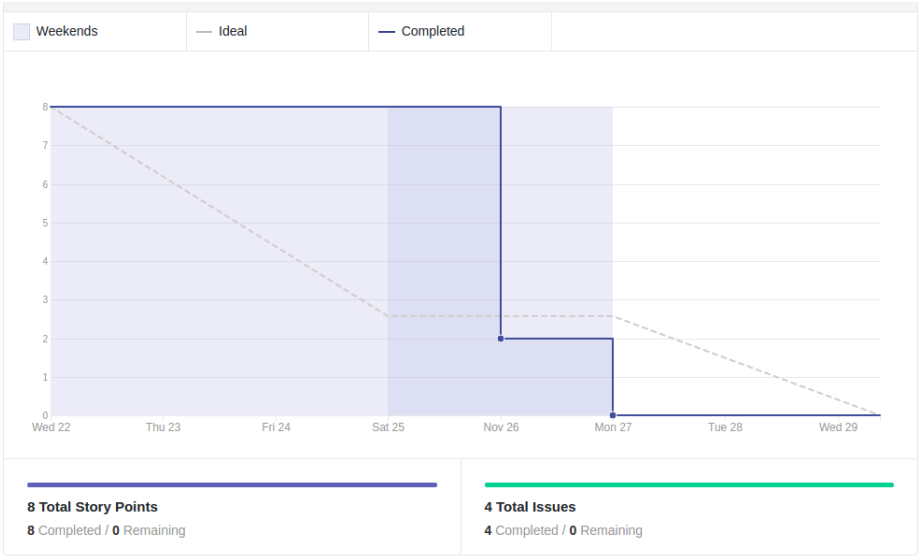

Figura A.7: Sprint 7.

Sprint 8 (28/11/17 - 06/12/17)

Los objetivos fueron:

- Cambio de parseo de log.
- Creación de tablas para los log en JavaScript.
- Muestreo de tabla en aplicación.
- Implementación de filtros en tabla
- Revisión de logs.
- Cambio de logo.

En este sprint, cabe destacar, el cambio en el parseo, dado que había incongruencias entre los log de pruebas y los reales, para evitar eso, se implementa con la librería common-csv [1], con ella cogemos los datos correspondientes a la columna concreta y nos evitamos que en las pruebas el usuario sea .ªlumno apellido1 apellido2z en producción sea .ªpellido1 apellido2, alumno".

Por otra parte, también es interesante la implementación de las tablas, donde el usuario de la aplicación puede ver los log filtrados y desgranar aun mas los datos con filtros adicionales.

El Sprint 8 se estimo en 10 días de trabajo y se realizo en esos 6 días.

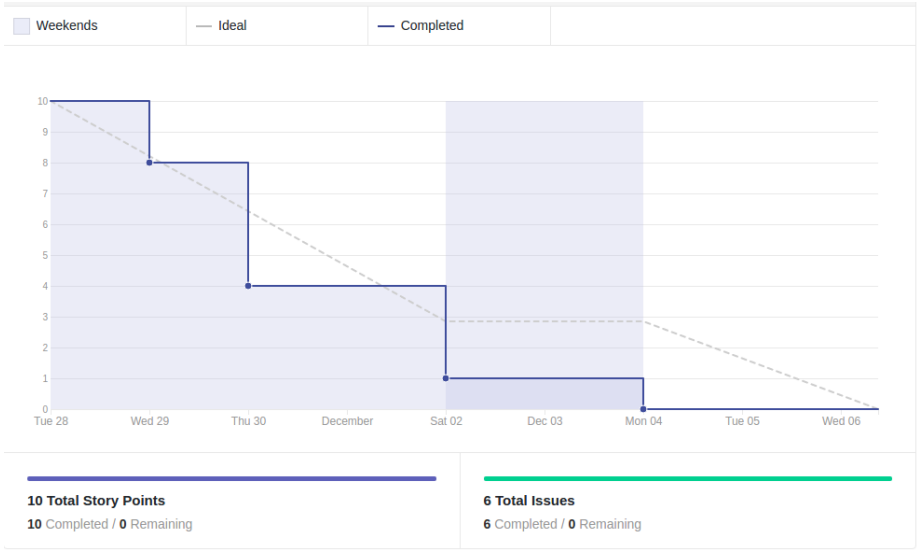

Figura A.8: Sprint 8.

Sprint 9 (06/12/17 - 13/12/17)

Los objetivos fueron:

- Sacar filtros del html a nuestra interfaz.
- Implementar funcionalidad filtros de tabla de logs y cargar la nueva gráfica con los log resultantes.
- Investigar la utilización de web scraping para nuestra aplicación.
- Crear botones en la interfaz para coger el csv de forma automática.
- Implementar funcionalidad de botón documento online.

En este sprint, cabe destacar, la investigación del web scraping, se estimo que se haría en 5 días mas 3 dias de implementación, en una mañana, se pudieron probar las librerías Jsoup, Selenium y htmlUnit. Las dos primeras se descartaron, Selenium por problemas de ubicación de exploradores porque no podemos saber en donde el usuario los tendrá instalados y lo descartamos y Jsoup solo parsea la web, no nos deja interaccionar con ella. Con la librería htmlUnt hicimos pruebas y conseguimos traernos los datos correspondientes, en la misma mañana pudimos hacer el trabajo que esperábamos hacer en 8 días.

El Sprint 9 se estimo en 15 días de trabajo y se realizo en esos 3 días.

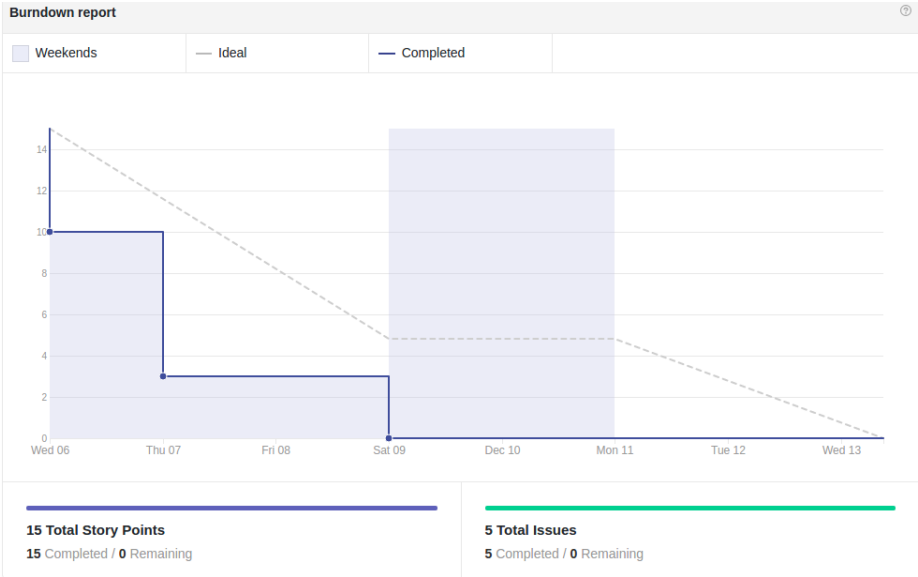

Figura A.9: Sprint 9.

Sprint 10 (13/12/17 - 20/12/17)

Los objetivos fueron:

- WebScripting funcional y refactorizado.
- Mostrar fecha y hora en la tabla de logs.
- Centrar botones inferiores.
- Descartar meses generados por la gráfica que no tengan logs.
- Modal de carga para la lectura de logs.
- Añadir tipos de gráficas.
- Arreglo de la muestra de fechas de la tabla log.
- Aumentar TimeOut de WebScraping.

El Sprint 10 se estimo en 12 días de trabajo y se realizo en esos 5 días.

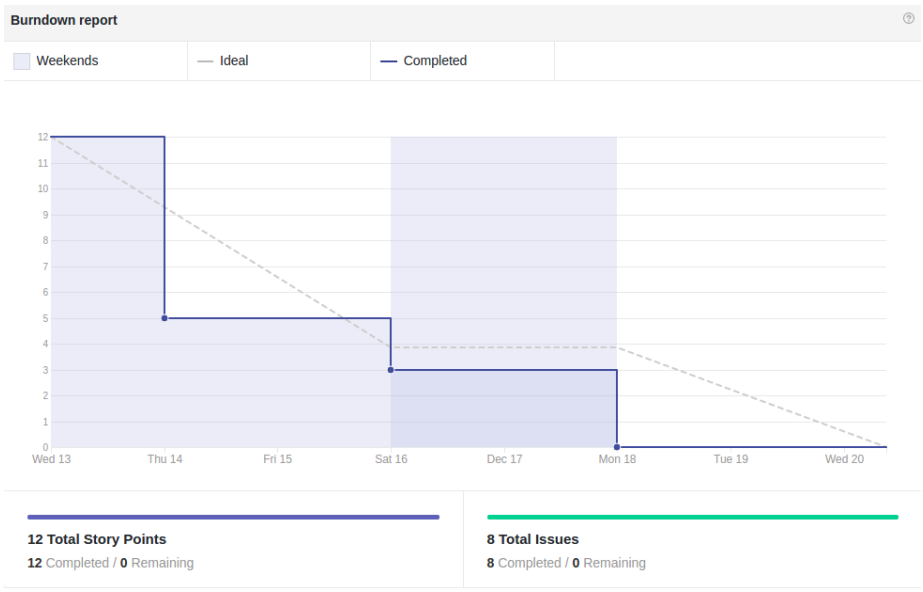

Figura A.10: Sprint 10.

Sprint 11 (20/12/17 - 12/01/18)

Los objetivos fueron:

- Documentación de código.
- Documentación de memoria.
- Refactorización.
- Arreglo de bugs para la entrega.

Este es el ultimo spring del proyecto, y el mas largo, se completara la memoria, anexos, documentación de código y se arreglaran bugs que se han observado durante el desarrollo y el testeo manual.

El Sprint 10

FALTA IMASLKEFÑSJFDÑASJF

A.3. Estudio de viabilidad

Podemos desglosar los costes del proyecto de la siguiente manera.

Viabilidad económica

En el desglose podemos diferenciarlo en las siguientes categorías.

Costes de personal

El proyecto se lleva a cabo por un programador a media jornada ya que esta de practicas en otra empresa 25H/S, empezando en Octubre del 2017 asta Enero del 2018.

Concepto	Coste
Salario mensual bruto	1.300
IRPF (10 %)	130
S.S(29.9 %)	388.88
Salario programador	1.688.88
Total 3 meses y medio(20H/S)	2955.54

Tabla A.1: Coste personal

Coste hardware

El proyecto se realiza con el ordenador personal y se considera la amortización a 5 años y se utiliza durante 3 meses y medio.

Concepto	Coste	Coste amortizado
Portatil	1.000	4.86
Total	1000	4.86

Tabla A.2: Coste hardware

Beneficios

El proyecto se ha pensado para ayudar al profesorado a ver la interacción de los alumnos en la asignatura. Se puede considerar un coste de 5 euros por licencia y que en la universidad hay, redondeando, 700 profesores estamos hablando de 3500 euros, la aplicación en un futuro cambiara y llegara a la 2.0 con lo cual habría que volver a pagar la nueva licencia.

La aplicación se puede extender a diferentes centros aparte de la UBU. Si lo observamos en ese sentido se podría cobrear 500 euros por licencia, para su utilización en esa universidad. Si tenemos 83 universidades es España, podríamos obtener unos beneficios de 83 * 500 = 41.500 euros.

Viabilidad legal

En esta sección hablaremos de la viabilidad del software utilizado, de si las licencias nos permiten la explotación del mismo o su propia utilización.

Herramientas	Licencia
commons-codec1.9	Apache License Version 2.0
commons-csv-1.5	Apache License Version 2.0
commons-logging-1.2	Apache License Version 2.0
gson-2.8.0	Apache License Version 2.0
htmlunit-2.28-OSGi	Apache License Version 2.0
httpasyncclient-4.1.2	Apache License Version 2.0
httpasyncclient-cache-4.1.2	Apache License Version 2.0
httpclient-4.5.2	Apache License Version 2.0
httpclient-cache-4.5.2	Apache License Version 2.0
httpcore-4.4.5	Apache License Version 2.0
httpcore-nio-4.4.5	Apache License Version 2.0
java-json	Json license
log4j-1.2.17	Apache License Version 2.0
slf4j-api-1.7.1	Apache License Version 2.0
httpclient-cache-4.5.2	Apache License Version 2.0
slf4j-log4j12-1.7.1	Apache License Version 2.0
UBULog 1.0	Eclipse Public License - v 1.0

Tabla A.3: Viabilidad legal

Para la aplicación desarrollada y para lo que esta pensado, por las licencias que se están utilizando no hay ningún problema para su utilización, mientras compartamos todo el código utilizado.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este proyecto esta orientado a profesores de la universidad de Burgos y en este apartado vamos ha hablar de los objetivos generales, de la especificación de requisitos(funcionales y no funcionales) y y especificaremos cada uno de ellos.

B.2. Objetivos generales

En el desarrollo del proyecto se quieren conseguir los siguientes objetivos:

- Desarrollar una aplicación de escritorio para poder logearse con ubuvirtual.
- Obtención de datos por el web service de moodle.
- Visualizar los datos correspondientes a la asignatura seleccionada.
- Facilitar la interpretación de los datos representándolo por diferentes tipos de gráficas.
- Desgranar los datos aportados para un análisis mas profundo de ellos.

B.3. Catalogo de requisitos

Ahora hablaremos de los requisitos funcionales y no funcionales, derivados de los objetivos generales de este proyecto. Se notara que varios puntos del apartado son iguales al anexo del tfg de Claudia Martínez Herrero [5]

Requisitos funcionales

- **RF-1 Autenticación de usuario:** Inicialmente el usuario debe identificarse con su correo de profesor y su contraseña para poder hacer uso de las funciones de la aplicación.
- **RF-2 Extracción de datos:** A partir de los datos de usuario, se obtendrán todos los datos referidos a los cursos en los que esté matriculado.
- **RF-3 Carga de datos:** El usuario debe poder cargar los datos correspondientes al log de la asignatura.
- **RF-4 Selección de datos:** El usuario debe poder seleccionar qué participantes de esa asignatura y en qué eventos quiere que se muestre el gráfico.
    - **RF-4.1 Filtrado de participantes y eventos:** Se podrá filtrar a los participantes según los grupos, el rol y por un campo de texto y los eventos por un campo de texto.
- **RF-5 Visualización de gráficas de logs:** El usuario podrá visualizar las registros en forma de gráfica.
- **RF-6 Visualización de Logs:** El usuario podrá visualizar las registros de manera gráfica en formato de tabla.
    - **RF-6.1 Filtrado de logs:** Se podrá filtrar logs por varios campo de texto cada uno asociado a una columna de la tabla.
- **RF-7 Cierre de sesión:** La aplicación cierra sesión.

Requisitos no funcionales

- **RNF-1** La aplicación debe ser de escritorio.
- **RNF-2** La aplicación debe funcionar en un ordenador con Java (versión 8)
- **RNF-3** La aplicación debe funcionar contra plataformas de Moodle 3.3 como es UBUVirtual.
- **RNF-4** La interfaz de usuario debe ser clara e intuitiva para el usuario.
- **RNF-5** La aplicación debe ser sencilla de utilizar, ahorrando pasos innecesarios que puedan resultar confusos para el usuario.
- **RNF-6** La aplicación será extensible.
- **RNF-7** La sección de gráficos debe permitir exportar los gráficos en formato de imagen: .png, .jpg... .

B.4. Especificación de requisitos

En esta sección se mostrará el diagrama de casos de uso resultante y se desarrollará cada uno de ellos.

Diagrama de casos de uso

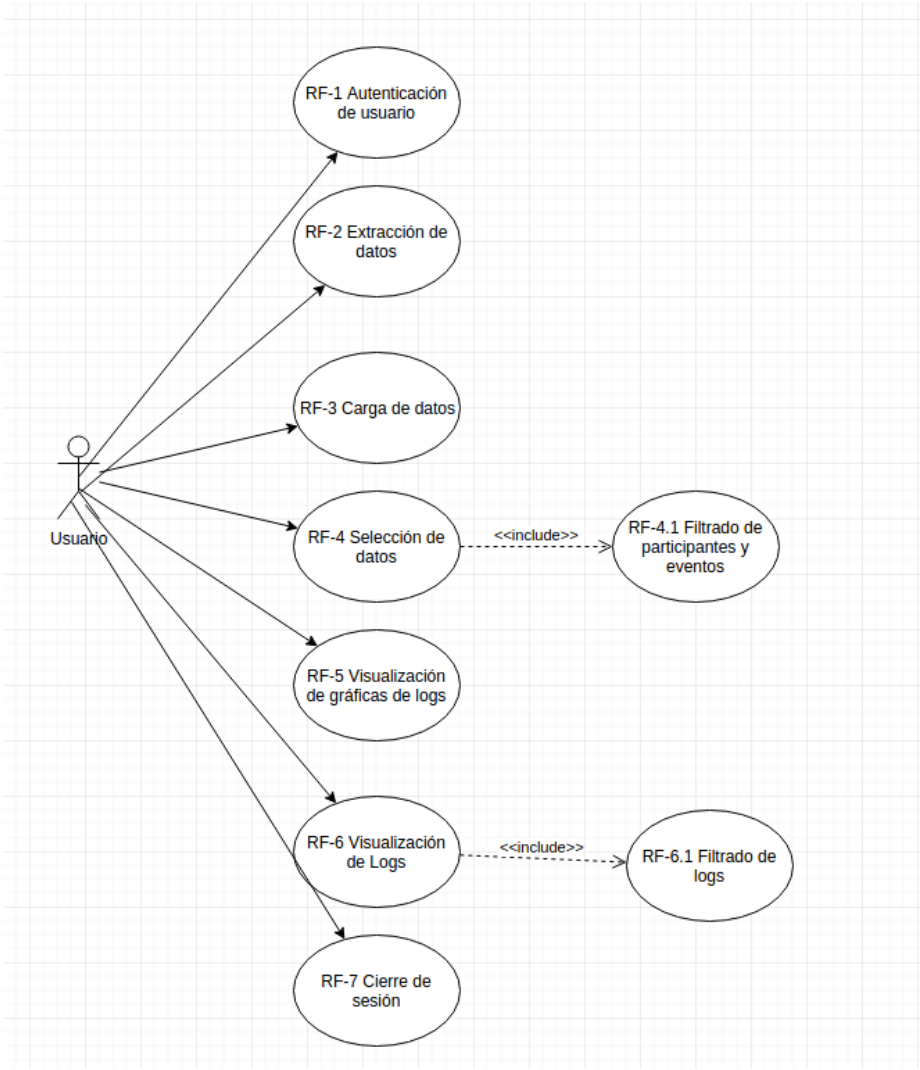

Figura B.1: Diagrama de casos de uso.

Actores

Solo interactuará con el sistema un actor, que se corresponderá con la figura del usuario.

Casos de uso

RF1	Autenticación de usuario
Descripción	Se introduce e-mail, contraseña y host (UBUVirtual).
Requisitos asociados	RF2.
Precondiciones	El usuario debe tener utilizar el e-mail de UBUVirtual.
Secuencia normal	1- Introducir los 3 campos del login. 2- Entrar en la aplicación.
Postcondiciones	Datos de usuario cargados. Se accederá a la pantalla de bienvenida.
Frecuencia	Alta.
Importancia	Alta.

Tabla B.1: RF1 - Autenticación de usuario

RF2	Extracción de datos
Descripción	El sistema carga los cursos en los que está matriculado el usuaio.
Requisitos asociados	RF1.
Precondiciones	El usuario se ha logueado correctamente.
Secuencia normal	1- Obtención de los cursos del usuario. 2- Carga de la lista de cursos. 3- Selección de curso por parte del usuario.
Postcondiciones	Se visualiza una nueva pantalla con el curso elegido.
Frecuencia	Alta.
Importancia	Alta.

Tabla B.2: RF2 - Extracción de datos

RF3	Extracción de datos
Descripción	El usuario cargara el fichero csv o ejecutara la descarga del log.
Requisitos asociados	RF1.
Precondiciones	1- El usuario se ha logueado correctamente. 2- El usuario selecciona una asignatura.
Secuencia normal	SI tiene csv: Carga documento csv. SI NO tiene csv: puede descargarlo manualmente o clickando en descarga automatizada.
Postcondiciones	Se cargan los datos de log en la aplicación.
Frecuencia	Media.
Importancia	Alta.

Tabla B.3: RF3 - Carga de datos

RF4	Selección de datos
Descripción	Se visualiza una pantalla con la lista de participantes del curso y la lista de eventos. El usuario debe seleccionar qué participantes que eventos o ambos desea visualizar.
Requisitos asociados	RF3, RF4.1.
Precondiciones	1- El log debe de estar cargado. 2- Se debe mostrar participantes y eventos disponibles. 3- Se debe clickar en participantes o eventos o ambos.
Secuencia normal	1- Se carga el log. 2- Se muestran los participantes y eventos disponibles. 3- Se hace click en participantes o eventos o ambos.
Postcondiciones	Se cargan los datos de log en la aplicación.
Frecuencia	Alta.
Importancia	Alta.

Tabla B.4: RF4 - Selección de datos

RF4	Filtrado de participantes y eventos
Descripción	El usuario podrá filtrar los participantes y eventos del curso.
Requisitos asociados	RF4.
Precondiciones	1- El log debe de estar cargado. 2- Se debe mostrar participantes y eventos disponibles.
Secuencia normal	1- Se carga el log. 2- Se muestran los participantes y eventos disponibles. 3- Se puede interaccionar con los filtros.
Postcondiciones	Se muestran los datos que coinciden con los filtros.
Frecuencia	Alta.
Importancia	Alta.

Tabla B.5: RF4.1 - Filtrado de participantes y eventos

RF5	Visualización de gráficas de logs
Descripción	La aplicación mostrara los datos seleccionados de manera gráfica.
Requisitos asociados	RF3, RF4.
Precondiciones	1- El log debe de estar cargado. 2- Se debe mostrar participantes y eventos disponibles. 3- Se hace click en participantes o eventos o ambos.
Secuencia normal	1- Se carga el log. 2- Se muestran los participantes y eventos disponibles. 3- Se hace click en participantes o eventos o ambos.
Postcondiciones	Se muestran los datos en la gráfica.
Frecuencia	Alta.
Importancia	Alta.

Tabla B.6: RF5 - Visualización de gráficas de logs

RF6	Visualización de logs
Descripción	La aplicación mostrara los logs cargados en forma de tabla.
Requisitos asociados	RF3, RF4.
Precondiciones	1- El log debe de estar cargado. 2- Se debe mostrar participantes y eventos disponibles. 3- Se hace click en participantes o eventos o ambos.
Secuencia normal	1- Se carga el log. 2- Se muestran los participantes y eventos disponibles. 3- Se hace click en participantes o eventos o ambos.
Postcondiciones	Se muestran los datos en la tabla logs.
Frecuencia	Alta.
Importancia	Alta.

Tabla B.7: RF6 - Visualización de logs

RF6.1	Filtrado de logs
Descripción	El usuario podrá filtrar logs del curso.
Requisitos asociados	RF6.
Precondiciones	1- El log debe de estar cargado. 2- Se debe mostrar participantes y eventos disponibles.
Secuencia normal	1- Se carga el log. 2- Se muestran los participantes y eventos disponibles. 3- Se puede interaccionar con los filtros.
Postcondiciones	Se muestran los datos que coinciden con los filtros.
Frecuencia	Alta.
Importancia	Alta.

Tabla B.8: RF6.1 - Filtrado de logs

RF7	Cierre de sesión de usuario
Descripción	Mediante un botón de salir el usuario podrá cerrar la aplicación.
Requisitos asociados	Ninguno.
Precondiciones	Sesión de usuario activa.
Secuencia normal	1- El usuario pulsa el botón de salir. 2- El sistema cierra la aplicación.
Postcondiciones	Aplicación cerrada.
Frecuencia	Media.
Importancia	Media.

Tabla B.9: RF7 - Cierre de sesión de usuario

Apéndice C

Especificación de diseño

C.1. Introducción

En esta parte de los anexos vamos a explicar el diseño de dato, diseño procedimental y diseño arquitectónico de nuestro proyecto.

C.2. Diseño de datos

Nuestro modelo de datos se corresponde con lo siguiente:

- **Chart**: tiene un tipo de gráfico, agrupación de fechas por meses de los log, y los valores de las coincidencias entre ellos. Se encarga de la construcción del gráfico en la aplicación
- **Course**: tiene un id, nombre, apellido, usuarios matriculados, grupos y roles asociados y tipo de actividad.
- **EnrolledUser**: tiene un id, nombre, apellido, nombre completo, primer acceso, ultimo acceso, descripción, ciudad, país, imagen pequeña, imagen grande, usuarios matriculados y grupos, roles y cursos asociados.
- **Event**: tiene nombre y logs asociados al evento.
- **Group**: tiene id , nombre y descripción.
- **Log**: tiene fecha , nombre del usuario que realiza la acción, nombre del usuario afectado, contexto, componente, evento, origen, ip y descripción.
- **MoodleUser**: tiene un id, nombre, apellido, e-mail, departamento, primer acceso, ultimo acceso, autenticación, descripción, imagen pequeña, imagen grande, token y cursos asociados.

- **Role**: tiene id , nombre.
- **TableLog**: no tiene atributos.

C.3. Diseño arquitectónico

Diseño de paquetes

El código de la aplicación esta organizado de la siguiente manera:

- **Paquetes java**: Se ha generado diferentes paquetes para el mantenimiento del código.
  - **Controllers**: Controladores de las vistas.

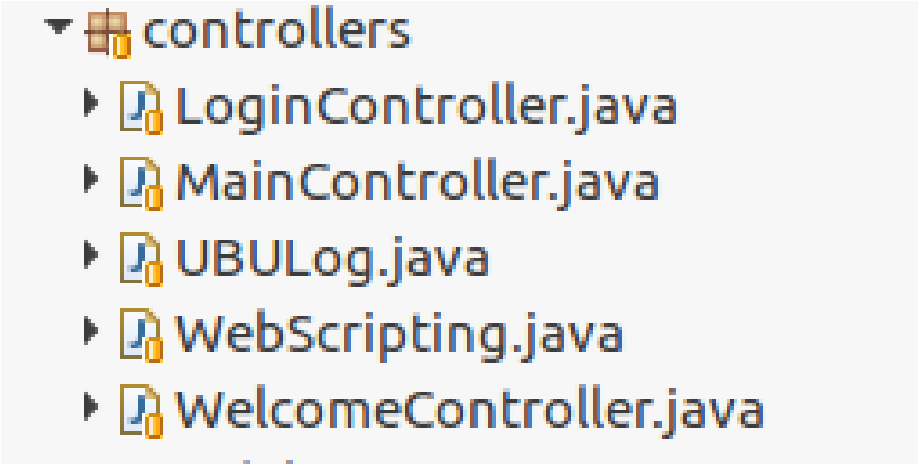

Figura C.1: Paquete controllers.

  - **Model**: modelo de la aplicación.

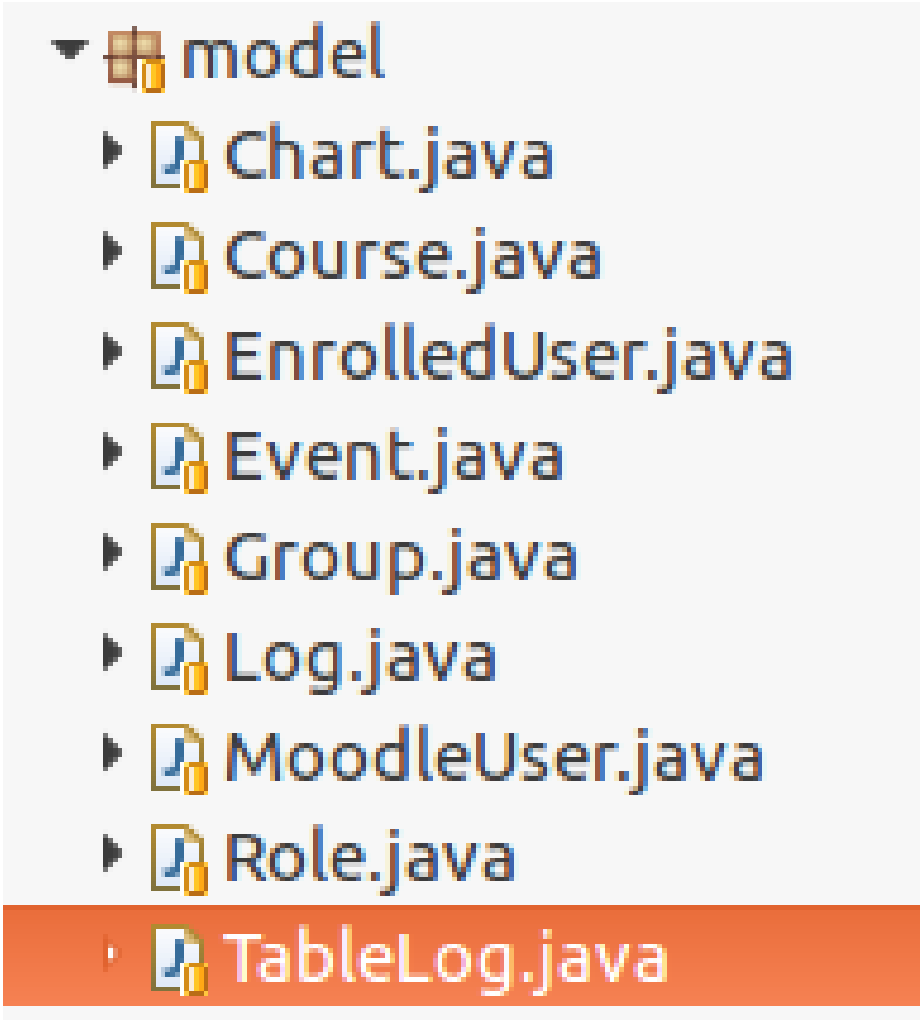

Figura C.2: Paquete model.

◇ **Parserdocument**: paquete para documentos.

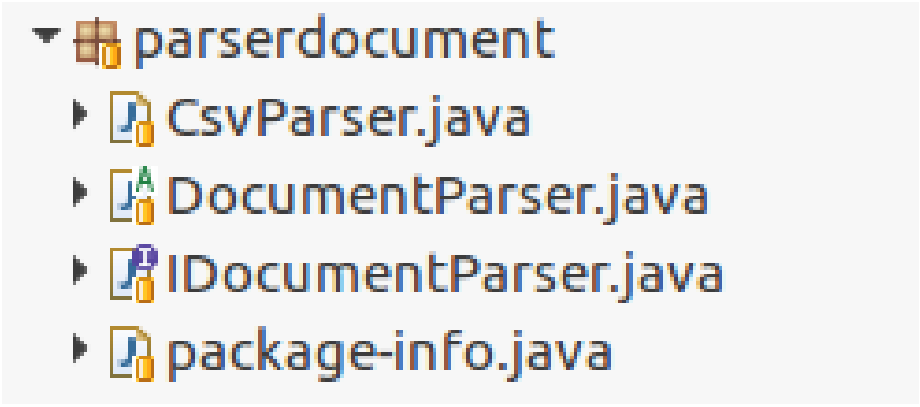

Figura C.3: Paquete parserdocument.

◇ **Ubulogexception**: Gestiona las excepciones.

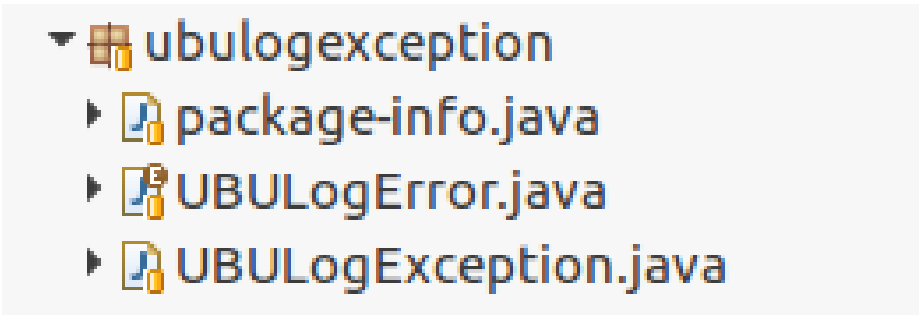

Figura C.4: Paquete ubulogexception.

◇ **Webservice**: servicios web para la obtención de datos en moodle.

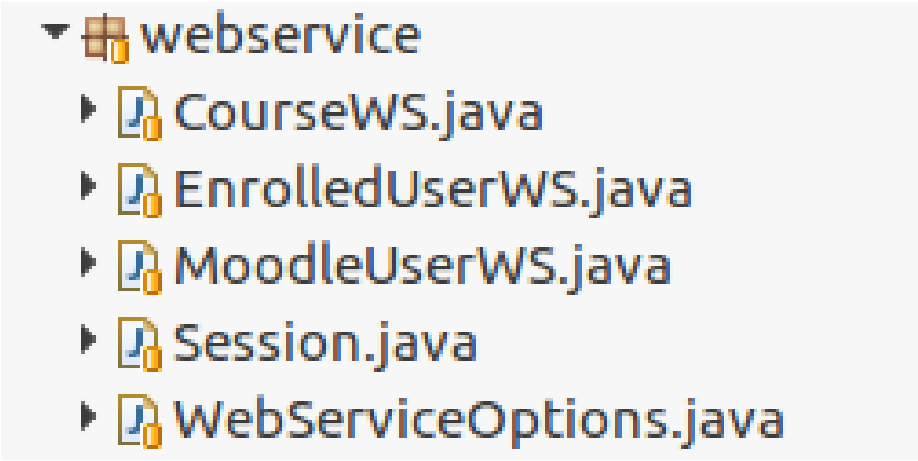

Figura C.5: Paquete webservice.

- **resources**: Los recursos que utilizaremos para el funcionamiento de nuestra aplicación.
  - **css**: Hoja de estilos en cascada (style.css) para la interfaz.
  - **view**: Archivos .fxml para las vista de la aplicación con java fx.
  - **img**: Imágenes de la aplicación.

Diseño de Clases

diagrama de Clases

- **Diagrama completo de clases**:

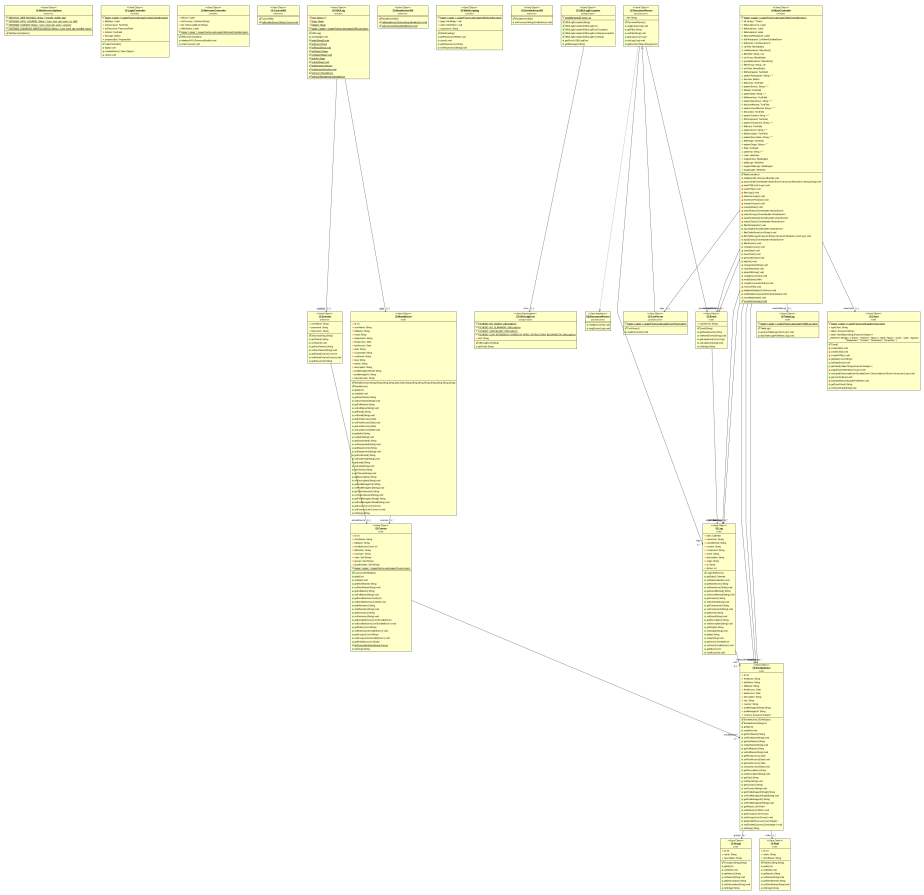

Figura C.6: Diagrama de clases.

diagrama de Paquetes

En esta subsección hacemos referencia a los anexos de Claudia Martínez Herrero. [5]

Modelo vista controlador

En esta subsección hacemos referencia a los anexos de Claudia Martínez Herrero. [5] añadiendo nueva formación a la tabla.

	Vista	Modelo	Controlador
RF1 Autenticación de usuario	Login.fxml	MoodleUser.java	Session.java LoginController.java MoodleUserWS.java MoodleOptions.java
RF2 Extracción de datos	NewMain.fxml Welcome.fxml	Course.java	WelcomeController.java MainController.java MoodleUserWS.java MoodleOptions.java
RF3 Carga de datos	NewMain.fxml	Log.java	MoodleUserWS.java MoodleOptions.java
RF4 Selección de datos	NewMain.fxml	Group.java Role.java Log.java EnrolledUser.java	MainController.java MoodleUserWS.java MoodleOptions.java
RF5 Visualización de gráficas de log	NewMain.fxml	EnrolledUser.java	MainController.java
RF6 Visualización de log	NewMain.fxml	EnrolledUser.java	MainController.java
RF7 Cierre de sesión	NewMain.fxml Welcome.fxml	MoodleUser.java	MoodleOptions.java MainController.java

Tabla C.1: Relación MVC entre las clases y vistas

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este anexo describe la documentación técnica de programación, incluyendo la instalación del entorno de desarrollo, la estructura de la aplicación, su compilación, ejecución, la configuración de los diferentes servicios y las pruebas del sistema.

D.2. Estructura de directorios

El repositorio del proyecto podemos observar la siguiente estructura:

- `/`: contiene los ficheros de configuración de Gradle [**?**], de los servicios de integración continua, en nuestro caso Travis-CI [4], el fichero README , la copia de la licencia y un fichero .xml que es el ANT para la creación del .jar.
- `/src/main/java/`: Encontraremos toda la lógica de la aplicación.
- `/src/test/java/`: El código para testear la aplicación.
- `/resources/`: Recursos de nuestra aplicación.
- `/resources/css/`: Estilos de la aplicación.
- `/resources/img/`: Imágenes de la aplicación.
- `/lib/`: Librerías de la aplicación.
- `/doc/`: documentación del proyecto.
- `/doc/javadoc/`: documentación *javadoc*.

- `/docs/proyectdoc/`: documentación en formato LaTeX.

D.3. Manual del programador

En esta sección explicaremos como montar el entorno de desarrollo, descargar el código fuente del proyecto, compilarlo, ejecutarlo y exportarlo. hay que tener en cuenta que este manual esta orientado a sistemas Linux como lo es Ubuntu 16.04 que es donde se ha desarrollado el proyecto, aunque también se podría desarrollar en S.O. Windows.
Para la instalación en sistemas Windows hay que leer la documentación de Claudia Martínez Herrero [5] referente al manual del programador.

Entorno de desarrollo

Para trabajar con el proyecto se necesita tener instalados los siguientes programas y dependencias:

- Java JDK 8.
- Git.
- Moodle 3.3.
- Eclipse.
- Plugin Gradle para Eclipse.

A continuación, se indica como instalar y configurar correctamente cada uno de ellos.

Java JDK 8

En esta instalación es importante descargar el JDK 8 de oracle ya que, otro jdk como puede ser el que viene por defecto en las librerías de Ubuntu no te proporcionan la Librería Java FX.
Deberemos abrir la consola y poner los siguientes comandos:

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
```

Figura D.1: Instalacion JDK 8 en Linux.

Git

Necesitamos git para la gestión del repositorio. Git nos permitirá clonar el proyecto en nuestro equipo, subir cambios, comprobar diferencias entre commit, etc.

```
oscar@oscar-GL62M-7RD:/media/oscar/Datos/workspace/tfg/GII-17.1B-UBULog-1.0$ sudo apt-get install git
[sudo] password for oscar:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
git ya está en su versión más reciente (1:2.7.4-0ubuntu1.3).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 233 no actualizados.
oscar@oscar-GL62M-7RD:/media/oscar/Datos/workspace/tfg/GII-17.1B-UBULog-1.0$
```

Figura D.2: Instalacion Git.

Moodle 3.3

Para la instalación de Moodle hacemos referencia a los anexos de Claudia Martínez Herrero [5] referente a la instalación de moodle en Windows ya que es idéntica.

Eclipse

Eclipse es el entorno de desarrollo seleccionado para este proyecto. Si el programador desea utilizar otro, no tendría ningún problema. Para la instalación deberemos descargar Eclipse en nuestro equipo. Lo descomprimimos y ejecutamos el archivo eclipse-inst, los pasos siguientes es igual que en Windows, se abrirá el gestor de instalación y solo hay que seguir las instrucciones.

Gadle

La mayoría de los IDE de desarrollo tienen integrado Gradle, como es el caso de Eclipse y no se necesita ningún tipo de instalación. En el caso de que no tenga, podemos instalarlo en nuestra maquina personal. Lo que deberemos hacer es:

```
oscar@oscar-GL62M-7RD:/media/oscar/Datos/workspace/tfg/GII-17.1B-UBULog-1.0$ sudo apt-get install gradle
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
gradle ya está en su versión más reciente (2.10-1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 233 no actualizados.
oscar@oscar-GL62M-7RD:/media/oscar/Datos/workspace/tfg/GII-17.1B-UBULog-1.0$
```

Figura D.3: Instalación gradle.

D.4. Compilación, instalación y ejecución del proyecto

Obtención del código fuente

Una vez hemos instalado todo lo necesario, deberemos clonar nuestro proyecto.
Deberemos entrar al repositorio GitHub y en la parte que pone Clone whith HTTPS, copiarnos la URL.

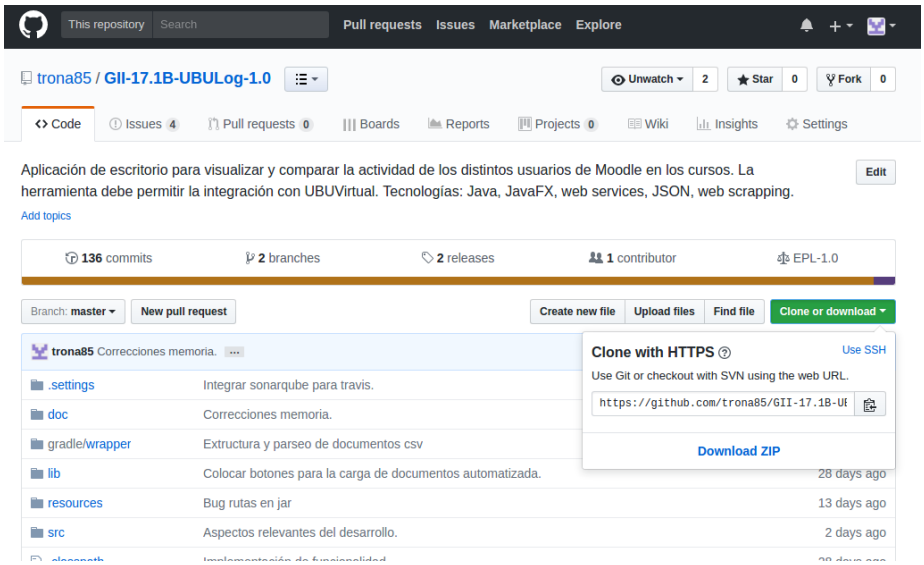

Figura D.4: Clonar proyecto con Git 1.

Abrimos la consola y nos vamos a nuestro workspace. Una vez ahí, ponemos el siguiente comando de git con la URL copiada anteriormente.

```
oscar@oscar-GL62M-7RD:/media/oscar/Datos/workspace/tfg$ git clone https://github.com/trona85/GII-17.1B-UBULog-1.0.git
```

Figura D.5: Clonar proyecto con Git 2.

Nos descargara el código de la aplicación y ya lo tendremos en nuestro equipo para poder trabajar con el.

Importar proyecto en Eclipse

Para importar nuestro proyecto deberemos abrir Eclipse. Una vez abierto deberemos:

1. Vamos a File–>import...
2. Nos saldrá una ventana en la que deberemos elegir la opción de gradle.

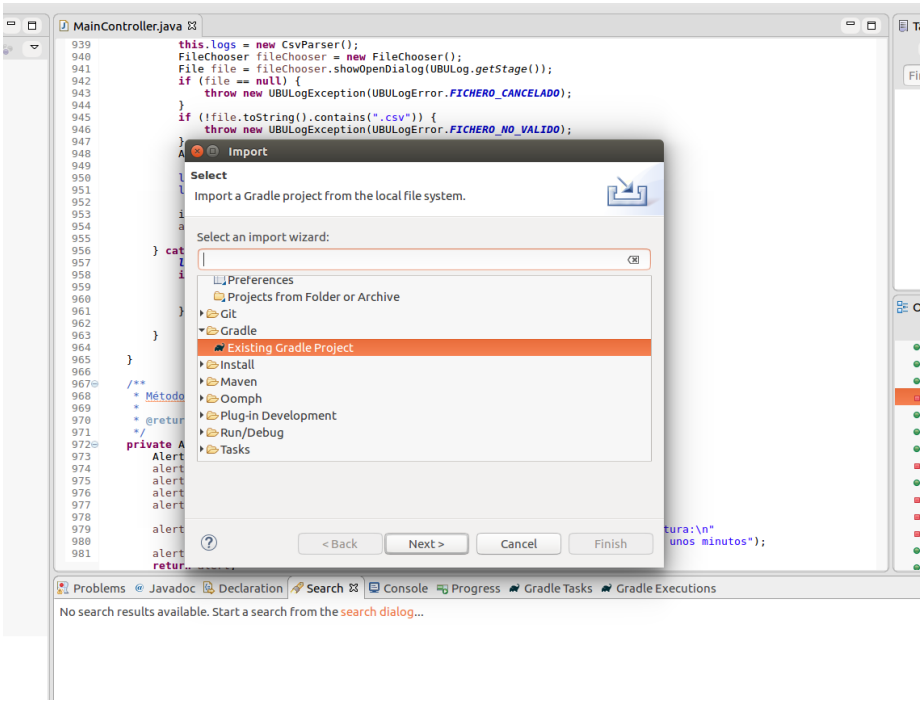

Figura D.6: Importar proyecto Gradle.

3. Nos saldra una ventana de información, después de su lectura se da a next.

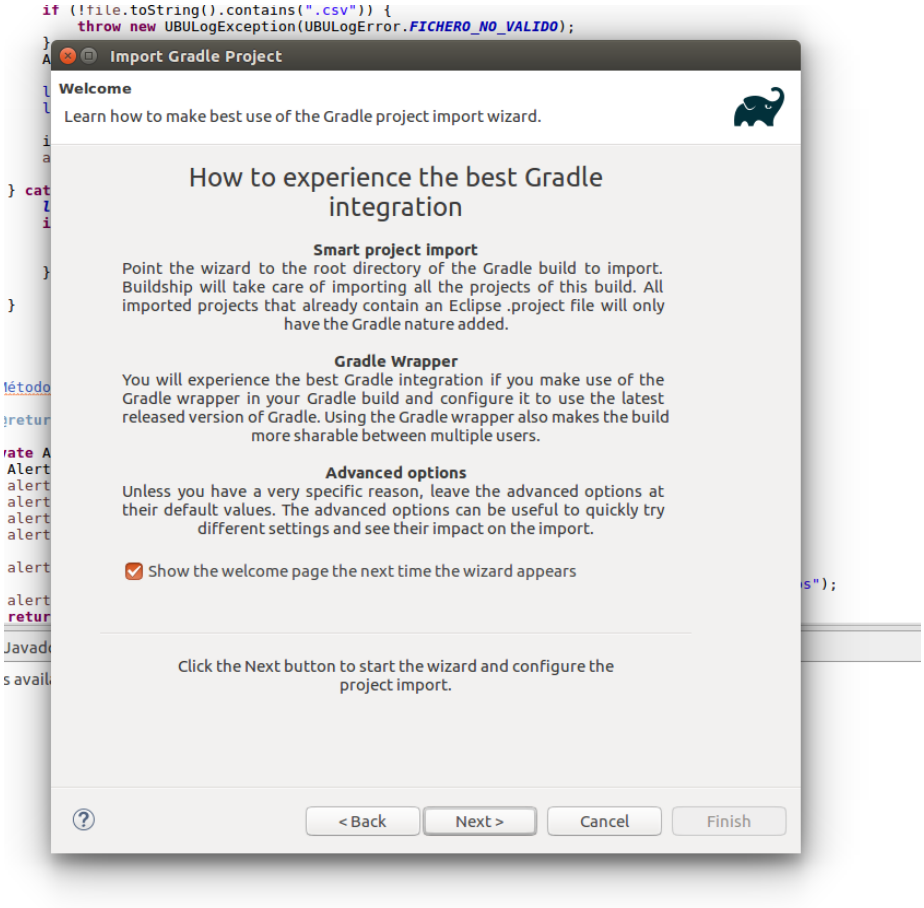

Figura D.7: Información Gradle.

4. En esta ventana nos ubicamos en la raíz del proyecto que queremos importar, igual que en la siguiente imagen, después pinchamos en finalizar.

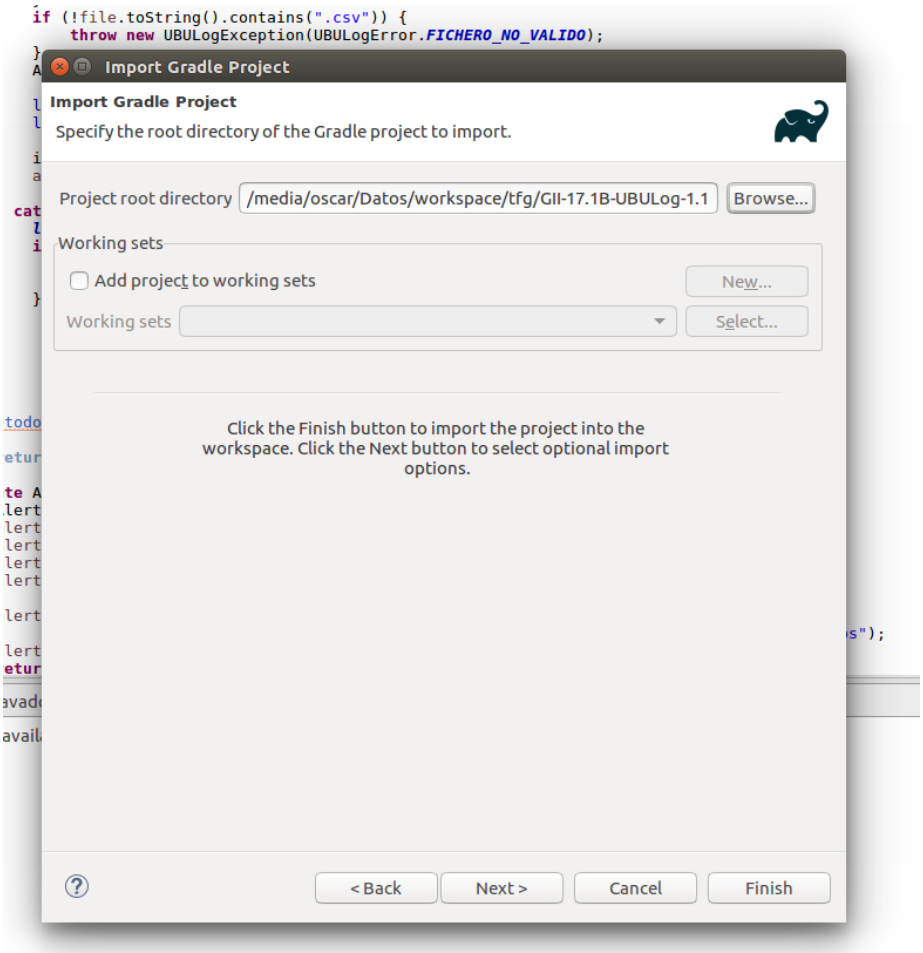

Figura D.8: Cargar proyecto Gradle.

5. Gradle empezara a descargar todas las dependencias del proyecto, esto puede tardar unos minutos. Cuando termine ya tendremos nuestro proyecto preparado para el desarrollo.

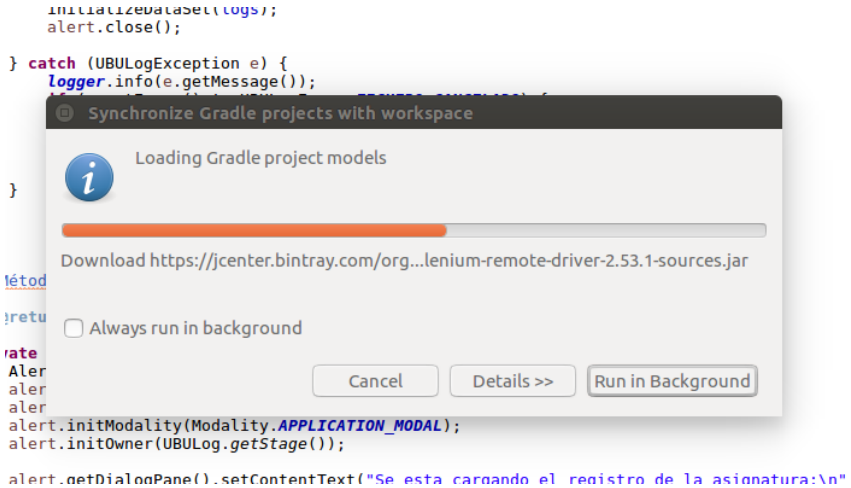

Figura D.9: Cargar proyecto Gradle.

Build.gradle

Build.gradle es el archivo donde debemos poner el tipo de aplicación que es, la versión y las dependencias necesarias de nuestro proyecto. [**?**]

Travis.yml

Travis.yml es el archivo donde tenemos la configuración para la integración continua y el análisis de la calidad del código con sonarqube. [4] [**?**]

Generar .jar

Para generar el .jar se ha creado un .xml en el proyecto que es un ANT, en el se tienen todas las dependencias, .class y recursos necesarios para la ejecución. Para generar el jar deberemos:

1. Abrimos UBULog.xml en eclipse.
2. La ventana de Eclipse que esta en la parte derecha, que pone OutLine tendremos una opción que es create runnable.

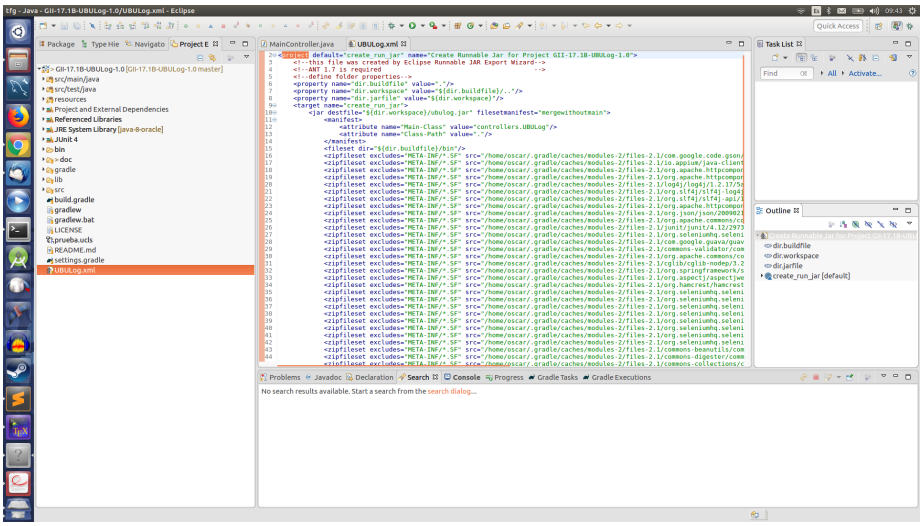

Figura D.10: Crear .jar.

Con el botón secundario pinchamos y ejecutamos el ANT Builder para generar el jar, en la consola del Eclipse nos saldrá esto.

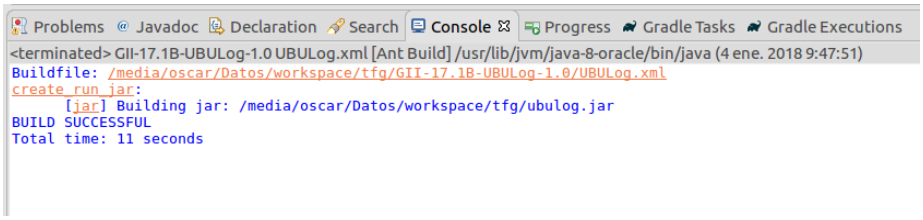

Figura D.11: Jar creado.

Una vez visto el resultado de la captura anterior, querrá decir que ya hemos generado el .jar.

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] Commons CSV – Home.

[2] GII-17.1B-UBULog-1.0 - trona85.

[3] Libreria para generar los graficos - Chart.js · Cocumentación.

[4] Travis CI - Integración continua.

[5] Claudia Martínez Herrero. *Anexo tfg, 2017* .