



---

# DOCUMENTACIÓN DEL PROYECTO

---

Gestión del programa



X DE FEBRERO DE 2017

José Manuel Lara Morilla  
José Manuel Luque Mendoza  
José Félix Gómez Rodríguez  
Feliciano Blanco Castaño  
Samuel González Zuloaga

## Índice de contenido

1. Introducción y contexto .....	2
2. Descripción del sistema .....	3
2.1. Planificación del trabajo .....	3
2.2. Cambios realizados .....	5
2.3. Funcionalidad.....	5
1- Aviso de cambios en el programa .....	5
2- TODO Añadir las otras funcionalidades.....	6
3. Entorno de desarrollo.....	7
3.1. Entorno de desarrollo.....	7
4. Gestión del código fuente .....	14
4.1. Gestión de las ramas en el código.....	14
4.2. Gestión de <i>Issues</i> .....	15
4.3. Políticas de nombre y estilo en el código fuente .....	¡Error! Marcador no definido.
5. Gestión de la construcción e integración continua .....	16
6. Entregables y Roles.....	16
6.1. Entregables .....	16
6.2. Roles.....	16
7. Herramientas utilizadas.....	17
8. Conclusiones y trabajo futuro .....	18

## 1. Introducción y contexto

A la hora de llevar a cabo la realización de un trabajo, por lo general se produce una lluvia de ideas por parte del equipo para ver cuáles son las posibles implementaciones que vamos a llevar a cabo.

De cara a este proyecto, el equipo tenía claro que para llevar a cabo el desarrollo de nuevas funcionalidades para ampliar la sección que se nos había asignado, había que decidir cuáles de estas vamos a llevar a cabo, ya que en la primera reunión del equipo se propusieron muchas ideas, pero a la hora de planificar el trabajo, nos dimos cuenta de que por temas de gestión de tiempo y del personal, todas las propuestas no se podían llevar a cabo.

De todas las propuestas que desde un principio se plantearon, decidimos llevar a cabo las siguientes:

- Implementación de un mensaje de notificación cuando se hace algún cambio en el programa del congreso.
- Implementar la posibilidad de imprimir el programa.
- Llevar a cabo copias de seguridad del programa, para salvaguardar cambios y poder recuperar un programa si un cambio no se introdujo correctamente.
- Poder cargar el programa en formato .xlsx vía online, y no de forma local.

Estas implementaciones se llevan a cabo para mejorar la página web

<http://congreso.us.es/splc2017/>, esta web ya tiene implementado un plugin que permite leer el fichero en formato Excel, y mostrar el contenido de este en forma de programa, como se puede ver en el siguiente ejemplo:

Monday	Tuesday	Wednesday	Thursday	Friday
Monday, 25 Sep				
8:15-8:45	REGISTRATION			
8:45-9:15	OPENING OF THE PRE-CONFERENCE			
9:15-10:30	<b>Foro Industrial</b> <i>Room: Salón de actos</i>	<b>Tutorial:</b> Using Feature Models to Manage Variability and Requirements Reuse - <i>Danilo Beuche</i> Room: SUM	<b>Tutorial:</b> Fostering a consistent SPL service ecosystem - <i>José A. Galindo and Pablo Fernandez</i> Room: AULA 1	
10:30-11:00	COFFEE BREAK			

Ilustración 1: Gestión del Programa

## 2. Descripción del sistema

### 2.1. Planificación del trabajo

Para la gestión del proyecto hemos utilizado la aplicación Trello, esta aplicación ha sido elegida por dos motivos principales:

- La facilidad de crear tareas y gestionarlas.
- La posibilidad de a través de un plugin utilizar la herramienta *Toggl* para contabilizar el tiempo que cada integrante del grupo ha dedicado a

En este ejemplo podemos ver cómo funciona Trello, ver las distintas etapas por las que puede pasar cada una de las etiquetas (actividades):

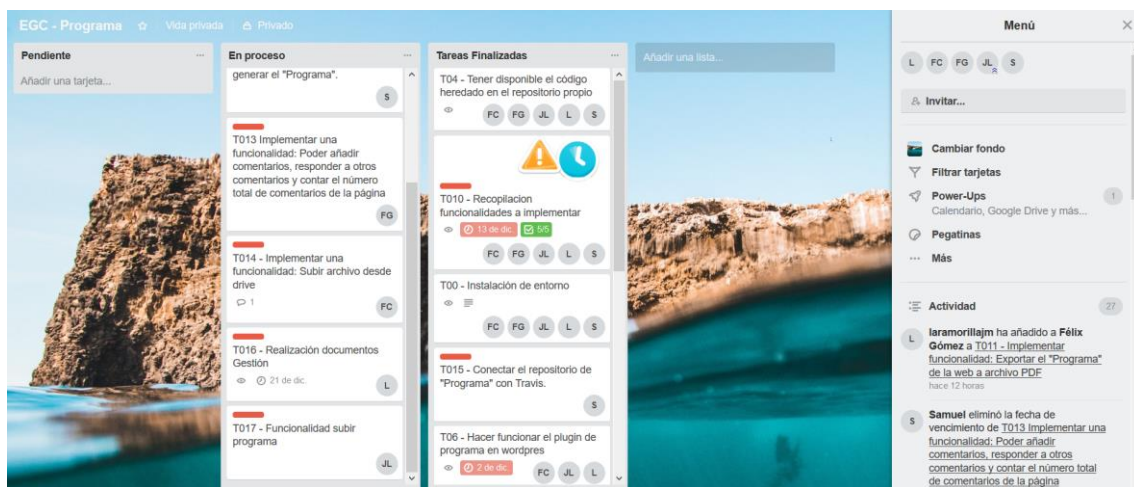


Ilustración 2: Trello

A la hora de crear una nueva tarea o *issues* (esto lo explicaremos más adelante), estos pasan por una serie de etapas. En el momento de creación, se ha de especificar una descripción referente a esta tarea, quien la va a desarrollar y para cuando ha de estar realizada.

Las etapas por los que pasa nuestra aplicación son:

- Pendiente: En este apartado especificamos todas las tareas que hay que realizar en el proyecto. Se le asigna a una persona del equipo, y se espera que esta persona mueva la etiqueta a la hora de empezar a desarrollar dicha tarea.
- En proceso: En este apartado se encuentran todas aquellas tareas que están en proceso de ejecución.
- Tareas Finalizadas: Se archivan todas las tareas que ya han sido realizadas.

En cuanto a las etiquetas de las *issues*:

- *Help wanted*: Destinada a incidencias para pedir ayuda.
- *Bug*: Destinada a un fallo en la programación del software.
- *Duplicate*: Destinada a identificar *issues* que ya han sido creadas.
- *Enhancement*: Destinada a solicitar una mejora del software.
- *Good First Issue*: Destinada a incidencias para un desarrollador del equipo junior, o con menor experiencia en el software.
- *Invalid*: Destinada a identificar incidencias incorrectas, refleja una petición que no tiene sentido.
- *Pending*: Destinada a *issues* que todavía no han sido gestionadas.
- *Won't fix*: Destinada a especificar que una incidencia no tiene solución posible. Se emplea cuando se describe como incidencia un comportamiento que parece ser defectuoso, pero que en realidad es una funcionalidad. También se puede emplear para una incidencia que describe un problema, para la que no existe solución posible. Hay que tener una buena razón para etiquetar una incidencia de esta manera.

TAREA	ASIGNADO A
<b>Instalación del entorno</b>	José Manuel Lara Morilla
<b>Definición de requisitos indispensables</b>	Samuel González Zuloaga
<b>Añadir información/enlaces wiki egc</b>	José Félix Gómez Rodríguez Samuel González Zuloaga
<b>Disponibilidad del código heredado en el repositorio</b>	José Manuel Lara Morilla José Manuel Luque Mendoza José Félix Gómez Rodríguez Feliciano Blanco Castaño Samuel González Zuloaga
<b>Realización de diapositivas <i>Milestone 2</i></b>	José Félix Gómez Rodríguez Samuel González Zuloaga
<b>Hacer funcionar el plugin de Programa</b>	José Manuel Luque Mendoza José Manuel Lara Morilla Feliciano Blanco Castaño
<b>Implementar API GET</b>	José Manuel Luque Mendoza José Manuel Lara Morilla Feliciano Blanco Castaño
<b>Realización informe retrospectiva</b>	José Félix Gómez Rodríguez
<b>Recopilación de funcionalidades a implementar</b>	José Manuel Lara Morilla José Manuel Luque Mendoza José Félix Gómez Rodríguez Feliciano Blanco Castaño

	Samuel González Zuloaga
<b>Funcionalidad exportar Programa a PDF</b>	José Manuel Lara Morilla José Félix Gómez Rodríguez
<b>Funcionalidad para mostrar el programa a partir de archivos en formatos no xlsx</b>	Samuel González Zuloaga
<b>Funcionalidad de comentarios sobre el programa</b>	José Félix Gómez Rodríguez
<b>Funcionalidad para subir programa de forma online</b>	Feliciano Blanco Castaño
<b>Realización de documentos de Gestión</b>	José Manuel Lara Morilla Feliciano Blanco Castaño

*Tabla 1: Planificación de tareas*

## 2.2. Cambios realizados

TODO

Desde un principio se nos facilito el código que correspondía al plugin funcionando del programa del congreso.

Actualmente estamos implementando funcionalidades, hoy la funcionalidad implementada es la de enviar un correo a los suscritos, si se produce un cambio en la planificación del programa.

## 2.3. Funcionalidad

### 1- Aviso de cambios en el programa

TODO: Falta configurar el servidor de correo.

Esta primera funcionalidad implantada permite a los usuarios recibir mediante correos electrónicos avisos sobre cambios que se produzca en el sistema.

Tal y como vemos en la siguiente ilustración esta es la vista actual que tiene el sistema, podemos ver el campo donde introducir el correo, además de los botones correspondientes para suscribirnos a este servicio.

FOTOOOOOOOOOOOOO

*Ilustración 3: Funcionalidad de avisos*

Ahora vamos a pasar a describir el código introducido para llevar a cabo esta funcionalidad.

```
68     <iframe name="suscribirse" style="display:none;"></iframe>
69     <form action="funcionmail.php" method="post" target="suscribirse">
70     Introduce aquí tu email si quieres saber cuando se ha modificado el programa!: <input type="text" name="email" /><br />
71     <input type="submit" name="submit" value="¡Enviarme!" />
72 </form>
73 <?php
```

*Ilustración 4: Botón funcionalidad avisos*

Este código es añadido a la clase principal en la cual se encuentra el plugin (*program-manager.php*), nos permite mostrar un botón en el que el usuario se registra para notificaciones.

```
<?php

$usuariosSuscritos = array();

static function enviarCorreo($post_ID) {
    if(get_post_field('post_content',$ID ) === '[write_long_program]'){

        array_push($usuariosSuscritos, $_POST['email']);

        mail($usuariosSuscritos,"modificaciones programa SPLC",'modificaciones en el post de programa SPLC');

        return $post_ID;
    }
}
add_action('publish_post', 'enviarCorreo');
?>
```

*Ilustración 5: Función para avisos*

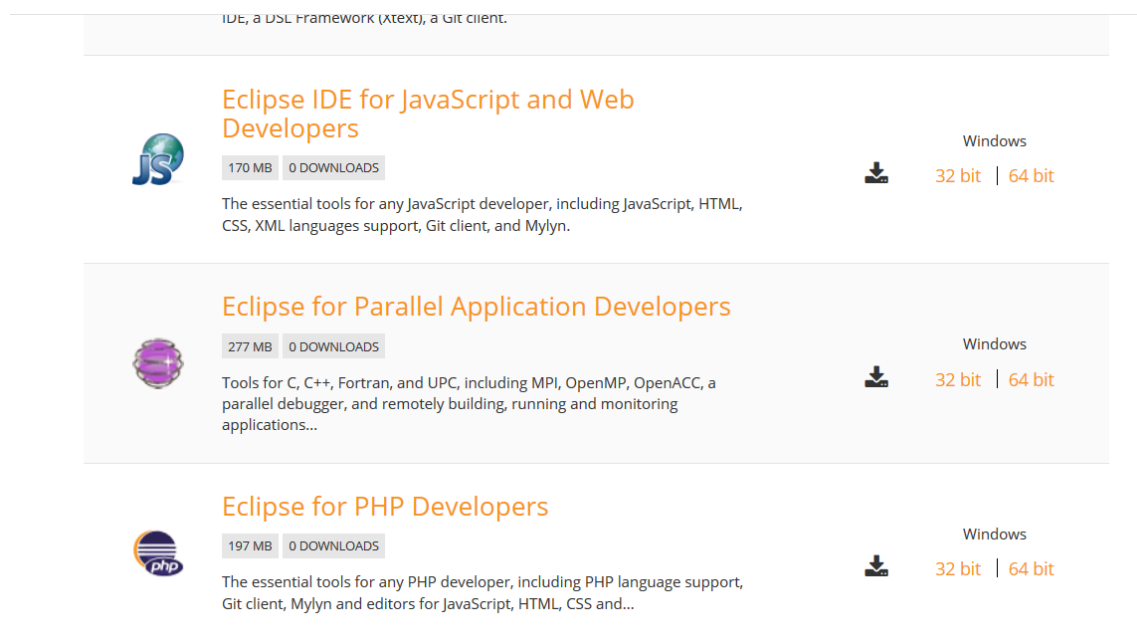
Este código implementa una función mediante la cual se introduce en el cuadro que aparece en la pantalla el correo electrónico en el que quieres que llegue las notificaciones. Funciona analizando el *\$post\_ID*, si este cambia envía un mail a todos los usuarios que se encuentren en la variable *\$usuariosSuscritos*.

## 2- TODO Añadir las otras funcionalidades

### 3. Entorno de desarrollo

#### 3.1. Entorno de desarrollo

El software utilizado para el desarrollo de este proyecto es eclipse, en su versión Oxigen. Lo podemos obtener desde el siguiente enlace: <https://www.eclipse.org/downloads/eclipse-packages/>



*Ilustración X: Descarga de eclipse*

En nuestro caso nos hemos descargado la versión para desarrolladores de PHP en su versión de *64bits*, hemos elegido esta opción porque vamos a trabajar en este lenguaje, y este paquete además trae utilidades que más adelante utilizaremos como es el cliente de GIT para la gestión de los *commit*.

Una vez descargado lo que único que tendremos que hacer es descomprimir el archivo zip descargado en una ruta de nuestro pc que elijamos. Se recomienda que esta ruta sea la del disco C.

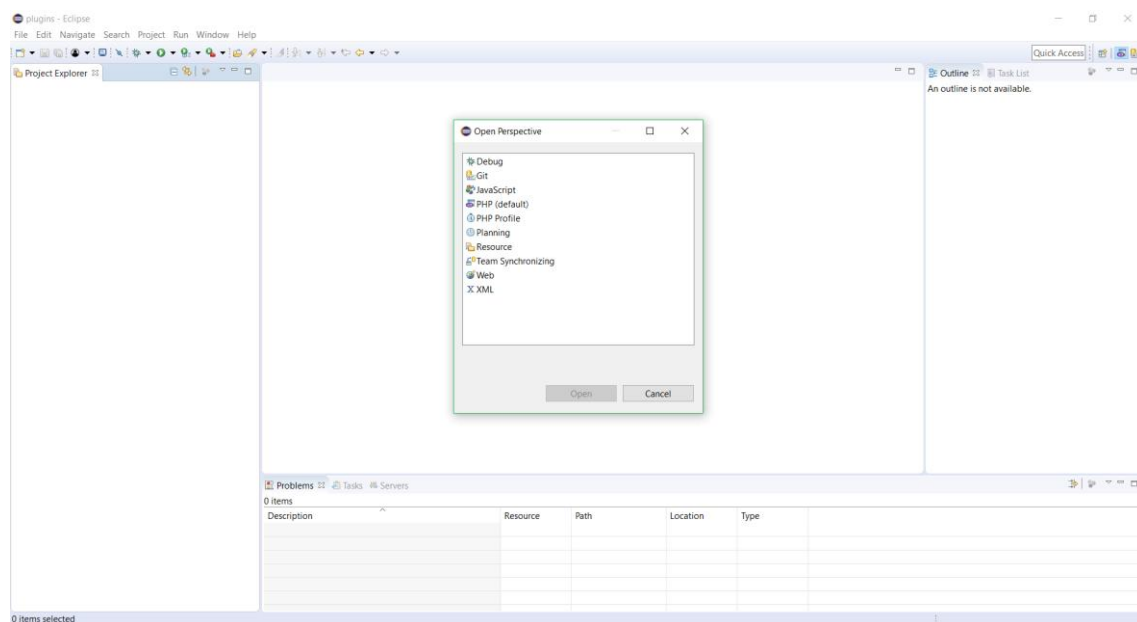


Este equipo > Disco local (C:)					
	Nombre	Fecha de modifica...	Tipo	Tamaño	
do os	\$WINDOWS.~BT	17/11/2017 13:39	Carpeta de archivos		
	Archivos de programa	05/12/2017 12:34	Carpeta de archivos		
	Archivos de programa (x86)	08/12/2017 19:40	Carpeta de archivos		
	Developer's configuration	20/12/2017 11:51	Carpeta de archivos		
	eclipse	14/12/2017 20:21	Carpeta de archivos		
ción	ESD	01/12/2017 1:54	Carpeta de archivos		
	PerfLogs	17/12/2017 22:12	Carpeta de archivos		
	Temp	08/12/2017 19:40	Carpeta de archivos		
	Usuarios	27/10/2017 17:53	Carpeta de archivos		
	Windows	17/12/2017 22:12	Carpeta de archivos		
	xampp	27/11/2017 14:03	Carpeta de archivos		

*Ilustración X: Ruta de instalación eclipse.*

Tras esto solo tendremos que iniciar el programa.

A continuación, lo que haremos será clonar el repositorio para trabajar sobre el código del plugin. Estos son los pasos que hay seguir, lo primero que haremos será abrir la vista del repositorio de GIT, window -> perspective -> open perspective -> other. En esta selección de vista elegimos GIT.



*Ilustración X: Abrir perspectiva de GIT*

Tras esto hacemos clic en Clone a Git repository, y seleccionamos Clone URI a continuación.

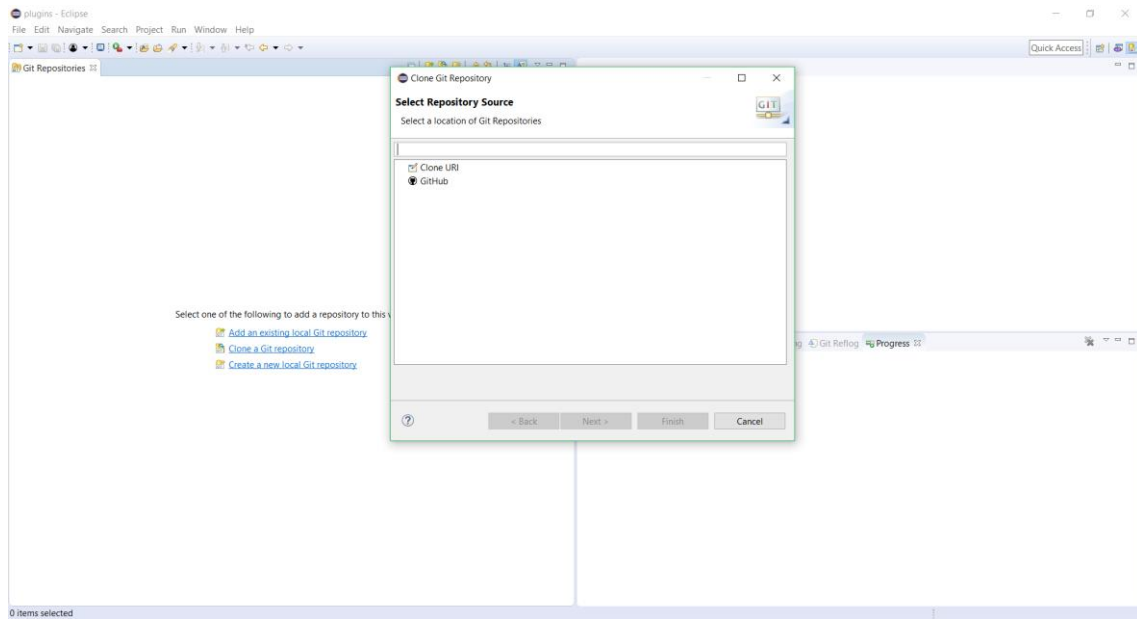


Ilustración X: Clone URI

Introducimos a URI de nuestro repositorio, y rellenamos los campos tal y como vemos

**Clone Git Repository**

**Source Git Repository**

Enter the location of the source repository.

**Location**

URI:  

Host:

Repository path:

**Connection**

Protocol:

Port:

**Authentication**

User:

Password:

☐ Store in Secure Store

Seleccionamos que rama queremos importarnos de nuestro repositorio

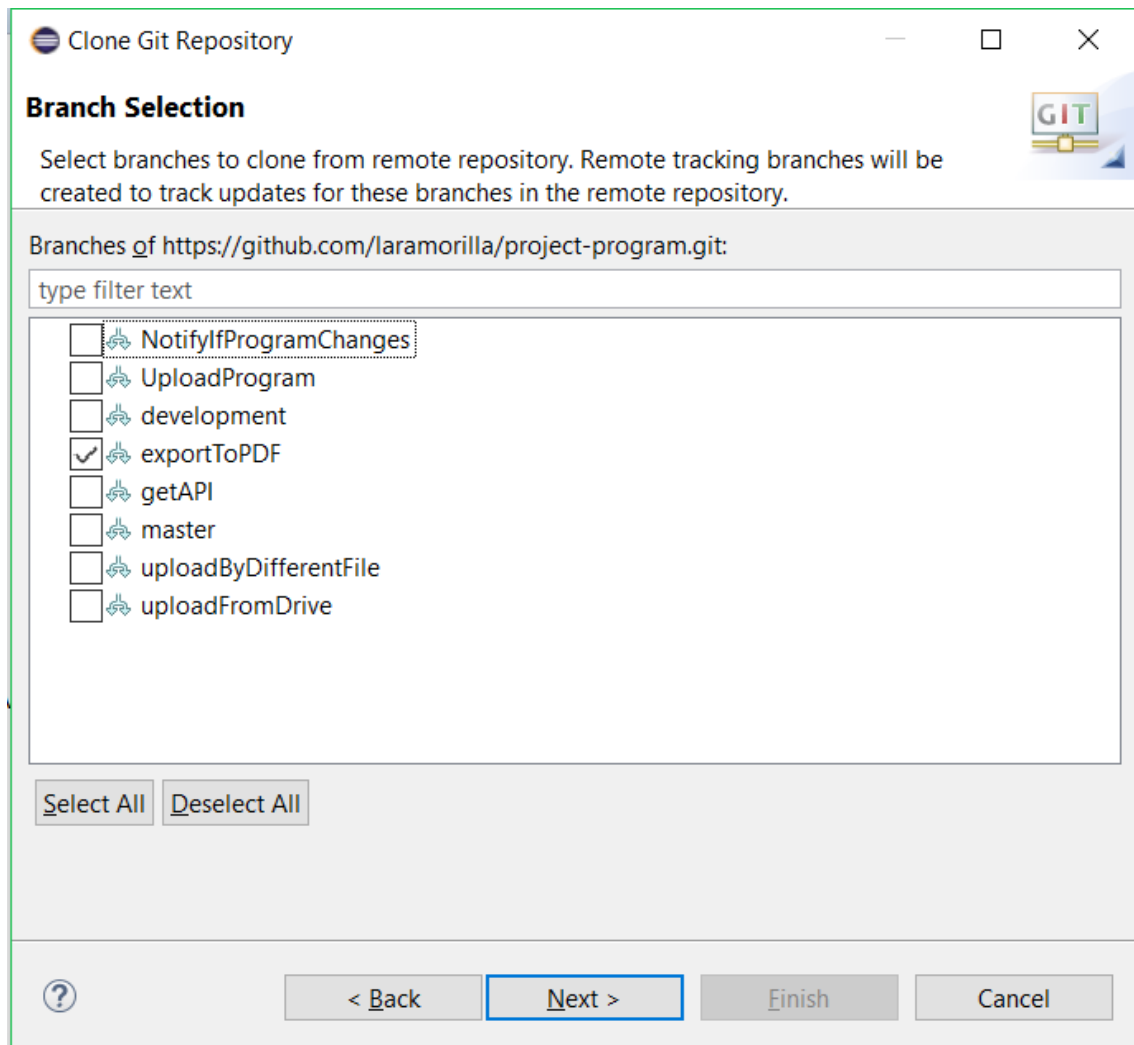
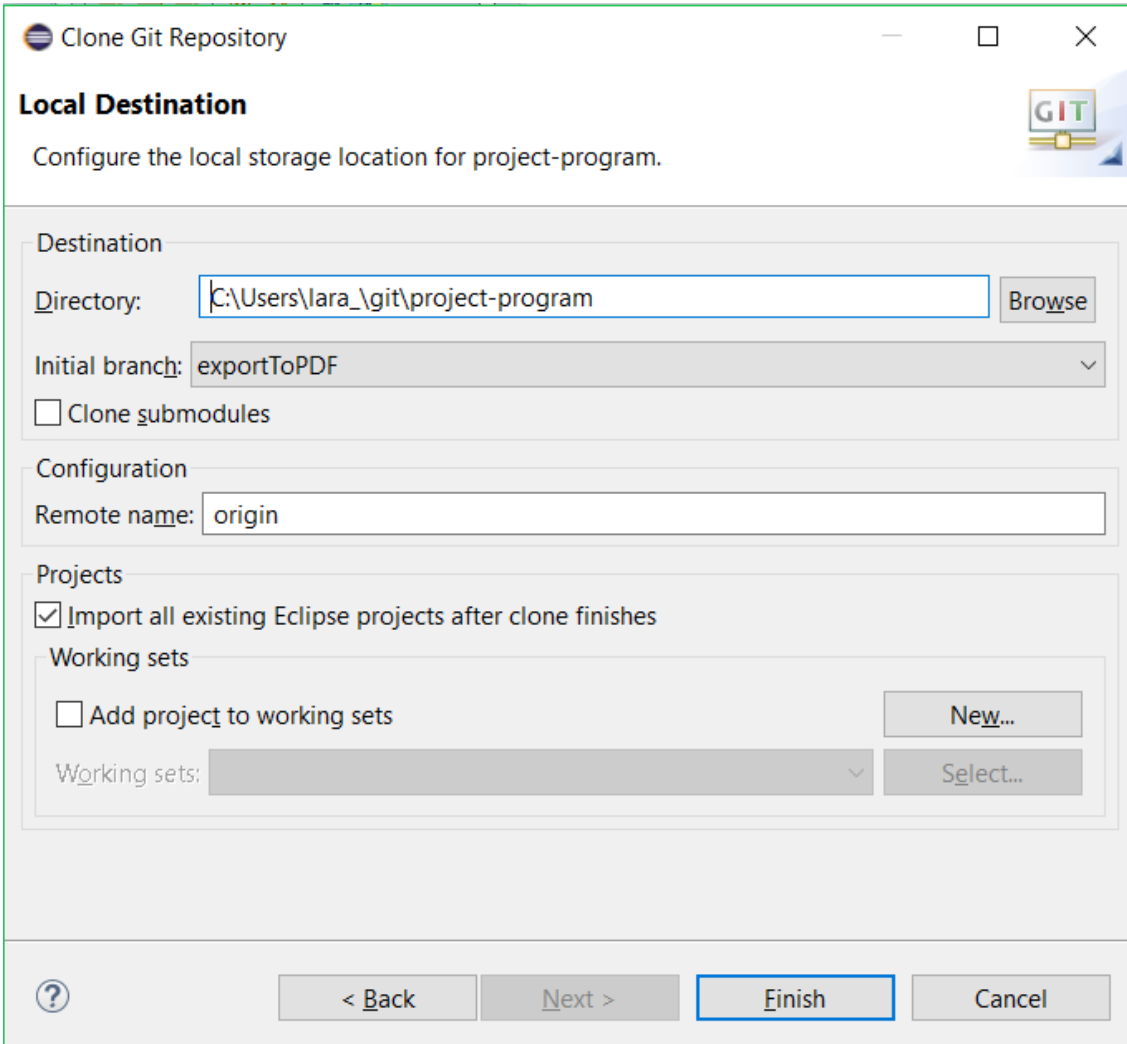


Ilustración X: Ramas de gitHub

Tras esto, seleccionamos la ruta donde queremos que se descargue el código y le damos a *finish* para terminar.



**Clone Git Repository**

**Local Destination**

Configure the local storage location for project-program.

**Destination**

Directory:

Initial branch:

☐ Clone submodules

**Configuration**

Remote name:

**Projects**

☒ Import all existing Eclipse projects after clone finishes

**Working sets**

☐ Add project to working sets

Working sets:

Ilustración X: Ruta de descarga del código

Como ultimo paso, desplegamos la rama y haciendo clic con el botón derecho, le damos a *import Project*, con esto tendremos ya en la vista java todo el código y podremos empezar a trabajar con este.

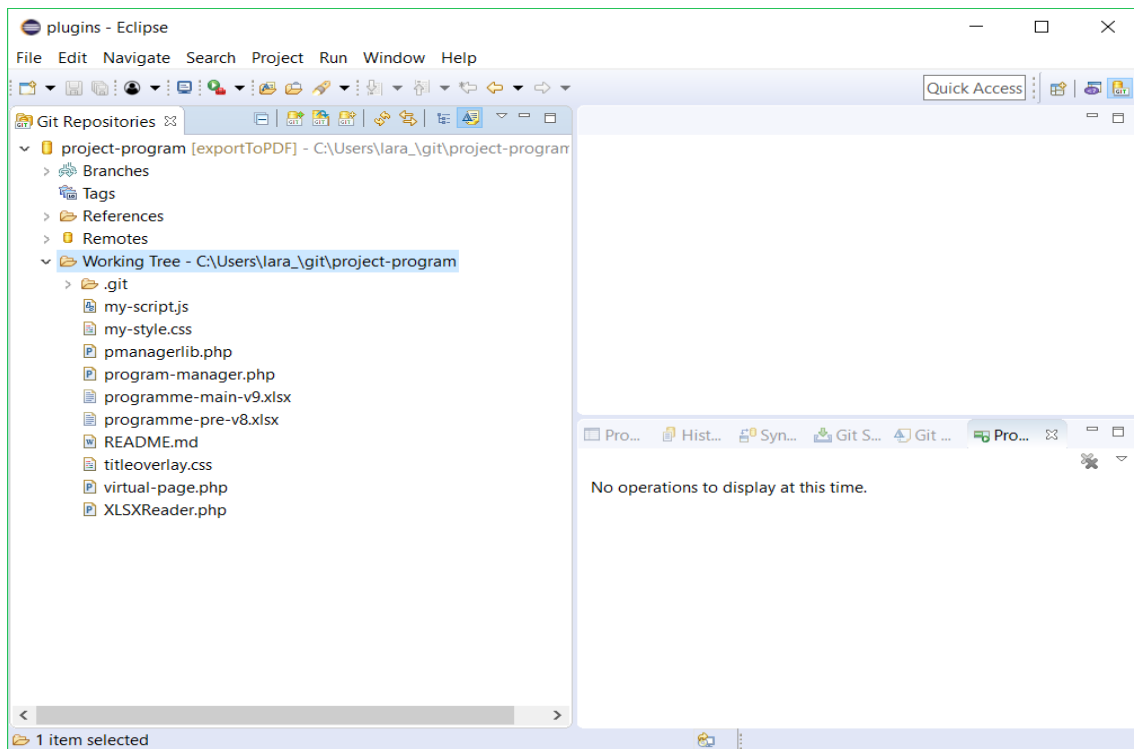


Ilustración X: Importar proyecto a la vista Java

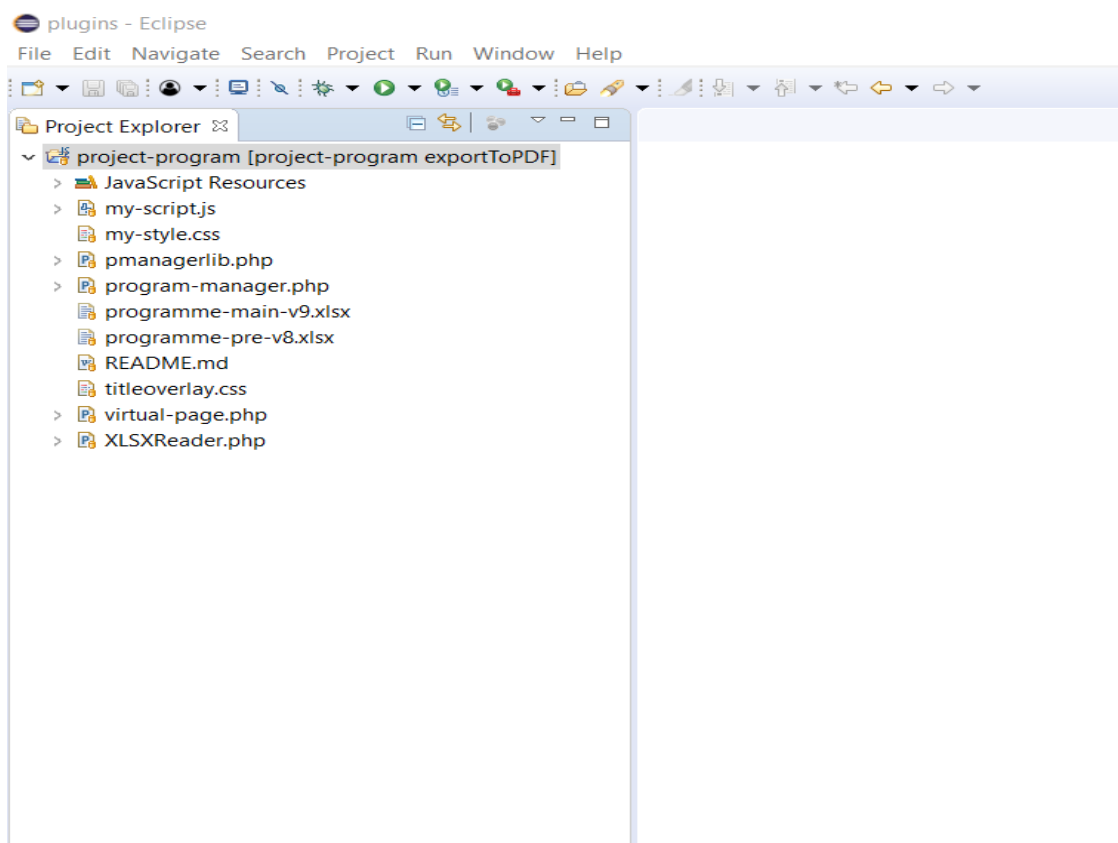


Ilustración X: Vista java del código

Hemos utilizado este software porque nos permite la gestión de forma sencilla de todo lo relacionado con GIT.

Otro software relevante que hemos utilizado es *xampp*, este software nos permite crear un servidor local web desde el cual podemos hacer pruebas locales de despliegue de la web. Se puede descargar desde el siguiente enlace: <https://www.apachefriends.org/es/download.html>

Además, se han utilizado GitHub para el control de versiones, la gestión de código, de incidencias y Travis para la automatización de las pruebas.

## 4. Gestión del código fuente

### 4.1. Gestión de las ramas en el código

La planificación de las ramas en nuestro proyecto se divide en dos principales y después tantas ramas como funcionalidades extras se vayan a realizar.

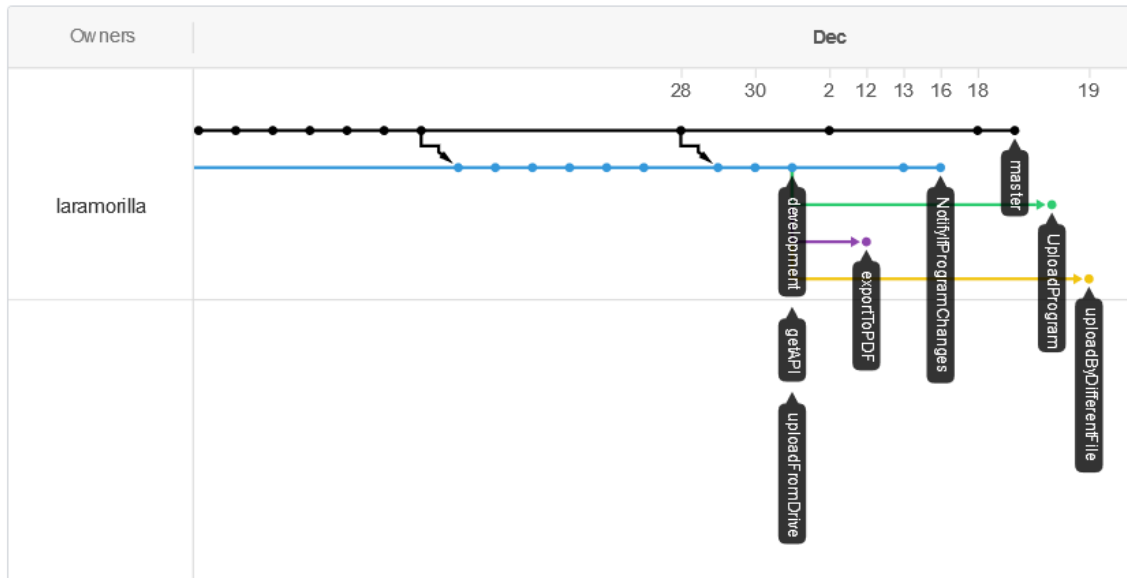


Ilustración X: Vista de GitHub

En esta imagen podemos ver algunas de las ramas de la que dispone nuestro proyecto. Vamos a proceder a describir alguna de ellas, cada una:

- **master:** Esta rama contiene el código que funciona y el cual se va a entregar al final de trabajo.
- **development:** Esta rama incluye los distintos cambios que se han introducido, se mantienen en esta rama hasta que se comprueben que funcionan con el sistema completo, tras esto se pasa el código a la rama **master**.
- **Otras:** Las demás ramas especifican una funcionalidad, por ejemplo, **exportToPDF** es una rama destinada a desarrollar todo el código para poder exportar el programa a PDF.

## 4.2. Gestión de Issues

Para llevar a cabo la apertura de una *issue* se han de seguir los siguientes pasos:

- Si se detecta un error, o hace falta una nueva funcionalidad, se comunica al equipo.
- El gestor de *issues*, en caso de que el propio miembro del equipo no pueda hacerlo en ese momento, crea una en GitHub y se la asigna a un desarrollador correspondiente.
- El desarrollador lleva a cabo una investigación para la solución de la *issue*:
  - si esta tiene que ver con una funcionalidad, se creará una rama desde la cual se desarrollará el código y se gestionará, tras llevar a cabo el desarrollo y las pruebas pondrá la solución y se cerrará la incidencia.
  - Si por el contrario la *issue* tiene que ver con un error del sistema, responderá a la incidencia con una solución y si es satisfactoria se cerrará la incidencia.
  - En caso de ser una *issue* referente a documentación se creará la *issue* y se cerrará cuando dicha documentación se realice y se suba al repositorio, haciendo referencia a dicha *issue*.

Para gestionar la conexión entre las incidencias y el código especificamos en cada commit a que *issue* hace referencia dicho código, y si ésta se cierra en ese momento o no.



## 5. Gestión de la construcción e integración continua

Vamos a utilizar una herramienta anteriormente citada, Travis CI. El funcionamiento básico de Travis CI es clonar el repositorio de GitHub a un nuevo entorno virtual y llevar a cabo una serie de tareas para construir y probar tu código. Si alguna de estas tareas falla, el *build* se considera *broken*. Si no falla ninguna tarea, el *build* se considera *passed* y Travis CI puede desplegar tu código a un servidor web, por ejemplo. Para tener una información más detallada se puede visitar el siguiente enlace, donde se explica en profundidad y se expone un ejemplo del mismo.

TODO introducir el ejemplo de nuestro código.

## 6. Entregables y Roles

### 6.1. Entregables

Los entregables son el código del proyecto con las funcionalidades requeridas ya funcionando. En nuestro caso, el entregable es el código que se encuentra en la rama *master*, que como se ha citado en los puntos anteriores es el código funcional de nuestro proyecto.

Como hemos dicho con anterioridad el despliegue lo hacemos con Travis CI, pero en nuestro caso, el entregable va a estar subido a *Docker*, para que de una forma sencilla se pueda integrar con el resto de proyectos de esta asignatura.

TODO insertar imagen de un despliegue.

Reseñar que el despliegue que se haga sobre la rama *development* de GitHub tiene que estar bajo supervisión del equipo. Si se quiere desplegar en la rama *master*, hay que hacer un aviso previo al equipo de integración ya que tienen que coordinar los distintos módulos del sistema.

### 6.2. Roles

Los roles para este trabajo no están divididos y enfocados a que cada uno trabaje por su cuenta.

En este proyecto todo el equipo trabaja de forma conjunta, si un miembro desarrolla un código el resto del equipo se encarga de revisarlo y dar el visto bueno.

En cuanto a los documentos de gestión, si tenemos dos encargados principales: José Manuel Lara Morilla y Feliciano Blanco Castaño, estos se encargan de la creación de dichos documentos. El resto del equipo llevará a cabo la labor de corregir o ampliar los apartados que cree oportunos.

## 7. Herramientas utilizadas

**Wiki de la asignatura** de la cual hemos obtenido toda la información necesaria para la realización de este proyecto.

**GitHub** es el repositorio que hemos utilizado para gestionar tanto el código fuente como las incidencias.

**Travis CI** es el servicio que se ha utilizado para la integración continua y la realización de las pruebas.

**Eclipse** es el entorno de desarrollo utilizado, por su simplicidad y por tener ya incorporado en su versión para desarrolladores PHP todo lo necesario para la gestión y desarrollo del código de este proyecto.

**Window** ha sido el sistema operativo elegido para utilizar todas las herramientas. Hemos elegido este sistema operativo porque facilita mucho la instalación y la interconexión entre los distintos programas.

**Docker** utilizado para desplegar el plugin final, hemos utilizado *Docker* porque permite iniciar el plugin e integrar distintos módulos de una manera sencilla y sin consumir apenas recursos, si comparamos con una máquina virtual.

**Virtual Box** es un software para poder tener varias máquinas virtuales, hemos utilizamos este software para iniciar la maquina virtual facilitada por el equipo de integración, mediante la cual hemos subido a *Docker* el plugin terminado.

**Xampp** nos permite tener un servidor web de forma local, hemos utilizado este software porque el equipo ya estaba familiarizado con él, además la aplicación incluye todo lo necesario, solo hay que instalar e iniciarl.

## 8. Conclusiones y trabajo futuro

TODO A la espera de la finalización del proyecto.