

Gestión de programa

Grupo 1

ID Opera: 17

- [Repositorio de código](#)
 - [Grupo de opera](#)

[Aguilar Cano, Francisco](#)

[Rodríguez Regidor, Juan Antonio](#)

Contenido

Resumen	3
Introducción y contexto	3
Descripción del sistema	4
Planificación del proyecto	5
Entorno de desarrollo	5
Gestión del cambio e incidencias	6
Gestión del código fuente	8
Gestión de la construcción e integración continua	11
Gestión de liberaciones, despliegue y entregas	12
Mapa de herramientas	12
Ejercicio de propuesta de cambio	13
Conclusiones y trabajo futuro	13
Lecciones aprendidas	14

Resumen

Nuestro cometido ha sido crear un plugin para wordpress que resolvía problemáticas surgidas en la web de un congreso. La problemática que existía era:

- Anteriormente se leía el programa desde un documento Excel, lo que no es muy cómodo ya que es un documento offline y cada vez que se realice un cambio en el programa hay que volver a subir el documento al servidor. Nuestra solución: cambiar la lectura del programa a una hoja de cálculo de Google Sheet, lo que hace que sea un proceso más cómodo.
- Cada vez que se accedía a la página que mostraba el programa se ejecutaba el método que leía el programa del documento Excel lo que es bastante ineficiente. Nuestra solución: cachear los resultados en la base de datos y realizar la lectura de la hoja de cálculo una sola vez, lo que hace el sistema mucho más eficiente.
- Era imposible volver a una versión anterior del programa, lo que era bastante incomodo ya que si querías deshacer algún cambio y volver a un estado anterior del programa tenías que volver a cambiar el documento Excel y resubirlo. Nuestra solución: guardar las distintas versiones del programa que han sido publicadas para poder volver a una versión anterior.

Introducción y contexto

Nos encontramos con una aplicación web realizada en wordpress sobre un congreso (<https://congreso.us.es/splc2017>) que tenía una serie de plugins instalados. Esta web ha sido mejorada añadiéndole la funcionalidad de realizar votaciones y una serie de mejoras más. El sistema está dividido en numerosos subsistemas y a cargo de cada uno de ellos hay un grupo de trabajo distinto. En concreto, nuestro grupo se ha encargado del subsistema de Gestión de programa.

Entre los numerosos plugins que tenía instalados la web del congreso, debemos centrarnos en "program-manager-wpp", cuya función era leer el programa del congreso desde un documento Excel y convertirlo a un HTML que lo representaba en la web. Cuando identificamos la problemática expuesta en el [apartado anterior](#) nuestra intención fue hacer modificaciones sobre la versión anterior del plugin, pero conforme fue pasando el tiempo, nos dimos cuenta de que no podíamos reutilizar nada de dicha versión y que la mejor idea era desarrollar nuestro propio plugin desde cero en el que incluyamos la funcionalidad expuesta en el [apartado anterior](#).

Descripción del sistema

Como se ha mencionado en el [apartado anterior](#) la misión de nuestro subsistema es la realización de un plugin para wordpress que finalmente será integrado en el subsistema realizado entre todos los subsistemas.

Para la lectura de la hoja de cálculo online utilizamos la API de Google Sheet y utilizamos la API key proporcionada a los desarrolladores para poder acceder a las hojas de cálculo públicas. Para acceder a una hoja de cálculo pública sólo necesitamos que en la interfaz de usuario del plugin nos proporcionen el ID de la hoja de cálculo del pre programa y del programa principal. Siendo una URI genérica de una hoja de Google Sheet, nos referimos a ID de la hoja de cálculo al parámetro "spreadsheetId":

`https://docs.google.com/spreadsheets/d/spreadsheetId/edit#gid=sheetId`

Otras funcionalidades a las que podemos acceder a través de la interfaz de usuario son la de elegir el programa que queremos que sea mostrada en la web de entre todos los programas que tenemos guardados en la base de datos en modo final, cambiar un programa que hemos cargado de modo borrador a modo final (pueda ser elegido para mostrar) y para cambiar de modo final a borrador.

Si queremos guardar un nuevo programa en la base de datos debemos marcar la casilla "Volver a cargar al guardar" y si por el contrario no queremos guardar un nuevo programa sino que lo que queremos es editar el resto de opciones debemos dejar dicha casilla sin seleccionar.

Program Manager Settings

ID Pre-Programa:

ID Programa Principal:

Seleccionar Versión a mostrar

Los borradores no pueden ser seleccionados para mostrar

Seleccionar Borrador para pasar a final

Seleccionar Final para pasar a borrador

Volver a cargar al guardar

Nuestro subsistema tiene una única dependencia: [Gestión de visualización de programa](#). Este subsistema se encarga de la visualización del programa que nuestro subsistema guarda en la base de datos.

Planificación del proyecto



Entorno de desarrollo

Las herramientas utilizadas en el entorno de desarrollo son las siguientes:

- XAMPP Versión 5.6.30-1
 - PHP Versión 5.6.30
 - MariaDB Database Version 10.1.21-MariaDB
 - PhpMyAdmin Versión 4.7.0
- Eclipse Oxygen for PHP Developers 4.7.0
 - Git Integration For Eclipse 4.6.0
- Composer Versión 1.5.6

El procedimiento de instalación del entorno de desarrollo es el siguiente:

1. Descargamos XAMPP a través de <https://www.apachefriends.org/download.html> siguiendo el asistente de instalación. Una vez descargamos abrimos la aplicación XAMPP Manager e iniciamos MySQL y Apache Web Server.
2. Ya podemos iniciar PhpMyAdmin a través de <http://localhost/phpmyadmin>, procedemos a ejecutar el script SQL de WordPress. Puede encontrarse aquí https://1984.lsi.us.es/wiki-egc/index.php/Lista_de_proyectos_a_realizar_17-18

3. Descargamos e instalamos Eclipse <https://www.eclipse.org/pdt/>.
 4. Instalamos composer siguiendo las siguientes pautas <https://getcomposer.org/doc/00-intro.md> e añadimos las librerías para la API de Google Sheet API v4 <https://developers.google.com/sheets/api/quickstart/php>
- Siguiendo estos pasos ya tendríamos el entorno de desarrollo instalado.

Gestión del cambio e incidencias

Para la **gestión de cambios**, se procederá de la siguiente manera:

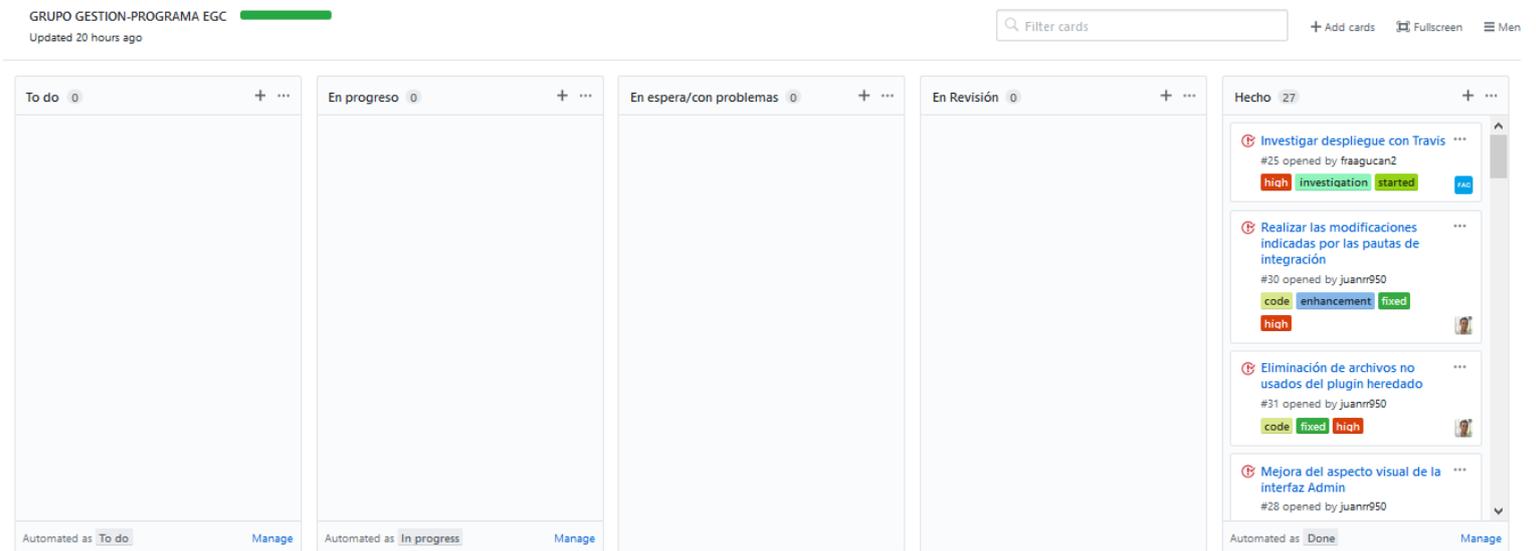
- En primer lugar se plantea el cambio dentro del grupo y se somete a votación.
- Si se rechaza termina el proceso, y si se acepta se traslada dicho cambio al Equipo de Integración.

Para la **gestión de incidencias** de nuestro proyecto vamos a utilizar la herramienta de gestión de incidencias que nos ofrece GitHub (Issues), ya que también va a ser nuestro gestor de código y nos resultará más cómodo de utilizar, además podemos visualizar en qué commit se solucionó cada incidencia añadiendo a la descripción del commit el número de la incidencia que se soluciona.

Cuando se cree una incidencia interna, el miembro del grupo deberá añadir:

- Título: lo más descriptivo posible.
- Descripción: en ella se desarrollará más el motivo de la incidencia.
- Se asignará a un miembro del grupo encargado para resolverla.
- Se hará uso de las etiquetas, pudiendo pasar por varios estados intermedios hasta su cierre en función de:
 - Temática
 - Documentation
 - Database
 - Code
 - Investigation
 - Prioridad
 - Critical
 - High
 - Medium
 - Low
 - Estado
 - New
 - Accepted (sólo para cambios)
 - Started
 - Fixed
 - Verified

- Duplicate (cambios y bugs)
 - Wontfix (cambios y bugs)
 - Tipo
 - Enhancement
 - Bug
- Cada Issue será situado en un proyecto interno al repositorio, siendo actualizado entre las distintas fases según corresponda a su avance. Las fases son:
 - TO DO
 - En progreso
 - En espera/ con problemas
 - En revisión
 - Hecho



Evidencias en el proyecto:

Fuente: <https://github.com/EGC-GESTIONPROGRAMA-1718/program-manager-wpp/projects/1>

Implementar feedback al ejecutar código de lectura y almacenamiento #22

Edit New issue

Closed juanrr950 opened this issue 3 days ago · 1 comment

juanrr950 commented 3 days ago

Se ha detectado que en el panel admin del plugin de wordpress no hay ningún tipo de feedback para saber lo que está pasando en el proceso de lectura y almacenamiento del programa.

Por lo que se propone realizar una mejora para tener algún tipo de indicación de lo que está pasando en dicho proceso.

juanrr950 added enhancement code high new labels 3 days ago

juanrr950 added this to the M4 Entrega y defensa de trabajos milestone 3 days ago

juanrr950 self-assigned this 3 days ago

juanrr950 added this to To do in GRUPO GESTION-PROGRAMA EGC via automation 3 days ago

juanrr950 moved this from To do to En progreso in GRUPO GESTION-PROGRAMA EGC 3 days ago

juanrr950 added started and removed new labels 3 days ago

juanrr950 added a commit that referenced this issue 3 days ago

Update : Mejora en Settings para obtener feedback 72dd7ds

juanrr950 added fixed and removed started labels 3 days ago

juanrr950 commented 3 days ago

Se da por completada la incidencia

Assignees: juanrr950

Labels: code, enhancement, fixed, high

Projects: Hecho in GRUPO GESTION-PRO..., M4 Entrega y defe...

Notifications: Subscribe

1 participant: juanrr950

Lock conversation

Fuente: <https://github.com/EGC-GESTIONPROGRAMA-1718/program-manager-wpp/issues/22>

En caso de que llegue alguna incidencia externa, se le añadirá la etiqueta “external” y:

- Si es un bug: se asignará a la persona encargada de dicho módulo y se solucionará.
- Si es una posible mejora o un fallo general: se convocará una reunión y se asignará a los miembros del equipo que se crea conveniente.

No hemos recibido incidencias externas, por lo que no hemos podido llevarlo a cabo.

Gestión del código fuente

Para la gestión del código fuente se ha hecho uso de Git y GitHub, de la siguiente forma:

Formato de commits: Los commits se harán cuando haya algo funcional que aportar al proyecto o solucione alguna incidencia y se hacen de la siguiente forma:

```
<Tipo> : <Título del commit>
<Cuerpo del commit>
<Pie del commit>
```

- <Tipo> : puede tomar los siguientes valores:
 - New (nuevo archivo)
 - Update(actualización de un archivo)
- <Título del commit>: lo más descriptivo posible.
- <Cuerpo del commit>: se indicará de forma más detallada el commit.
- <Pie del commit>: indicará el issue al que hace referencia, de la siguiente forma: Closes ó Refers #(número de Issue).

Evidencias en el proyecto:

Update: solucionado la aparición de tildes en BD

Se ha añadido codificación utf8 en la conexión con la base de datos para que aparezcan correctamente las tildes.

Closes #18

Fuente: <https://github.com/EGC-GESTIONPROGRAMA-1718/program-manager-wpp/commit/4d3cb51d6fcf8baa938b83fb05b1e08b03e5e6bb>

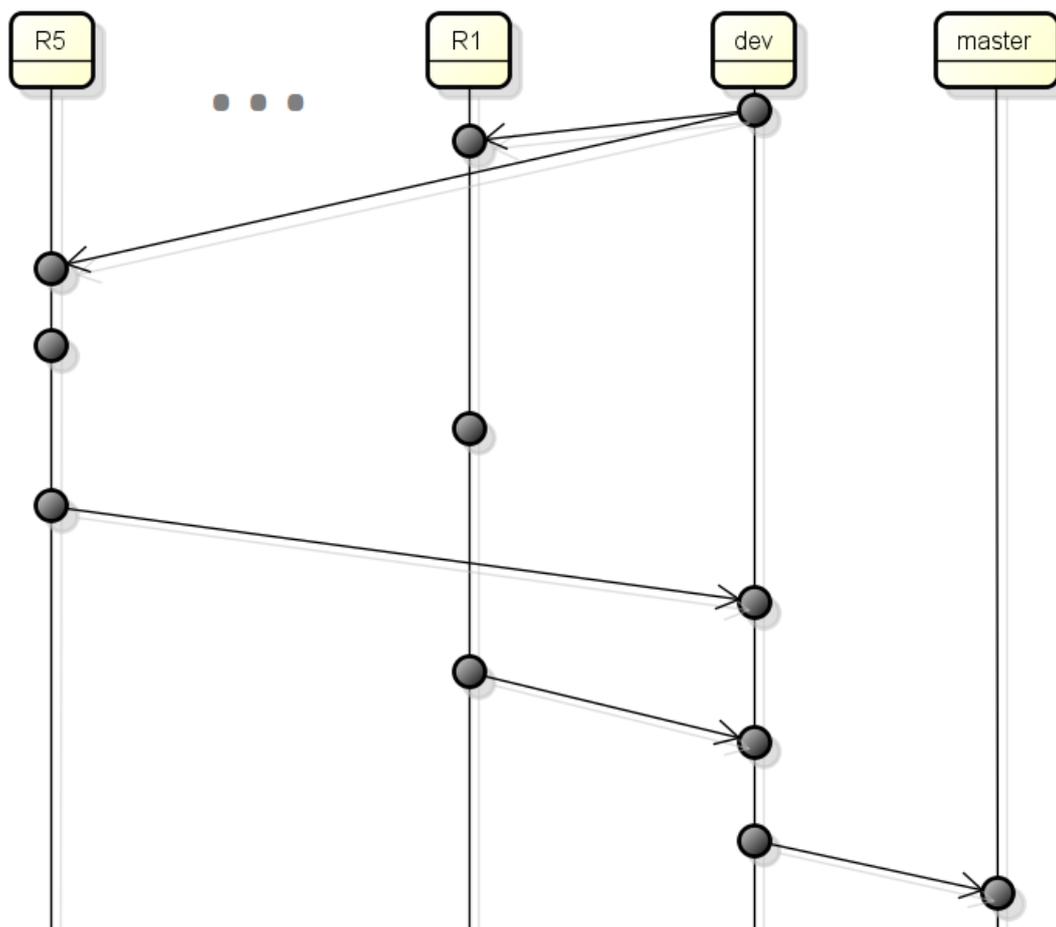
New: procesado del pre-programa

Añadida la funcionalidad para leer la hoja de cálculo del pre-programa desde Google Sheet e insertarlo en la base de datos.

Refers #7 #15

Fuente: <https://github.com/EGC-GESTIONPROGRAMA-1718/program-manager-wpp/commit/cb58159fc03ac9187a75d061b6620a742edec02f>

Gestión de ramas: Gestionaremos el proyecto añadiendo una rama para cada miembro del equipo, los progresos de estas ramas se volcarán a la rama de desarrollo y de esta se volcará a la rama master, que contendrá el código funcional estable.



Evidencia en nuestro proyecto:

Default branch				
<code>master</code>	Updated 20 hours ago by fraagucan2	✓	Default	Change default branch

Your branches				
<code>dev</code>	Updated 20 hours ago by fraagucan2	✓	0 0	New pull request
<code>fraagucan2</code>	Updated 20 hours ago by fraagucan2	✓	11 0	New pull request

Active branches				
<code>dev</code>	Updated 20 hours ago by fraagucan2	✓	0 0	New pull request
<code>fraagucan2</code>	Updated 20 hours ago by fraagucan2	✓	11 0	New pull request
<code>juarodreg</code>	Updated 21 hours ago by juanrr950	✓	2 0	New pull request

Stale branches				
<code>feltrimar</code>	Updated 4 months ago by jagalindo		31 0	New pull request
<code>fraordmed</code>	Updated 4 months ago by jagalindo		31 0	New pull request
<code>pabpinjim</code>	Updated 4 months ago by jagalindo		31 0	New pull request

Fuente: <https://github.com/EGC-GESTIONPROGRAMA-1718/program-manager-wpp/branches>

Gestión de la construcción e integración continua

Para la gestión de las dependencias utilizamos *Composer* y para integración continua *Travis CI*.

Fichero de gestión de dependencias en el proyecto (composer.json):

```
{
    "require" : {
        "google/apiclient" : "^2.0",
        "phpunit/phpunit" : "^6.5"
    }
}
```

Como se puede observar utilizamos dos dependencias: la API de Google y PHPUnit para las pruebas unitarias.

Fichero de integración continua dentro del proyecto (.travis.yml):

```

language: php

before_deploy:
  # Set up git user name and tag this commit
  - git config --local user.name "YOUR GIT USER NAME"
  - git config --local user.email "YOUR GIT USER EMAIL"
  - git tag "$(date +%Y%m%d')"

deploy:
  provider: releases
  api_key: ${GH_TOKEN}
  skip_cleanup: true
  on:
  tags: false

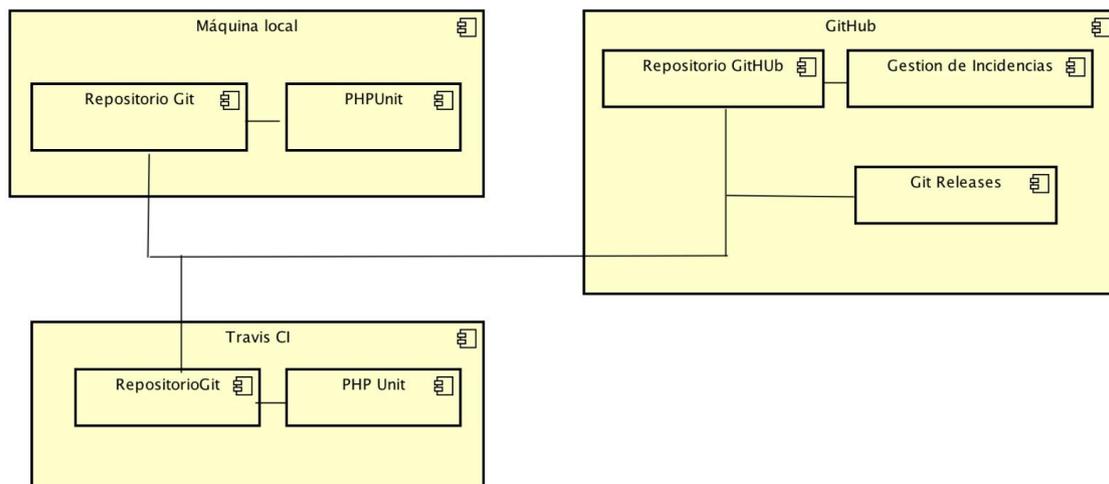
```

El mecanismo que seguimos para la integración continua es el siguiente: cada vez que se hace un push a master, Travis CI ejecuta los tests de PHPUnit que tenemos en el proyecto y si los pasa correctamente construye el proyecto y lo despliega en GitHub Releases.

Gestión de liberaciones, despliegue y entregas

Cada vez que se despliega el subsistema Travis asigna como nombre de versión una cadena de la siguiente forma: "YYYYMMDD" donde "YYYY" es el año actual, "MM" es el mes actual y "DD" el día actual en el que se realiza el despliegue.

Mapa de herramientas



Los cambios en los archivos del código se hacen desde la máquina local la cual se comunica con gitHub para enviar las modificaciones en forma de commits.

La gestión de incidencias se realiza desde la interfaz web de github, la cual se comunica mediante con el repositorio de código mediante referencias a las incidencias cuando se realiza un commit poniendo #ID_COMMIT.

La integración con gitHub se hace desde TRAVIS CI el cual lee el fichero .travis.yml escrito en la maquina local y solicitado al repositorio de gitHub, posteriormente despliega en gitHub Releases.

Ejercicio de propuesta de cambio

El cambio a realizar será que cuando se crea un programa nuevo en la base de datos se guarde en modo “draft” en lugar de en modo “final”.

Conclusiones y trabajo futuro

La realización de este trabajo ha sido una actualización del plugin de gestión de programa de las conferencias SPLC 2017, el cual leía un programa de conferencias realizado en excel y lo mostraba.

Con el nuevo plugin la lectura se hace a partir de hoja de Google Sheet la cual no es necesario tener una licencia de pago como en Microsoft Excel. También es una hoja más dinámica ya que pueden participar varias personas a la vez y no es necesaria la instalación de un software en el sistema.

Como mejora hemos realizado el almacenamiento del programa en una base de datos para poder tener varios programas con distintas versiones y en estado final o borrador.

Al realizar este plugin hemos aprendido a usar varias herramientas y tecnologías como relizar un plugin para Wordpress, consumir datos de la API de Google Sheet desde PHP. A parte del uso de gestión de incidencias de una forma estructurada, la gestión del código a traves gitHub y la integración continua con Travis CI.

Como mejora para un trabajo futuro proponemos:

- La autenticación de tipo OAuth2 de Google para que se pueden leer hojas privadas de un determinado usuario.
- Previsualizar de una forma rápida los programas en estado borrador (draft).
- Separación en diferentes páginas de la interfaz de administración del plugin.

Lecciones aprendidas

Llegados a este punto hemos entendido y aceptado que la gestión de la configuración es muy importante, ya que mejora el tiempo de realización de un proyecto. Como por medio de un sistema de control de versiones como git se mejora mucho la realización de código en equipo y la integración continua automatiza muchos procesos que de otra manera habría que hacerlos a mano y llevarían más tiempo.

A mitad del proyecto tuvimos que tomar la decisión de olvidarnos casi completamente del código heredado ya que por falta de documentación, código indocumentado y diferentes versiones sin saber cuál es la funcional, el plugin no funcionaba ni éramos capaces de hacerlo funcional. Por lo que una lección aprendida muy importante es documentar bien los proyectos