Documento del proyecto

EGC-G2

2017/2018

Autenticación

Jose Carlos García Rodríguez

Jose Ángel Domínguez Espinaco

Tania Salguero Álvarez

Damián Serrano Fernández

14/01/2018

Índice

Resumen

En este resumen vamos a describir como se ha llevado a cabo el proyecto de la asignatura de Evolución y Gestión de la Configuración (EGC).

El proyecto general consiste en integrar varios subsistemas para ofrecer una herramienta de voto online.

Al inicio del curso se propusieron todos los subsistemas que son necesario para llevar acabo la aplicación online de votos. Dando a elegir a cada grupo un subsistema. Cada subsistema engloba un proyecto ya definido del año anterior, dando a elegir a cada grupo, coger dicho proyecto o realizar uno nuevo desde el principio.

Nuestro grupo a elegido hacerlo desde el principio, ofreciendo al sistema de votación una login para el acceso al sistema y un registro de usuarios.

El objetivo principal es integrarnos con los demás subsistemas y hacer funcionar el sistema de votación en su totalidad. Para lograr dicho objetivo cada grupo con su subsistema deberá comunicarse y coordinarse con otros subsistemas que sean necesarios.

Nuestro grupo de trabajo ha valorado y tomado decisiones tanto internamente cómo con los demás grupos de los demás subsistemas teniendo que establecer procesos y herramientas ya constituidas para crear políticas de trabajo y seguir unas pautas para el desarrollo de nuestro subsistema y para la integración con los demás subsistemas.

Para ello hemos seguido las clases de teoría y las prácticas de la asignatura y además se han aportado herramientas de gestión de código, gestión de incidencias y depuración para desarrollar nuestro subsistema y poder realizar la integración continua con los demás subsistemas.

Introducción y contexto

Trabajaremos para hacer un portal de las jornadas de EGC a partir de un portal de un congreso que permita tener un sistema de participación para poder votar. Ref:

https://1984.lsi.us.es/wiki-egc/index.php/Lista_de_proyectos_a_realizar_17-18

Para el desarrollo del proyecto hemos llegado a un acuerdo para especificar cada punto importante, basándonos en la documentación proporcionada por los profesores.

- Elementos de control: se enumeran los elementos de gestión de la configuración y cómo se va a gestionar la configuración de cada uno de ellos.

- Entorno de desarrollo: describimos el entorno de desarrollo utilizado, la versión usada y el procedimiento para instalar el sistema para que tenga funcionabilidad con el resto de subsistemas. Hemos usado:

- Lenguaje/Herramienta:

    - Plugin para Eclipse Neon.3 (4.6.3) de Python 2.7
    - Django 1.11,
    - Django REST API
    - Sistema de gestión de bibliotecas: pip
    - Bibliotecas: MySQL, Django REST API
    - Necesita Base de datos: Sí (MySQL)

- Gestión del código fuente: describe cómo gestionamos el código fuente durante el desarrollo del proyecto. Se creó un repositorio común con subgrupos, uno por cada subsistema para facilitar la integración o el trabajo de cada grupo en común con los demás subsistemas.

- Gestión de la construcción y de la integración: describe las técnicas de integración continua, principalmente usamos Travis CI para automatizar el trabajo de las tareas.

- Gestión del cambio, incidencias y depuración: se describen las herramientas y los procesos que se usan y como se usan.

- Gestión de liberaciones, despliegue y entrega: se describe el procedimiento usado para desplegar en una máquina los subsistemas de forma que estén integrados.

- Gestión de la variabilidad: se describen los mecanismos usados y los niveles en los que se gestiona la variabilidad.

- Mapa de herramientas: se describe las herramientas usadas en el nuestro proyecto.

El sistema a desarrollar consta de varios subsistemas que se dividen por cada grupo de clase formado en el aula, éstos son los subsistemas y enlaces a la wiki de la asignatura:

- Equipo de integración general - 17_18 - G2
- Gestión del programa - 17_18 - G2
- Gestión del registro - 17_18 - G2
- Gestión de visualización del programa - 17_18 - G2
- Gestión de integración con redes sociales - 17_18 - G2
- Autenticación - 17_18 - G2
- Administración de votaciones - 17_18 - G2
- Administración de censos - 17_18 - G2
- Cabina de votaciones - 17_18 - G2
- Cabina de telegram - 17_18 - G2
- Almacenamiento de votos - 17_18 - G2
- Recuento de votos - 17_18 - G2
- Visualización de resultados - 17_18 - G2

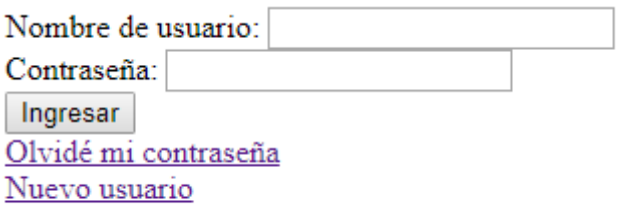
Nosotros nos encargaremos del subsistema de AUTENTICACIÓN, que consiste en la autenticación de los usuarios para acceder al sistema y poder realizar varias operaciones. También somos los encargados de realizar el registro de usuarios para los que no estén registrados en el sistema.

Descripción del sistema

Nuestro grupo se encargará del subsistema de Autenticación para el portal del congreso US.

El subsistema se encargará principalmente de dar acceso a los usuarios ya existentes y realizar un registro para los nuevos usuarios que quieran acceder al sistema de votación. Así como la recuperación de la contraseña si a un usuario se le olvida.

La ventana principal del subsistema de autenticación es un formulario que nos pide el usuario y la contraseña, que tendrá que estar previamente registrado en el sistema.

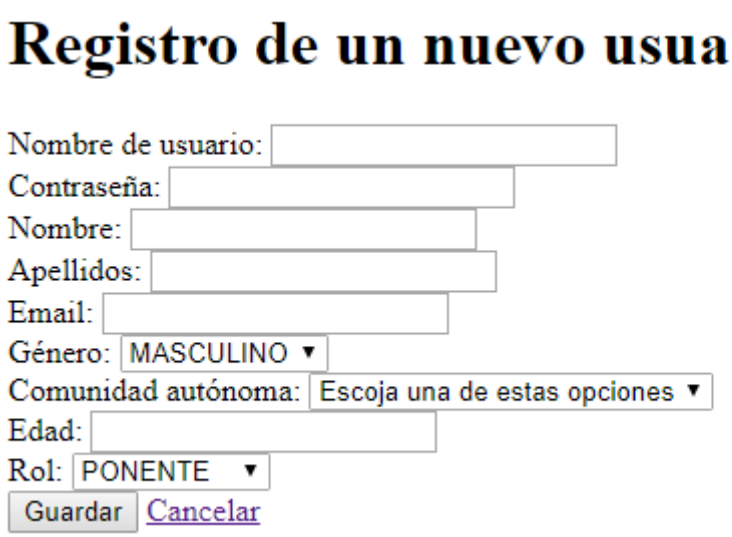


Una vez autenticado se guardan dos cookies en el sistema, una con identificador “user”, donde guarda el nombre de usuario autenticado y otra con el identificador “token” en la que se almacenará un token generado a partir de

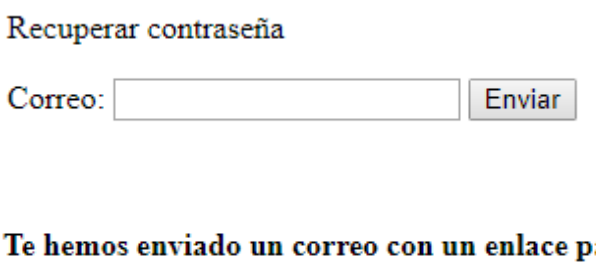
su nombre de usuario y contraseña.(Texto recogido de la memoria del año anterior).

Nuestro subsistema ofrece una API REST con peticiones GET para la verificación del usuario autenticado sea correcta. Las peticiones tienen el siguiente formato:

Registro en el sistema: Nuestro subsistema ofrece también un registro para usuarios no registrados, donde se les solicita una serie de datos.



Recuperar contraseña: El subsistema ofrece la opción de si a un usuario se le olvidó la clave poder recuperarla. Lo que hace es llevarte a un formulario donde introduciremos el email asociado a esa cuenta. Nos llegara un correo con un enlace para cambiar la contraseña y listo.



Planificación del proyecto

Para la planificación del proyecto hemos realizado al principio de cada Milestone una reunión para poner en común todas las tareas que hay que realizar en ese Milestone. Seguidamente se asignan a los miembros del equipo para que finalmente que el coordinador del grupo las suba como issue en GitHub. En términos generales el rol que ha desempeñado cada miembro del grupo ha sido el siguiente:

- Jose Ángel: Coordinador del proyecto, encargado del despliegue del sistema, de la configuración con la BD y responsable de la rama master.
- Tania: Encargada de la implementación de la API del proyecto.
- Jose Carlos: Encargado de la implementación de la API del proyecto.
- Damián: Encargado de las plantillas del proyecto y de la implementación de Travis.

Cabe destacar que algunos miembros han participado en los roles de otros miembros para ofrecer su ayuda.

Con respecto al registro del tiempo dedicado en cada tarea utilizamos Toggl.

Entorno de desarrollo

El entorno de desarrollo que hemos utilizado es Eclipse Neón junto con Django REST Framework en la versión Neon.3 Release (4.6.3), es interesante decir que utilizamos Python 2.7 como plugin de Eclipse y como biblioteca pip.

1. Primero, se debe descargar de internet Python 2.7 e instalarlo en la carpeta c:\Python27 puede hacerlo en la siguiente url: https://www.python.org/download/releases/2.7.7/
2. Segundo, en Windows , ir a Equipo (botón derecho) > Propiedades > Configuración avanzada del sistema > Variables de entorno… > En Variables del sistema Editar la variable Path (añadir ;C:\Python27\Scripts) .Se debe descargar de internet el archivo get-pip.py (saldrá un texto, deberá ‘guardarlo como’ para descargarlo), puede hacerlo en la siguiente url: https://pip.pypa.io/en/stable/installing/ y ejecutar en cmd, en el directorio donde descargarmos el archivo anterior, ‘get-pip.py’
3. Por último y después de instalar pip, por cmd y aconsejablemente en el directorio donde tenemos el proyecto escribimos ‘pip install’ seguido de los siguientes comandos:

- django==1.11
- djangorestframework==3.7.3
- django-rest-auth==0.9.2
- django-allauth==0.34.0
- numpy==1.13.3
- pytz==2017.3

Ahora vamos con la instalación de PyDev que será nuestro IDE para Python en Eclipse. (Importante: PyDev requiere Java 8 y Eclipse 4.6 (Neon) para soportar Python 2.6 o superior)

Para la instalación de PyDev seguimos el siguiente enlace: http://www.pydev.org/manual_101_install.html

Para comprobar que se instaló bien hacemos lo siguiente:

- Ir a 'window > preferences' y comprobar si hay una opción PyDev. Configurar el intérprete de Python 2.7 1. Ir a window > preferences > PyDev > Interpreter – Python
- Elegir el intérprete que hemos instalado, para ello basta pulsar Auto Config El Auto Config intentará encontrarlo (python.exe) en PATH, pero puede fallar, si asi es usted deberá digirirlo a la ruta donde se instaló el Python.exe
- Los paths se encuentran en SYSTEM PYTHONPATH. El System libs debe contener al menos los directorios Lib y Lib/site-packages.
- Pulsar Aceptar.

Gestión del cambio, incidencias y depuración

se describirá el proceso de gestión de incidencias, cambio y depuración que ha seguido en el proyecto. También deberá enlazar partes de su proyectos donde se evidencie que ha seguido ese proceso.

La gestión de incidencias debería contener explícitamente dos apartados. Uno de cómo se han gestionado la incidencias internas y otro el cómo se han gestionado y se ofrece protocolo para gestionar las incidencias externas tanto las recibidas como las que se reporten a otros subsistemas.

Cuando una incidencia esté relacionada con un commit, señalar el commit dentro de la propia incidencia y viceversa.

Gestión del código fuente

La gestión del código de nuestro proyecto la hacemos a través de Git, ya sea por comandos o por su interfaz (hemos llegado a utilizar GitHub Desktop y GitKraken, llegamos a utilizar este último porque nos daba más facilidad a la hora de ver archivos en conflictos) Para realizar un commit en nuestro proyecto debe de haber corregido, eliminado, añadido u optimizado algún archivo, ya sea archivo de desarrollo, integración, etc. La estructura para realizar un commit es la siguiente: Título del commit: tipo: aquí ponemos el título Cuerpo del commit: aquí describimos el commit. Pie del commit: Closes #<número de la incidencia en GitHub> No siempre se podrá poner el pie del commit, ya que no siempre está relacionado con alguna incidencia. El commit se puede realizar a dos ramas:

- Mi propia rama: cada componente del grupo tiene su rama propia, además de dos ramas más que son dev (desarrollo) y master. Si lo subo a mi rama es que no ha habido una característica completa y funcional añadida (por parte del desarrollo) o modificación en archivos importantes como Dockerfiles para que funcione el despliegue.
- Rama dev: si se sube aquí es porque se tiene una nueva funcionalidad completa o modificación correcta en archivos importantes para el proyecto.

Una vez algo está en dev, el coordinador del grupo es el encargado de pasarlo a master cuando compruebe que efectivamente todo está correcto en los archivos de la rama dev.

Gestión de la construcción e integración continua

Nuestra aplicación va a ser desplegada en un servidor remoto, por lo que para la integración continua usaremos Travis CI.

El funcionamiento de Travis CI es recoger el código de nuestra aplicación , lo compila y realiza una serie de test para probar el correcto funcionamiento. Seguimos una serie de pasos para configurarlo.

Nos registramos en la página oficial y luego integración nos dio permisos para el repositorio en común EGC-G2-Trabajo-1718 donde esta nuestro repositorio Autenticacion para poder hacer las pruebas pertinentes.

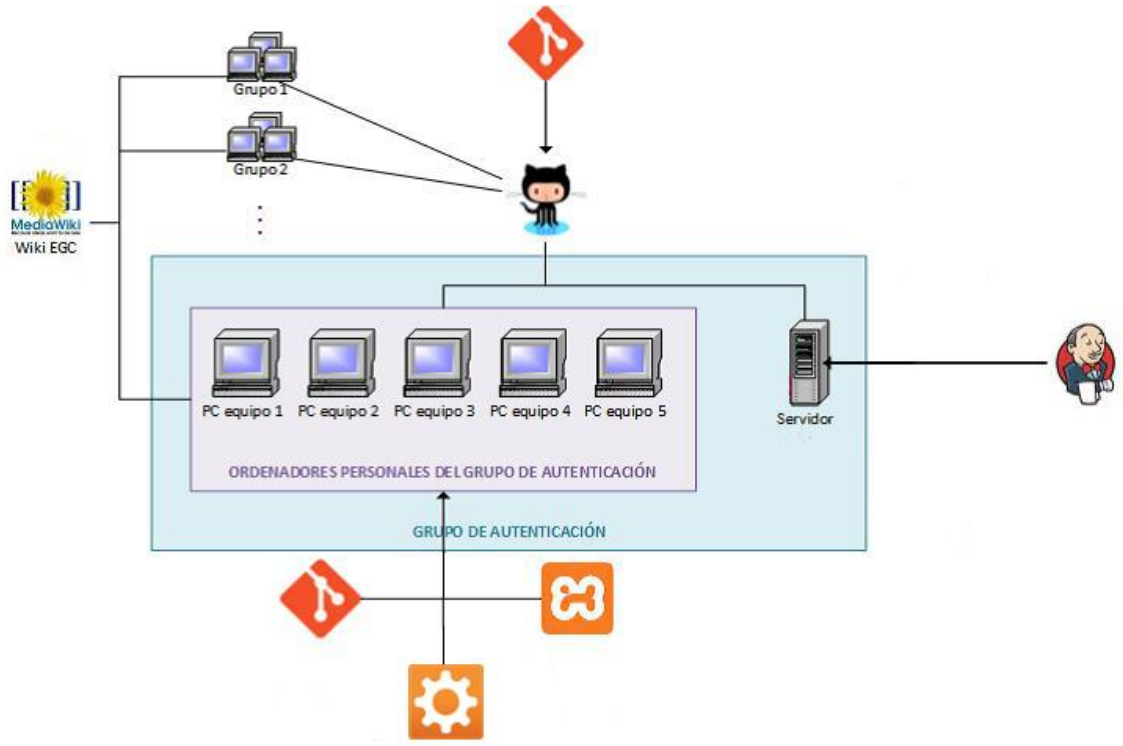
Una vez tengamos los permisos, debemos crear un archivo llamado .travis.yml en la raíz de nuestro proyecto, donde contiene una serie de comando para hacer funcionar los test. Integración nos dio una plantilla donde solamente debíamos añadir el lenguaje, la versión y el script de las pruebas de nuestro proyecto.

Gestión de liberaciones, despliegue y entregas

se explicarán los procesos, técnicas y herramientas para la gestión de las liberaciones, despliegue y entregas del proyecto. Evite poner información de las herramientas en sí que se pueda encontrar en fuentes bibliográficas o internet. Si es del caso haga referencia a ellas. Céntrese en los aspectos particulares de su proyecto en concreto:

- Proceso definido para las liberaciones: Hemos utilizado GitHub con una rama por cada miembro del equipo y dos ramas generales llamadas dev y master. En la rama dev se comprueba el proyecto pasándole las pruebas definidas anteriormente y en master se sube el proyecto con las pruebas realizadas con éxito. Para realizar una liberación de una versión estable utilizamos el objeto "tag" que nos proporciona git de manera que "taggeamos" el commit de la rama master que consideremos que sea la versión estable de nuestro sistema.
- Proceso definido para el despliegue: Hemos utilizado la herramienta Docker para realizar el despliegue de manera local y en remoto. Los pasos para ejecutar nuestro proyecto en local con Docker son los siguientes:
    - $ sudo docker-compose build
    - $ sudo docker-compose run --rm djangorest python manage.py makemigrations principal
    - $ sudo docker-compose run --rm djangorest python manage.py migrate
    - $ sudo docker-compose up
- Proceso definido para las entregas: Para las entregas de cada Milestone mostrábamos la parte del proyecto que en ese momento estaba más avanzada cogiéndola desde GitHub
- Política de nombrado e identificación de los entregables: Para el nombrado de las versiones estables de nuestro proyecto lo haremos por su fecha de liberación, es decir, si liberamos el proyecto el 11/01/2018 se llamará "autenticacion 20180111".

Mapa de herramientas



Todos los grupos hemos utilizado la wiki de la asignatura de EGC, por lo que todos tenemos una base en común donde describiremos las características del proyecto.

Nuestro grupo ha realizado el proyecto desde cero en Python.

Para la gestión de código fuente hemos utilizado git como gestor de GitHub para alojar el código del proyecto.

Todos los grupos tenemos un repositorio común llamado “EGC-G2-Trabajo-1718”, en este repositorio solo los coordinadores de cada grupo tienen acceso.

De este repositorio sale otro repositorio para cada grupo (EGC-G2-Trabajo-1718/subsistema), donde subsistema es el subsistema que ha elegido cada grupo para la aplicación online de votos.

Nosotros nos hemos organizado de tal forma que tenemos una rama cada integrante y solo el coordinador puede subirlo a master.

Para la incidencias se utiliza GitHub mediante el modulo “issues”.

Hemos preparado una maquina virtual con la configuración preparada para que el proyecto funcione. Las herramientas principales son:

- Docker
- Lenguaje/Herramienta: Plugin para Eclipse Neon.3 (4.6.3) de Python 2.7, Django 1.11, Django REST API

- Sistema de gestión de bibliotecas: pip
- Bibliotecas: MySQL, Django REST API
- Necesita Base de datos: Sí (MySQL)

Ejercicio de propuesta de cambio

Nuestro cambio será añadir un atributo más al modelo Usuario como prueba del proceso que se debería realizar. Los pasos a seguir serían:

- El coordinador del grupo asigna una issue en github con lo que se debe realizar y se la asigna al responsable (en este caso sería un responsable de desarrollo)
- El componente del grupo responsable de esta issue recibirá el aviso de que se le a asignado dicha issue por el correo que tenga asignado en github.
- El desarrollador deberá hacer un merge con la rama master, si es que no tiene dicha versión ya, y empezará a trabajar en dicha issue. Mientras esta desarrollando pueden aparecerle errores con lo que primero se mira en las issues finalizadas por si ese mismo error estuviera ahi resuelto, si no estuviera, debería de hacer una nueva issue con dicho error. (Y cuando se llegara a resolver dar la solución y cerrar la issue)
- Una vez esta comprobado que se funciona por el desarrollador, se realiza un merge a la rama dev, donde el coordinador hará la última comprobación de dicha mejora para ver si es apta para ir a la rama master.

Conclusiones y trabajo futuro

Con el presente trabajo hemos aprendido principalmente a trabajar de forma coordinada con un gran número de personas, pues no solamente hemos tenido que trabajar codo a codo todos los miembros del grupo, sino que también hemos tenido que ponernos de acuerdo con el resto de grupos de la clase.

Por otro lado, la experiencia realizando el trabajo nos ha ayudado a ser más autodidactas, pues hemos tenido que aprender muchas nuevas tecnologías (algunas elegidas por nosotros y otras impuestas desde otros grupos) de forma autónoma.

Además, hemos conocido a fondo una herramienta muy útil e indispensable para gestionar los trabajos, GitHub, la cual nos permite una gestión del código y de las incidencias de forma mucho más profesional y eficiente de lo que veníamos haciéndolo anteriormente. También nos ha proporcionado un control del estado del proyecto y de los errores que nos han ido surgiendo, los cuales hemos detallado en las issues para no tener que volver a solucionarlos en un futuro, algo que nos ha servido para ahorrar tiempo.

Los aspectos que creemos que serían útiles implementar en un futuro, es la posibilidad de añadir códigos captcha, lo cual aumenta la seguridad e impide el

acceso a bots. Además sería de utilidad implementar la posibilidad de autenticarse a través de redes sociales, como por ejemplo Facebook, pues sería mucho más sencillo para el usuario utilizar su cuenta ya existente, que tener que crear una nueva cuenta.