RECUENTO DE VOTOS

grupo 1

ID de opera: 24

Miembros del

- Ronda Lucena, Arturo
- Roldán Sánchez, Pablo
- Rodríguez Martin, Javier
- De Tovar Vázquez, Ernesto
- Acha Burgos, Álvaro
- Huertas Vera, Francisco Javier

Enlaces de interés:

- repositorio de código
- sistema desplegado
- Espacio de Opera

Descripción del sistema:

El objetivo del sistema es calcular los resultados de las votaciones. Para ello hemos tomado como referencia el proyecto del año anterior [http://1984.lsi.us.es/wiki-egc/index.php/Recuento_y_modificaci%C3%B3n_G1_1617].

Se ha decidido implementar en Python usando el Framework Django en vez de JavaScript, como estaba el anterior, debido a que el equipo tenía más experiencia en esas tecnologías. El uso de Django nos permite un fácil manejo de los modelos de datos y de la implementación de una Api-Rest para la consulta de resultados.

El sistema consta de distintos tipos de recuentos como "Aprobación simple", "Aprobación múltiple" y "Borda simple".

Respecto al resto de subsistemas interactuamos con dos. En primer caso con "Almacenamiento de Votos" de los cuales obtenemos las votaciones mediante una conexión SQL, y "Visualización de Resultados" el cual consume nuestra Api-Rest para obtener los resultados de las votaciones.

Los cambios desarrollados al proyecto son:

-El recuento "aprobación múltiple" y "Borda Simple", anteriormente solo se tenía el Aprobación simple
-La implementación en Python, anteriormente estaba desarrollado en JavaScript
-El despliegue en Docker, anteriormente no se usaba esta tecnología

Planificación del proyecto

El proyecto se ha ido planificando de forma incremental a lo largo de los milestones, manteniendo como guía las decisiones de la primera reunión de equipo (ver Diario de grupo).

En cuanto al reparto de tareas, este se ha llevado acabo mediante el uso de las issues en GitHub y para complementar se ha usado el servicio de mensajería instantánea de Telegram, Las issues están todas recogidas en el repositorio facilitado por el equipo de integración (ver en el enlace). El equipo ha empleado este sistema tanto para comunicar las necesidades del proyecto en forma de issues, tanto para los problemas y/o soluciones encontradas en forma de comentarios dentro de las ya existentes.

Las issues han sido asignadas a miembros concretos del equipo en función de sus capacidades para el desarrollo de la tarea y de su disponibilidad en el momento de necesidad, todo esto bajo el apoyo del servicio de mensajería instantánea para facilitar su organización. Además, en las diferentes issues de forma secundaria han podido participar otros miembros del equipo aparte del o de los principales.

Entorno de desarrollo

Para el entorno de desarrollo propusimos un estándar en nuestra primera reunión, de forma que aseguramos que el proyecto funcionaría en cualquier máquina. Para crear este entorno estándar basta con seguir los pasos del apartado "Resumen de creación del entorno estándar", expuesto más abajo.

Nuestro estándar consistió en trabajar desde una máquina virtual con una distribución Debian de linux instalada, la cual se puede obtener desde https://drive.google.com/file/d/1ybgjy01ZwroYWk0jbF7EKWkhY-366oDI/view. La clave de acceso para el usuario debian es debian. La máquina cuenta con un entorno de desarrollo integrado PyCharm y con la versión de python 2.7. Además de estas herramientas, es necesario instalar git y pip (sudo apt-get install git python-pip) que usaremos para clonar el proyecto y para gestionar las dependencias respectivamente.

Una vez preparado este entorno, podemos clonar el repositorio (git clone https://github.com/Proyecto-EGC-G1/RecuentoVotos-EGC-G1) e instalar las dependencias del proyecto. En la rama master podemos encontrar distintas versiones listas para producción, mientras que en la rama developer encontraremos la última versión en desarrollo del proyecto. Por tanto, si lo que se desea es instalar el entorno de desarrollo, debemos obtener esta última rama. Para instalar las dependencias usamos pip desde la carpeta donde hemos clonado el repositorio (pip install --no-cache-dir -r requirements.txt). Con esta gestión de las dependencias, siempre encontraremos el archivo requirements.txt las versiones adecuadas de cada dependencia en el estado del proyecto clonado. Las dependencias actuales son:
- Django versión 1.11.7
- djangorestframework versión 3.7.3
- mysql-connector versión 2.1.6

Si se han seguido estos pasos, podremos ejecutar el servidor desde la carpeta donde hemos clonado el repositorio (python manage.py runserver), aunque hasta que no instalemos el subsistema de la base de datos no podremos usarlo.

Instalación del subsistema de la base de datos
Para configurar este subsistema del que depende nuestro sistema vamos a usar docker-compose. Para ello tendremos que instalar tanto docker como docker-compose en nuestra máquina virtual. Si estamos instalando docker en nuestra propia máquina, deberíamos elegir aquí nuestro sistema y seguir los pasos indicados https://www.docker.com/community-edition. Si en cambio estamos usando el estándar de desarrollo propuesto, introducimos en la máquina virtual lo siguientes comandos para ello:
wget https://download.docker.com/linux/debian/dists/stretch/pool/stable/amd64/docker-ce_17.12.0~ce-0~debian_amd64.deb
sudo dpkg -i docker-ce_17.12.0~ce-0~debian_amd64.deb

Ahora podemos eliminar el paquete docker-ce_17.12.0~ce-0~debian_amd64.deb. Para usar docker tendremos que usar sudo cada vez, o introducir a nuestro usuario en el grupo de docker y reiniciar la sesión. Instalamos docker-compose:
sudo apt-get install curl
sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

Terminada la instalación, podemos acceder mediante http://localhost:8000/recvotes/{pollID} (por ejemplo, http://localhost:8000/recvotes/10).

Resumen de instalación del entorno de desarrollo estándar
- Descargar https://drive.google.com/file/d/1ybgjy01ZwroYWk0jbF7EKWkhY-366oDI/view
- Importar a VirtualBox o a otro software de virtualización.
- Iniciar con credenciales debian / debian
- Abrir consola (click en Konsole, que se encuentra en el escritorio), copia en ella el siguiente bloque e introduce de nuevo la contraseña cuando la pida:

```
cd ~ \
 && sudo apt-get install -y git python-pip curl \
 && git clone https://github.com/Proyecto-EGC-G1/RecuentoVotos-EGC-G1 \
 && cd RecuentoVotos-EGC-G1 \
 && pip install --no-cache-dir -r requirements.txt \
 && wget
https://download.docker.com/linux/debian/dists/stretch/pool/stable/amd64/docker-
ce_17.12.0~ce-0~debian_amd64.deb \
 && sudo dpkg -i docker-ce_17.12.0~ce-0~debian_amd64.deb \
 && rm docker-ce_17.12.0~ce-0~debian_amd64.deb \
 && sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-
compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose \
 && sudo chmod +x /usr/local/bin/docker-compose
```

- El proceso de instalación ha terminado. Puedes editar el proyecto con instalando tu editor favorito o con PyCharm, que se encuentra instalado en la máquina. Para iniciar el sistema copia en el terminal el siguiente bloque:

```
cd ~/RecuentoVotos-EGC-G1 \
 && sudo docker-compose up -d \
 && python manage.py runserver
```

- Instala tu navegador favorito o abre firefox. Puedes enviar peticiones a http://localhost:8000/recvotes/{pollID}. (Por ejemplo, http://localhost:8000/recvotes/10.)

Gestión del cambio, incidencias y depuración

Para la realización de los cambios necesitamos tener una comunicación con todos los componentes del grupo. Para la comunicación informal del grupo de trabajo utilizamos la aplicación de mensajería Telegram, al igual que con el grupo de integración u otros grupos. Las decisiones más importantes las tratamos en reuniones presenciales.

Por otro lado, utilizaremos las issues de GitHub para la gestión de incidencias. Esta funcionalidad la utilizaremos para asignación de una tarea, para arreglar algún problema, para la asignación de código, entre otras. Para ello crearemos las incidencias con un título corto y entendible, le añadiremos una descripción en la que detallaremos el problema en cuestión o la tarea a realizar, asignaremos al miembro del equipo que sea el encargado de realizarla y utilizaremos las etiquetas que hemos añadido al conjunto por defecto que nos ofrece git.

Para el control de las versiones tenemos dos ramas en el repositorio, master y develop. La rama develop la utilizamos para desarrollar el código, en esta rama pueden subir todos los miembros, en cambio en la rama master, que la utilizaremos para subir el código y los archivos que estén en la versión final, sólo puede realizar cambios el coordinador, Arturo Ronda.

Cuando un miembro recibe una incidencia realiza la depuración de la siguiente manera:

1. Recibirá un email de GitHub notificándole de la asignación de la incidencia.
2. Analizará la incidencia, escribirá un comentario al que la redactó si necesita más información y también se comunicará mediante Telegram para que haya más control con la incidencia.
3. Con la información necesaria, el encargado de la incidencia se pondrá a realizarla y cambiará la label de tipo Status a In Progress.
4. Cerraremos las issues referenciándolas en los commits que estén relacionados con la issue.

Gestión del código fuente

La gestión del código se realiza mediante GitHub en el cual se tiene alojado un repositorio[https://github.com/Proyecto-EGC-G1/RecuentoVotos-EGC-G1], no se tiene una herramienta específica para trabajar con git, algunos usamos el terminal[https://git-scm.com/] pero tambien se usa GitKraken[https://www.gitkraken.com/]

Para realizar un commit, la persona debe haber terminado un único bloque de trabajo proveniente de una issue que tenga asignada, este bloque ya puede ser la issue al completa o una parte de ella.

El mensaje del commit debe tener un Título que defina en que consiste el bloque de trabajo anteriormente mencionado, si es necesario añadir un texto explicativo si hay algo que mencionar como tener algún problema o si ha realizado algún cambio fuera de la zona esperada del proyecto. Por último, deberá hacer referencia a la issue sobre la que está trabajando y si el commit finaliza la issue se debe cerrar en el commit.

Ejemplo de referencia a issue: https://github.com/Proyecto-EGC-G1/RecuentoVotos-EGC-G1/commit/3ce98197d405367dc37edfbd721d61821db84697

Ejemplo de cierre de issue: https://github.com/Proyecto-EGC-G1/RecuentoVotos-EGC-G1/commit/96d95430dcc7cfb2cb57b5d12c3239ad877183de

En el repositorio se tienen dos ramas, develop y master.
En la rama develop se realizan los commits que tienen que ver con el avance del proyecto, esta rama es en la que trabaja todo el equipo simultáneamente.

En la rama Master solo se usa para releases del proyecto provenientes de la rama develop. De esto se encarga el director del proyecto, el cual una vez se haya alcanzado una versión estable del proyecto la pasará a esta rama, y le añadirá el Tag correspondiente con el identificador de la release.

Gestión de la construcción e integración continua

Utilizaremos Travis CI tanto para la construcción automática como para la integración continua. Con esta herramienta realizamos la construcción cada vez que se hace un commit al repositorio de GitHub.

Para la integración continua, Travis verifica que después de cada commit el proyecto funcione correctamente. Para ello, instala todas las dependencias, ejecuta los scripts de la base de datos, ejecuta los test de la aplicación comprobando que todos funcionan correctamente y crea la imagen en Docker.

Gestión de liberaciones, despliegue y entregas

Proceso definido para las liberaciones:

Para la liberación de una nueva versión el coordinador del equipo evaluara si el cambio acumulado hasta el commit ha considerar como nueva versión es completamente funcional y constituye un cambio suficiente. Además, solo el coordinador deberá crear nuevas versiones en la rama master salvo que le sea imposible realizarlo, para lo cual se delegará en otro miembro del equipo.

Proceso definido para el despliegue:

El proceso de despliegue se realiza de forma automática gracias a la implementación de Travis en el proyecto como ya se ha explicado en puntos anteriores, por lo tanto, el proceso de despliegue viene definido por el proceso de liberación de versiones.

Proceso definido para las entregas:

Las entregas se realizarán a través del repositorio de GitHub asignado por el equipo de integración (ver enlace).

Política de nombrado e identificación de los entregables:

Los entregables o versiones del proyecto serán identificados mediante Tags con el formato numérico “A.B.C” colocados por el coordinador del equipo aumentando uno de los valores según lo siguiente:

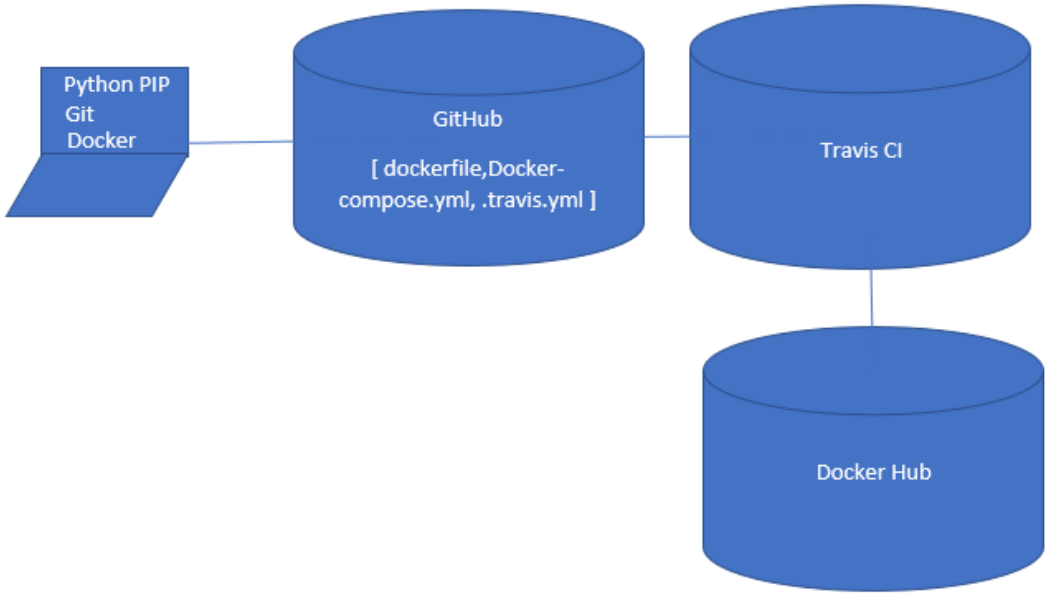
> A -> implica un cambio tan grande en el proyecto que habrá una perdida de compatibilidad respecto a las versiones anteriores.
> B -> implica un cambio funcional en la aplicación añadiendo algo nuevo.
> C -> implica un arreglo sobre las funcionalidades existentes en el proyecto pudiendo ser simplemente una mejora sobre alguna de ellas.

Mapa de herramientas

Para alojar el código en remoto y llevar a cabo la gestión de incidencias se ha utilizado la plataforma de desarrollo colaborativo GitHub, usando git como software de control de versiones.

Para gestionar la construcción se ha usado PIP como sistema de gestión de paquetes.
Se ha usado Travis CI como servicio de integración continua. La configuración del mismo se encuentra en el archivo ".travis.yml", alojado en el repositorio de código.

Por último, para automatizar el despliegue y llevar a cabo la integración con el resto del sistema, se ha usado Docker. La configuración del subsistema desarrollado está presente dentro de los archivos "dockerfile" y "docker-compose.yml", alojados ambos en el repositorio de código.



Ejercicio de propuesta de cambio

Supongamos que desde visualización quieren recibir el json con otro formato e integración lo aprueba. Por ejemplo, en vez de recibir el resultado de cada QuestionOption como:
"result": {"quantity": 2}
hayan decidido que, como el objeto resultado es simple y solo tiene un atributo en este momento, es mejor tenerlo de esta otra forma:
"result": 2

El primer paso sería abrir la issue en nuestro repositorio, https://github.com/Proyecto-EGC-G1/RecuentoVotos-EGC-G1/issues. Lo apropiado sería que alguien de integración o de visualización se encargara de redactarnos el problema y el cambio solicitado, pero lo más probable es que esta propuesta de cambio se hiciera de manera más informal y que nosotros mismos abriéramos la issue.

Una vez redactada la issue, uno de nuestros miembros se la asignaría o quizás la asignara el project manager al miembro que crea mejor capacitado para resolverla. Dicho miembro haría una pequeña tarea de investigación para ver cómo puede parsear con el framework usado una propiedad de un objeto relacionado como si fuera la propiedad del objeto que lo relaciona, leyendo la documentación de la misma (http://www.django-rest-framework.org/api-guide/relations/). Bajo decisión de ese miembro queda explicar los cambios que cree conveniente hacer en la issue antes de hacerlos o proceder a cambiar el código, pero una vez que se ponga con esta tarea debe moverla en el tablero kanban para reflejar que está en progreso.

Para realizar el cambio, si fuera a necesitar varios días, o varios momentos en el día, lo apropiado sería que se creara una rama para ello o se hiciera en una rama personal del miembro del equipo. De esta forma cualquiera puede acceder al último instante de este cambio, aunque no esté completo, sin que ello implique que lance el proceso de Travis, el cual comprobaría una rama en un instante de desarrollo inacabado y publicando una imagen errónea del proyecto en DockerHub.

En este caso, el cambio en el código consistiría simplemente en cambiar la línea 11 del archivo recuento/serializers.py a result = serializers.PrimaryKeyRelatedField(read_only=True). Cuando ya está terminado el cambio y se ha comprobado, se sube a la rama developer, bien directamente o bien haciendo merge de la rama en la que se ha hecho el cambio. El commit debe indicar que cierra la issue para que efectivamente esta quede cerrada y se actualice la tarea en el tablero kanban.

Al hacer esta acción, Travis pasará los test y generará y subirá la nueva imagen de Docker.
El nuevo cambio se podrá observar en el entorno de producción en cuanto se actualice la imagen en este. https://g1recvotos.egc.duckdns.org/recvotes/1/

Conclusiones y trabajo futuro

Como conclusión en este trabajo nos hemos dado cuenta como equipo lo complejo que es llegar a un objetivo común con otros equipos, mientras cada uno se dedica a realizar su propio proyecto, esto nos lo hemos encontrado en situaciones que requerían la atención de el resto de equipos “para beneficio del avance de nuestro grupo”. No es que critiquemos con esto a nuestros compañeros, sino que incluso a nosotros nos ha llegado a costar en varias ocasiones, pues no es sencillo entender las necesidades de otro equipo cuando no conoces en muchas ocasiones como están realizando su parte.
Esto nos ha llevado a darnos cuenta de lo fundamental que es compartir información con otros equipos y mantener al grupo global informado de la situación individual de cada uno, no solo mediante el uso de reuniones, sino mediante la publicación de documentos explicativos de la situación tecnológica y del progreso de cada parte.

Como mejoras al proyecto que dejamos al próximo año enumeramos las que debido al desarrollo desde cero de muchos equipos no hemos podido implementar:

- Descifrado de la información de los votos (requiere de un cifrado previo)
- Sistemas de recuento Borda funcionales ya que el simple esta implementado, pero no ha podido ser llevado a la practica por incompatibilidad con el modelo de datos actual.
- Uso de Json en lugar de conectar con la base de datos de integración evitando así conflictos de modelado en el desarrollo de nuevos sistemas de recuento.
- Sistema de autenticación de usuarios antes de realizar un recuento.
- Estadísticas relativas al resultado de la votación.