ALMACENAMIENTO DE VOTACIONES

Documento del proyecto

Enlaces de interés:

https://pypi.python.org/pypi/almvotes/1.0 - URL de PyPi que contiene la librería
https://github.com/Proyecto-EGC-G1/Almacenamiento-Votos-EGC-G1 - Repositorio de GitHub
http://opera.eii.us.es/egc/public/trabajo/ver/id/93 - Portal Ópera de nuestro grupo

ID Opera: 20 – Grupo 1

Almagro Martos, Héctor
Calbet González, María Victoria
González González, Juan Pablo
Rodríguez Aguilar, Daniel
Sánchez Paredes, Juan Carlos
Vázquez Zambrano, Pablo

14/01/2018

Índice de contenidos

1. Resumen

Almacenamiento de votos es un proyecto que nace con el objetivo de ser una capa abstracta dentro del sistema de votación, que gestione el correcto guardado de las votaciones que los usuarios realicen cuando estén dando uso al sistema, es decir, cuando haya una votación abierta en la que los ciudadanos estén citados a votar.

Ante el problema de la seguridad que debe existir dentro de un sistema de votaciones para que este sea verídico, almacenamiento de votos consigue procesar, cifrar y guardar el voto correctamente dentro de la base de datos del sistema de votaciones. Gracias al lenguaje de programación Python y al sistema de cifrado RSA, se consigue resolver esta problemática planteada, con un resultado más que satisfactorio.

2. Introducción y contexto

Almacenamiento de votos es una capa que convive junto con las demás del sistema de votaciones, encontrándose esta por debajo de la cabina de votaciones (encargados de crear las votaciones en las que lo ciudadanos estarán citados a votar). El código que hemos creado en Python proporciona la funcionalidad necesaria para que un usuario sea capaz de votar, que su voto esté cifrado, a buen recaudo, y que sea guardado correctamente en el lugar que corresponde de la base de datos del sistema central.

Almacenamiento es una de las capas más importantes, puesto que contiene algo crucial, como es la seguridad y fiabilidad de los resultados de una votación. Esto es muy importante en cualquier tipo de censo de población, donde unas elecciones, una encuesta o cualquier otro tipo de votación debe ser 100% legítima y fiable, asegurando que sea contado el voto de todos y cada uno de los ciudadanos y por supuesto sin duplicidades ni otro tipo de problemas. Votar es un derecho y que sea legítima dentro de un marco igualitario es democracia.

3. Descripción del sistema

El sistema que almacenamiento de votaciones propone y que se ha desarrollado, es un programa con el lenguaje de programación Python, encargado de realizar automáticamente todas las funciones que anteriormente se han descrito en el apartado "1. Resumen" y en el apartado "2. Introducción y contexto".

La estructura del sistema está compuesta por capas. En lo que nos concierne dentro de almacenamiento de votos, existen dependencias con autenticación, que estarán controlando la autenticidad de los usuarios, y con cabina de votaciones, los cuales estarán lanzando votaciones para los ciudadanos. La capa abstracta inferior a cabina, somos nosotros, encargados de recibir esos datos, procesarlos, y guardarlos.

Para ello, se ha creado una librería con los métodos necesarios de generación de voto, guardado de voto, cifrado RSA de voto y guardado de voto en base de datos, con las restricciones y excepciones pedidas. Al ser un sistema automatizado y que cabina de votaciones debe implementar en su subsistema, se llegó a la conclusión de que la mejor forma de implementarlo era convirtiendo almacenamiento de votos en una librería, la cual cabina integra en su subsistema y permite enlazar todo el proceso que sigue un voto hasta que llega a la base de datos central, se cifra y se almacena.

El uso de Travis CI es importante, dado que nos permite automatizar todo el proceso de construcción, integración y entrega del sistema nada más se genera un commit dentro de GitHub, permitiendo así tener todo el sistema actualizado y construido a tiempo.

Los cambios que se han desarrollado para el proyecto son los siguientes:

- Se decidió usar el lenguaje Python, en vez de PHP.
- Se decidió no usar código heredado.

4. Planificación del proyecto

Se dispone a presentar la planificación del trabajo y las tareas que se han realizado.

- **Milestone 1:**
    - Preparación del ecosistema de trabajo.
- **Milestone 2:**
    - Proceso para gestión de incidencias.
    - Proceso para la gestión del código.
    - Presentación del proyecto en clase.
- **Milestone 3:**
    - Proceso para gestión de código.
    - Implementación del subsistema.
    - Pruebas de funcionamiento.
    - Defensa del código.
- **Milestone 4:**
    - Conexión entre incidencias y código en GitHub.
    - Conversión del código en formato librería.
    - Automatización de la construcción mediante Travis.
    - Automatización de las pruebas mediante test de funcionalidad.
    - Automatización de la integración en el sistema.
    - Automatización de entrega y despliegue.
    - Desplegado de forma aislada.
    - Implementación de restricción: El sistema deberá verificar, antes de almacenar un voto, que no exista un voto del mismo usuario para la misma votación.
    - Implementación de restricción: El identificador anónimo de un usuario de un voto se generará a partir del nombre del usuario que realiza la votación y un sistema de encriptación propio, para mantener el anonimato del votante.
    - Tratamiento de errores: Hacer excepciones de las comprobaciones.
    - Conexión del programa con la base de datos central.

Puede encontrar una descripción mucho más completa de las tareas en el documento adjunto "Diario de grupo". El reparto de tareas por integrantes del grupo queda descrito también en el diario de grupo y en el repositorio de GitHub. Mas adelante, en la sección 6, encontrará las evidencias de dicho reparto de tareas y cómo acceder a él, así como su estructura.

5. Entorno de desarrollo

Se ha usado Eclipse como entorno de desarrollo integrado. En su versión Oxygen con el plugin PyDev, el cual permite el desarrollo en Pyhton.

La versión de Python utilizada es la 2.7.9, en adición del framework Django 1.8, y el framework Tastypie 0.14.0.

Para el montaje en local el subsistema se ha utilizado XAMPP en su versión 3.2.2 con una base de datos SQL MariaDB en su versión 10.0.31.

Para montar el entorno los pasos necesarios a seguir son los siguientes:

- Descargar e instalar Eclipse y el plugin PyDev, que permitirá trabajar con Python en Eclipse.
- Instalar XAMPP, Git, Python, Django y Tastypie.
- Una vez instalado XAMPP activar el servidor Apache. Activar también SQL con MariaDB.
- Abrir una ventana de comandos Git y escribir el siguiente comando: *git clone https://github.com/Proyecto-EGC-G1/Almacenamiento-Votos-EGC-G1.git*
- Importar el proyecto a Eclipse.
- Arrancar XAMPP, activar Apache y MariaDB.
- Importar el script SQL de la base de datos.
- Compilar el proyecto.

6. Gestión del cambio, incidencias y depuración

Descripción del proceso de gestión de incidencias que se ha elegido para el proyecto.

a) Gestión de incidencias internas:

Cada incidencia tendrá asociadas uno o más tags diferentes en función del tipo de tarea que haya que hacer con el fin de terminar la incidencia:

- **Bug**: Si consiste en arreglar un fallo.
- **Documentation**: Cuando haya que rellenar documentación.
- **Enhancement**: Si se trata de una mejora del subsistema.
- **Implementation**: Si la tarea a realizar consiste en una implementación del código de nuestro subsistema.
- **Organization**: Si consiste en una tarea que facilite el trabajo (por ejemplo, la creación del workspace).
- **Study**: Cuando haya que realizar un estudio, por ejemplo, sobre una herramienta o sobre un tipo de implementación en el código.
- **Urgent**: Si la tarea debe de realizarse lo antes posible.

Conforme avanza la incidencia, se deberá de actualizar mediante un comentario explicando brevemente el estado de la misma. Al igual, cuando se finalice una issue, tendrá que ir acompañada con un comentario que lo afirme, y puede que de forma detallada, si así se requiere. También, si la incidencia está relacionada con un commit, se enlazará issue y commit.

b) Gestión de incidencias externas:

Para las incidencias externas se ha decidido seguir exactamente el mismo formato descrito en el apartado anterior. Así mismo, las incidencias externas recibidas se notificarán a nuestro equipo de la misma forma. Puede ver una evidencia de ello en el siguiente enlace:
https://github.com/Proyecto-EGC-G1/Almacenamiento-Votos-EGC-G1/issues/33

Para evidenciar el uso de dicho sistema de gestión de incidencias, puede acceder a la sección Issues de nuestro GitHub desde el siguiente enlace:
https://github.com/Proyecto-EGC-G1/Almacenamiento-Votos-EGC-G1/issues

6.1. Gestión del código fuente

Principalmente, se encontrará la rama original/master, de la cual se crearán diferentes ramas en función de las actividades o incidencias (Issues) a realizar.

Es decir, habrá una rama por cada tarea o incidencia, nombrada por el nombre descriptivo de la tarea o por la incidencia o issue que se encarga de resolver la misma (Issue#xy). Una vez la tarea esté completamente realizada, se eliminará la rama del repositorio con el fin de no saturar el repositorio de las mismas.

Una vez que en una rama se ha terminado la funcionalidad y se ha probado que funciona, se procederá a hacer merge con la rama master.

6.2. Gestión de la construcción e integración continua

Para la construcción automática del proyecto se ha utilizado el servicio Travis CI, que permite una sincronización entre un repositorio alojado en GitHub y el sistema de construcción Travis.

Travis es capaz de detectar automáticamente el momento en el que se ha realizado un commit en el repositorio correspondiente. En el momento que eso ocurre, Travis inicia una construcción (build o compilación) en sus servidores de la red, de forma totalmente autónoma y transparente para el usuario.

Travis permite ver en todo momento mediante logs el estado de una construcción, permitiendo así comprobar una construcción correcta o con errores, y en el caso de existir algún error, la visualización de su traza para la posterior corrección del mismo.

6.3. Gestión de liberaciones, despliegue y entregas

- **Proceso definido para las liberaciones**: Subida al repositorio de GitHub y posterior construcción automatizada del proyecto mediante Travis CI. El siguiente paso es la construcción de la librería a través de PyPi.
- **Proceso definido para el despliegue:** Almacenamiento de votos es una librería, por tanto, no necesitamos un despliegue como tal en un servidor de aplicaciones. Cabina de votaciones es la encargada de ello tal como explicaremos en el siguiente punto.
- **Proceso definido para las entregas:** Cabina de votaciones obtendrá nuestra librería y la trasladará automáticamente a su proyecto, una vez ellos construyan su imagen, instalará en el dockerfile la librería de almacenamiento de votos.

- **Política de nombrado e identificación de los entregables:** No se ha determinado seguir una política de nombrado para los entregables ni para la identificación de los mismos. En todo caso se identifica con un nombre intuitivo al cual puede acceder cualquier miembro de los distintos equipos.

7. Mapa de herramientas

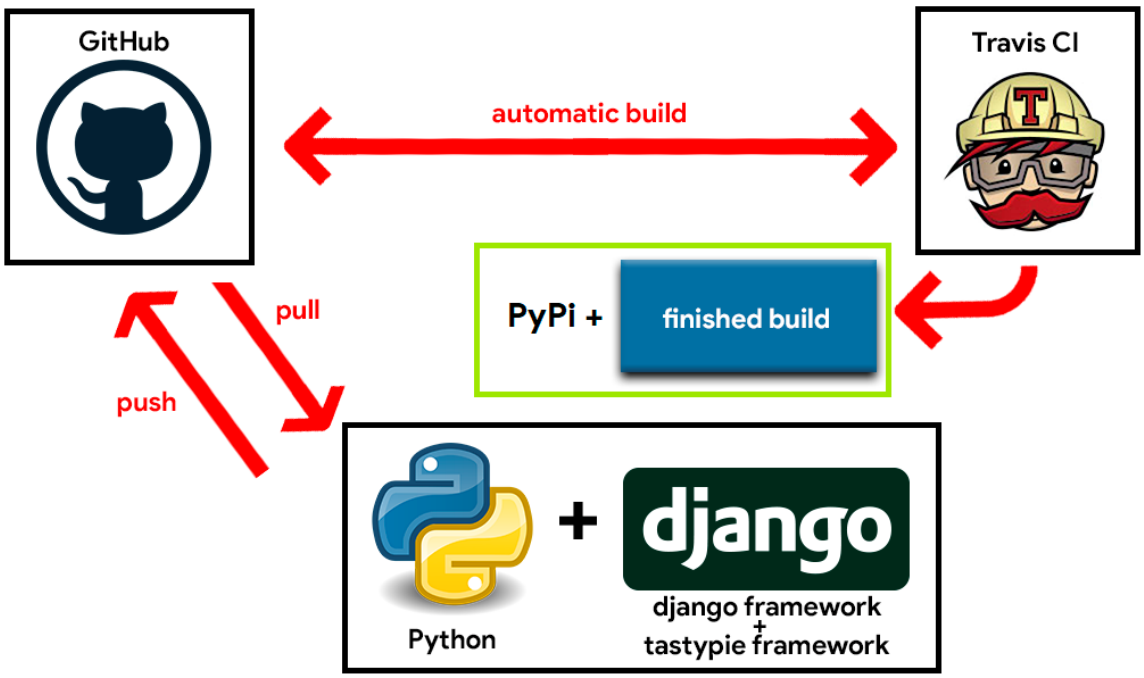


fig.1 – Mapa de herramientas.

Nuestro mapa de herramientas para la gestión de la configuración se constituye de un funcionamiento relativamente sencillo. Se comienza desde el desarrollo del proyecto en Python y los framework necesarios, dicha entidad establece relación bidireccional con el repositorio GitHub, con el cual el proyecto estará totalmente sincronizado y actualizado en la nube con el uso de Git.

A partir de aquí, se establece nuevamente una relación bidireccional entre GitHub y el constructor automatizado Travis CI, el cual leerá los datos de los commits en GitHub para automatizar la construcción del subsistema.

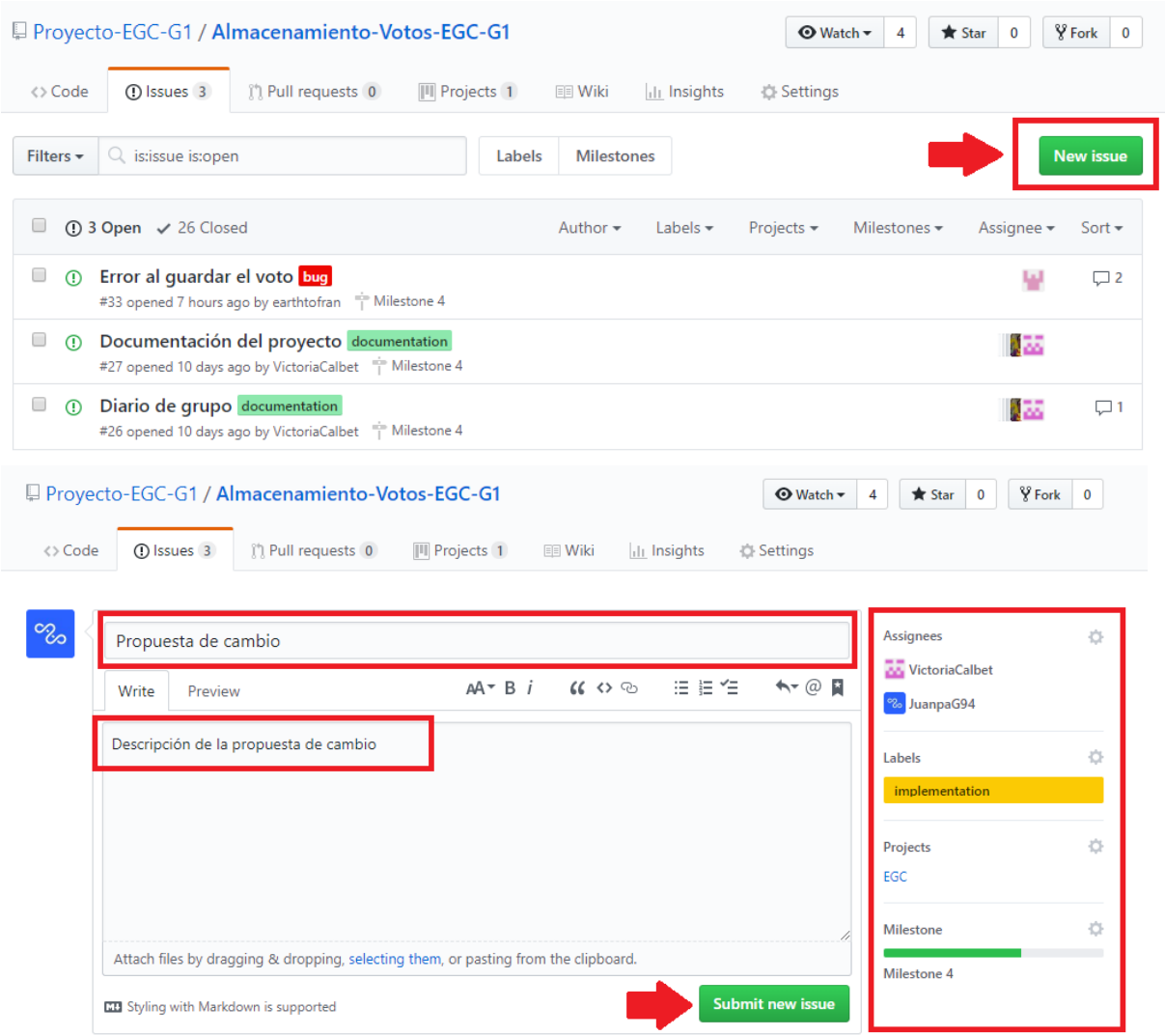
Terminada dicha tarea, se generará una build compilada, y PyPi se encargará de convertir dicha build en librería, para que el equipo de cabina de votaciones pueda implementarla en su subsistema.

8. Ejercicio de propuesta de cambio

La propuesta de cambio que proponemos es añadir un cierto límite de edad mínima requerida a la hora de votar. En todas las votaciones se puede requerir por ejemplo que el ciudadano votante sea mayor de edad. Nuestro ejercicio consistirá en añadir dicha restricción al sistema.

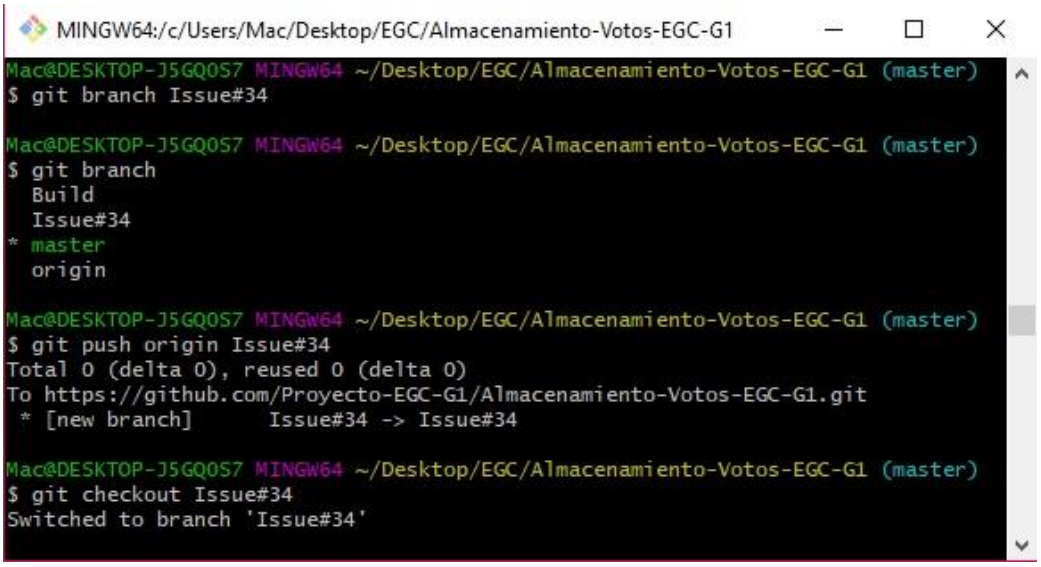
Los pasos que realizar son:

- Abrir una nueva Issue en el repositorio de Github:
    - Asignar la tarea a los miembros implicados.
    - Seleccionar el label necesario para la Issue.
    - Seleccionar el Milestone y proyecto al que pertenece.



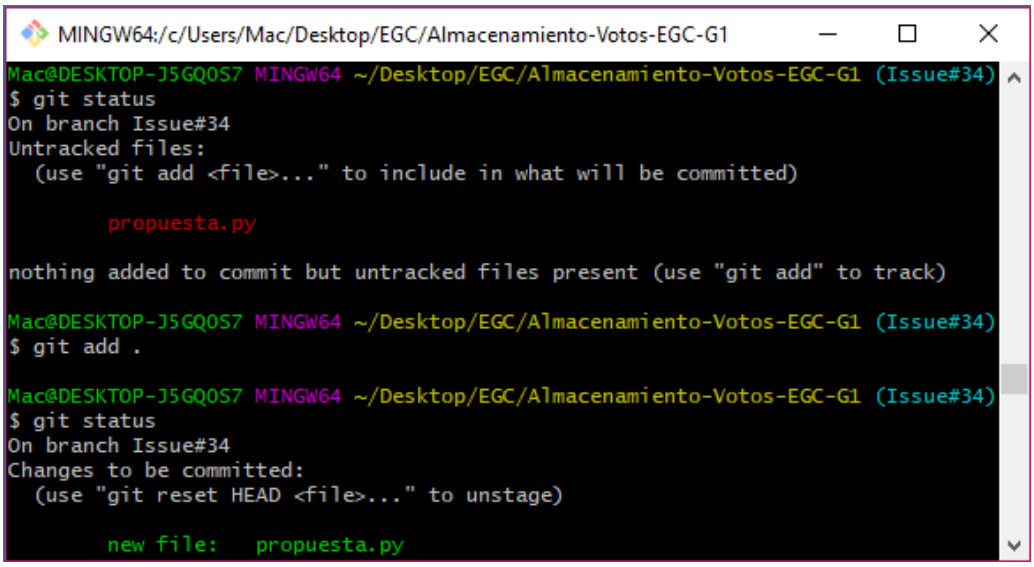
- Añadir la nueva Issue al tablero Kanban, según corresponda.

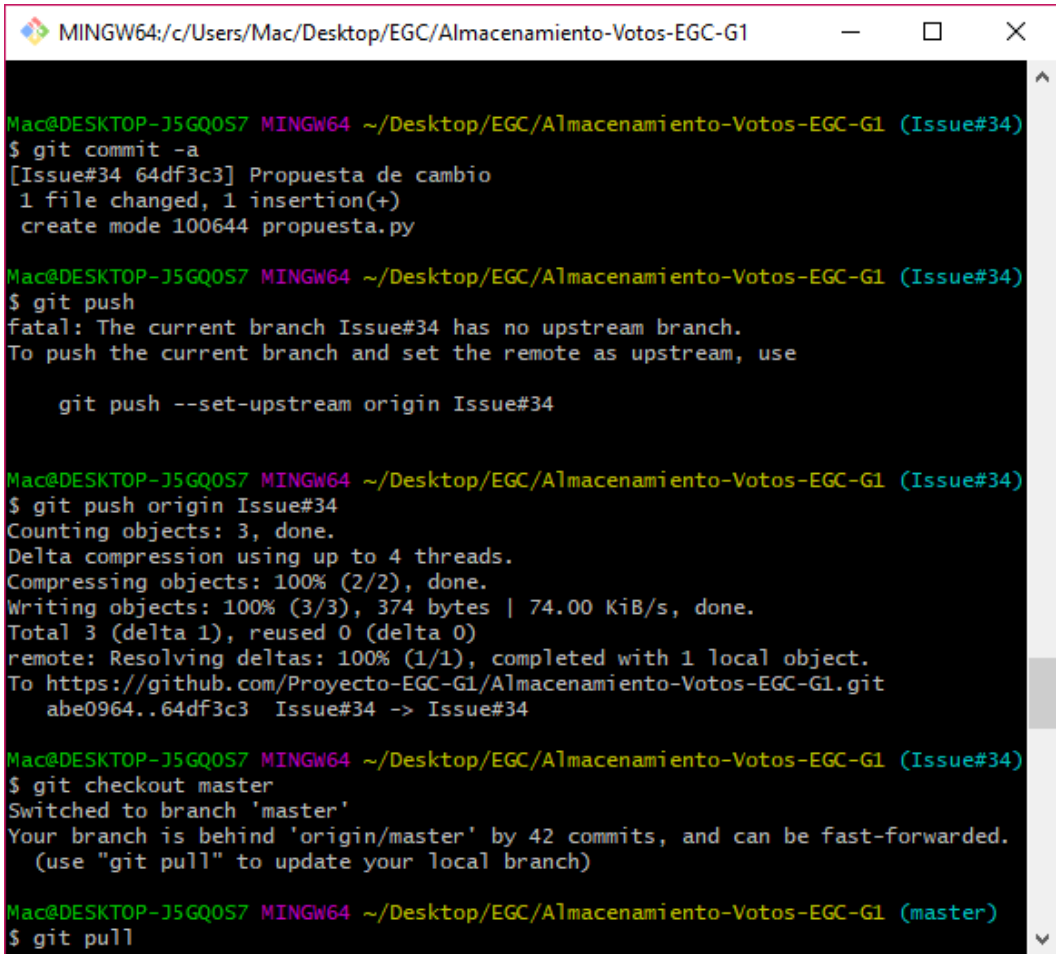
- Crear una nueva rama para la funcionalidad:
  - Comienzo de la implementación de la funcionalidad.

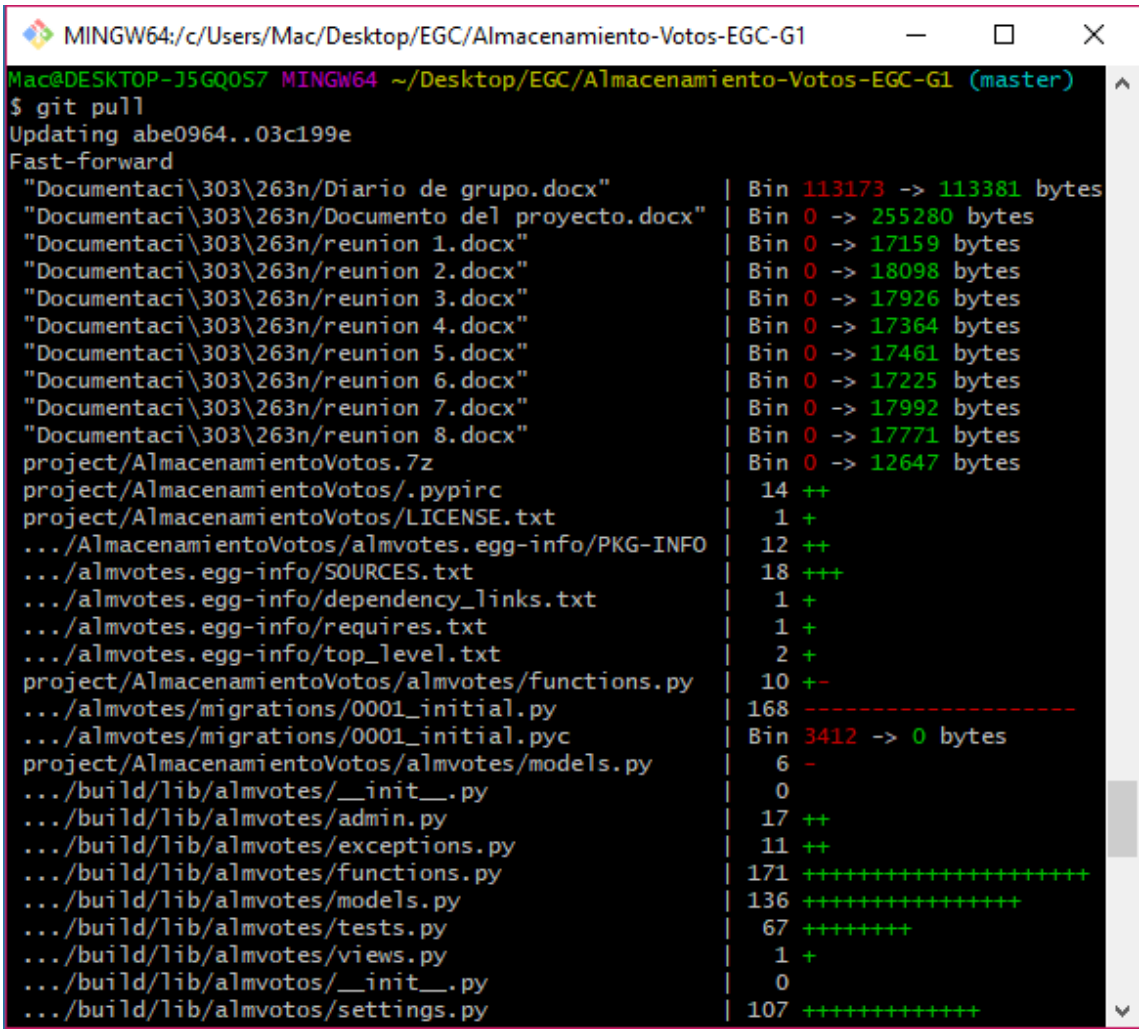


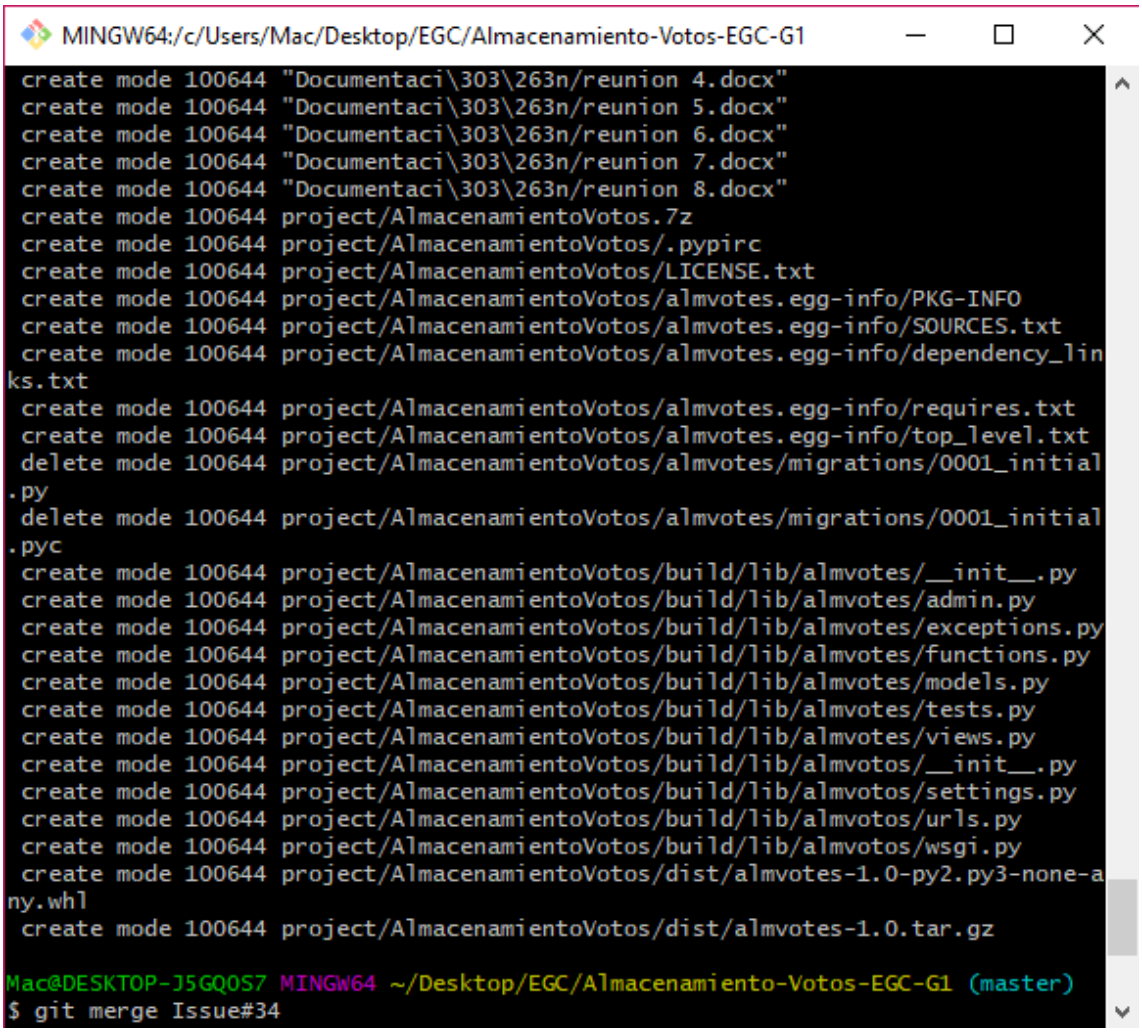
- Una vez está implementado se sube a la rama mediante un commit con el formato de commits definido. Se enlaza el commit con la Issue.

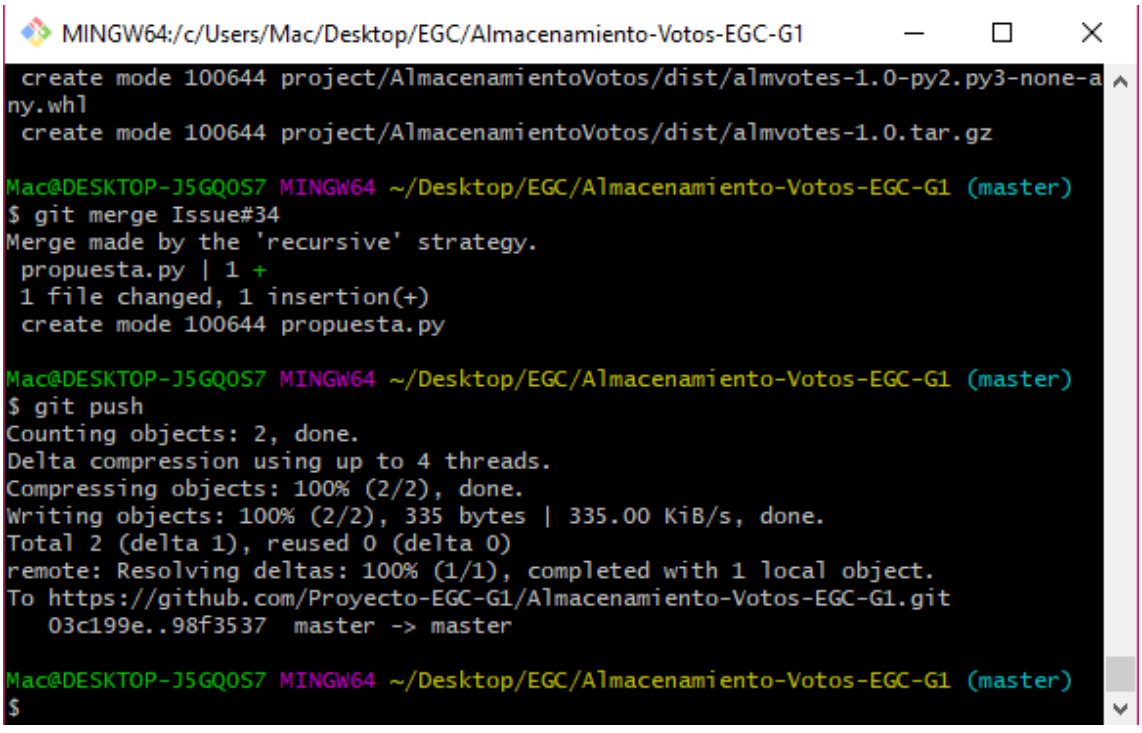
- Una vez se ha probado que funciona, será necesario hacer merge de dicha rama con la rama master, siguiendo el formato definido para los commits nuevamente



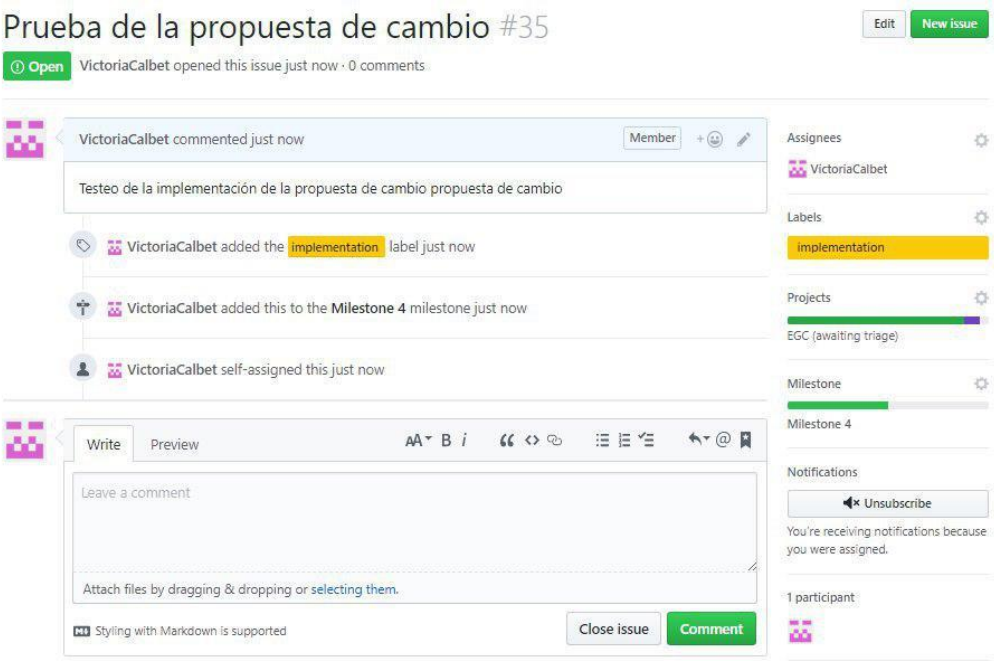


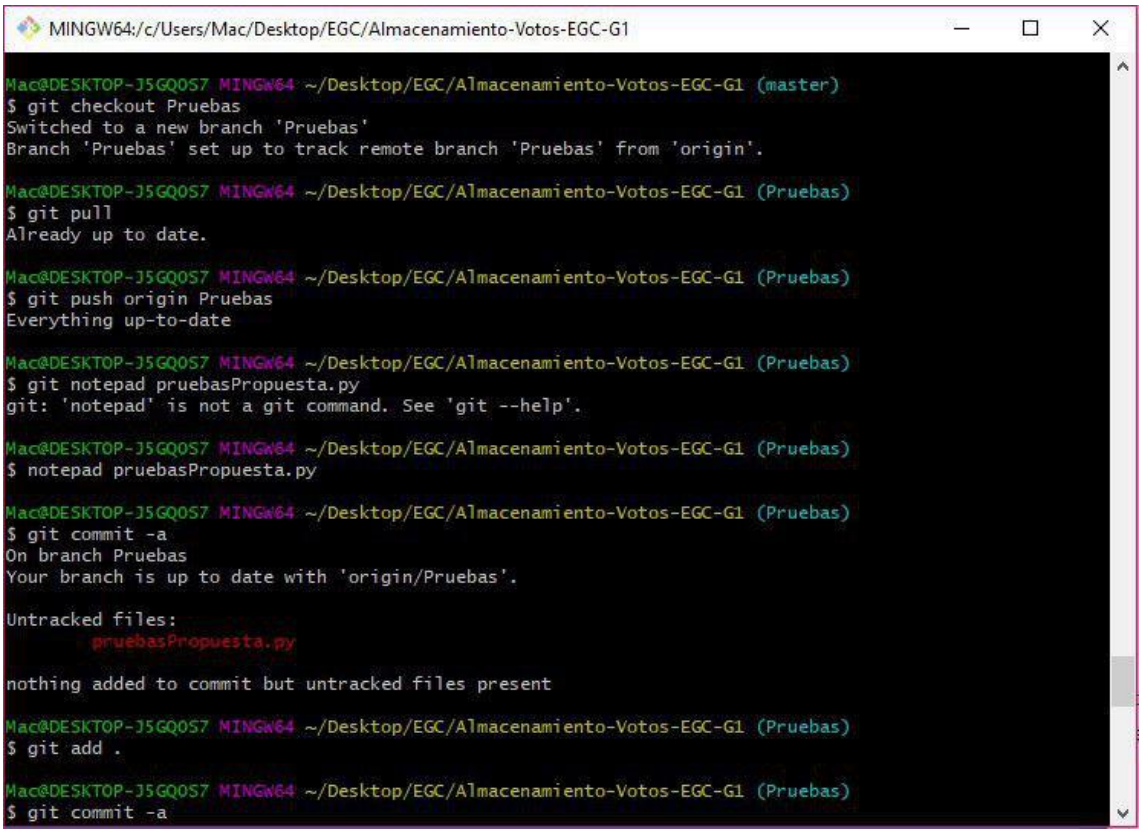

MINGW64:/c/Users/Mac/Desktop/EGC/Almacenamiento-Votos-EGC-G1
Mac@DESKTOP-J5GQOS7 MINGW64 ~/Desktop/EGC/Almacenamiento-Votos-EGC-G1 (master)
$ git pull
Updating abe0964..03c199e
Fast-forward
"Documentaci\303\263n/Diario de grupo.docx" | Bin 113173 -> 113381 bytes
"Documentaci\303\263n/Documento del proyecto.docx" | Bin 0 -> 255280 bytes
"Documentaci\303\263n/reunion 1.docx" | Bin 0 -> 17159 bytes
"Documentaci\303\263n/reunion 2.docx" | Bin 0 -> 18098 bytes
"Documentaci\303\263n/reunion 3.docx" | Bin 0 -> 17926 bytes
"Documentaci\303\263n/reunion 4.docx" | Bin 0 -> 17364 bytes
"Documentaci\303\263n/reunion 5.docx" | Bin 0 -> 17461 bytes
"Documentaci\303\263n/reunion 6.docx" | Bin 0 -> 17225 bytes
"Documentaci\303\263n/reunion 7.docx" | Bin 0 -> 17992 bytes
"Documentaci\303\263n/reunion 8.docx" | Bin 0 -> 17771 bytes
project/AlmacenamientoVotos.7z | Bin 0 -> 12647 bytes
project/AlmacenamientoVotos/.pypirc | 14 ++
project/AlmacenamientoVotos/LICENSE.txt | 1 +
.../AlmacenamientoVotos/almvotes.egg-info/PKG-INFO | 12 ++
.../almvotes.egg-info/SOURCES.txt | 18 +++
.../almvotes.egg-info/dependency_links.txt | 1 +
.../almvotes.egg-info/requires.txt | 1 +
.../almvotes.egg-info/top_level.txt | 2 +
project/AlmacenamientoVotos/almvotes/functions.py | 10 +-
.../almvotes/migrations/0001_initial.py | 168 ----------
.../almvotes/migrations/0001_initial.pyc | Bin 3412 -> 0 bytes
project/AlmacenamientoVotos/almvotes/models.py | 6 -
.../build/lib/almvotes/__init__.py | 0
.../build/lib/almvotes/admin.py | 17 ++
.../build/lib/almvotes/exceptions.py | 11 ++
.../build/lib/almvotes/functions.py | 171 +++++++++++++++++++++
.../build/lib/almvotes/models.py | 136 ++++++++++++++++
.../build/lib/almvotes/tests.py | 67 ++++++++
.../build/lib/almvotes/views.py | 1 +
.../build/lib/almvotos/__init__.py | 0
.../build/lib/almvotos/settings.py | 107 +++++++++++++


MINGW64:/c/Users/Mac/Desktop/EGC/Almacenamiento-Votos-EGC-G1
create mode 100644 "Documentaci\303\263n/reunion 4.docx"
create mode 100644 "Documentaci\303\263n/reunion 5.docx"
create mode 100644 "Documentaci\303\263n/reunion 6.docx"
create mode 100644 "Documentaci\303\263n/reunion 7.docx"
create mode 100644 "Documentaci\303\263n/reunion 8.docx"
create mode 100644 project/AlmacenamientoVotos.7z
create mode 100644 project/AlmacenamientoVotos/.pypirc
create mode 100644 project/AlmacenamientoVotos/LICENSE.txt
create mode 100644 project/AlmacenamientoVotos/almvotes.egg-info/PKG-INFO
create mode 100644 project/AlmacenamientoVotos/almvotes.egg-info/SOURCES.txt
create mode 100644 project/AlmacenamientoVotos/almvotes.egg-info/dependency_links.txt
create mode 100644 project/AlmacenamientoVotos/almvotes.egg-info/requires.txt
create mode 100644 project/AlmacenamientoVotos/almvotes.egg-info/top_level.txt
delete mode 100644 project/AlmacenamientoVotos/almvotes/migrations/0001_initial.py
delete mode 100644 project/AlmacenamientoVotos/almvotes/migrations/0001_initial.pyc
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotes/__init__.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotes/admin.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotes/exceptions.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotes/functions.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotes/models.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotes/tests.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotes/views.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotos/__init__.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotos/settings.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotos/urls.py
create mode 100644 project/AlmacenamientoVotos/build/lib/almvotos/wsgi.py
create mode 100644 project/AlmacenamientoVotos/dist/almvotes-1.0-py2.py3-none-any.whl
create mode 100644 project/AlmacenamientoVotos/dist/almvotes-1.0.tar.gz
Mac@DESKTOP-J5GQOS7 MINGW64 ~/Desktop/EGC/Almacenamiento-Votos-EGC-G1 (master)
$ git merge Issue#34

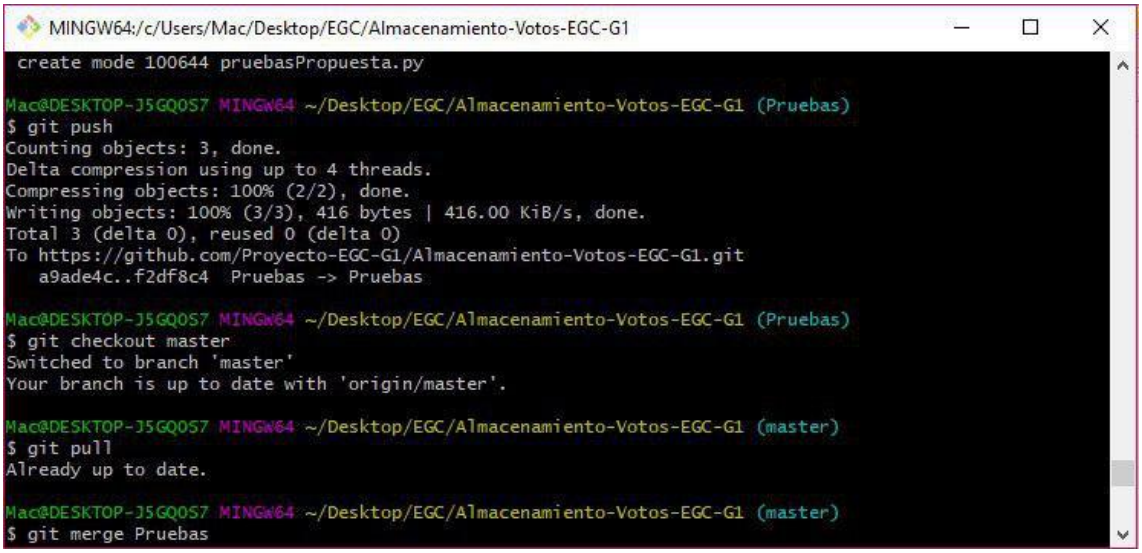

MINGW64:/c/Users/Mac/Desktop/EGC/Almacenamiento-Votos-EGC-G1
create mode 100644 project/AlmacenamientoVotos/dist/almvotes-1.0-py2.py3-none-any.whl
create mode 100644 project/AlmacenamientoVotos/dist/almvotes-1.0.tar.gz
Mac@DESKTOP-J5GQOS7 MINGW64 ~/Desktop/EGC/Almacenamiento-Votos-EGC-G1 (master)
$ git merge Issue#34
Merge made by the 'recursive' strategy.
propuesta.py | 1 +
1 file changed, 1 insertion(+)
create mode 100644 propuesta.py
Mac@DESKTOP-J5GQOS7 MINGW64 ~/Desktop/EGC/Almacenamiento-Votos-EGC-G1 (master)
$ git push
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 335 bytes | 335.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Proyecto-EGC-G1/Almacenamiento-Votos-EGC-G1.git
03c199e..98f3537 master -> master
Mac@DESKTOP-J5GQOS7 MINGW64 ~/Desktop/EGC/Almacenamiento-Votos-EGC-G1 (master)
$

- Realización de tests de funcionalidad en la rama de testing y creación de su correspondiente Issue.

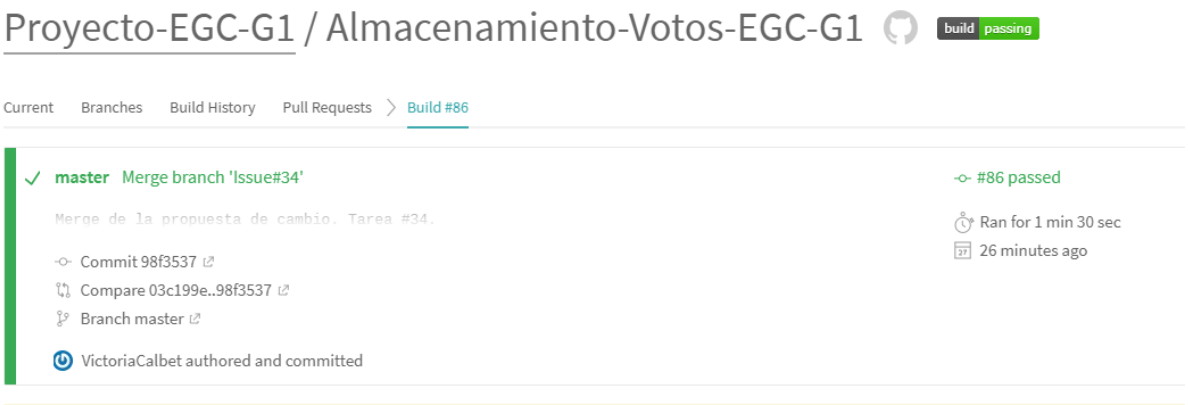




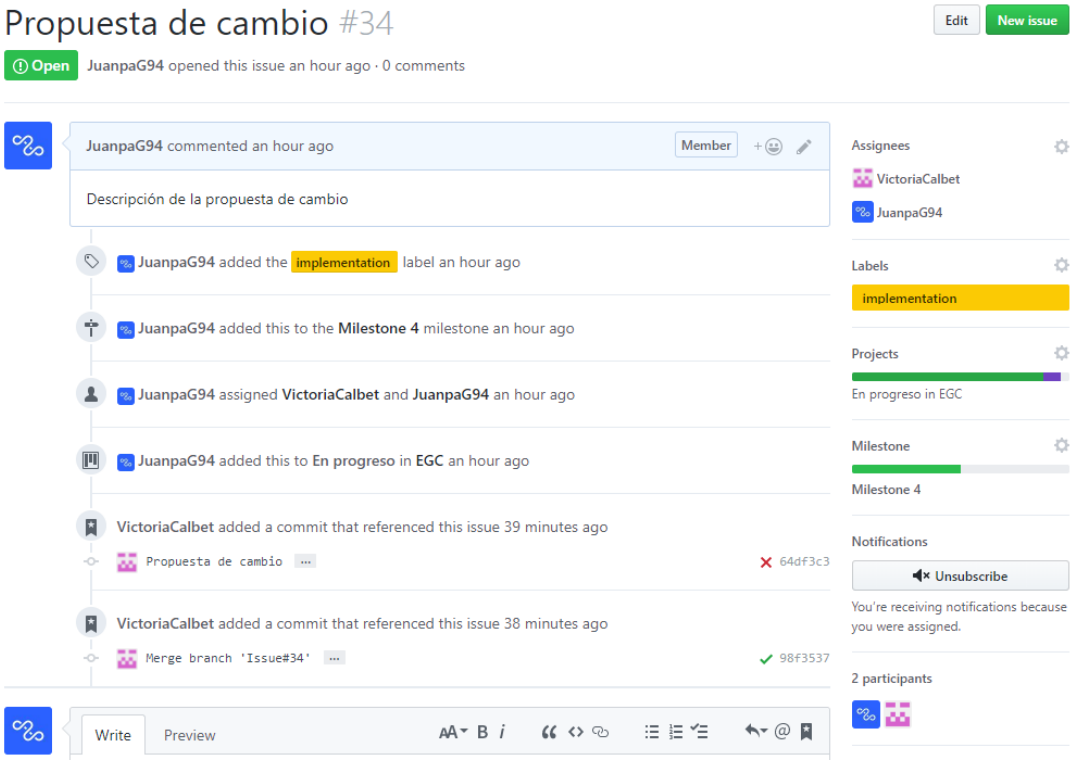
- Merge de los tests de funcionalidad con la rama master. Una vez realizado, hacer commit con el formato ya definido.

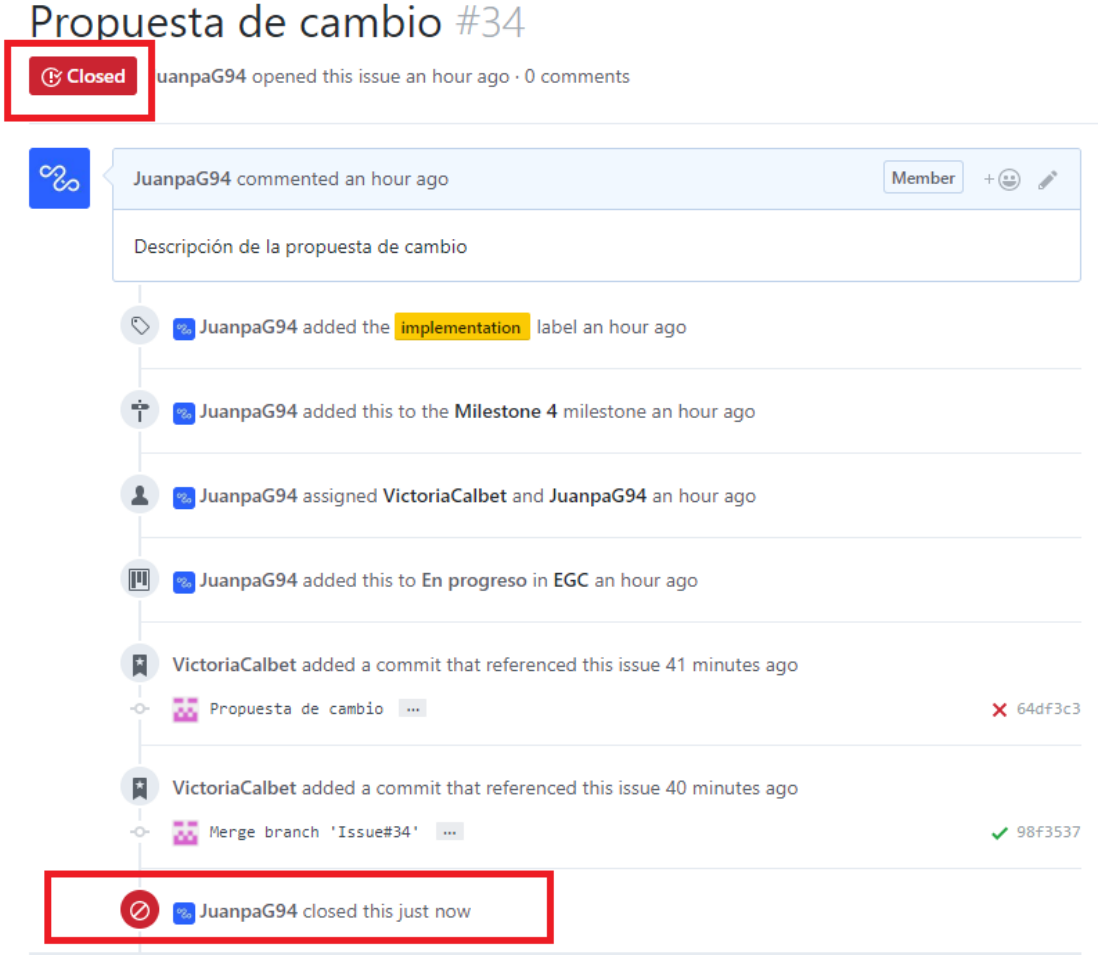


- Dirigirse a Travis CI para comprobar que se realiza la construcción del proyecto de manera correcta.

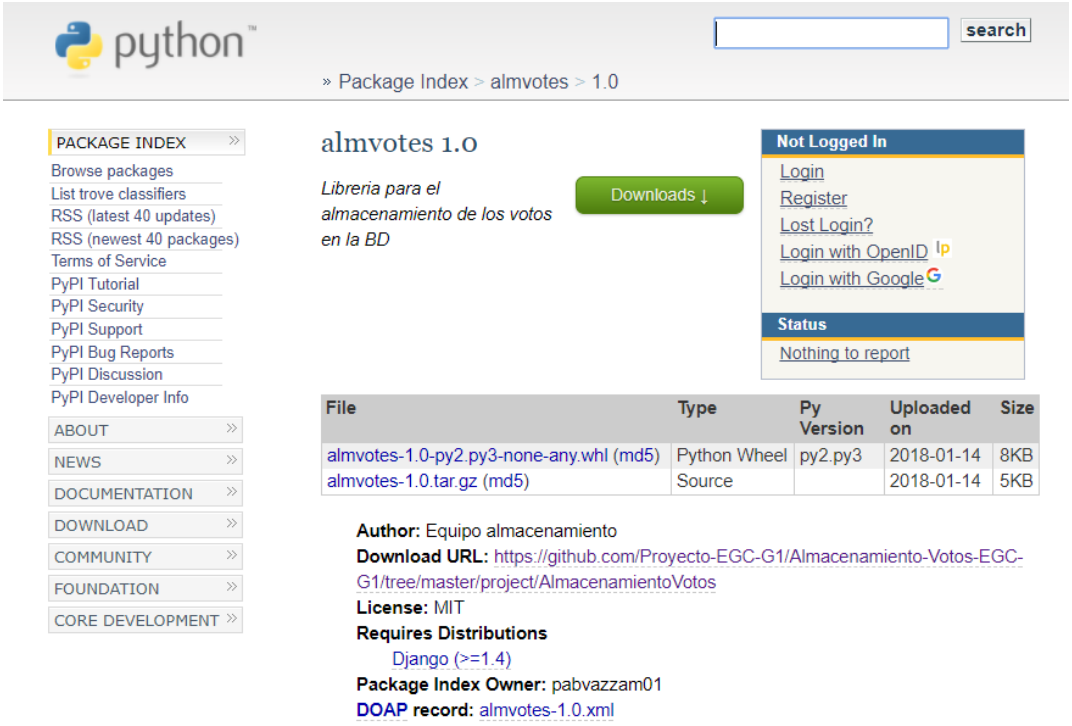


- Cerrar la Issue correspondiente (en el tablero Kanban se moverá automáticamente la Issue a la columna “Hecho”).


Propuesta de cambio #34
Edit
New issue
Open
JuanpaG94 opened this issue an hour ago · 0 comments
JuanpaG94 commented an hour ago
Member
Descripción de la propuesta de cambio
JuanpaG94 added the implementation label an hour ago
JuanpaG94 added this to the Milestone 4 milestone an hour ago
JuanpaG94 assigned VictoriaCalbet and JuanpaG94 an hour ago
JuanpaG94 added this to En progreso in EGC an hour ago
VictoriaCalbet added a commit that referenced this issue 39 minutes ago
Propuesta de cambio
64df3c3
VictoriaCalbet added a commit that referenced this issue 38 minutes ago
Merge branch 'Issue#34'
98f3537
Write
Preview
Assignees
VictoriaCalbet
JuanpaG94
Labels
implementation
Projects
En progreso in EGC
Milestone
Milestone 4
Notifications
Unsubscribe
You're receiving notifications because you were assigned.
2 participants


Propuesta de cambio #34
Closed
uanpaG94 opened this issue an hour ago · 0 comments
JuanpaG94 commented an hour ago
Member
Descripción de la propuesta de cambio
JuanpaG94 added the implementation label an hour ago
JuanpaG94 added this to the Milestone 4 milestone an hour ago
JuanpaG94 assigned VictoriaCalbet and JuanpaG94 an hour ago
JuanpaG94 added this to En progreso in EGC an hour ago
VictoriaCalbet added a commit that referenced this issue 41 minutes ago
Propuesta de cambio
64df3c3
VictoriaCalbet added a commit that referenced this issue 40 minutes ago
Merge branch 'Issue#34'
98f3537
JuanpaG94 closed this just now

- Abrir nueva Issue para la subida de la librería y realizar los pasos a seguir para rellenar la Issue, tal como se creó la anterior.
- Cuando todo se ejecute, debemos hacer una nueva subida de la librería en PyPi con los nuevos cambios.



- Cerrar la Issue correspondiente.

9. Conclusiones, trabajo futuro y lecciones aprendidas

a) Como conclusiones principales, podemos concretar que la creación de un sistema de votaciones es bastante complejo, hay que tener bien atadas las restricciones y casos de uso que pueden surgir en un sistema de este tipo, así como una fuerte barrera de seguridad que impida la manipulación de dichas votaciones, algo crucial para que una votación sea legítima y fiable.

b) También es necesario opinar sobre ciertos aspectos de la asignatura: La organización de una asignatura de esta índole es complicada, y de cara al futuro deberían atarse mejor los cabos de dicha organización y la gestión de los contenidos. Han existido muchos conflictos entre los alumnos con respecto a los repartos de roles y responsabilidades en general y diversos cambios de última hora que han podido ralentizar el proceso de desarrollo del proyecto.

c) Mejoras que se proponen para el futuro:

- Unificación más robusta de los lenguajes de programación del sistema (por ejemplo, construir todo únicamente en Python).
- Mejorar el despliegue, como por ejemplo usar Docker o algún servidor de aplicaciones, en lugar de XAMPP.
- Usar base de datos no relacional (NoSQL) las cuales permiten mayor escalabilidad y rapidez que una BD SQL relacional.

d) Listado de lecciones aprendidas:

Descripción del problema 1: Es difícil establecer días de reuniones con el equipo.
Causa: Los miembros del grupo tienen horarios dispares de trabajo y tiempo libre.
Acción Correctiva: Crear un Doodle en el que cada miembro puede poner las horas y días libres en los que puede acudir a una reunión.
Resultado obtenido: Se simplifica el trabajo de organizar una reunión, simplemente se debe acudir al horario y escoger un día en que la mayoría del grupo pueda asistir.
Lección aprendida: Es de gran utilidad antes de comenzar a planificar reuniones, establecer un horario de disponibilidad del equipo de forma que simplifique la organización de las reuniones.

Descripción del problema 2: Problemas al tratar de conseguir la construcción en Travis.
Causa: Una mala configuración de los scripts.
Acción Correctiva: Revisar los scripts, la configuración, leer la documentación para aclarar las dudas.
Resultado obtenido: Se consigue corregir el problema con éxito y la construcción en Travis ya avanza y termina correctamente.
Lección aprendida: Es importante revisar correctamente la documentación y revisar todos los parámetros del proyecto, así como las dependencias necesarias para este.

Descripción del problema 3: Problemas de comunicación con otros subsistemas.
Causa: Falta de comunicación con otros subsistemas en tiempos razonables de cara a la entrega final, por tanto, no hubo margen de maniobra para modificar algunos aspectos que exigían.
Acción Correctiva: Rechazar la propuesta.
Resultado obtenido: Se mantiene la versión anteriormente establecida, la cual mantenía un correcto funcionamiento.
Lección aprendida: Hay que ponerse en contacto con cierta antelación de cara a tener un mayor margen de maniobra de cara a la entrega y al desarrollo de soluciones.