14/01/2018

**ID Opera: 20**

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

**almacenamiento de votaciones**

**Documento del proyecto**

**Índice de contenidos**

1. **Resumen**
2. **Introducción y contexto**
3. **Descripción del sistema**
4. **Planificación del proyecto**
5. **Entorno de desarrollo**
6. **Gestión del cambio, incidencias y depuración**
   1. **Gestión del código fuente**
   2. **Gestión de la construcción e integración continua**
   3. **Gestión de liberaciones, despliegue y entregas**
7. **Mapa de herramientas**
8. **Ejercicio de propuesta de cambio**
9. **Conclusiones, trabajo futuro y lecciones aprendidas**

### 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.

### 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.

### 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”.

Para ello, se ha creado una API 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 API, 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 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.

La estructura del sistema junto con los distintos subsistemas que la componen es la siguiente:

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

***Revisar***

*Se explicará el sistema desarrollado desde un punto de vista funcional y arquitectónico. Se hará una descripción tanto descriptiva como técnica de sus componentes y su relación con el resto de subsistemas. Habrá una sección que enumere explícitamente cuáles son los cambios que se han desarrollado para el proyecto.*

### 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.

### 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.
* Descargar e instalar Git
* Descargar e instalar XAMPP
* 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
* Compilar el proyecto

***Revisar***

## Gestión del cambio, incidencias y depuración

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

1. **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.

1. **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*](https://github.com/Proyecto-EGC-G1/Almacenamiento-Votos-EGC-G1/issues)

### 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.

### 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.

### 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
* Proceso definido para el despliegue
* Proceso definido para las entregas
* Política de nombrado e identificación de los entregables.

### Mapa de herramientas

Se dará un esquema de cómo se conectan las herramientas que se usan en el proyecto, qué relaciones tienen o qué relaciones propondría añadir. No olvide explicar bien el mapa de herramientas. Se trata del mapa de herramientas de gestión de la configuración. El mapa de herramientas que se usen para el desarrollo (como bases de datos u otros) no es necesario que aparezca aquí.

### Ejercicio de propuesta de cambio

Se presentará un ejercicio con una propuesta concreta de cambio en la que a partir de un cambio que se requiera, se expliquen paso por paso (incluyendo comandos y uso de herramientas) lo que hay que hacer para realizar dicho cambio. Debe ser un ejercicio ilustrativo de todo el proceso de evolución y gestión de la configuración del proyecto.

### Conclusiones, trabajo futuro y lecciones aprendidas

Se enunciarán algunas conclusiones y se presentará un apartado sobre las mejoras que se proponen para el futuro (curso siguiente) y que no han sido desarrolladas en el sistema que se entrega