Decide - Locaste - Censo

GRUPO 2

ID de Opera: 107

Miembros del grupo:

- **Jimeno Cordero, Carlos**
- **Lineros Fernández, Juan Manuel**
- **Ramos Castro, Eva**

Enlaces de interés:

- Repositorio de GitHub
- URL del sistema desplegado

RESUMEN

Este proyecto se ha llevado a cabo con el objetivo de aprender sobre la evolución y gestión de la configuración de un proyecto, materia sobre la que trata la asignatura, utilizando para ello un sistema heredado denominado Decide. Este sistema ha sido evolucionado por el equipo de desarrollo de manera organizada y se han incluido nuevas funcionalidades al mismo, además de automatizar algunos procesos entre los cuales se incluye validación (*testing*) y despliegue (*deployment*) del sistema. Esto nos ha servido para comprobar las ventajas que tiene el uso de este tipo de herramientas y para aprender nuevos aspectos sobre el desarrollo en equipo.

ÍNDICE

1. Introducción y contexto

Este proyecto trata sobre la evolución de un sistema heredado, denominado Decide, y los procesos de gestión y automatización del mismo. Decide es una plataforma electrónica de votación de software libre donde los usuarios pueden registrarse y participar en las diferentes votaciones del mismo y consultar los resultados una vez finalizan. El código fuente se encuentra disponible en la plataforma de GitHub. Este sistema ha sido desarrollado por la empresa Wadobo y es una simplificación de una herramienta comercial mucho más completa denominada nVoting.

Como proyecto para la asignatura Evolución y Gestión de la Configuración, los alumnos trabajarán como equipos de desarrollo que se encargarán de tomar este sistema y evolucionarlo, es decir, ampliarlo incluyendo nuevas funcionalidades, mejoras y corrección de errores si fuera necesario. Además, toda la evolución se debe llevar a cabo haciendo uso de metodologías y herramientas que permitan la gestión del mismo, y la automatización de algunos procesos como puede ser el despliegue de la plataforma. El cómo se ha llevado a cabo, las herramientas utilizadas y las metodologías aplicadas se describirán en cada uno de los apartados correspondientes de este documento.

2. Descripción del sistema

El sistema Decide está compuesto por distintos subsistemas, cada uno encargado de una funcionalidad concreta. Se integran entre sí haciendo uso de llamadas API, lo cual permite que cada uno de ellos pueda utilizar la tecnología que considere adecuada así como el lenguaje de programación. No obstante, es necesaria una plataforma base en la que se reúnan las rutas de todas esas APIs. A continuación se muestra un diagrama con las relaciones entre cada uno de los subsistemas:

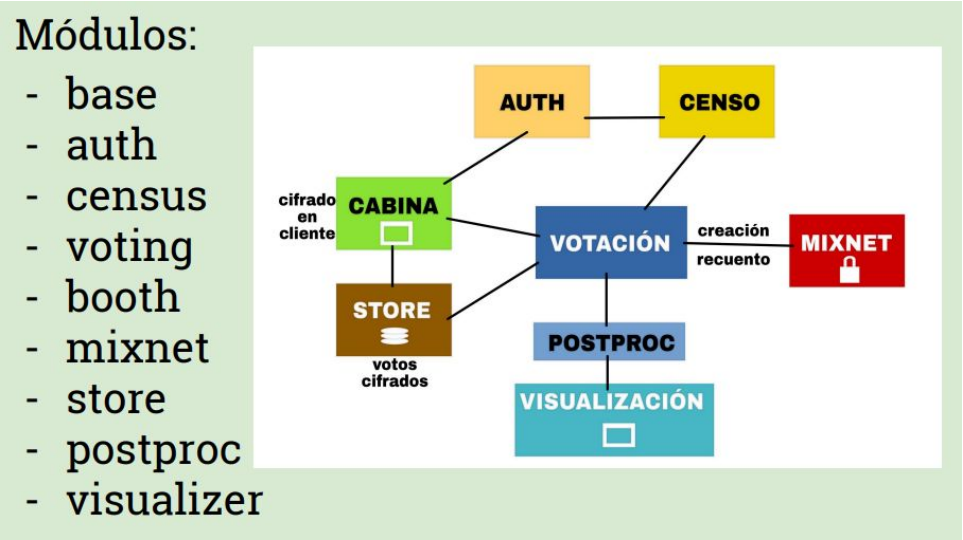


Diagrama relacional de los módulos de Decide

La utilidad y funcionamiento de cada uno de los módulos se puede consultar en el siguiente enlace.

Nosotros hemos trabajado en la evolución del módulo de Censo (*census*), el cual se encarga de gestionar quién puede votar y quién no en una votación determinada, el modelo usado para *census* consta de dos atributos:

- voting_id : Entero con el id de una votación.
- voter_id : Entero con el id de un usuario.

Las mejoras que se han implementado en *census* son las siguientes:

- FrontEnd para el módulo de *census*: Una vista accesible desde una URL en el propio sistema para que los usuarios puedan gestionar los censos de las votaciones de forma visual e intuitiva. Permite visualización, creación y eliminado de censos de la votación elegida.
- Importación y exportación de censos en formato Excel: Funcionalidad añadida al FrontEnd mencionado que permite poder importar y exportar la información de censo de la votación elegida.
- Restricciones añadidas en la creación de censo: Las votaciones pueden llevar restricción de sexo, edad y si son públicas o privadas, las cuales deben ser revisadas a la hora de crear los censos. Esta mejora es una integración con los subsistemas de *auth* y *voting*.
- Ampliación de funcionalidad en la API de *census*: Nueva funcionalidad que permite obtener las votaciones en las que está censado un usuario elegido.

Para llevar a cabo el proceso de evolución y gestión hemos hecho uso de las siguientes herramientas: Git, GitHub, PyCharm, PostgreSQL, Docker, Travis y Heroku. En los siguientes apartados se irá mencionando qué papel ha tenido cada una de estas herramientas.

3. Planificación del proyecto

El desarrollo de este proyecto se ha llevado a cabo a través de un proceso iterativo con tres *milestones* donde al final de cada uno de ellos se ha presentado una versión funcional pero mejorada del sistema Decide. Antes de empezar, los alumnos se dividieron en grupos para trabajar en los diferentes módulos de Decide. En nuestro caso, nos encargamos del módulo censo (*census*). A continuación se explica brevemente la planificación de cada milestone, para más detalles se puede consultar el documento “Diario de Grupo”.

- ***Milestone* 1:** Puesto que era el inicio del proyecto, el equipo se reunió para decidir sobre las mejoras que implementaríamos en el sistema, pero primero debíamos familiarizarnos con las tecnologías y el *framework* de Django, así que las tareas que planificamos fueron las siguientes:
    a. Preparación del entorno de desarrollo. Asignado a todos los miembros.
    b. Estudio del *framework* Django. Asignado a todos los miembros.
    c. Diseño y actualización del diario de grupo. Asignado a Juan Manuel Lineros.
    d. Implementación de una primera versión de FrontEnd para el módulo *census*. Asignado a Juan Manuel Lineros.
- ***Milestone* 2:** En este punto del desarrollo ya teníamos unas nociones básicas sobre cómo trabajar con el sistema, así que empezamos a implementar funcionalidades más complejas y a hacer las primeras integraciones con otros módulos. Las tareas que planificamos para este milestone son:
    a. Continuación del estudio de Django. Asignado a todos los miembros.
    b. Implementación de un FrontEnd funcional para el módulo *census*. Asignado a Juan Manuel Lineros.
    c. Implementación de restricciones en la creación de censos (integración con los módulos *voting* y *auth*). Asignado a Carlos Jimeno.
    d. Implementación de nueva funcionalidad a la API de *census* para obtener el listado de censos de un usuario (integración con postproc). Asignado a Juan Manuel Lineros.
    e. Actualización del diario de grupo. Asignado a Juan Manuel Lineros.
- ***Milestone* 3:** Siendo la última iteración del proyecto, aquí nos centramos en completar las funcionalidades que habíamos acordado al iniciar el proyecto y decidimos añadir algunas más que pensamos que podrían ser interesantes. Además, se llevó a cabo la documentación requerida del proyecto como parte de la entrega final.
    a. Continuación del estudio de Django. Asignado a todos los miembros.

b. Implementación de las restricciones faltantes en la creación de censos (integración con los módulos *voting* y *auth*). Asignado a Carlos Jimeno.
c. Implementación de la creación de censos a través de hojas de Excel. Asignado a Eva Ramos.
d. Implementación de exportación de censos a hojas de Excel. Asignado a Eva Ramos.
e. Actualización de los tests de verificación del módulo *census*. Asignado a Carlos Jimeno y Eva Ramos.
f. Elaboración de la documentación del proyecto. Asignado a todos los miembros.
g. Actualización del diario de grupo. Asignado a todos los miembros.
h. Bug en FrontEnd de censo. Asignado a Eva Ramos.
i. Refactorización del FrontEnd de censo. Asignado a Eva Ramos.
j. Corrección de bugs y modificación del flujo de creación de censos. Asignado a Carlos Jimeno.

4. Entorno de desarrollo

Nuestro entorno de desarrollo está compuesto por los siguientes elementos:

- **Python 3.6:** Decide está desarrollado con el lenguaje de programación Python usando el *framework* Django, por lo que necesitamos instalar el paquete de herramientas de Python para poder ejecutar aplicaciones escritas en dicho lenguaje dentro de nuestra máquina. Para la instalación de Python 3.6 en el sistema operativo Windows 10 se puede consultar este videotutorial donde se explica paso a paso el proceso, pero teniendo en cuenta que el enlace de descarga que aparece en el video ya no está disponible porque ha sido sustituido por la versión 3.7, así que se debe descargar desde este enlace (pulsar en el enlace “Download” de la última versión 3.6 que aparezca en la lista).
- **PyCharm Community Edition 2018.2.4:** PyCharm es un IDE para el desarrollo de aplicaciones Python. en nuestro caso hemos utilizado la “Community Edition”, ya que es la versión gratuita. Para descargarlo simplemente acceder a este enlace y pulsar en el botón “DOWNLOAD” que aparece en la versión “Community”. En este enlace se puede ver un videotutorial sobre cómo instalar el IDE. A partir del minuto 1:50 se dedica a la configuración del IDE, pero esto no es relevante para la instalación. En el video se crea un nuevo proyecto, pero en nuestro caso estaremos importando uno ya existente, por lo que deberemos hacer click en “File”, luego en “Open…” y seleccionar la carpeta donde se encuentre el proyecto Decide.

- **PostgreSQL 10:** La persistencia de información de Decide se realiza con una base de datos PostgreSQL, por lo que necesitamos tener instalado este sistema gestor de base de datos en nuestra máquina para hacer funcionar la aplicación. En este enlace se puede descargar la versión 10 de PostgreSQL (hacer click en "Download" en la columna "Windows x86-64"). Aquí se puede ver un videotutorial sobre cómo instalar PostgreSQL una vez descargado (la parte a partir del minuto 4:50 se debe ignorar, el Stack Builder no es necesario que se instale).
- **Git:** Se trata de una herramienta gratuita de control de versiones y gestión de repositorios de código. Se puede descargar desde este enlace. y se puede ver el proceso de instalación en este videotutorial.

Una vez instaladas las herramientas anteriores, deberemos descargarnos el proyecto Decide desde aquí. Para ello, ejecutaremos la herramienta Git Bash y seguiremos los siguientes pasos:

- Escribiendo el comando "pwd" averiguaremos en qué directorio nos encontramos actualmente, lo cual será necesario para saber dónde se va a descargar el proyecto.
- A continuación, escribiremos "git clone https://github.com/EGC-Decide/locaste". Esto creará una carpeta "locaste" en el directorio actual con todo el contenido del proyecto Decide. Si al ejecutar este comando te solicita los credenciales de tu cuenta GitHub, simplemente escríbelos y pulsa Intro. Si no tienes cuenta en GitHub, puedes crearla aquí.

Ahora que ya tenemos el proyecto Decide en nuestra máquina, ejecutaremos PyCharm e importamos el proyecto pulsando en "File", "Open…" y seleccionando la carpeta "locaste" que se ha descargado con Git. Cuando esté importado, deberemos realizar los siguientes pasos de configuración:

- En la terminal que aparece en la parte de abajo de PyCharm deberemos escribir "pip install -r requirements.txt". Esto instalará las dependencias requeridas por el proyecto Decide para su correcto funcionamiento.
- En Windows, seleccionaremos desde el menú la aplicación "pgAdmin 4" y se nos abrirá una aplicación web en el navegador.
- Haciendo click derecho sobre "Login/Group Roles" y seleccionando "Create", "Login/Group Role…" se nos abrirá una nueva ventana donde deberemos introducir la información del usuario que utilizará el sistema Decide. Para ello, rellenamos el campo "Name" de la pestaña "General" con el nombre de usuario ("decide") y el campo "Password" de la pestaña "Definition" con la contraseña del mismo usuario ("decide" también).

- A continuación hacemos click derecho sobre “PostgreSQL 10” y seleccionamos “Create”, “Database…”. En la ventana que se nos abre, introduciremos el nombre de la base de datos (“decide”) en el campo “Database” de la pestaña “General”, y en esa misma pestaña escogeremos el usuario “decide” creado anteriormente en el desplegable de “Owner”.
- Volvemos a PyCharm. Nos duplicamos el archivo “local_settings.example.py”. y le cambiamos el nombre a “local_settings.py”. Dentro, tenemos que cambiar algunos datos: todas las direcciones que aparecían como “10.5.0.1”, hay que cambiarlas por “localhost” y además en la sección “DATABASES” hay que modificar los datos de “NAME” y “USER” para que aparezca “decide”. Finalmente, añadimos un campo “PASSWORD” con la contraseña “decide”. Debería quedar así:

Estructura final de DATABASES

- En la terminal de PyCharm ejecutamos los siguientes comandos:
  - “cd decide” para situarnos en dicha carpeta, lo cual será necesario para ejecutar los comandos a continuación.
  - Para migrar la estructura de la base de datos: “python manage.py migrate”.
  - Crear super usuario: “python manage.py createsuperuser”. Aquí nos solicitará los datos del usuario administrador del sistema.
  - Lanzar aplicación: “python manage.py runserver”.
- Entramos en la aplicación desde la ruta 127.0.0.1:8000/admin y nos logueamos con el super usuario que acabamos de crear. Desde aquí podemos comenzar a crearnos usuarios y votaciones para poder empezar a usar la aplicación.

5. Gestión de incidencias

Para el proceso abstracto de gestión de incidencias se puede consultar el siguiente enlace, además la gestión de commits viene definida en este otro enlace. Ambos procesos se han usado de forma común por todos los subsistemas del grupo Locaste. La única diferencia que hemos seguido en censo es el formato de los

commits, para los cuales se ha modificado ligeramente el que aparece en el segundo enlace, dando el siguiente formato:

“Census [<*tag*>]: Título. Cuerpo #*Issue* si la hubiera’’

5.1 Incidencias Internas:

Se ha utilizado el proceso definido en el primer enlace.

5.2 Incidencias Externas:

Para las incidencias que nos han sido reportadas de otros subsistemas, por lo general siempre se han puesto en contacto con el coordinador de Censo, mediante el grupo de telegram de los coordinadores. A continuación, se ha intentado resolver el problema o la duda si es que ya está implementado lo que se solicita. En caso contrario, algún miembro del submódulo interesado nos ha creado una incidencia en GitHub y en censo nos hemos puesto de acuerdo internamente para ver a quien se le asignaba.

Para las incidencias que hemos tenido que reportar a otros subsistemas se ha seguido el mismo proceso que para las que nos han sido reportadas: primero nos ponemos en contacto con el submódulo de interés, intentamos resolver el problema sin implementar nada nuevo o corregir lo ya existente si no es necesario, y en caso contrario creamos la incidencia para ese submódulo y les dejamos a ellos que se la asignen a quien crean conveniente.

En cuanto a las herramientas utilizadas, se ha hecho uso de GitHub para la creación de incidencias y la gestión de las mismas. Además, se ha usado la funcionalidad del tablero Kanban “Proyectos” que permite ver en qué estado se encuentra cada tarea arrastrando su ficha a las distintas columnas. Algunas de estas columnas tenían automatización, por ejemplo al crear una incidencia automáticamente se añade a la columna “To do”.

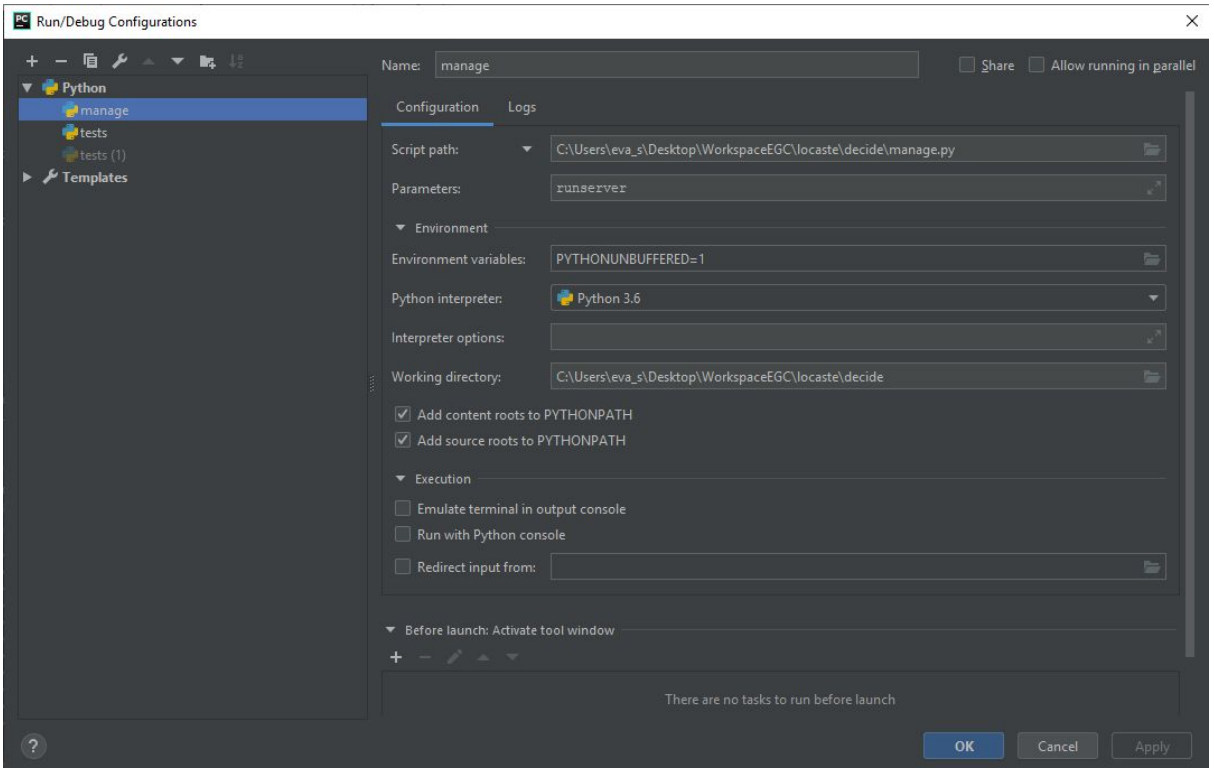
Algunos ejemplos de incidencias de censo son los siguientes: #70, #57, #24.

6. Gestión de depuración

En cuanto a la depuración de errores, casi todos los que hemos encontrado o se nos han reportado, los hemos registrado como incidencias y compañeros de otros submódulos tambíen nos han creado incidencias. Además, el proceso de depuración enunciado a continuación se ha utilizado cuando ha sido necesario a nivel personal para poder implementar los requisitos que teníamos asignados.

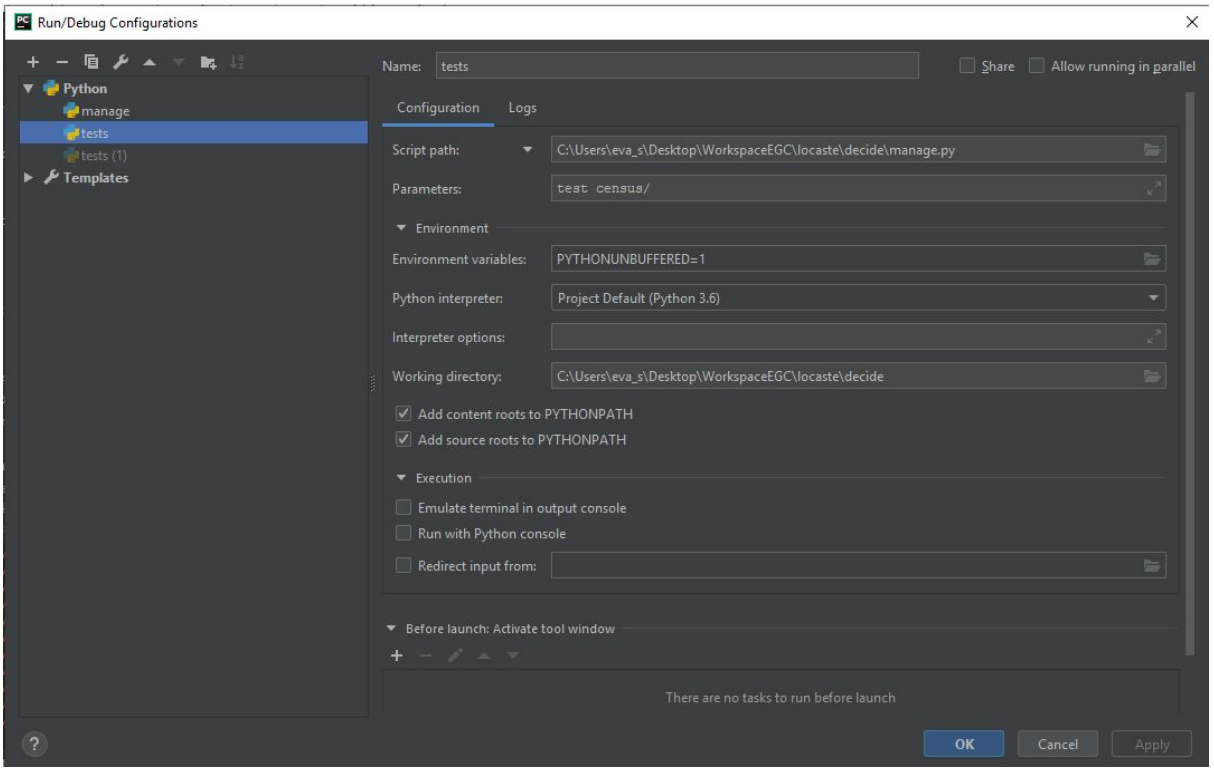
Para la depuración del código hemos utilizado 3 configuraciones distintas.

- Para la depuración de errores del BackEnd hemos creado una configuración en PyCharm. Esta configuración es equivalente a ejecutar “python manage.py runserver”, solo que en PyCharm si no tienes la configuración creada, no te permite darle al bichito de debuggear.



Configuración 1 - runserver

- Para la depuración de errores en los test hemos creado una configuración similar a la anteriormente explicada, permitiendo depurar únicamente los tests de *census*.



Configuración 2 - test

- En la depuración del FrontEnd hemos usado la palabra reservada de javascript “debugger” que junto a la herramienta para desarrolladores del navegador permite depurar el código javaScript.

Algunos ejemplos de incidencias que han requerido depuración en censo: #129, #106.

7. Gestión del código fuente

El proyecto Decide/locaste inicialmente surgió a partir de crear una organización en GitHub en la que 6 miembros cada uno de un subsistema distinto fueron asignados administradores de la misma.

Cada uno de estos miembros creó un equipo dentro de la organización y añadió a todos los miembros de su submódulo. Además, a cada uno de esos equipos se le asignó una rama del repositorio con el nombre de su submódulo y se modificaron los permisos de esa rama para que solo los miembros del equipo pudieran hacer *push* a la misma.

El proceso que se ha seguido en el submódulo censo para la evolución del código ha consistido en realizar todos los cambios en una única rama, la asignada a nuestro equipo. Para ello se ha intentado hacer *commits* con cambios que estén

relacionados con únicamente una incidencia, y a ser posible, resolviendo la incidencia en un solo *commit.*

Los comandos utilizados para subir nuestros cambios a la rama *census* han sido:

- git pull
- git add <*ruta de archivos nuevos/modificados/eliminados*>
- git commit
- git push

A la hora de unificar nuestros cambios con la rama dev, se han ejecutado las siguientes instrucciones desde la carpeta del repositorio:

- git checkout census
- git pull
- git checkout dev
- git pull
- Se han comprobado que los tests de todos los submódulos siguen funcionando, y si no, se han corregido los fallos con un nuevo commit.
- git merge census
- Si no han surgido conflictos, el *mergeo* debería ser automático, y si han sucedido, git nos pide que los arreglemos y hagamos un commit.

Finalmente para pasar los cambios de dev a master se debe encargar un coordinador mediante un *pull request.*

Las tecnologías usadas han sido Git y Github.

Algunas evidencias: commit

8. Gestión de la construcción e integración contínua

El proceso de gestión de la integración puede consultarse aquí. Se ha definido y usado de manera global entre todos los subsistemas de locaste.

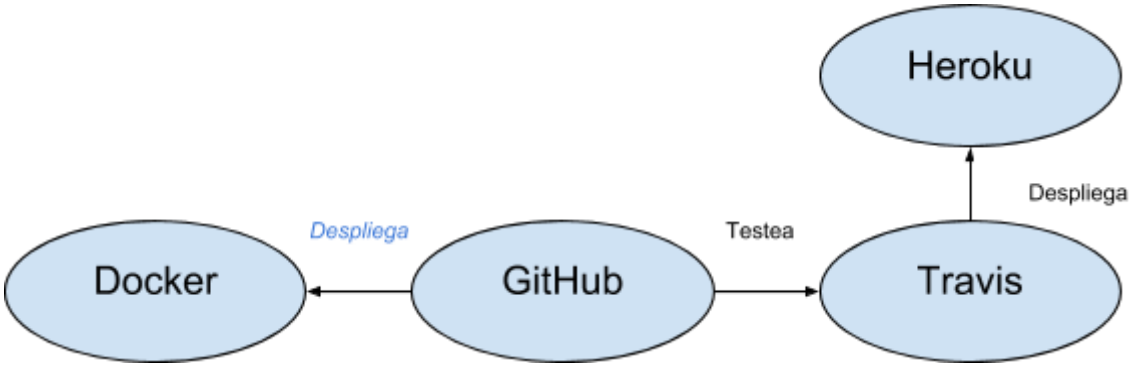
La herramienta utilizada es Travis.

9. Gestión de liberaciones, despliegue y entregas

- Entregas:

    - Se realizará una única entrega el 13/01/2019.
- Identificación de los entregables:
    - Para el nombrado o identificación de las versiones se usará el siguiente esquema.
- Despliegue:
    - Para el proceso de despliegue se puede consultar el siguiente enlace.
- Liberaciones:
    - Se realizará una única liberación el 13/01/2019.
- Licencia:
    - La licencia escogida se puede consultar aquí.

10. Mapa de herramientas



Este mapa de herramientas representa la herramientas usadas en el proyecto.
En un primer lugar nos encontramos con la herramienta GitHub. Esta herramienta ofrece diversos módulos para la gestión del código, en particular usaremos el repositorio de código además del gestor de incidencias y el tablero kanban “Proyectos”.

El repositorio de GitHub se relaciona con la herramienta Travis, la cual ejecuta los test de forma automática cada vez que se realiza un commit,.avisando al autor del mismo con los resultados de ejecutar los tests.

Heroku es una plataforma donde se realiza el despliegue con una versión estable de la rama master. Está herramienta está relaciona con Travis, ya que es la encargada de pasar los tests en el sistema antes de hacer el despliegue automáticatico. Todo esto usando siempre la rama master únicamente.

En último lugar tenemos la herramienta Docker relacionada con el repositorio GitHub. Esta herramienta automatiza el empaquetamiento de aplicaciones dentro de contenedores de software para su posterior ejecución de forma rápida y contenida. Al no ser usada se ha indicado la acción "*despliegue*" en cursiva y en azul.

11. Ejercicio de propuesta de cambio

Ejercicio de cambio: Restricción de URL de acceso al FrontEnd del módulo de Censo.

1. Realizar *issue*, existiendo la posibilidad de asignar esta *issue* a un desarrollador. Debe ser etiquetada con las *tags* "enhancement" (tipo de *issue*), "Census" (módulo) y "neutral" (prioridad de la *issue*).
2. El desarrollador que vaya a realizar esta *issue* debe asignarse a dicha *issue* (en caso de estar ya asignado este paso no hará falta).
3. El desarrollador deberá estudiar la viabilidad del cambio propuesto. Si se va a llevar a cabo, la *issue* será cambiada de columna pasándola a WIP. Además, deberá asegurarse que los cambios serán realizados en la rama correcta. Para ello deberá realizar los siguientes pasos:
    a. Abrir el programa Git Bash en la ruta de la carpeta del repositorio.
    b. Ejecutar el comando "git branch" para comprobar en qué rama estamos situados. Si no estamos en la rama "Census", ejecutamos "git checkout census" y nos cambiamos a esta.
    c. Ejecutar "git pull". Con este comando incorporas los cambios de un repositorio remoto al repositorio local, de la rama en la que te encuentras.
4. El desarrollador realizará los cambios pertinentes. Una vez finalizado y probado se realizará un commit, posteriormente moverá la tarjeta a la columna "IN REVIEW". Para hacer el commit se seguirá los siguiente pasos.
    a. Ejecutar "git pull". Con este comando incorporas los cambios de un repositorio remoto al repositorio local, de la rama en la que te encuentras.
    b. Ejecutar "git status". Con este comando veremos los ficheros modificados.
    c. Ejecutar "git add" {ruta fichero modificado a subir}. Con este comando añadimos el fichero modificado para realizar un commit.
    d. Ejecutar "git commit"
        i. Establecer cabecera con el siguiente formato. "Census[feat]: Restricción de URL de acceso al Frontend del módulo de Censo.
        {Cuerpo de un mensaje}

Resolves #{nº incidencia}”

e. Ejecutar “git push”. Con este comando se suben los commits locales a remoto.

5. El mismo desarrollador u otro realizará pruebas sobre el cambio realizado, si dichas pruebas son satisfactorias se cerrará la issue y se pasará a la columna DONE. Indicando así que el desarrollo ha llegado a su fin satisfactoriamente.

12. Conclusiones y trabajo futuro

Posibles mejoras para el subsistema Censo:

- Mejorar el control de restricciones en la creación de censos: actualmente no se está usando la forma correcta que sugiere Django para controlar las restricciones en los campos de un modelo.
- Añadir las urls que creamos convenientes al index de la página web o a la vista de votaciones si la hubiera.
- Personalización de mensajes de Error.
- Restricción de URL de acceso al FrontEnd del módulo de Censo.

Conclusiones:

Esta experiencia ha sido muy nutritiva, además de haber supuesto un gran reto.
La experiencia nos ha dado la posibilidad de:

- Evolucionar un código que no ha sido desarrollado por nosotros mismos con la dificultad añadida de estar desarrollado con tecnología poco usada a lo largo de la carrera.
- Coordinarnos con diversos compañeros para la integración de los subsistemas.
- Obtener soltura con herramientas usadas en la asignatura. Esto es beneficioso para nosotros ya que dichas herramientas son usadas en el mundo laboral.