Miembros

Cordero Vela, Francisco Manuel  
Lázaro Domínguez, José Manuel

Mogrovejo Campero, Miguel Ángel

Benítez Rodríguez, Daniel

Romero Adame, José Ángel

Cabina telegram

Proyecto para EGC

Índice

[1. Control de versiones 2](#_Toc501194204)

[2. Resumen 3](#_Toc501194205)

[3. Gestión del código fuente 3](#_Toc501194206)

[3.1. Herramientas 3](#_Toc501194207)

[3.2. Gestión de ramas 3](#_Toc501194208)

[3.3. Procedimientos de gestión de código 4](#_Toc501194209)

[3.4. Roles 4](#_Toc501194210)

[3.5. Aprobación de cambios 5](#_Toc501194211)

[4. Gestión de incidencias 5](#_Toc501194212)

[4.1. Incidencia interna 5](#_Toc501194213)

[4.2. Incidencias externas 5](#_Toc501194214)

[4.3. Estados de las incidencias 6](#_Toc501194215)

[4.4 Prioridad de las incidencias 6](#_Toc501194216)

[5. Entorno de desarrollo 6](#_Toc501194217)

[5.1. Máquina virtual 6](#_Toc501194218)

[5.2. NodeJS 7](#_Toc501194219)

[4.3. JavaScript V8 7](#_Toc501194220)

[5.4. Git 7](#_Toc501194221)

[5.5. Npm 7](#_Toc501194222)

[5.6. Docker 7](#_Toc501194223)

# Control de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Autor | Fecha | Cambios |
| 1.0.0 | Miguel Ángel Mogrovejo Campero | 10/12/2017 | Esquema del documento, apartados principales hasta la fecha. |
| 1.0.1 | Miguel Ángel Mogrovejo Campero | 12/12/2017 | Continuando los apartados principales hasta la fecha. |
| 1.0.2 | Miguel Ángel Mogrovejo Campero | 16/12/2017 | Añadiendo nuevos apartados. |

# Resumen

El objetivo principal es dar una solución a la problemática de realizar encuestas por el móvil, esto dentro del proyecto sobre votación de la asignatura EGC (Evaluación y Gestión de la Configuración), para ello se ha decidido utilizar la aplicación Telegram que es gratuita y utilizar el servicio de bots que te da esta plataforma, estos bots se pueden programar con una API abierta que te ofrece Telegram.

Para llevar a cabo esta tarea, hemos decidido usar las siguientes herramientas:

* Para la gestión del código se utiliza Git.
* Para la gestión de las incidencias se utiliza GitHub.
* Para la comunicación entre el grupo usamos Discord y Telegram.
* Para la modificación del código usamos Sublime Text 3.
* Para la Implementación usamos Nodejs.

# Funcionalidad del bot

Nuestra idea principal es que el bot tenga las siguientes funcionalidades:

* Una funcionalidad que sirva para listar las encuetas existentes, para que el usuario puede elegir la encueta en la que quiere participar.
* Una funcionalidad que liste todas las preguntas con sus respuestas de una de las encuestas, una vez hecho esto el usuario podrá seleccionar las respuestas de las distintas preguntas que se le mostraran por pantalla.
* Una funcionalidad que envié dichas respuestas para luego ser almacenadas y tratadas, cuando el usuario terminé su encuesta podrá usar un comando para enviar las respuestas seleccionadas a las preguntas.

# Gestión del código fuente

## Herramientas

Para la gestión del código se usará la herramienta de control de versiones Git, así como la plataforma GitHub para la coordinación de todos los miembros del equipo. las operaciones de gestión de código se hacen a partir de la consola de comandos, ya que así se nos enseñó en las prácticas de la signatura Egc.

## Gestión de ramas

La gestión de ramas en el repositorio remoto y sobre el que se articulará el trabajo del equipo son dos: “master” y “dev”.

* Rama master: aquí se mantendrán las versiones estables del sistema. La última versión de esta rama será nuestra línea base a la hora de desarrollar el código. Esta rama solo se conecta con la rama dev, es decir, solo se pueden obtener modificaciones de esta rama, y no de cualquier forma, sino siguiendo un procedimiento de control de cambios.
* Rama dev: En esta rama es donde se trabajará y se subirá las modificaciones antes de pasar a la rama master. Una de las reglas que sea decidido es que no se deberá subir código que contenga errores de compilación o errores que no permitan el funcionamiento del proyecto.

## Procedimientos de gestión de código

De forma local, se puede trabajar directamente en la rama dev para cambios menores, pero para el resto de cambios, se define el siguiente procedimiento de gestión de código:

* Lo primero que se debe hacer antes de comenzar a trabajar es moverse de la rama master a la rama dev usando el comando: git checkout dev
* Cuando un trabajador quiera implementar algún cambio, debe crear una rama nueva a partir de la última versión en dev.
* Con cada commit se debe implementar algún cambio, o se debe hacer referencia a la incidencia que se ha tratado.
* En las modificaciones realizadas que no han supuesto la implementación completa de un cambio, no es necesario que sean subidas al repositorio remoto si es el caso de que se continuará trabajando en esta modificación mas adelante, en caso contrario como por ejemplo que no se haya terminado la modificación por encontrar incidencia q no se han sabido resolver, si se deberá subir los cambios al repositorio remoto de esa rama.
* En las modificaciones realizadas que, si suponen la total implementación del cambio, se debe realizar un commit que haga referencia a la resolución de la incidencia o del cambio en cuestión, todos los miembros asignados deben comprobar la validez del cambio. Una vez aceptado, se pueden unir los cambios con la rama dev.

## Roles

Existen varios roles en la gestión del código que son asignados cuando una incidencia es aprobada, salvo en casos particulares:

* Desarrollador: miembros del equipo que producen incrementos en el subsistema. Todos los miembros son desarrolladores. Se necesita la aprobación de al menos otro desarrollador, a parte del responsable, para poder pasar el incremento a la rama de desarrollo.
* Responsable de un incremento: Una vez aprobado un cambio, se asignan uno o varios desarrolladores a producir el incremento. El responsable debe, según lo definido en el proceso anterior, crear la rama auxiliar y mantener esa rama hasta su unión con la rama de desarrollo. También puede delegar esta tarea a uno de los desarrolladores, de manera que quede constancia de tal aprobación. El responsable también es un desarrollador.
* Jefe de proyecto: este rol es fijo durante el desarrollo del proyecto. También es un desarrollador. El jefe del proyecto es el encargado de revisar los cambios menores de la rama de desarrollo, así como de autorizar y realizar el paso a una versión estable.

## Aprobación de cambios

En la práctica, la aprobación de los cambios se va hacer de manera informal, Se pedirá a los otros miembros del desarrollo que comprueben el incremento, y estos darán su aprobación. Esto se puede hacer por mensajería instantánea usando Telegram para agilizar el proceso, se puede tomar capturas de la conversación si se cree necesario para dejar constancia de dicha aprobación.

# Gestión de incidencias

## Incidencia interna

Cuando un miembro del equipo encuentra un error en la aplicación, debe crear una nueva incidencia en GitHub de la forma que se nos ha enseñado en la asignatura Egc:

* Una introducción general al contexto donde se produce el fallo.
* Pasos a ejecutar: será necesario que describa, con más o menos detalle, los pasos que producen el error en el sistema, así puedan otros miembros del equipo reproducirlo y confirmar su existencia.
* El resultado esperado si no se diese el error.

Debe quedar claro cada parte de la información. Cualquier dato adicional que sea añadido, como capturas de pantalla o cualquier otro intento del miembro para acotar la búsqueda del problema será bien recibido.

## Incidencias externas

Cuando un participante externo deja una incidencia. En primer lugar, se le da mayor prioridad y se deja en segundo plano las tareas que se estén realizando en el momento de recibir dicha incidencia para través de la herramienta de gestión de incidencias de GitHub dar una solución lo antes posible.

Algunos pasos que seguir son:

* Primero, identificar el problema. Para ello se deberán recopilar la información necesaria aportada por el creador de la incidencia.
* Segundo, una vez obtenida la información, se intentará reproducir el error detectado. Si no se encuentra dicho error o si se es incapaz de reproducirlo, se puede debatir sobre invalidar la incidencia o volver a recabar información de está. Todo ello debe etiquetarse correctamente.
* Tercero, una vez que se ha identificado el fallo del sistema, se puede proceder, a asignar a un miembro del equipo y a comenzar a resolver la incidencia. Este proceso es exactamente igual al de una incidencia interna, y debe consensuarse en el grupo.
* Cuarto, Cuando se resuelve la incidencia, se debe referenciar la corrección a la misma y se debe pedir explícitamente al participante que verifique la correcta resolución de la incidencia. En caso negativo, se repetiría.

## Estados de las incidencias

Cuando se creada una incidencia, para identificar fácilmente su contenido usaremos las siguientes etiquetas:

* bug: para informar de un error en el funcionamiento del código.
* enhancement: para proponer una nueva funcionalidad.
* help wanted y question: para dudas sobre el funcionamiento en general, sea de organización, de la arquitectura del sistema, de la configuración del entorno de desarrollo, etc.
* duplicate: para señalar información o archivos que estén duplicados.
* Invalid: para señalar problemas con archivos inválidos.
* tarea: para asignar tareas a miembros del equipo.

## 5.4 Prioridad de las incidencias

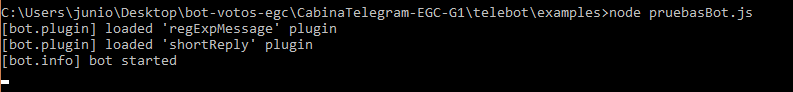
La prioridad implica la urgencia con que debe resolverse una incidencia se añadirá una etiqueta que ponga alta, media y baja prioridad, y es, en general, decidida en consenso por los miembros del grupo. La prioridad se puede cambiar en cualquier momento si así se considera necesario, por el mismo método que se decidió la prioridad original.

# Gestión de construcción

La construcción se realiza usando Nodejs, primero hay que descargar el proyecto desde github luego, se debe usar el gestor de paquetes NPM para en la dirección que esta ubicado el proyecto descargado instalar los paquetes necesarios de node, si no se realiza esto no se podrá ejecutar los scripts con nodejs.

Hay que tener cuidado con las versiones tanto de node como de npm ya que si se usan versiones inestables o versiones muy antiguas, dará fallos en el instalador de paquetes y no permitirá la completa instalación de los paquetes que requiere node.

Una vez se ha realizado estos pasos se podrá desplegar el bot usando node en consola para ejecutar los archivos javascript necesarios, si se a realizado todo lo anterior debería verse el siguiente mensaje:



Una ves recibido el mensaje “bot satarted” el bot debe de estar funcionando en telegram.

Como dato adicional en el repositorio de GitHub se ha creado una carpeta con una serie de guías que puede ayudar con problemas como por ejemplo al actualizar o instalar node.

# 7. Entorno de desarrollo

## 7.1. Máquina virtual

Como programa de virtualización hemos utilizado Virtual Box, ya que es uno de los más conocidos, utilizado previamente por el grupo y bastante fiable. El hecho de utilizar una máquina virtual ayuda a trabajar con entornos exactamente iguales con esto descartamos fallos que se puedan producir por el uso de versiones distintas del sistema operativo. El equipo ha trabajado con diferentes versiones de Virtual Box al tenerlo instalado de asignaturas anteriores sin problemas.

Usamos una maquina virtual de Linux versión Ubuntu(32-bit), con 2G de Memoria RAM y 2 procesadores.

## 7.2. NodeJS

Nodejs nos proporciona una manera fácil para construir programas de red escalables, hemos utilizado la versión 8.9.3, ya que es la última versión estable de Node. Para instalar paquetes de Node se ha utilizado npm, el cual es un instalador de paquetes bastante sencillo.

## 6.3. JavaScript V8

Node ejecuta V8 JavaScript, el motor V8 JavaScript es el motor JavaScript subyacente que Google usa con su navegador Chrome, aunque node le da otros propósitos para usarlo en la parte del servidor (No está restringido a ejecutarse en un navegador).

## 6.4. Git

Como hemos utilizado GitHub para compartir el código, es necesario el uso de Git para mantener el control de versiones y el código compartido entre todos los miembros del grupo.

## 6.5. Npm

Es gestor de paquetes de Node, la versión que usamos es 5.6.0 que viene al instalar nodejs. Es el ecosistema más grande de librerías de código abierto.

## 

## 6.6. Docker

Es un proyecto de código abierto para automatizar el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción del SO.

# Gestión de la construcción y la integración continua

Para la integración continua usaremos la herramienta Travis CI. El funcionamiento básico de Travis CI es clonar el repositorio de GitHub a un nuevo entorno virtual y llevar a cabo una serie de tareas para construir y probar tu código. Si alguna de estas tareas falla, el build se considera broken. Si no falla ninguna tarea, el build se considera passed y Travis CI puede desplegar tu código a un servidor web, por ejemplo.

Los pasos a seguir son los siguientes:

1. hacer un fork del repositorio de GitHub.
2. Usar tu cuenta de GitHub entrando en https://travis-ci.org y acepta la confirmación de permisos de GitHub.
3. Una vez logueado en Travis CI y sincronizado tus repositorios de GitHub, ve a tu perfil y activa el repositorio que quieras construir.
4. Añadir un fichero .travis.yml a tu repositorio para indicar a Travis CI lo que tiene que hacer.
5. Hacer commit y push del .travis.yml para iniciar una build de Travis CI.
6. Comprobar en la página de estado de la construcción si tu build ha pasado o falla.