Cabina de Slack

Grupo 1

ID opera: [27](http://opera.eii.us.es/egc/public/grupo/ver/id/108)



Miembros:

[Luna Zayas, Eduardo](https://github.com/edubsteep)

[Ardoy Álvarez, Alejandro Manuel](https://github.com/alex395)

[Ruiz Budia, Antonio Jesús](https://github.com/budia95)

[Bersabé Gutiérrez, José Ignacio](https://github.com/Igna695)

Enlaces de interés:

* Repositorio de código: [enlace](https://github.com/Proyecto-EGC-G1/CabinaSlack-EGC-G1)
* Canal de Slack: [enlace](https://egc1718.slack.com/messages/C8LCE9A2Y/team/U7TSQT1S8/)
* Repositorio de docs: [enlace](https://drive.google.com/open?id=1WCo_AA7Q1REUg57mD8A0SixA4vL5EF9r)
* Gestión de incidencias: [enlace](https://github.com/Proyecto-EGC-G1/CabinaSlack-EGC-G1/issues)
* Integración continua: [enlace](https://travis-ci.org/Proyecto-EGC-G1/CabinaSlack-EGC-G1)
* Opera: [enlace](http://opera.eii.us.es/egc/public/grupo/ver/id/108)

Contenido

[1. Resumen 3](#_Toc503700395)

[2. Introducción y contexto 4](#_Toc503700396)

[3. Descripción del sistema 5](#_Toc503700397)

[3.1 Planificación del trabajo 5](#_Toc503700398)

[3.2 Cambios realizados 6](#_Toc503700399)

[3.3 Modelo relacional del sistema de votaciones 6](#_Toc503700400)

[3.4 Funcionalidad 7](#_Toc503700401)

[4. Entornos de desarrollo 7](#_Toc503700402)

[4.1 Instalación del entorno de desarrollo 8](#_Toc503700403)

[4.2 Instalación herramientas en Ubuntu 12](#_Toc503700404)

[4.3 Creación de bot 16](#_Toc503700405)

[4.4 Conclusión 19](#_Toc503700406)

[5. Gestión del código fuente 20](#_Toc503700407)

[5.1 Roles 20](#_Toc503700408)

[5.2 Gestión de ramas en el código 20](#_Toc503700409)

[5.3 Gestión Labels en GitHub 20](#_Toc503700410)

[5.4 Caso práctico 21](#_Toc503700411)

[5 Gestión de la construcción e integración continua 23](#_Toc503700412)

[6 Gestión del cambio e incidencias 24](#_Toc503700413)

[7 Gestión de liberaciones, despliegue y entregas 25](#_Toc503700414)

[8 Mapa de herramientas 26](#_Toc503700415)

[9 Conclusiones 26](#_Toc503700416)

# Resumen

Al inicio, según los proyectos propuestos en clase, se nos ocurrió la idea de pertenecer al grupo de Cabina de Slack, ya que nos interesaba la idea de realizar un bot y la posibilidad de conocer esta aplicación la cual hasta el momento era de total desconocimiento para nosotros. Para la realización de un bot es necesario el uso de las herramientas que nos proporciona la API de slack para la cual se puede iniciar una conversación, o pertenecer a un canal, solamente que, en vez de recibir los mensajes de otra persona, lo recibe un programa, el cual nos aporta funcionalidades según lo que se haya programado.

Esta solución nos permite visualizar votaciones de manera sencilla. Slack es una aplicación de mensajería instantánea gratuita la cual cualquier persona puede hacer uso de ella desde cualquier terminal móvil u ordenador, solo teniendo la url del espacio de slack.

Uno de los principales problemas fue el desconocimiento de esta tecnología, así como, desconocimiento de los entornos que hemos utilizado. Otro de los problemas ha sido la falta de soltura con GitHub y Ubuntu, ya que en ambas teníamos poca experiencia de uso.

Por último, podemos destacar el aprendizaje para la realización del bot, ya que era un tema bastante interesante y útil en el cual queríamos adentrarnos.

# Introducción y contexto

Como hemos explicado anteriormente, hemos desarrollado un bot para la aplicación llamada Slack, la cual es completamente gratuita para cualquier usuario. Este bot nos permite acceder a las votaciones que estén disponibles para realizar las votaciones en la web del congreso. La elección de este trabajo vino por la falta de interés en los demás subsistemas disponibles el día de elección en clase, ya que o bien, no sabíamos la funcionalidad de este, o bien no nos motivaba su función.

Nuestra aplicación permite acceder a la base de datos del sistema de votaciones integrado en la página del congreso de splc 2017. Nuestro proyecto se centra en apartado de cabina de votaciones, permitiendo al usuario mostrar las votaciones disponibles y realizar la votación correspondiente. Una vez que realiza la votación, debemos comunicarnos con Cabina de Votaciones, para emitir el voto de este usuario.

Como funcionalidad extra hemos añadido comandos de ayuda, así como avisos en la aplicación.

Las principales ventajas de usar la herramienta de bots de Slack son:

* Ofrece salas de chat organizadas por temas, así como grupos privados y mensajes directos.
* Gratuita.
* Posee un cuadro de búsqueda que permite acceder a todo el contenido de la aplicación.
* Integra una gran cantidad de servicios a terceros y respalda las integraciones hechas por la comunidad.

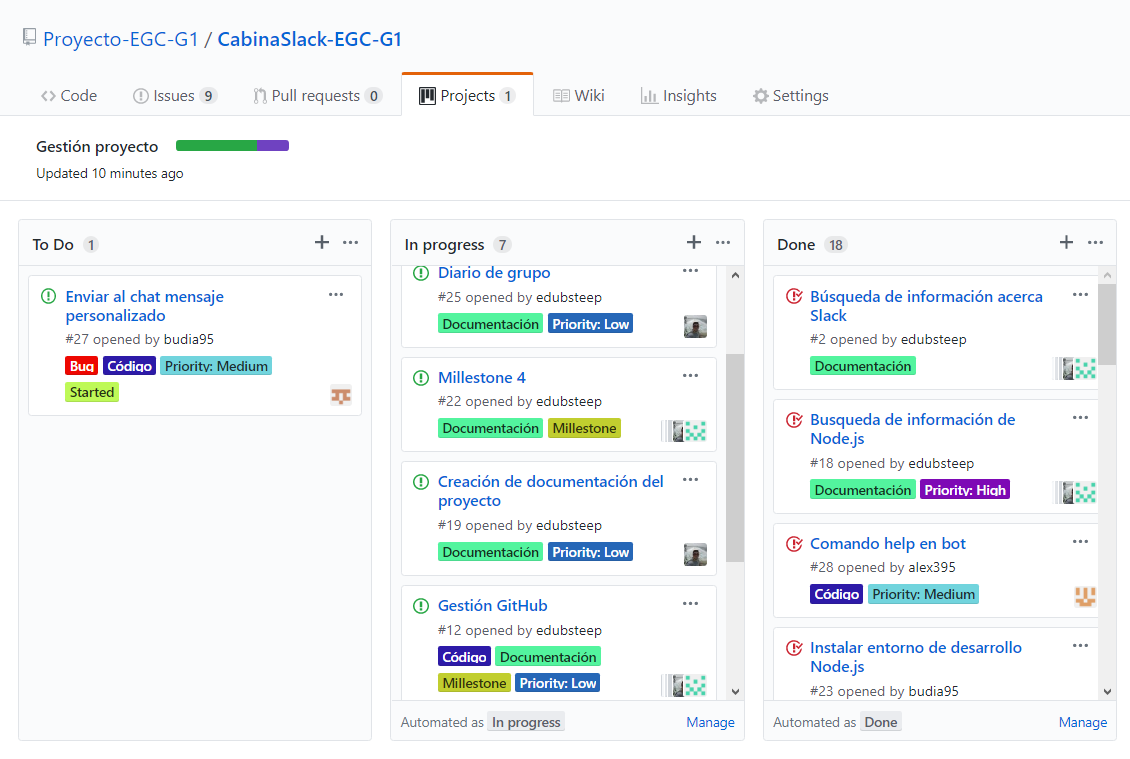
Además de estas ventajas, usar un bot nos permite realizar cualquier proceso en tiempo real y automáticamente sin la necesidad de la presencia de una personan que revise el funcionamiento de la aplicación del usuario que lo esté utilizando, junto con la facilidad de uso y rapidez de uso.

Como repaso final, esto supone una mejoría en cuanto a experiencia de uso puesto que tendríamos la posibilidad de visualizar votos desde cualquier lugar utilizando nuestro teléfono móvil sin necesidad de tener un PC en un momento concreto.

# Descripción del sistema

## Planificación del trabajo

Para tratar la gestión de tareas en nuestro proyecto hemos hecho uso de los tableros Kanban que nos proporciona GitHub, dentro del apartado de “Project”, en nuestro proyecto creado. Las columnas designadas en nuestro proyecto son ToDo, In Progress y Done, que identifican como su propio nombre indica, las tareas por hacer, las tareas cuya realización están en progreso y por último, las tareas finalizadas. Este tablero únicamente es visible por los participantes del repositorio, los cuales pueden acceder y gestionar los Issues que se creen. A continuación presentamos un ejemplo gráfico de la gestión de este tablero:



Al crearse una nueva tarea o issue primero se asigna a la columna “To Do” sin ningún participante. Posteriormente se asigna al integrante que se estime oportuno, pasando a la columna “In Progress” con el posterior desarrollo de la tarea recientemente añadida. Una vez finalizada esa tarea se procede al cierre del issue y su posterior cambio en el tablero Kanban a la columna “Done”.

La planificación completa y detallada se encuentra en el documento adjunto llamado “Diario de grupo” o también se puede acceder mediante el siguiente enlace:

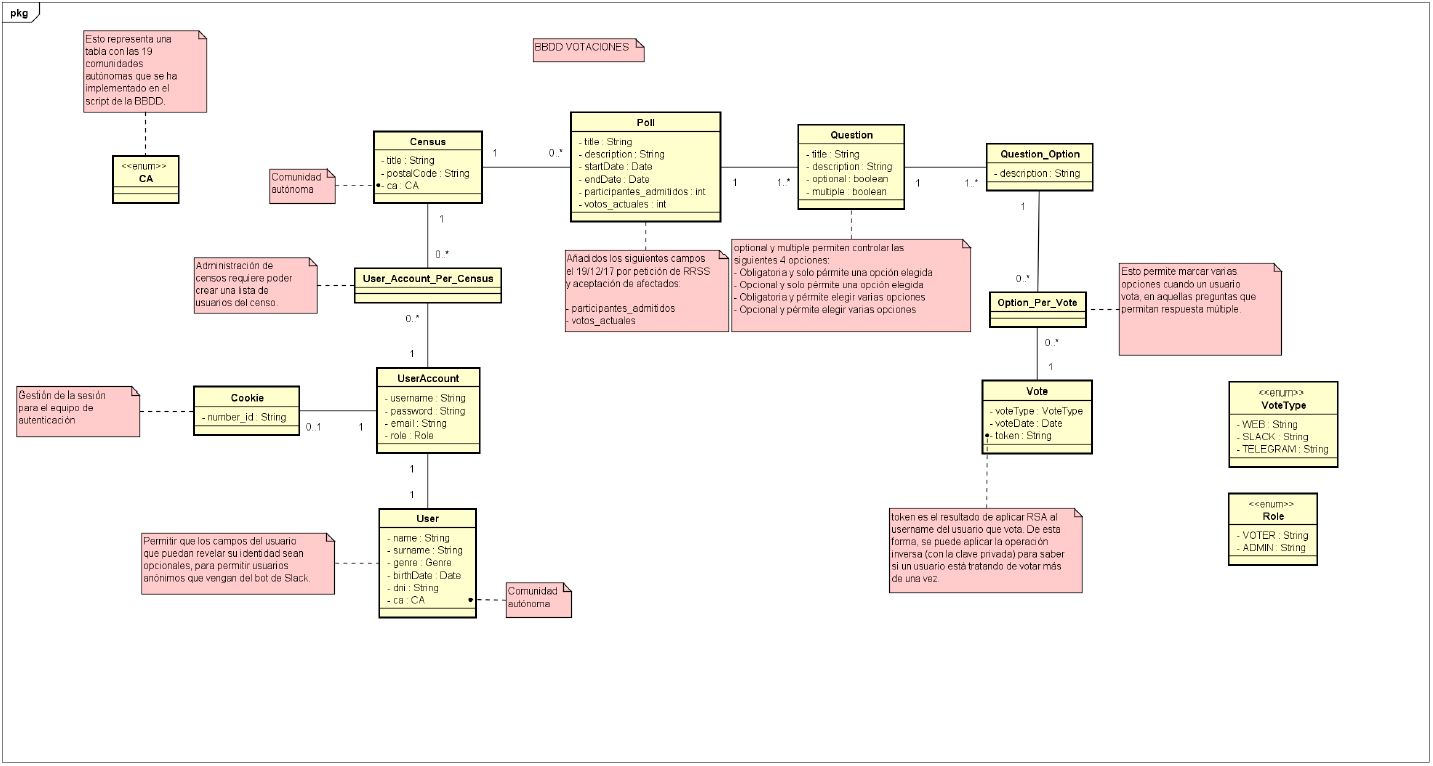
[Enlace a GitHub](https://github.com/Proyecto-EGC-G1/CabinaSlack-EGC-G1/blob/master/docs/Diario%20de%20grupo.docx)

## Cambios realizados

Nuestro proyecto contaba con una gran desventaja, ya que era la primera vez que esta cabina nunca había sido realizada, no existía la posibilidad de analizar código heredado, por lo tanto, no podemos realizar cambios a ningún proyecto anterior.

Otro de los cambios realizados fue la necesidad de cambiar el lenguaje y tecnología de nuestro proyecto, ya que tras buscar información nuestra decisión fue realizar el bot en Python pero a la hora de comenzar a programarlo, no encontramos la manera de conectar el código con la base de datos, por lo que decidimos a última hora cambiar el entorno de desarrollo a Node.js, el cual nos permitía realizar el bot y la conexión con la base de datos.

## Modelo relacional del sistema de votaciones



Como podemos ver el modelo relacional, un usuario unicamente iteractua con nosotros si decide realizar la votación por Slack, a parte de la Web y Telegram.

En si, nuestro subsistema no se integra con cualquier otro, unicamente muestra el contenido que tendrá la base de datos con las votaciones existentes, cuyo subsistema encargado será Administración de votaciones.

## 3.4 Funcionalidad

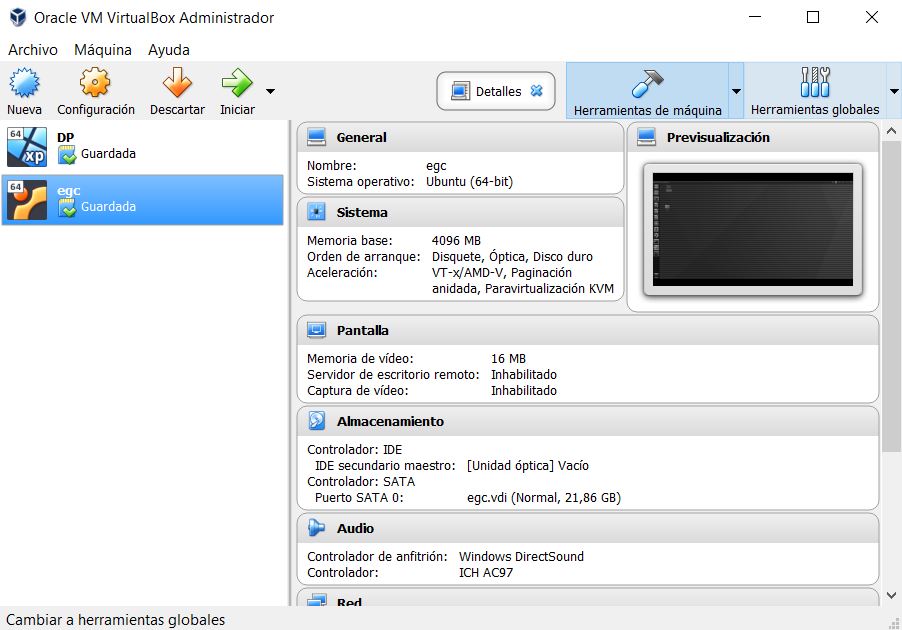
# Entornos de desarrollo

Nuestro entorno base para el desarrollo está definido por los siguientes elementos de configuración:

* Node.js (Versión 4.2.6): <https://nodejs.org/en/blog/release/v4.2.6/>
* Visual Studio Code (Versión 1.19.2): <https://code.visualstudio.com/updates/v1_19>
* Ubuntu (Versión 16.04): <https://www.ubuntu.com/download/desktop>
* Slack (Versión 3.0.3): <https://slack.com/intl/es-es/downloads/windows>
* MySQL (Versión 5.7): <https://dev.mysql.com/doc/refman/5.7/en/mysql-nutshell.html>
* Git (Versión 2.7.4): <https://git-scm.com/download/linux>
* Virtual Box (Versión 5.2.4): <https://www.virtualbox.org/wiki/Downloads>
* Docker (Versión 17.05.0-ce): [https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-16-04-es](https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-16-04-es%20)
* Docker-compose (Versión 1.18): [https://docs.docker.com/compose/install/#install-compose](https://docs.docker.com/compose/install/%23install-compose)
* Npm (Versión 5.6.0): <https://www.npmjs.com/package/download>

## Instalación del entorno de desarrollo

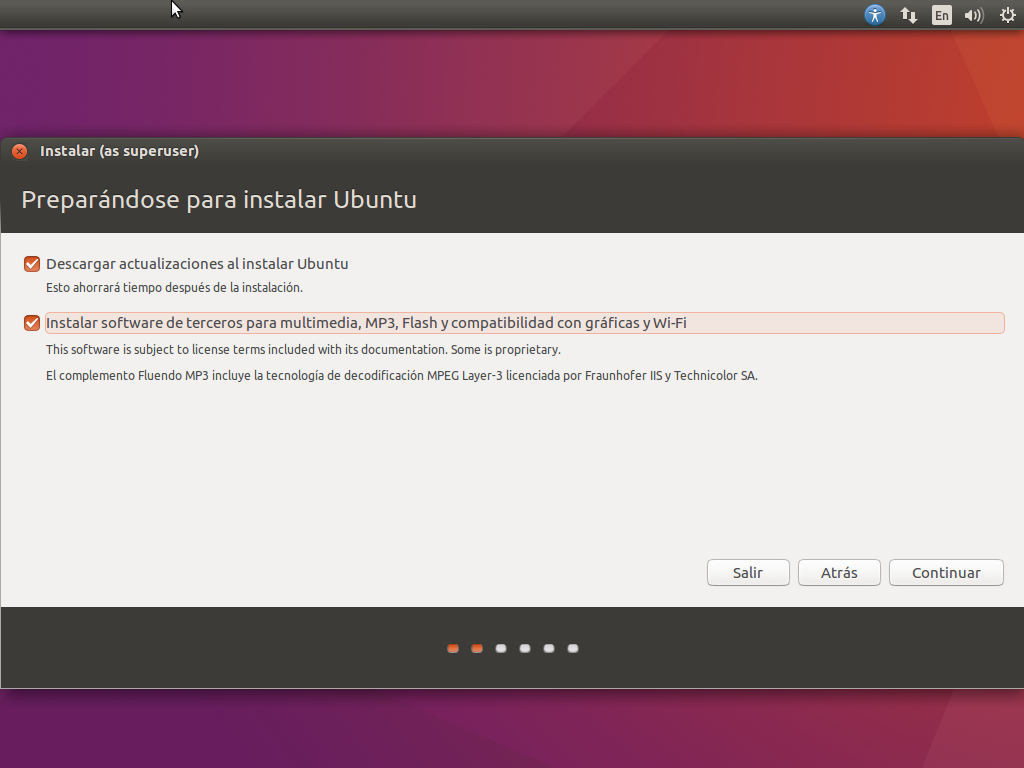
En primer lugar, utilizaremos un Virtual Box (Versión 5.2.4) para poder instalar una máquina virtual de Ubuntu. Para ello añadimos una nueva maquina virtual, declaramos que queremos que sea Ubuntu y iniciamos la máquina:



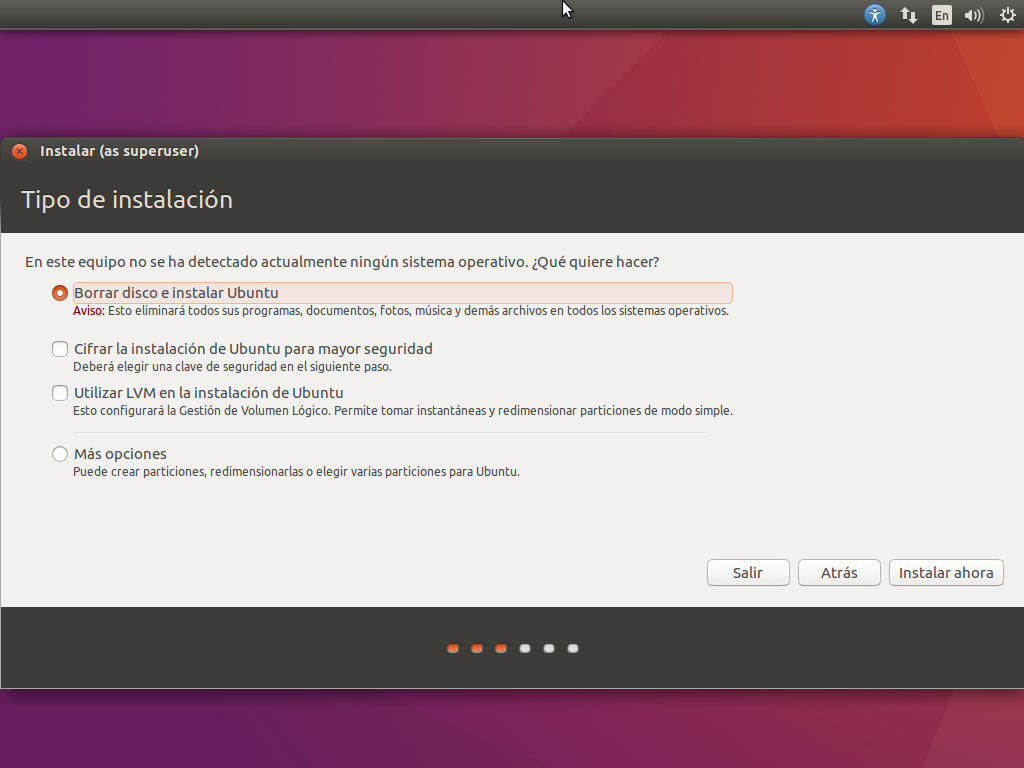
Para la instalación de Ubuntu procedemos a descargar el .iso en la página anteriormente indicada y la añadimos a la máquina virtual. Tras esto procedemos a la instalación de Ubuntu tal y como muestra la siguiente imagen:



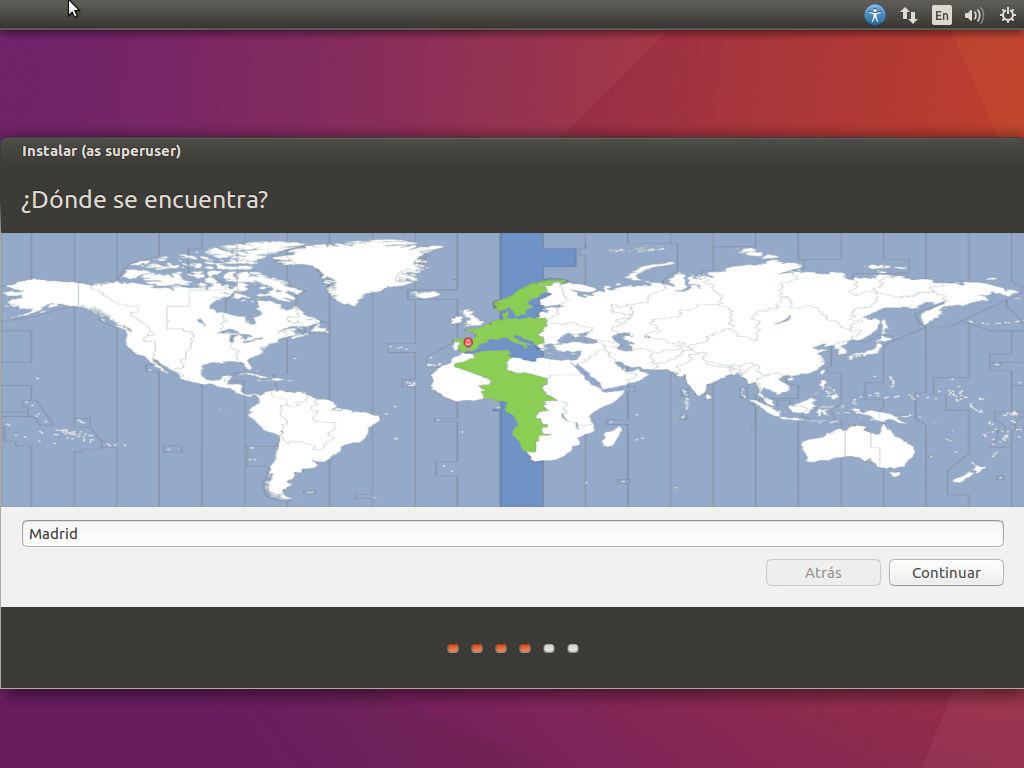
En la siguiente pantalla nos da la opción de descargar las actualizaciones e instalar software de terceros para reproducir archivos multimedia y otros. No obstante ninguno de los dos es obligatorio ya que podemos hacerlo una vez que el sistema ya esté instalado:



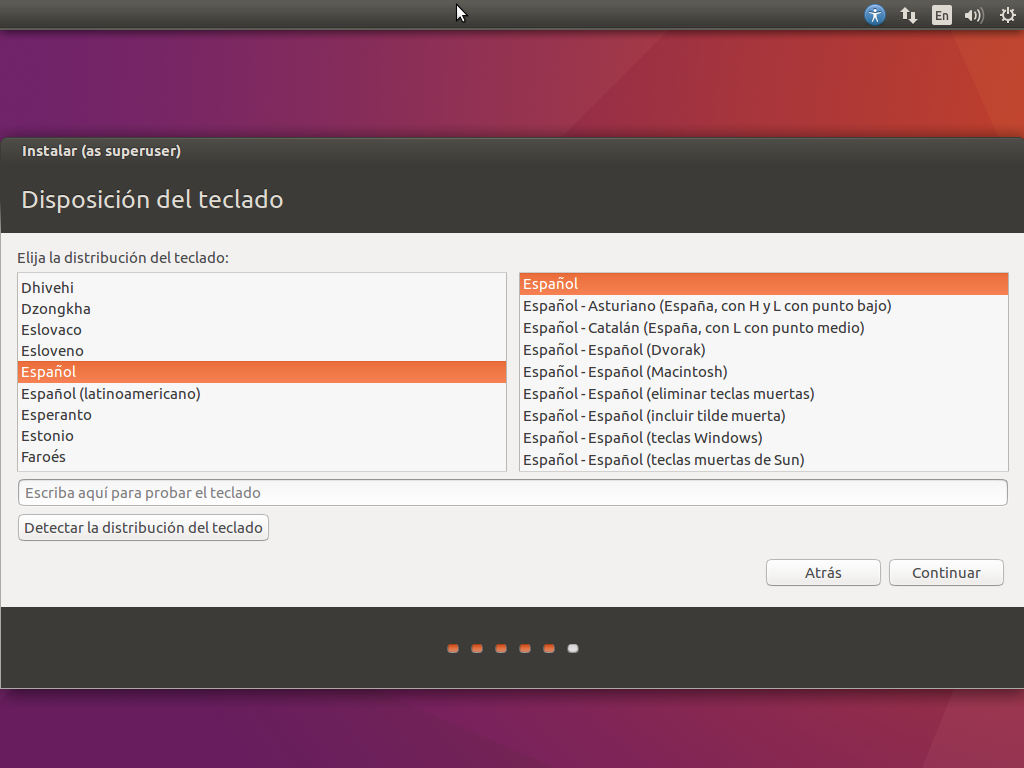
A continuación nos saldrá el asistente de instalación, en este caso vamos a usar el disco completo .vdi creado al crear la maquina virtual por lo que dejamos la opción por defecto y pulsamos en “instalar ahora”:



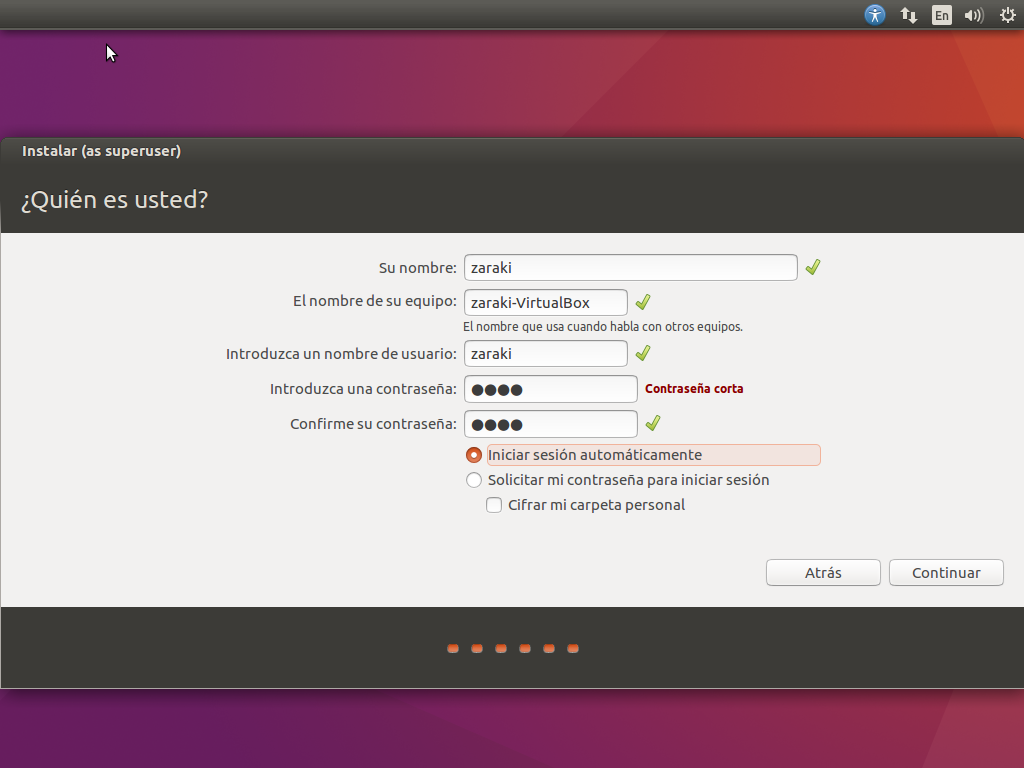
Tras la instalación elegimos nuestra zona horaria y pulsamos en “continuar”



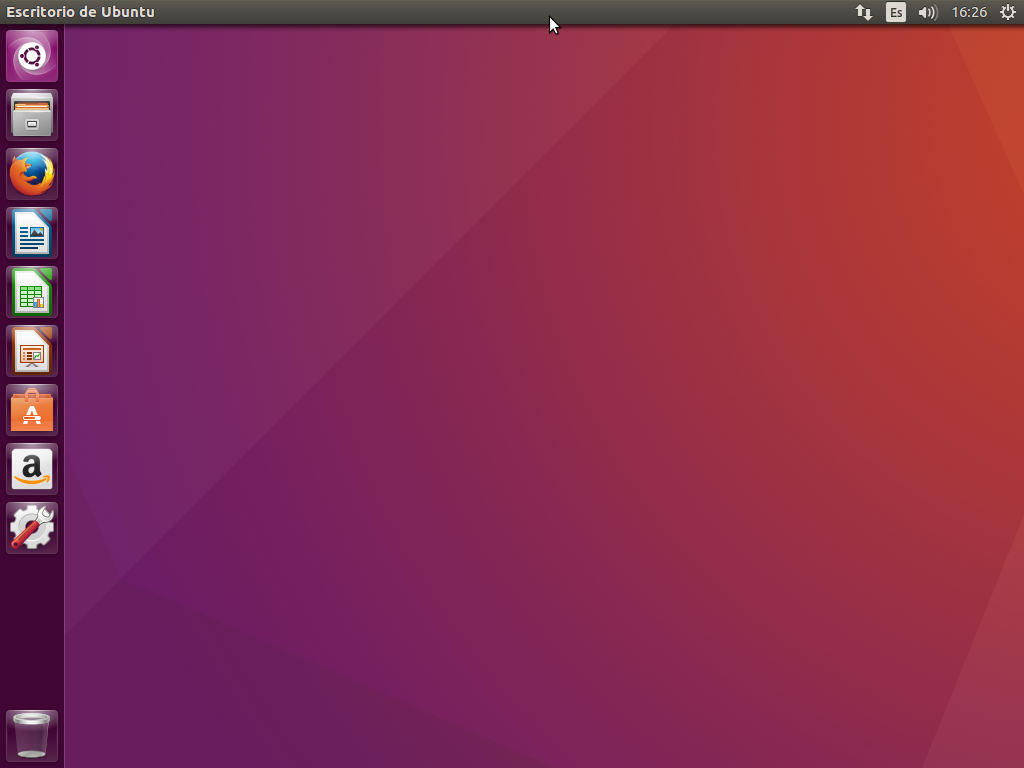
A continuación elegimos nuestra disposición del teclado y “continuar”:



Por último nos saldrá una última pantalla en la que debemos poner nuestro nombre de usuario y contraseña entre otras opciones:



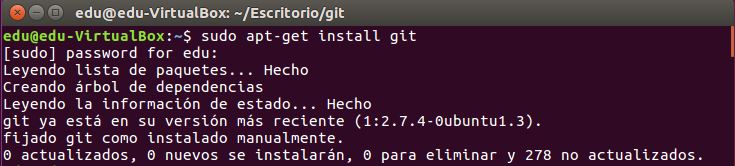
Finalmente reiniciamos el sistema, arrancamos desde el disco duro y ya tendremos Ubuntu 16.04 funcionando.



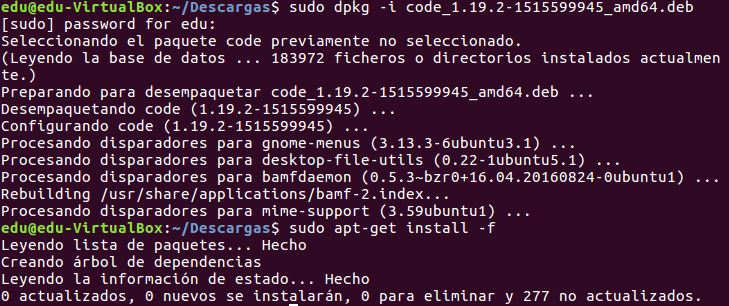
## Instalación herramientas en Ubuntu

Para el correcto funcionamiento del proyecto en Ubuntu, es necesario instalar varias herramientas para el correcto funcionamiento. A continuación vamos a proceder a nombrar cada una de estas herramientas, así como, los comandos necesarios para la instalación en una terminal de Ubuntu:

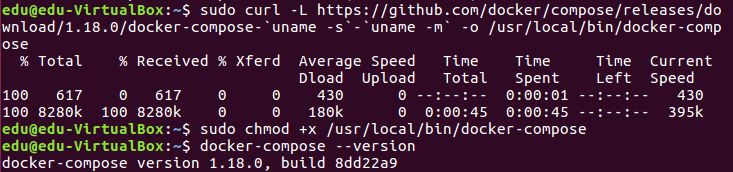
* Git: la instalación de git es muy sencilla, ya que con un simple comando, podemos instalar todas las extensiones de git. Este comando es “sudo apt-get install git”.



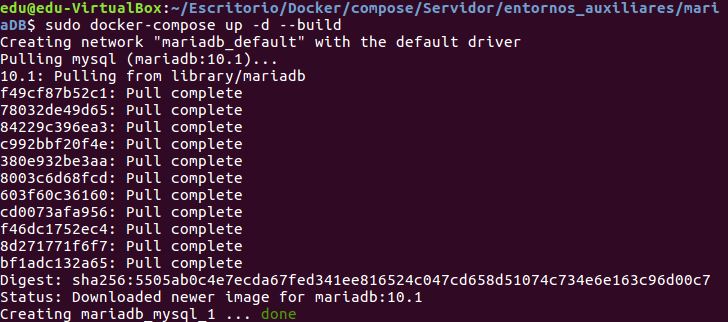
* Visual Studio: para la instalación de Visual Studio es necesario la descarga del paquete con enlace en el apartado 4. Posteriormente debe ejecutar 2 comandos en la terminal para completar su instalación “sudo dpkg -i nombre\_de\_app.deb” y “sudo apt-get -f install”.



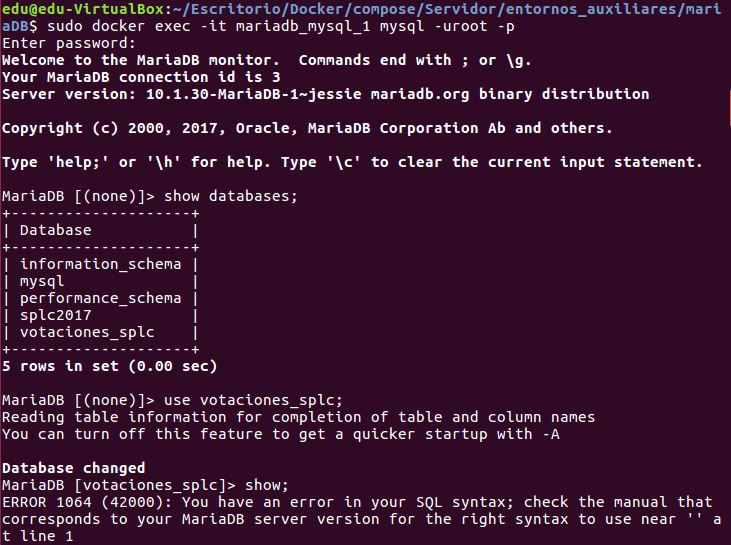
* Docker-compose: para poder ejecutar el contenedor dado por el equipo de integración, debemos de instalar el Docker-compose, el cual es una herramienta para la ejecución de contenedores. Para la instalación son necesarios los comandos, con su correspondiente representación gráfica:
  + “sudo curl -L 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”



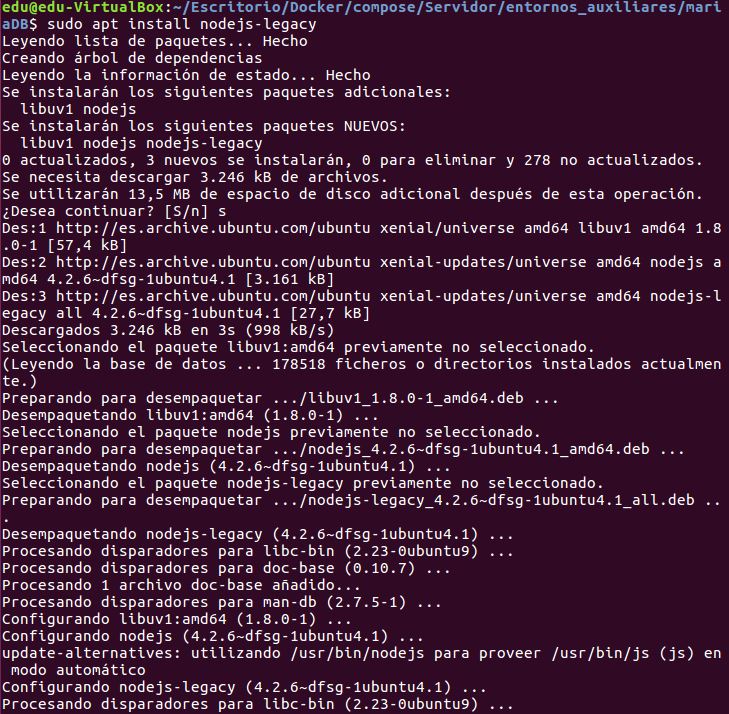
* MariaDB: Para la ejecución del contenedor, hay que ejecutar el comando “docker-compose build up -d –build” únicamente para la construcción de este, después solo se utilizaría “docker-compose up -d”. La estructura de la respuesta será la siguiente:



Para poder acceder a la base de datos debemos ejecutar el comando “sudo docker exec -it mariadb\_mysql \_1 mysql -uroot -p”

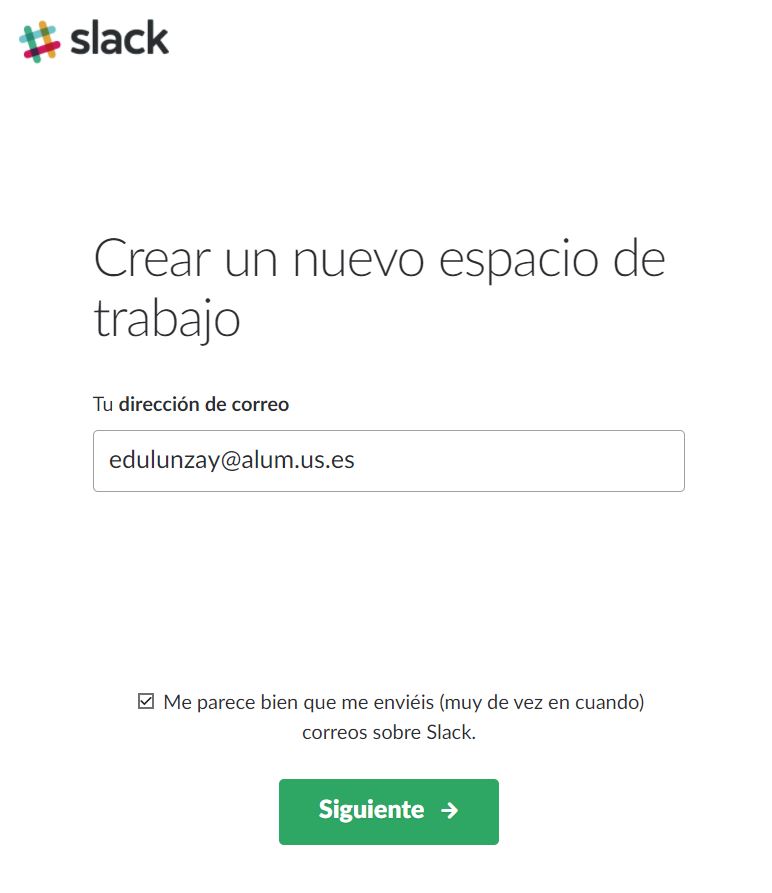


* Node.js: para la instalación de nodejs, es necesario la ejecución del comando “sudo apt install nodejs-legacy”. Esto instalará todas las herramientas necesarias para el funcionamiento de nodejs.

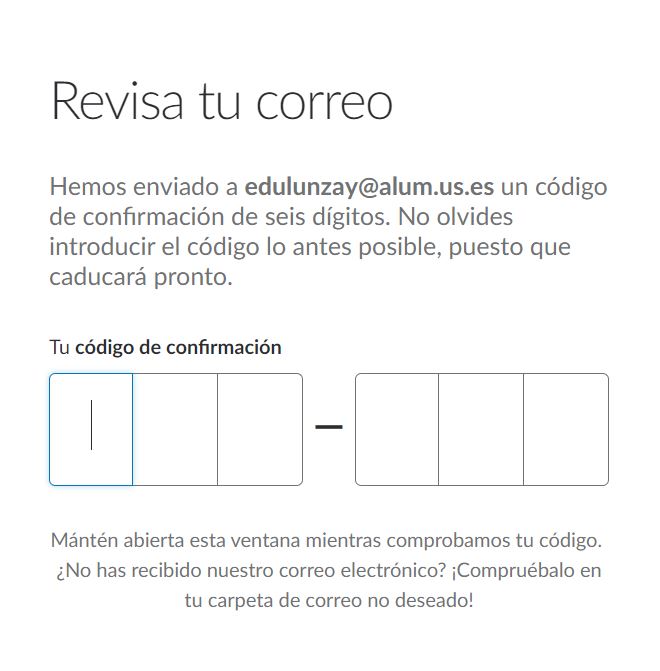


## Creación de bot

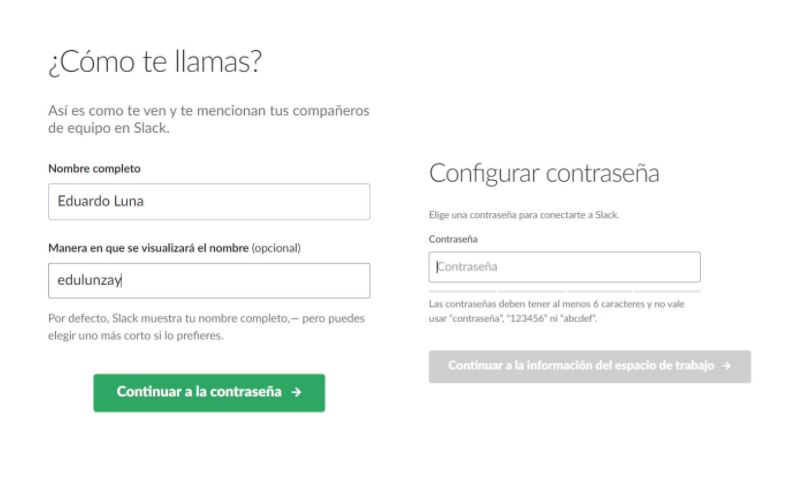
Para la creación del bot, es necesario crear un espacio de trabajo en Slack donde poder alojarlo. Para ello es necesario introducir un correo electrónico tal y como muestra la imagen:



Posteriormente hay que introducir un código de confirmación, que hemos recibido en el correo que hemos introducido anteriormente.



A continuación debemos de introducir un nombre de usuario y una contraseña, así como, los intereses por los que creo el espacio:





Tras esto, se procede a la elección del nombre del espacio de trabajo y creación final del mismo:



Por último, ya tenemos acceso a slack para poder crear canales, añadir miembros, así como, añadir bots.



## Conclusión

Para finalizar este apartado, todo lo referente a la programación y creación del código se realiza mediante código JavaScript y ejecutándolo mediante Nodejs.

# Gestión del código fuente

## Roles

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

* Desarrollador: todos miembros del equipo que producen incrementos en el subsistema. Una vez finalizado el proceso de desarrollo y comprobado el correcto funcionamiento deberá hacer un merge a la rama master
* Jefe de proyecto: este rol es fijo durante el desarrollo del proyecto. También es un desarrollador. Este describe las pautas para la gestión del código fuente, incidencias y cambios.

## 5.2 Gestión de ramas en el código

En la planificación inicial del proyecto se decidió las ramas a crear para el uso en la gestión del código:

* master: rama en la que se encuentra el código final y estable, preparado para incluirlo en el entregable.
* develop: rama que cuenta con los últimos cambios añadidos y a la que se suben éstos.
* documentation: rama dedicada a la subida de documentos en el repositorio.
* testing: rama dedicada a la realización de pruebas del código.

Para la gestión del código (y de las ramas) se utiliza Git. Un breve resumen de la utilización de estos comandos se puede ven en el siguiente enlace:

<https://elbauldelprogramador.com/mini-tutorial-y-chuleta-de-comandos-git/>

## 5.3 Gestión Labels en GitHub

Tras algunas opiniones e ideas para poder llevar a cabo una mejora del proyecto con respecto a las labels hemos propuesto una serie de pasos a seguir para que esto pueda llevarse a cabo con una eficiencia mayor a la anterior.

Nos enfrentamos ante un posible cambio, corrección, prueba o mejora del código fuente y procedemos a hacer el cambio. El desarrollador que se disponga a hacer este cambio deberá crear una 'issue' con su correspondiente justificación en ella. Una de las mejoras aplicadas es la incorporación de etiquetas para distinguir las distintas issues creadas.

Cuando se crea una issue, se deberá obligatoriamente añadir una etiqueta según el cambio que se vaya a realizar:

* bug: si este cambio se trata de la corrección de un error en el código.
* enhancement: si consiste en mejorar algún tipo de metodología.
* question: en el caso que se quiera resolver alguna duda sobre el cambio que vaya a realizarse antes de ponerse a trabajar sobre él.
* help wanted: un desarrollador del equipo de trabajo puede 'pedir ayuda' a otro compañero que pueda contribuir en la realización de la issue que ha creado.

A continuación, el desarrollador que haya creado la nueva 'issue' deberá asignársela a sí mismo o ser asignada por algún otro desarrollador.

Creadas la issues correspondientes, el desarrollador puede empezar a trabajar en el hito que se ha marcado a realizar en el proyecto. Todo esto está gestionado por GitHub; por ello, 'subiremos' y 'descargaremos código' de nuestro proyecto en la herramienta con el uso de commits.

## Caso práctico

En el caso práctico se va a exponer cómo se ha realizado una iteración en el código. En primer lugar, uno de los miembros del equipo crea un “Issue” en el repositorio de nuestro subsistema (<https://github.com/Proyecto-EGC-G1/CabinaSlack-EGC-G1>) en el que se propone la funcionalidad a implementar, tal como se aprecia en la imagen:

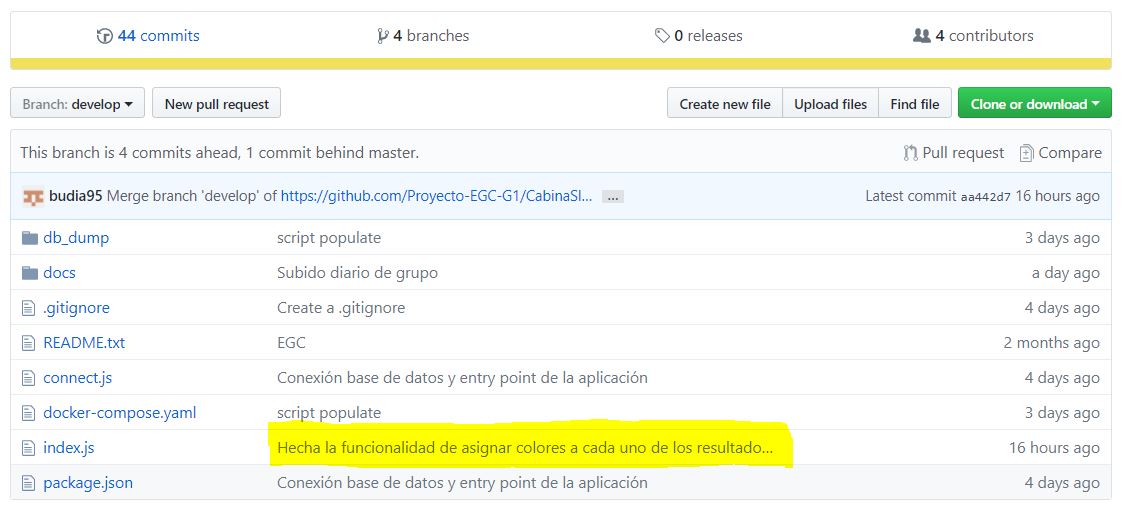


Una vez creado el issue añadimos campos correspondientes a la plantilla que hemos planteado en la gestión de las labels (punto 5.2) y a continuación rellenamos según requiera la incidencia las etiquetas que se correspondan a nuestra petición de cambio.

Tras la asignación de issue, el desarrollador se dispone a trabajar sobre lo pedido para la mejora del proyecto. Siempre que se realice un nuevo cambio, sea de mayor o menos impacto, deberá realizarse en la rama “develop”.

A continuación el issue deberá trasladarse por los diferentes estados en el que se encuentre la tarea dentro del proyecto. Este paso quedará definido en el apartado 7 “Gestión del cambio, incidencias y depuración”.

Una vez realizado el cambio en nuestro proyecto haremos la subida a GitHub mediante Git, tal y como aparece en la siguiente imagen:



Una vez realizado todos estos pasos, el desarrollador encargado del issue, deberá verificar el cambio subido en la rama “develop” y realizar las pruebas necesarias para dar un veredicto. Tras la verificación y correcto funcionamiento, se hará un merge a la rama “master” y el cambio quedará integrado en nuestro proyecto sin posibilidad de cambios.

Para finalizar la gestión, el desarrollador encargado deberá proceder al cierre del issue y automáticamente, este pasará al estado “done” en nuestro proyecto:



# Gestión de la construcción e integración continua

Con respecto a la integración continua, nuestro sistema se integra con los demás mediante Travis CI, en el que mediante un archivo llamado .travis.yml que incluye la configuración para la ejecución del proyecto incluido en nuestro repositorio de GiHub. Si alguna de estas tareas falla el build producido se considera como “Failed”. Si no falla ninguna tarea Travis toma esta tarea como “Passed”, pudiendo desplegar para su integración completa.

Los pasos a seguir para la integración en Travis son los siguientes:

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

Por último nos gustaría destacar la falta de comunicación con el subsistema de almacenamiento de votos, ya que nos proporcionó el código para efectuar el almacenamiento de nuestros votos con poca antelación a la entrega del proyecto, y en código Python, por lo cual no teníamos tiempo disponible para la realización de cambio de lenguaje de este.

# Gestión del cambio e incidencias

Los cambios se controlan mediante el sistema de control de versiones GitHub junto con la herramienta proporcionada por GitHub en la gestión del proyecto. Mediante la técnica del tablero Kanban se reparten las tareas en distintas columnas como se explica con anterioridad (en dichas columnas se pueden apreciar los cambios que se pretenden realizar). Para los cambios ya realizados basta con acceder a la sección issues de GitHub y acceder a ellos en la sección *Closed*. Para un nivel mayor de detalle de los cambios realizados se puede acceder al historial de commits al repositorio.

Como se acaba de explicar, los cambios tienen tres estados:

* To Do
* In Progress
* Done

En cuanto a roles no hay una diferenciación de privilegios a la hora de gestionar los cambios, cualquier miembro puede realizar cualquier acción sobre los cambios.

Por lo general, los cambios se extraen todos de los requisitos definidos en un inicio, luego cualquier cambio que entre al estado “To Do” será realizado tarde o temprano. Tampoco se diferencian prioridades para seleccionar un cambio. El único factor que puede determinar el desarrollo más o menos tardío de un cambio podría ser el tamaño que se asigna a cada tarea (priorizando los que más tiempo puedan consumir).

La depuración se realiza de la misma forma para incidencias internas como para aquellas que procedan de otro subsistema. El equipo analizará el origen de la issue, buscando su causa y las posibles soluciones que existan a esta. Deberá de comunicarse con el equipo de integración para dar el positivo de realizar el cambio. De todas las posibles soluciones el equipo selecciona la más convincente y la aplica. Tras esto se notifica al resto del equipo o al equipo del subsistema externo de que la incidencia ha sido solucionada. En caso de tratarse de un error de código, el encargado de solucionarlo será el que desarrolló dicho módulo.

# Gestión de liberaciones, despliegue y entregas

Se consideran entregables el conjunto de código ya liberado mediante GitHub. Dichos entregables sirven para identificar versiones concretas del software, y así poder subsanar fallos identificando la construcción que está usando el cliente.

El despliegue se realizará entregando la máquina virtual donde se aloja el proyecto o, en su defecto, el contenedor de docker donde se encuentre el proyecto. Esta máquina tendrá todo lo necesario para la ejecución del proyecto.

Una vez dentro de la máquina virtual debe situarse en la carpeta donde se encuentre el proyecto para poder ejecutar el archivo “index.js” en el que se encuentra el bot.

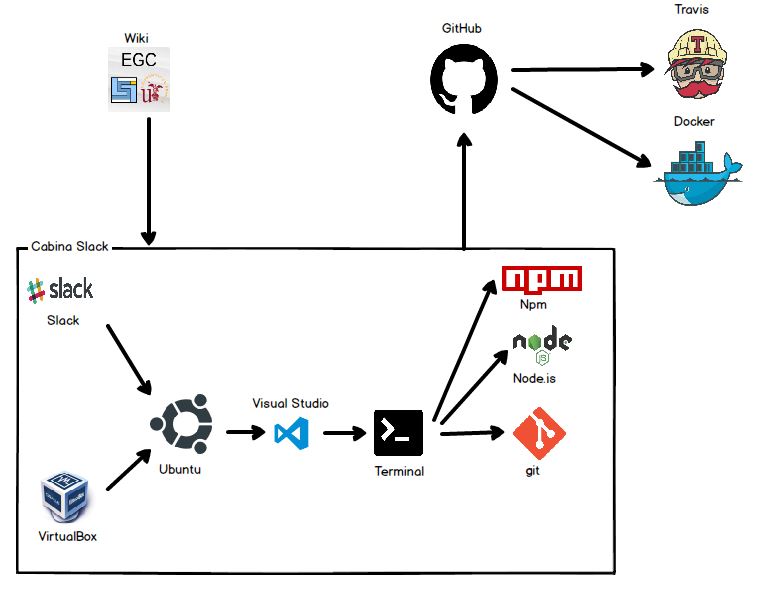
Por último, se despliega en GitHub por defecto en su rama develop, cuando se considere por consenso de todos los miembros del grupo que la versión en desarrollo es correcta y pasa los tests en Travis lanzados automáticamente al detectar dicho despliegue, el coordinador del grupo realiza un “merge” con su rama estable.

Durante la entrega del proyecto, cada uno de los integrantes del grupo revisará cada entregable de manera grupal.

En esta situación, un miembro será el encargado de preparar la máquina virtual, comprobando que todo funciona correctamente. Luego de esto, el mismo miembro se encargará de subirla donde corresponda.

Los demás miembros del equipo se centrarán en revisar la documentación del proyecto y el diario de equipo, cuidando el formato y procurando darle un toque profesional y técnico. Cuando todo esté revisado, se subirá al portal Opera un archivo comprimido con ambos entregables.

# Mapa de herramientas



# Conclusiones y trabajo futuro

Tras la realización del proyecto, realmente estamos muy orgullosos de los objetivos cumplidos en el mismo, debido a que el equipo de Cabina de Slack del año pasado ni siquiera existía, por lo que tuvimos que partir desde 0. Por otro lado, la creación de un bot de Slack ha sido un acontecimiento realmente bueno, ya que hemos aprendido a integrar un elemento tan curioso como la que es un bot además de aprender a utilizar una herramienta como es Slack, la cual se usa bastante en el ámbito laboral.

Como ya hemos comentado antes, no disponíamos de código heredado el cual pudiéramos basarnos e incrementar su funcionalidad, por lo cual nos enfrentamos a un gran reto.

Finalmente, mencionar que nunca nos hemos llegado a enfrentar a un proyecto de esta envergadura y con una planificación o coordinación en común.

Como propuesta de mejora para un futuro sería añadir la funcionalidad de poder guardar el resultado de las votaciones.