

agora@us

Grupo 9: Creación y Administración de Censos



2016/2017

Grupo 2 (tarde)

Id Opera: 9

Pablo Romero Vázquez

Santiago Fraga Martín-Arroyo

Nicolás Lorenz Rosado

Simón Egea Guerrero

Rubén Barrientos Mohedano

# Historial de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Autor** | **Descripción** | **Fecha** |
| 1.0 | Santiago Fraga Martín-Arroyo | Realización del esquema del documento y los apartados de gestión del código deficiente, gestión de incidencias deficientes y mejora de la documentación | 30/01/2017 |
| 1.1 | Pablo Romero Vázquez | Realización del apartado de .gitignore y corrección del documento | 30/01/2017 |
| 1.2 | Santiago Fraga Martín-Arroyo | Corrección de errores y realización del apartado de Uso de Jenkins solo en local | 31/01/2017 |
| 1.3 | Nicolás Lorenz Rosado | Creación del apartado de Integración con otros subsistemas | 31/01/2017 |
| 1.3 | Santiago Fraga Martín-Arroyo | Unión de los distintos apartados, corrección del formato y revisión. | 01/02/2017 |
|  |  |  |  |

Índice

[Historial de versiones 2](#_Toc473717543)

[Tabla de figuras 4](#_Toc473717544)

[Resumen 6](#_Toc473717545)

[Mejoras realizadas 7](#_Toc473717546)

[Gestión del código deficiente 7](#_Toc473717547)

[Gestión de incidencias deficiente 8](#_Toc473717548)

[No se usa el .gitignore 11](#_Toc473717549)

[Uso de Jenkins solo en local 12](#_Toc473717550)

[Integración con otros sistemas 20](#_Toc473717551)

[El .war, el despliegue y la construcción se genera de forma manual 25](#_Toc473717552)

[Mejora de la documentación 34](#_Toc473717553)

# Tabla de figuras

[Ilustración 1Commits de uno de los miembros 7](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717491)

[Ilustración 2Incidencia en github 8](#_Toc473717492)

[Ilustración 3 Incidencia #23 8](#_Toc473717493)

[Ilustración 4 Labels de una incidencia 9](#_Toc473717494)

[Ilustración 5 Archivo gitignore 11](#_Toc473717495)

[Ilustración 6 Ejemplo de git rm 11](#_Toc473717496)

[Ilustración 7Login de jenkins 12](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717497)

[Ilustración 8Login de Jenkins 12](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717498)

[Ilustración 9Panel principal de jenkins 13](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717499)

[Ilustración 10Administracion de Jenkins 13](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717500)

[Ilustración 11Instalar plugin Jenkins 14](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717501)

[Ilustración 12Configurar plugin de maven Jenkins 14](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717502)

[Ilustración 13Configurar jdk en jenkins 15](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717503)

[Ilustración 14Crear tarea en Jenkins 15](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717504)

[Ilustración 15Configurar tarea en jenkins 1 16](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717505)

[Ilustración 16Configurar disparadores en Jenkins 17](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717506)

[Ilustración 17Configurar origen del codigo en Jenkins 17](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717507)

[Ilustración 18Configurar pasos previos en Jenkins 18](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717508)

[Ilustración 19Configurar proyecto en Jenkins 18](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717509)

[Ilustración 20Ver tarea en jenkins 19](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717510)

[Ilustración 21 Demostración conexión 1 20](#_Toc473717511)

[Ilustración 22 Demostración conexión 2 21](#_Toc473717512)

[Ilustración 23 Antiguo método de crear censo 21](#_Toc473717513)

[Ilustración 24 Modificación de servicio 22](#_Toc473717514)

[Ilustración 25 Modificación controlador 22](#_Toc473717515)

[Ilustración 26 Adición de línea para crear un censo 22](#_Toc473717516)

[Ilustración 27 Datos del fichero a usar 23](#_Toc473717517)

[Ilustración 28 Selección del método 23](#_Toc473717518)

[Ilustración 29 Demostración del método 23](#_Toc473717519)

[Ilustración 30Heroku login 25](#_Toc473717520)

[Ilustración 31Heroku create 25](#_Toc473717521)

[Ilustración 32Heroku instalación plugin 25](#_Toc473717522)

[Ilustración 33Heroku deploy 26](#_Toc473717523)

[Ilustración 34Heroku panel de control 26](#_Toc473717524)

[Ilustración 35Heroku application error 27](#_Toc473717525)

[Ilustración 36App Engine 28](#_Toc473717526)

[Ilustración 37Jelastical creación del entorno 28](#_Toc473717527)

[Ilustración 38Jelastical entorno creado 29](#_Toc473717528)

[Ilustración 39Jelastical creación de la base de datos 29](#_Toc473717529)

[Ilustración 40Jelastical despliegue 30](#_Toc473717530)

[Ilustración 41Versión de MYSQL Server 30](#_Toc473717531)

[Ilustración 42Versión de maven 30](#_Toc473717532)

[Ilustración 43Versión de git 31](#_Toc473717533)

[Ilustración 44Creación base de datos 31](#_Toc473717534)

[Ilustración 45Despliegue war 31](#_Toc473717535)

[Ilustración 46Archivo host 31](#_Toc473717536)

[Ilustración 47Configuración archivo hosts 31](#_Toc473717537)

[Ilustración 48Vaciado de DNS 32](#_Toc473717538)

[Ilustración 49ADMCensus 32](#_Toc473717539)

[Ilustración 50Copy Artifact plugin 32](#_Toc473717540)

[Ilustración 51Deploy to Websphere container plugin 32](#_Toc473717541)

[Ilustración 52Jobs 33](file:///D:\Universidad\Cuarto\EGC\Trabajo\Documentacion%20de%20las%20mejoras.docx#_Toc473717542)

# Resumen

En la siguiente tabla se muestran todas las mejoras que el profesor de la asignatura nos ha recomendado realizar, a quien se le ha asignado dicha mejora y si se ha realizado correctamente.

|  |  |  |  |
| --- | --- | --- | --- |
| Titulo | Descripción | Asignado a | Solucionado |
| Gestión del código deficiente | Uno de los miembros del grupo solo realiza un commit a lo largo del desarrollo | Pablo Romero Vázquez | SI |
| Gestión de incidencias deficiente | La gestión de incidencias es demasiado pobre y no se pone en práctica con naturalidad | Grupo | SI |
| No se usa .gitignore | Cada vez que se hace un commit se suben archivos que no son necesarios como los .class, .classpath o .project | Pablo Romero Vázquez | SI |
| Uso de Jenkins solo en local | Jenkins solo se usa en local | Grupo | SI |
| Integración con otros sistemas | El sistema no interacciona con otros subsistemas | Nicolás Lorenz Rosado | SI |
| El .war se genera de forma manual | La generación del war se realiza de forma manual | Simón Egea Guerrero y Rubén Barrientos Mohedano | SI |
| El despliegue y construcción se hace manual | Tanto el despliegue como la construcción del sistema se hace de forma manual | Simón Egea Guerrero y Rubén Barrientos Mohedano | SI |
| Mejora de la documentación | Se debe de mejorar la documentación en su totalidad | Pablo Romero Vázquez y Santiago Fraga Martín-Arroyo | SI |

# Mejoras realizadas

A continuación, se detallan las mejoras que se han conseguido realizar y como se han llevado a cabo.

## Gestión del código deficiente

Uno de los miembros del grupo solo había realizado un “commit” durante todo el desarrollo del subsistema, debido a esto se decidió que realizase el archivo “.gitignore” y algunos test que faltaban por realizar.

Tras la realización de las tareas anteriormente mencionadas dicho miembro ha aumentado sus “commits” hasta llegar a 7, tal y como podemos observar en la siguiente figura.



Ilustración Commits de uno de los miembros

## Gestión de incidencias deficiente

Debido a que no teníamos soltura a la hora de gestionar las incidencias generadas en nuestro subsistema se optó por realizar una mejora de esta sección la cual viene detallada a continuación.

Para gestionar las incidencias se deberá exponer dicha incidencia de la forma más clara posible y evitando ambigüedades. Si es necesario se deben de poner los pasos a seguir para reproducir el error en otra máquina distinta.

Comenzamos situándonos en el apartado issues de GitHub:

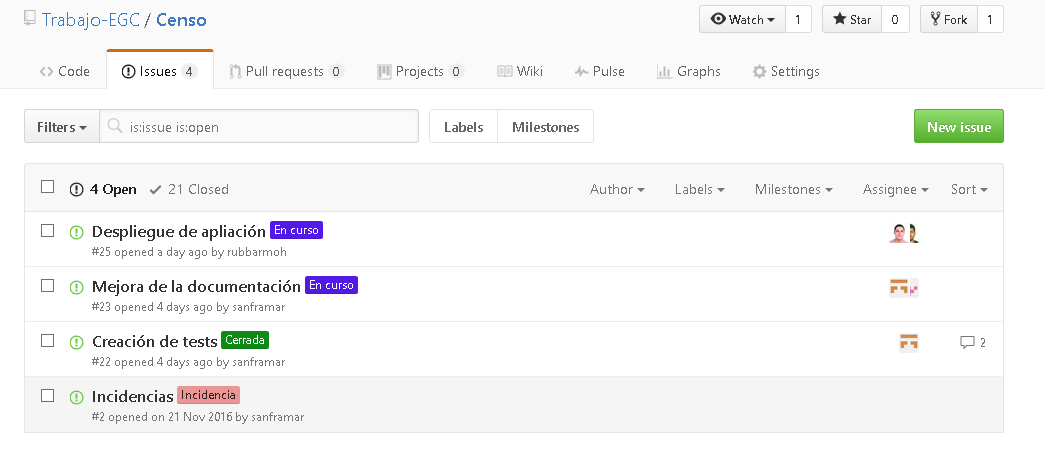


Ilustración Incidencia en github

Como se observa en la imagen, tenemos 25 incidencias, 4 abiertas y 21 cerradas. En este caso vamos a ver la incidencia “Mejora de la documentación”.

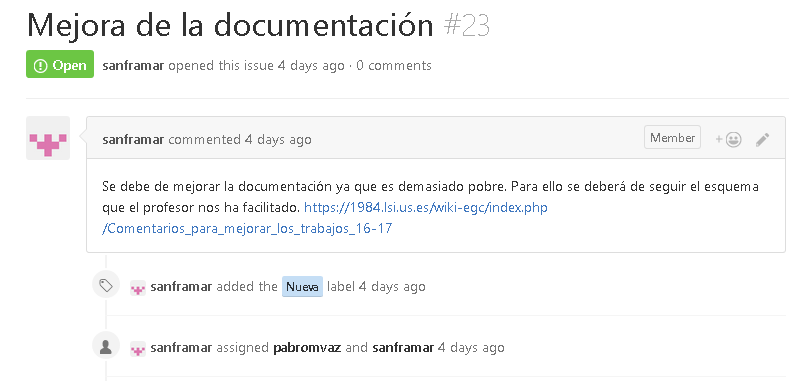


Ilustración Incidencia #23

Tal y como puede observarse, la incidencia sigue el esquema anteriormente mencionado, compuesto de un título, una descripción y una etiqueta.

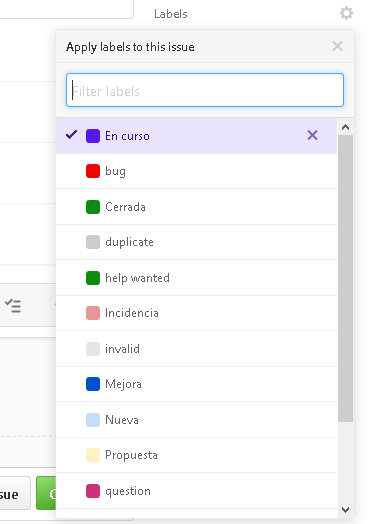


Ilustración Labels de una incidencia

En el partado “Label”, podremos asignar la etiqueta que creamos oportuna a la incidencia. En caso de no existir la etiqueta que queramos, se puede crear.

En nuestro caso, la evolución de las etiquetas de una incidencia seria la siguiente:

* Nueva: incidencia recién creada y en la que aún no se está trabajando
* En curso: la incidencia ya tiene a un miembro del equipo asignado y está trabajando en ella.
* Revisar: la incidencia ya está implementada en una rama distinta a la rama master para ser probada por un miembro distinto al que trabajó en ella.
* Cerrada: la incidencia funciona correctamente y se ha implementado en la rama master.

Cabe destacar que en la incidencia se puede mencionar a un determinado compañero mediante el comando “@nombreDelCompañero” para que GitHub le notifique. Además, también pueden referenciarse incidencias en los commits mediante los comandos “Fixed”, “Fixes” o “Close” + #numeroDeLaIncidencia. Si por algún casual vuelven a surgir conflictos, la incidencia puede ser reabierta.

## No se usa el .gitignore

Con el objetivo de eliminar del repositorio archivos innecesarios y evitar que volvieran a ser subidos, se ha creado un fichero “.gitignore”.

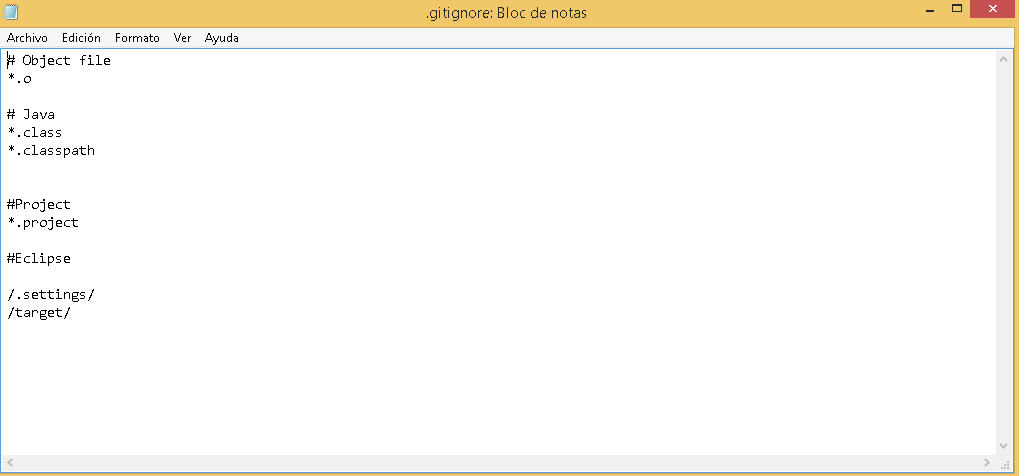


Ilustración Archivo gitignore

En la ilustración podemos observar aquellos archivos que no deseamos que sean subidos. Cabe destacar que en “#Eclipse” hemos indicado que no se suban las carpetas “settings” y “target”, de forma que se ignora todo lo que contenga.

Previamente, para borrar los archivos del repositorio, podemos hacerlo eliminando manualmente los archivos y haciendo un push al repositorio o bien podemos hacerlo con los comandos que nos proporciona git y realizando el push. En nuestro caso, hemos usado los comandos de git. Dichos comandos son “git rm nombreDelArchivo” y “git rm -r nombreDeCarpeta” en caso de que querer borrar los archivos tanto del repositorio como del sistema y “git rm --cached nombreDeArchivo” si queremos borrar los archivos en el repositorio pero queremos mantenerlos en el sistema. Como los archivos que no se subirán a GitHub son los que genera automáticamente eclipse, se optó por borrarlos también del sistema y una vez creado el archivo “.gitignore”, ya no volverían a ser subidos.



Ilustración Ejemplo de git rm

## Uso de Jenkins solo en local

Se ha configurado Jenkins para que automáticamente se descargue el código de nuestro repositorio en Github y ejecute las pruebas para comprobar que todo funciona correctamente, dicha configuración viene detallada a continuación.

La primera vez que entremos en Jenkins nos pedirá un usuario y contraseña, deberemos de poner la cuenta con la que nos registramos a la hora de instalar Jenkins.

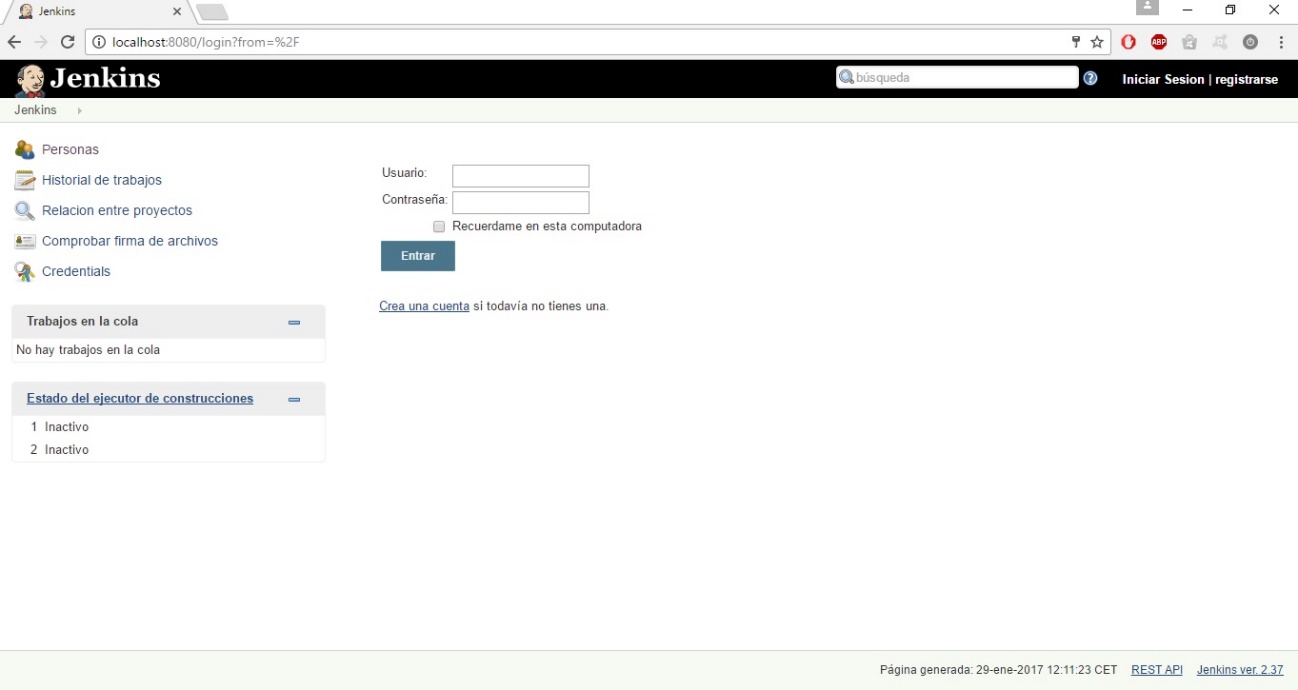


Ilustración Login de jenkins

Ilustración Login de Jenkins

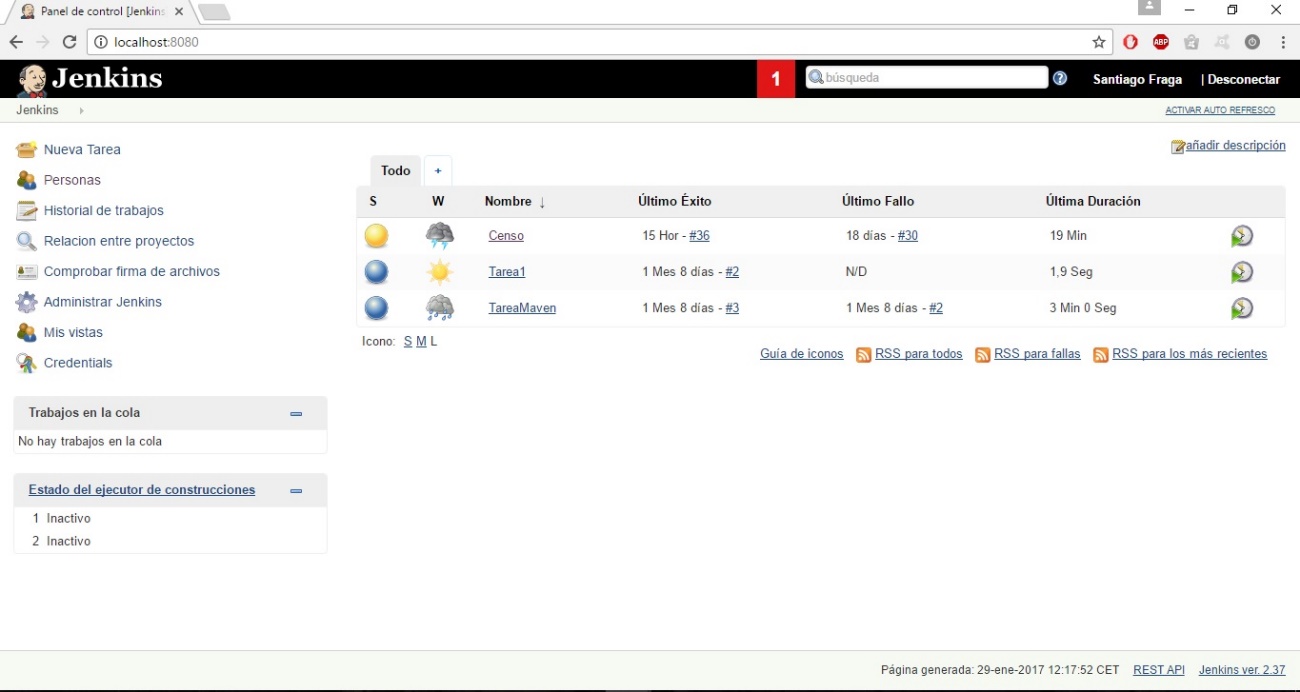
Una vez logueados en nuestro Jenkins, nos saldrán todos los proyectos que tenemos y una pequeña información sobre si se ejecutaron bien o si por el contrario hubo algún tipo de problema.

Ilustración Panel principal de jenkins

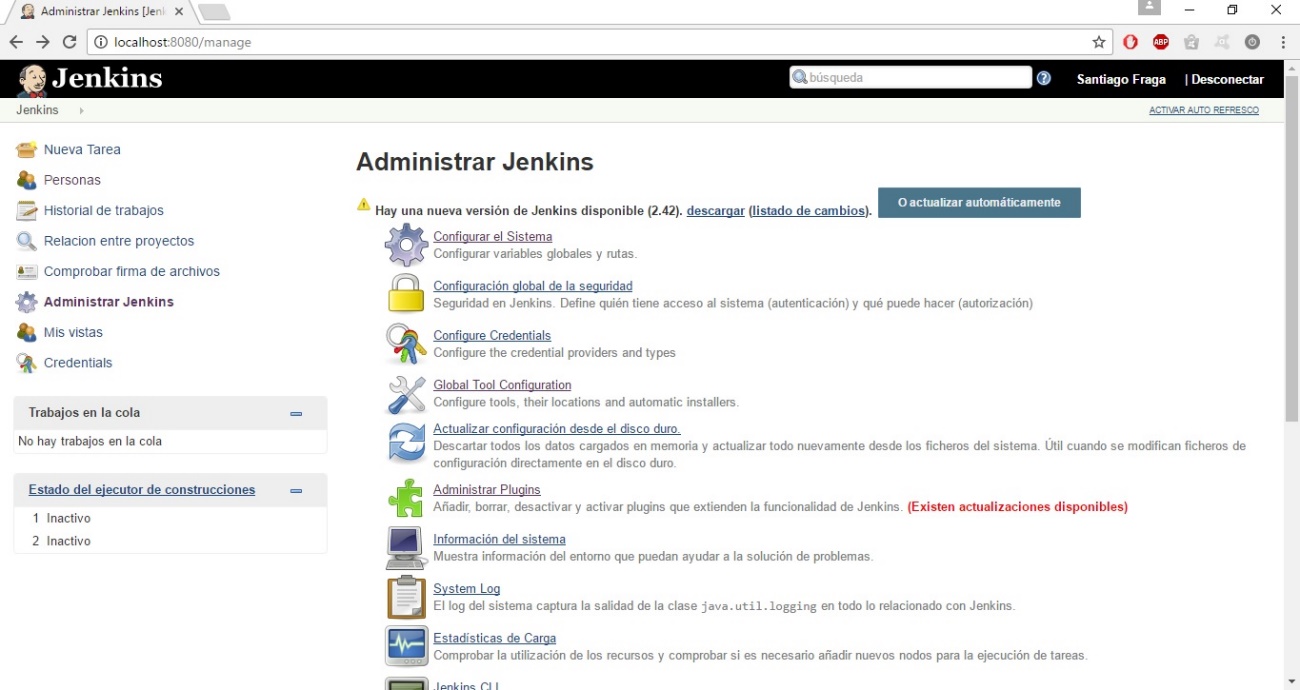
Debido a que nuestro proyecto usa Maven deberemos de introducir el plugin para Maven en nuestro Jenkins. Para ello iremos a “Administrar Jenkins” situado en la parte izquierda de la pantalla.

Ilustración Administracion de Jenkins

Una vez en la siguiente ventana nos iremos a la sección de “Administrar plugins”.

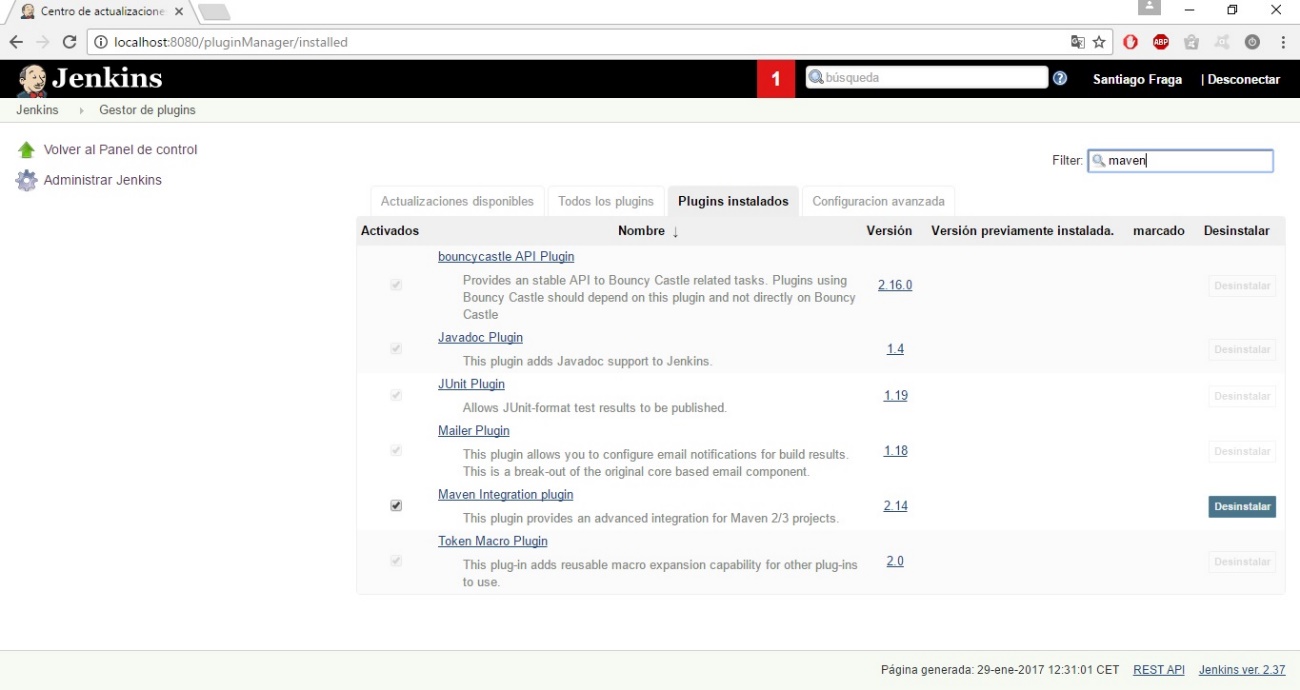
En esta ventana escribiremos en el buscador que nos proporciona Jenkins “maven” y seleccionaremos el plugin llamado “Maven Integration plugin”. Una vez seleccionado Jenkins lo descargará y se reiniciará para hacer efectivos los cambios.

Ilustración Instalar plugin Jenkins

Una vez completado dicho paso volveremos a la ventana de “Administrar Jenkins” pero esta vez accederemos a la opción de “Global Tool Configuration” para configurar mínimamente el plugin de Maven.

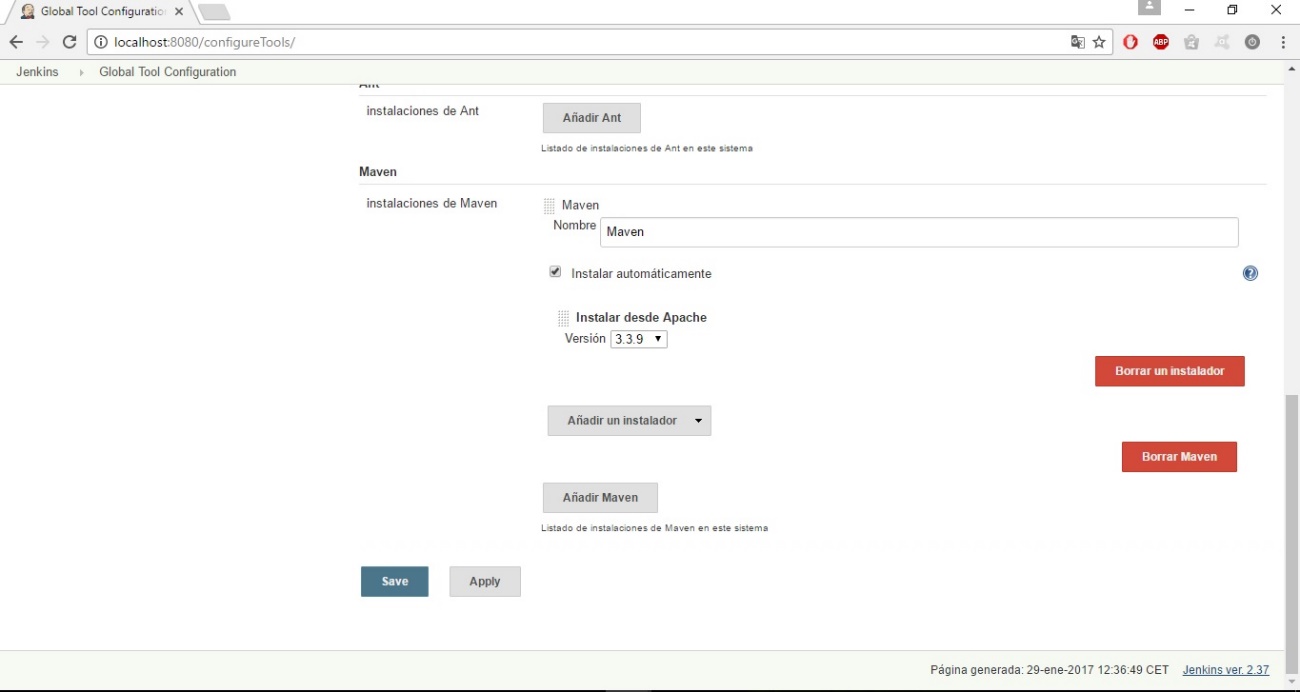
Tras acceder a dicha ventana nos iremos a la sección de “Maven” la cuál configuraremos tal y como se muestra en la figura siguiente:

Ilustración Configurar plugin de maven Jenkins

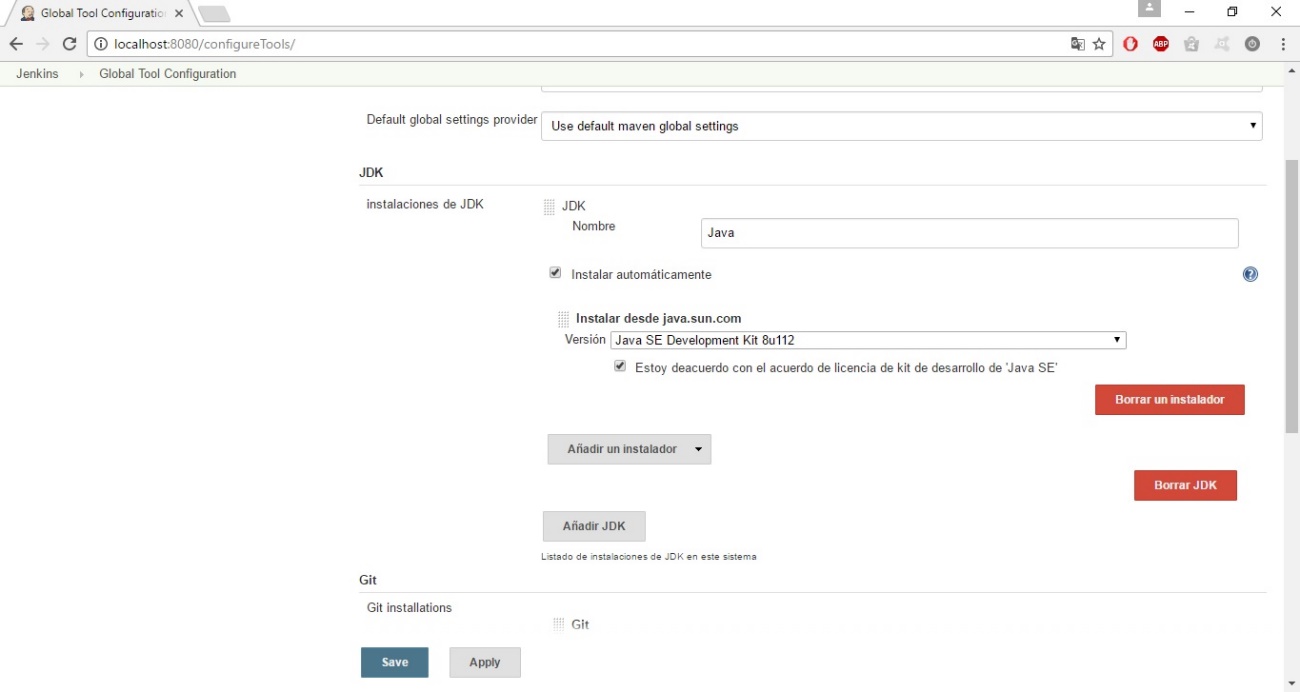
Una vez realizado este paso ya tendremos nuestro maven configurado para Jenkins. Ahora, en la misma ventana debemos de configurar el JDK ya que nuestro proyecto está realizado casi en su totalidad con Java. Para configurar dicho JDK nos iremos a la sección “JDK” y la configuraremos tal y como se muestra en la imagen:

Ilustración Configurar jdk en jenkins

Tras realizar este último paso ya tendremos nuestro Jenkins listo para ejecutar nuestro proyecto.

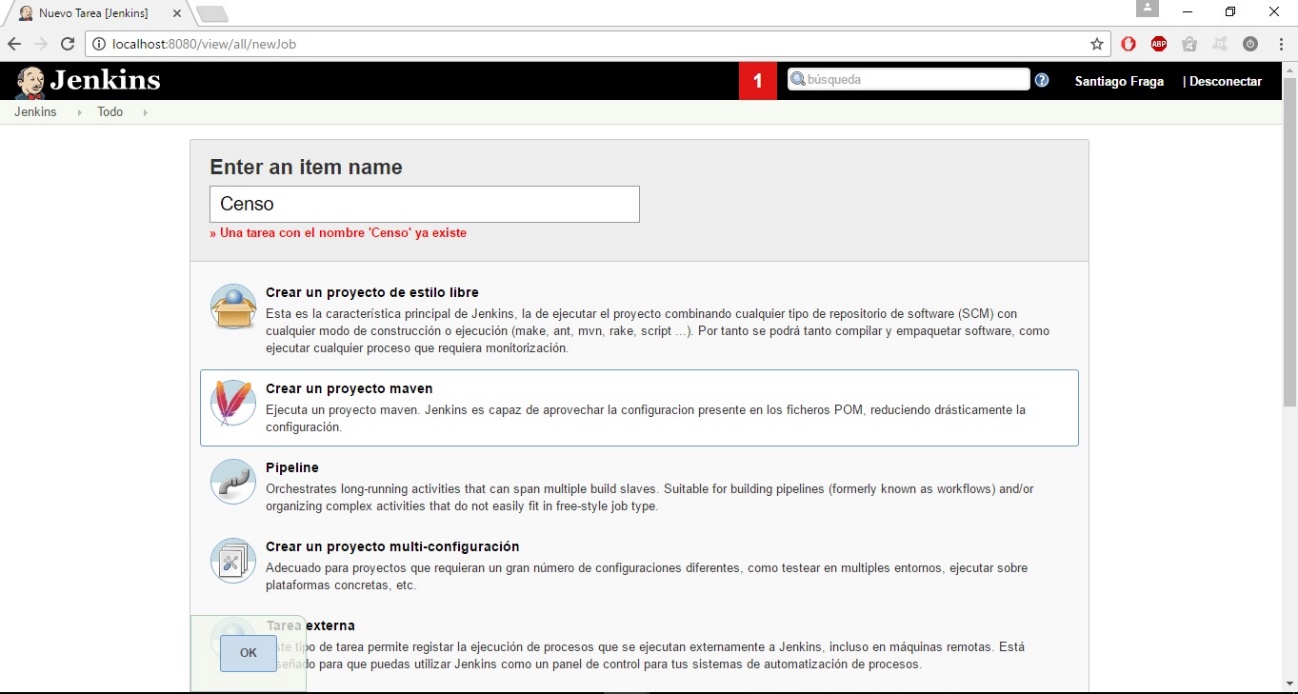
Ahora regresaremos a la ventana principal y le daremos a “Nueva tarea” situada en la parte izquierda de la pantalla.

Ilustración Crear tarea en Jenkins

Debido a que en nuestro caso se trata de un proyecto maven seleccionaremos la opción de “Crear un proyecto maven”. Posteriormente, le daremos un nombre y pulsaremos en ok.

Una vez realizada la siguiente acción nos aparecerá la siguiente pantalla para configurar nuestra tarea:

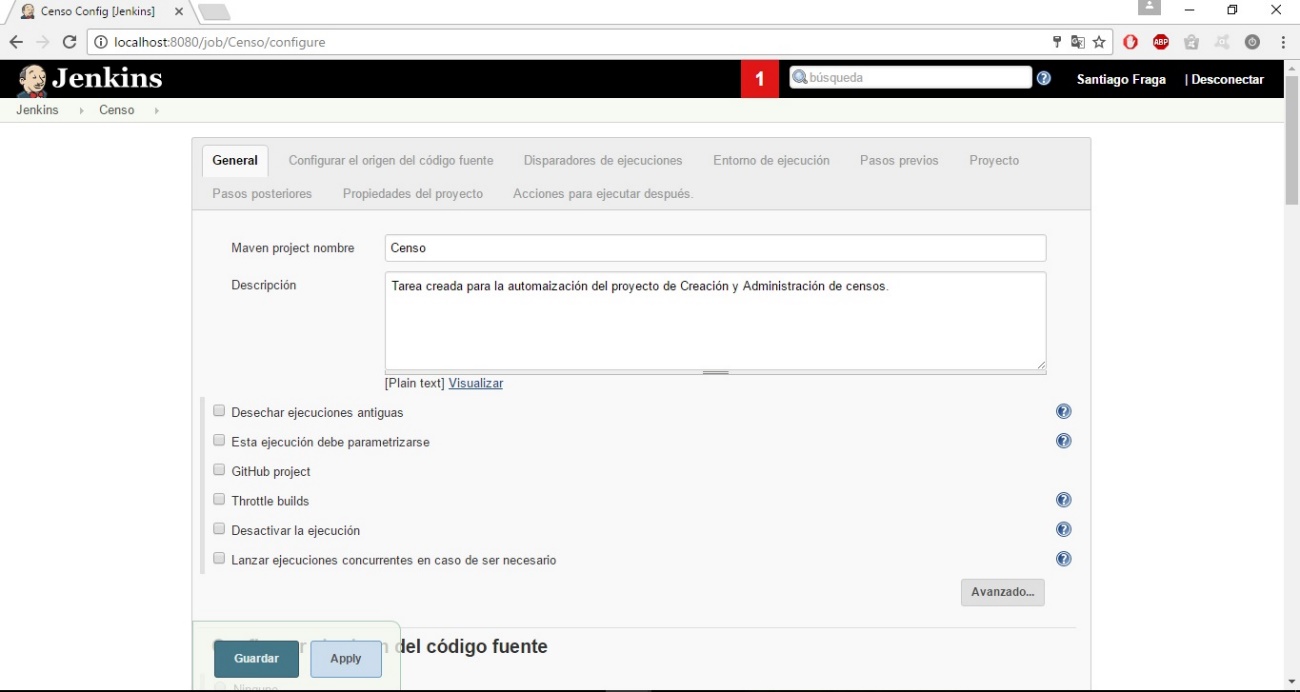


Ilustración Configurar tarea en jenkins 1

En la sección “General” solo deberemos de introducir una pequeña descripción de la tarea que hemos creado.

La siguiente sección a configurar es “Configurar el origen del código fuente”, en dicha sección le diremos a Jenkins que nos coja el código de nuestro repositorio de Github y para ser más concretos de nuestra rama master.

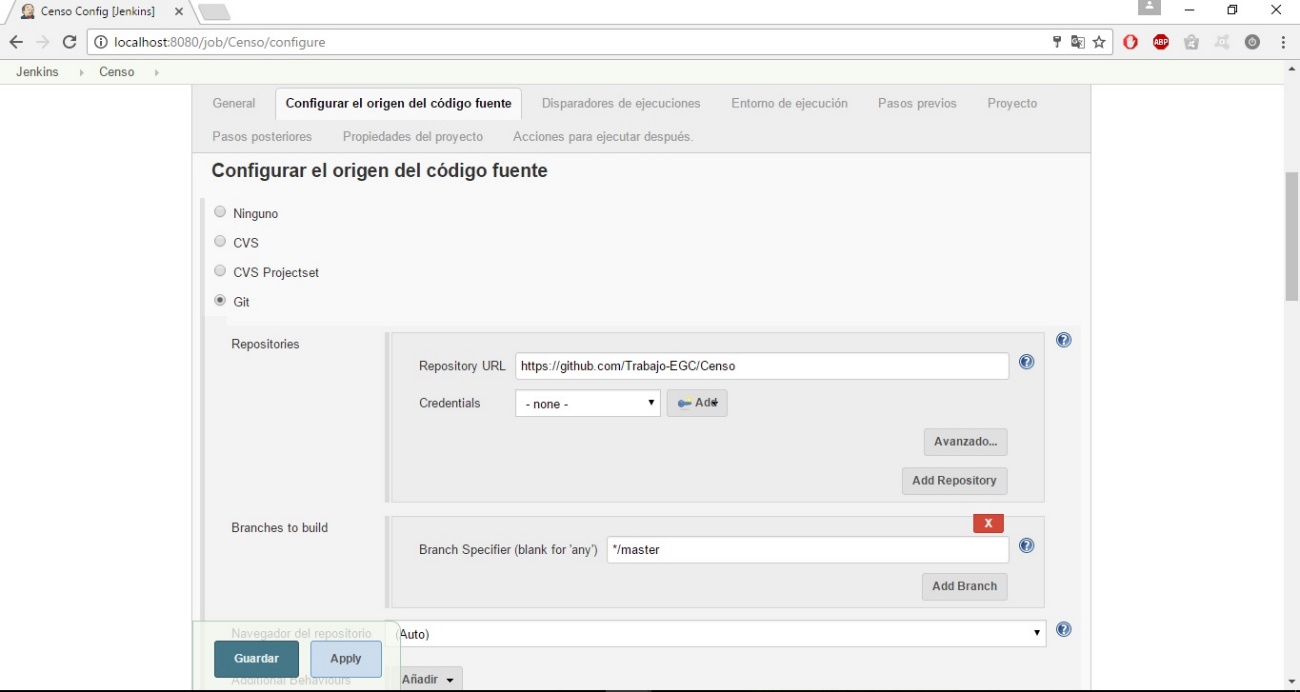
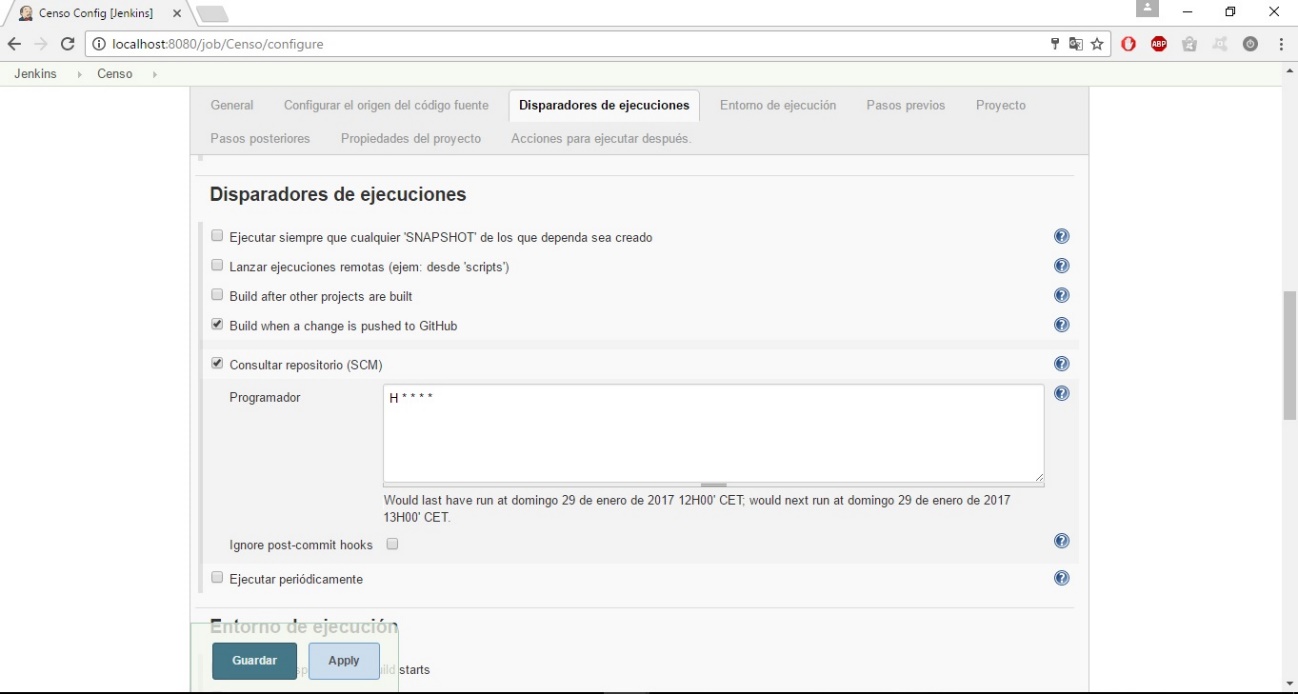
El siguiente apartado que debemos de configurar es el de “Disparadores de ejecuciones” en el cuál le diremos a Jenkins que se debe de ejecutar cada vez que se haga un “push” a nuestro repositorio o cada x tiempo.

Ilustración Configurar disparadores en Jenkins

Ilustración Configurar origen del codigo en Jenkins

Después de configurar esta sección toca irnos a la sección de “Pasos previos” la cuál configuraremos tal y como se nos muestra en la siguiente imagen:

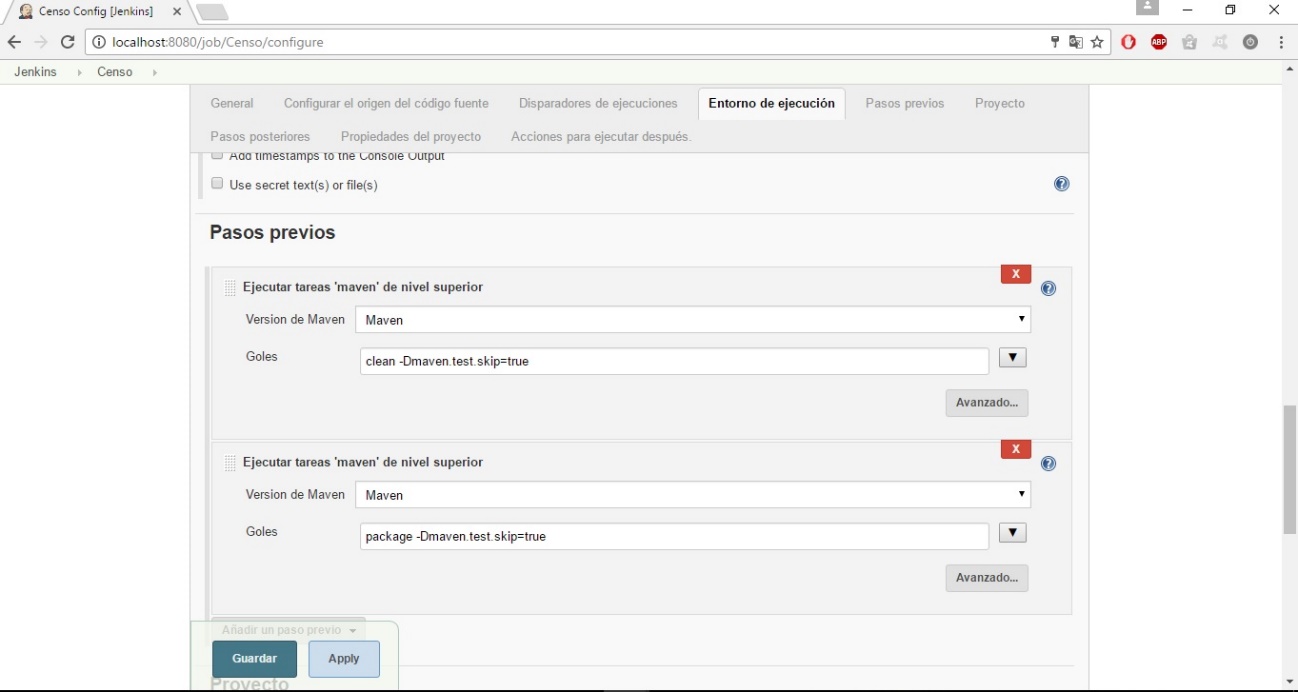
Tras este paso lo único que nos quedaría por configurar sería la sección de “Proyecto” ya que debemos de decirle donde se encuentra nuestro archivo “pom.xml” que es el que usa Maven para configurar todas las dependencias y demás. Esto lo hacemos de la siguiente forma:

Ilustración Configurar pasos previos en Jenkins

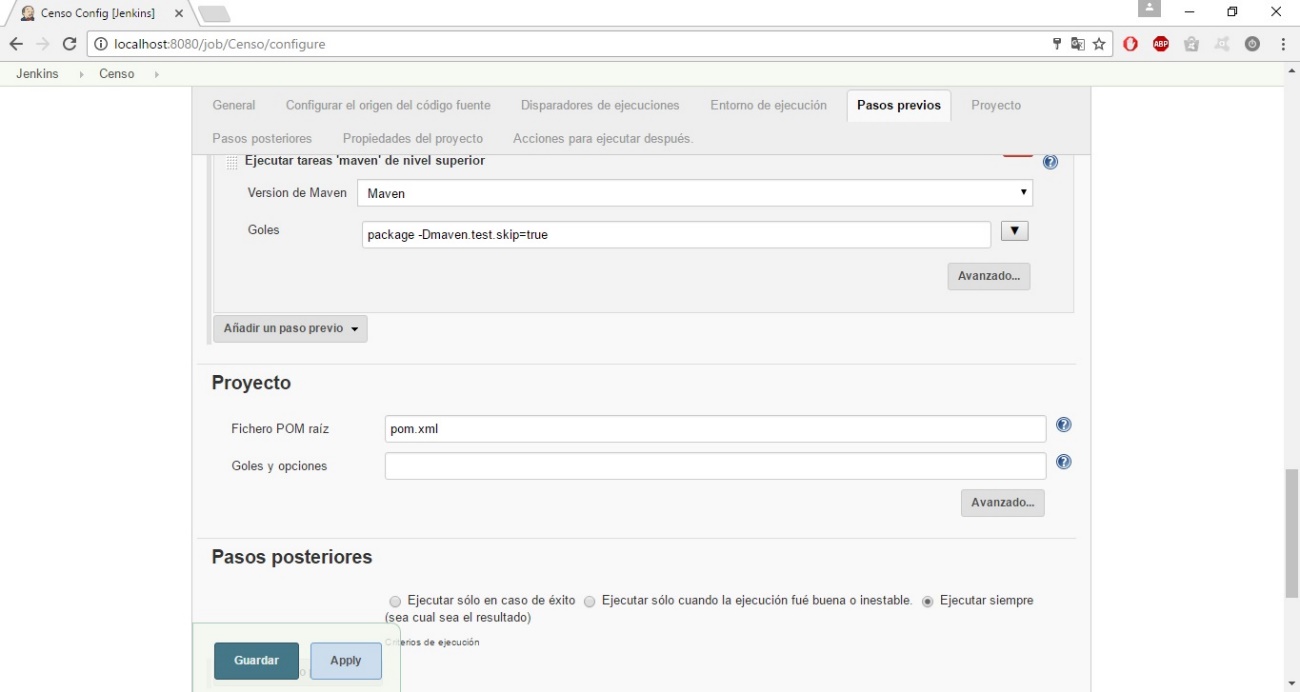


Ilustración Configurar proyecto en Jenkins

Después de realizar este paso ya tendremos nuestro Jenkins configurado para la ejecución de nuestra aplicación.

Podemos ver con más detalle nuestra tarea si pinchamos en ella desde la ventana principal de Jenkins.

Ilustración Ver tarea en jenkins

En dicha ventana podemos observar las distintas ejecuciones de nuestra tarea así como si ha dado fallo. Además, podremos forzar una ejecución dándole a “Construir ahora”.

## Integración con otros sistemas

Se ha mirado con que otros proyectos se podía realizar la integración y se encontró que podíamos hacerlo con cabina de votación y creación y administración de votaciones.

Detallaré como ha ido la integración con ambas:

**Cabina**

Lo primero que se hizo fue hacer un clone del proyecto en Git en la carpeta del workspace donde trabajo para así poder investigar sobre que podíamos integrar con nuestro proyecto.

Después de inspeccionar dicho código no encontramos nada para poder realizar la integración así que se pasó directamente a buscar maneras de integrar la otra opción.

**Votación**

Al igual que con el proyecto de Git, se realizó un clone del proyecto para poder trabajar con él y poder investigar el código.

En este caso, al ejecutar los dos proyectos en paralelo se puede acceder de uno a otro, viendo que están conectados:

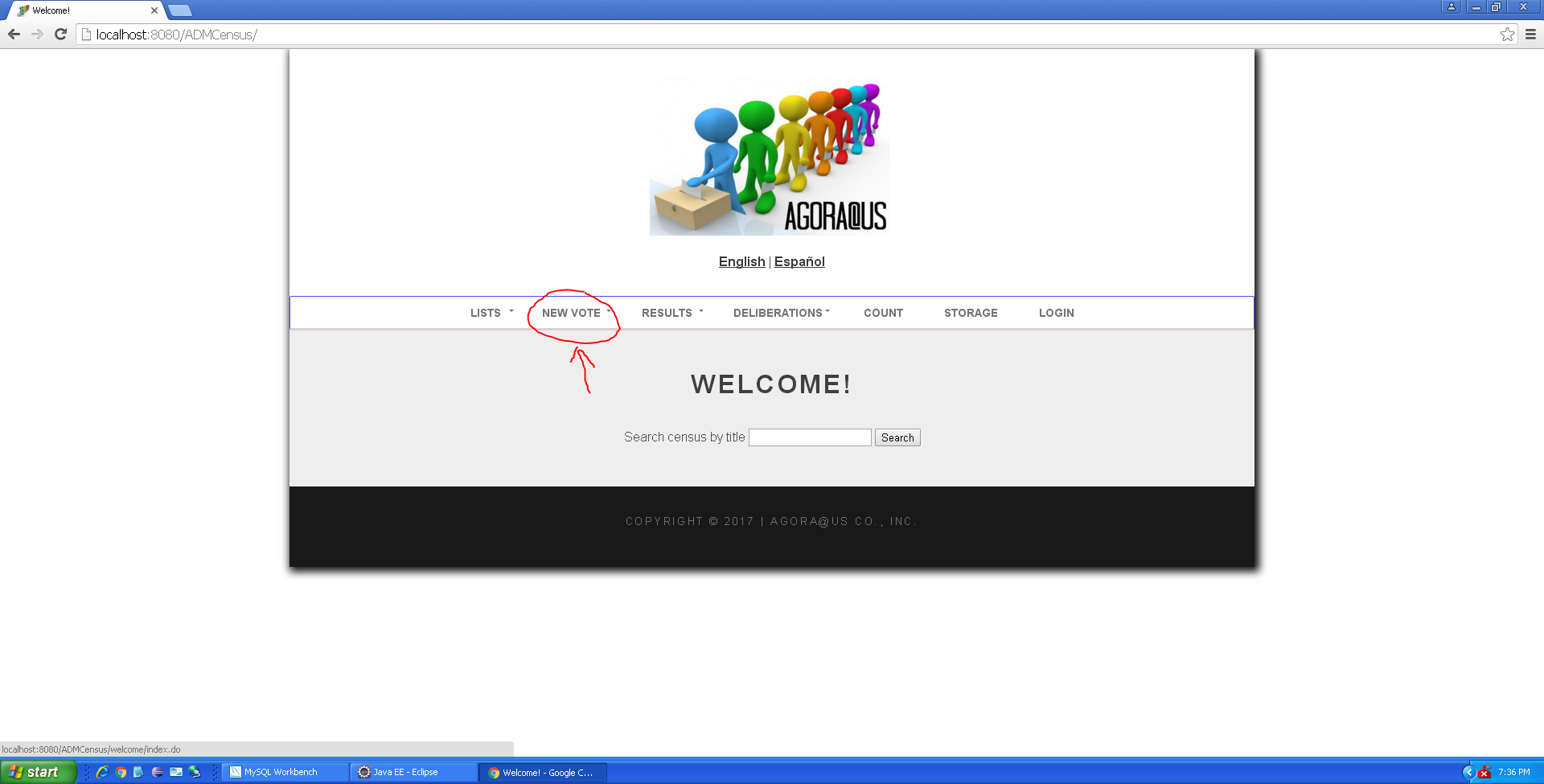


Ilustración Demostración conexión 1

Si le damos a New Vote, nos lleva al sistema de votación, y si desde ahí le damos a volver al menú principal nos llevaría de vuelta a nuestro proyecto:

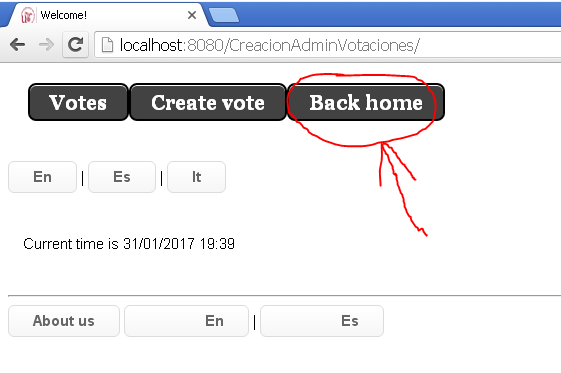


Ilustración Demostración conexión 2

Aunque al igual que el otro proyecto no podemos integrar servicios así que vamos a realizar una simulación como si recibiéramos los datos desde votación y los usaremos en los servicios que hagan uso de estos datos.

Por ejemplo, vemos que para crear un censo necesitamos idVotación, tipoVotación, etc

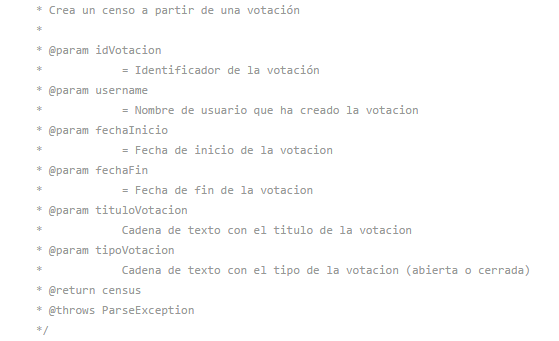


Ilustración Antiguo método de crear censo

Esos datos los recibiremos desde un fichero, el cual vamos a leer y extraer la información para poder así trabajar con ella.

Se modifican el servicio visto antes, el test para que coincida con el nuevo servicio y por último el controlador.

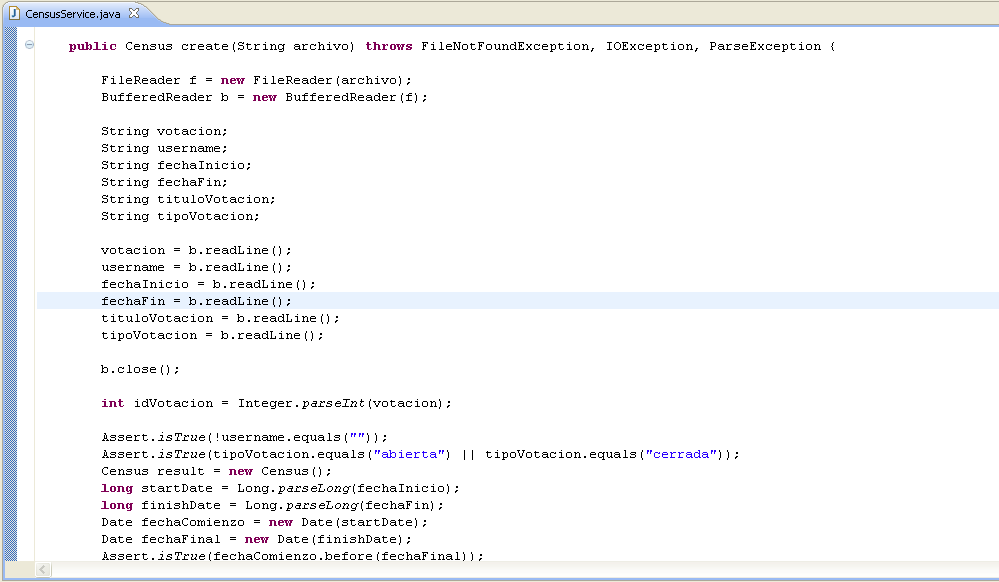


Ilustración Modificación de servicio

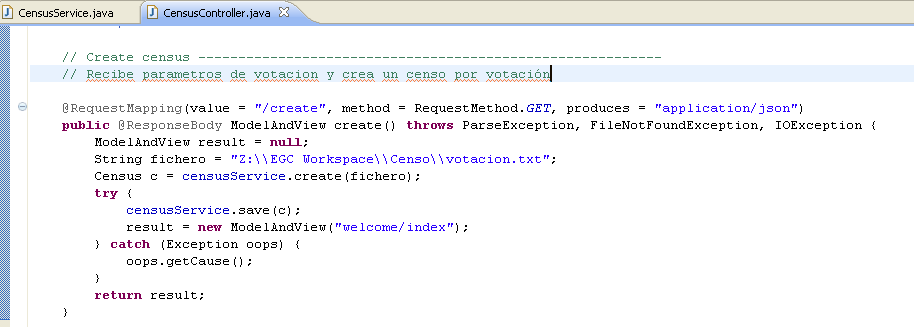


Ilustración Modificación controlador

Una vez tengamos todo esto modificado, modificamos la vista principal para poder hacer uso del servicio:



Ilustración Adición de línea para crear un censo

Ahora solo queda ejecutar el servidor y probar que el servicio, leyendo datos de un fichero funciona:

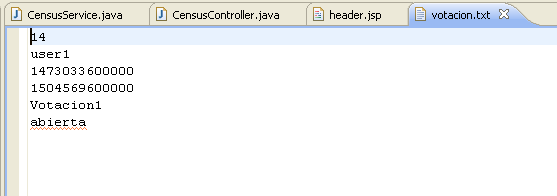


Ilustración Datos del fichero a usar

Seleccionamos el método crear censo desde archivo:

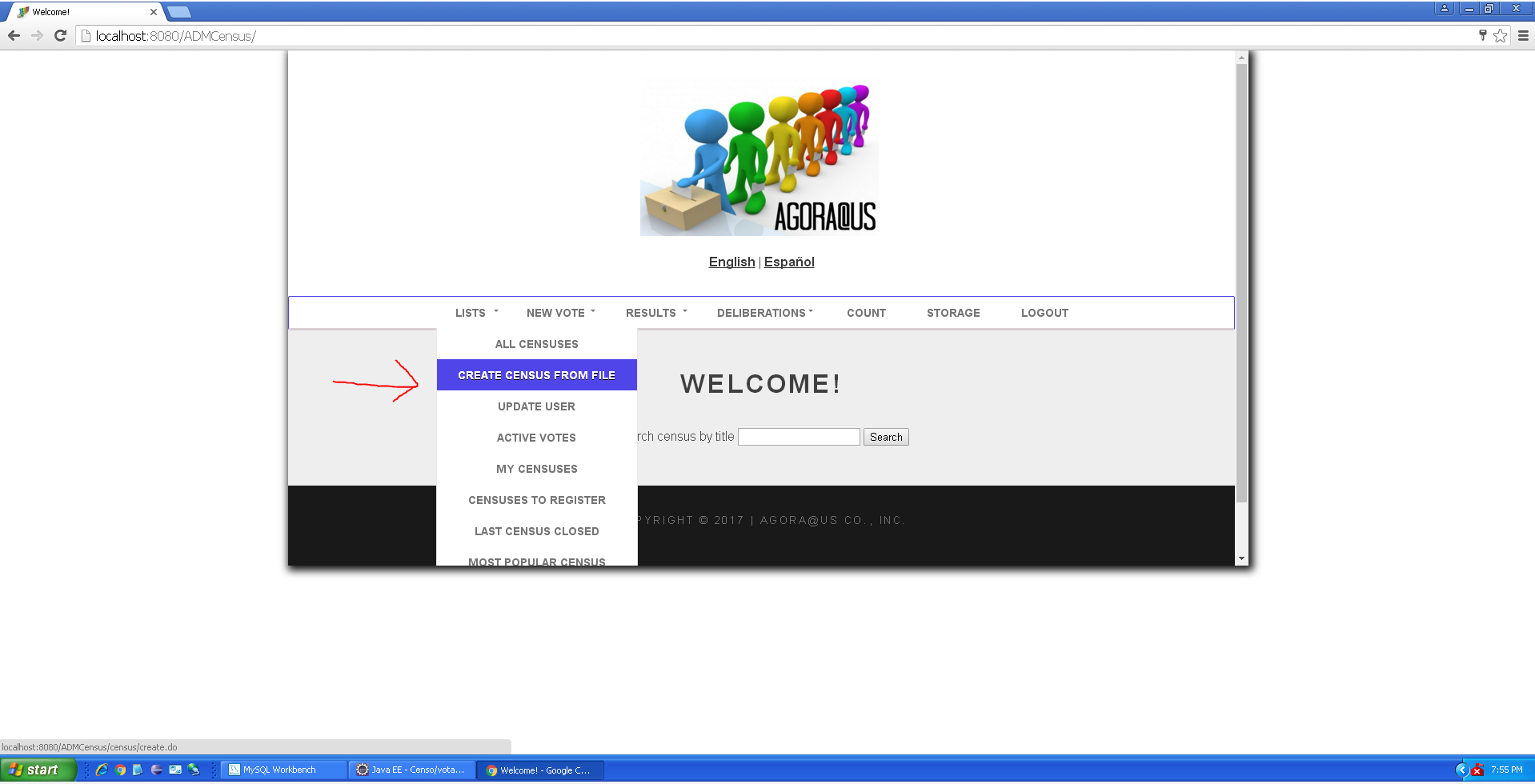


Ilustración Selección del método

Vamos al listado de censos y vemos que tenemos nuestro nuevo censo creado con datos recibidos de un fichero:

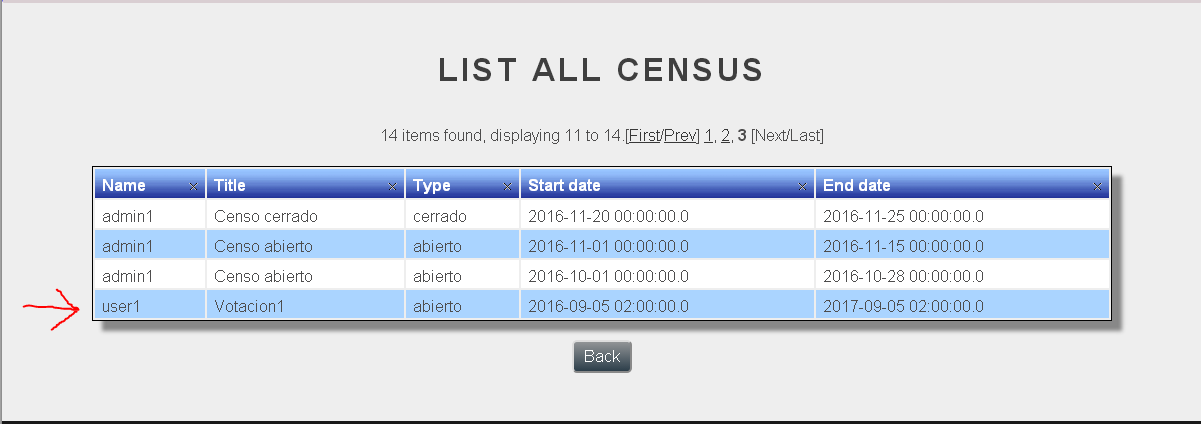


Ilustración Demostración del método

Indicar que todo esto es una simulación de integración para poder demostrar cómo funcionaría nuestro proyecto con la integración adecuada si los servicios de otros proyectos funcionaran correctamente.

## El .war, el despliegue y la construcción se genera de forma manual

Para realizar el despliegue de nuestro subsitema primero hemos buscado varias páginas que ofrecen servicios de computación en la Nube que soportan distintos lenguajes de programación.

El primero que hemos probado ha sido Heroku. Permite realizar el despliegue en pocos pasos. Primero debemos acceder a <https://www.heroku.com/> donde nos debemos registrar y entonces podemos encontrar una guía de como desplegar nuestra aplicación y también podemos descargar el instalador del cliente. Una vez instalado el cliente nos vamos a acceder introduciendo el comando heroku login en la consola.

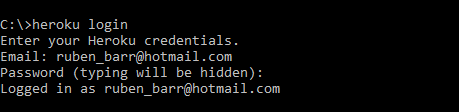


Ilustración Heroku login

A continuación, debemos crear una aplicación, para ello vamos a ir a la ruta donde se encuentra nuestro proyecto y vamos a introducir el comando heroku create “Nombre app”.



Ilustración Heroku create

Para poder subir nuestro war debemos instalar un plugin de heroku, para ello vamos a introducir el comando heroku plugins:install heroku-cli-deploy.

C:\Users\Simon\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Captura3.png

Ilustración Heroku instalación plugin

Una vez instalado el plugin vamos a introducir el comando heroku war:deploy “ruta del war” --app “nombre app”.



Ilustración Heroku deploy

A continuación, podemos acceder a heroku y veremos el panel de control de nuestra aplicación.

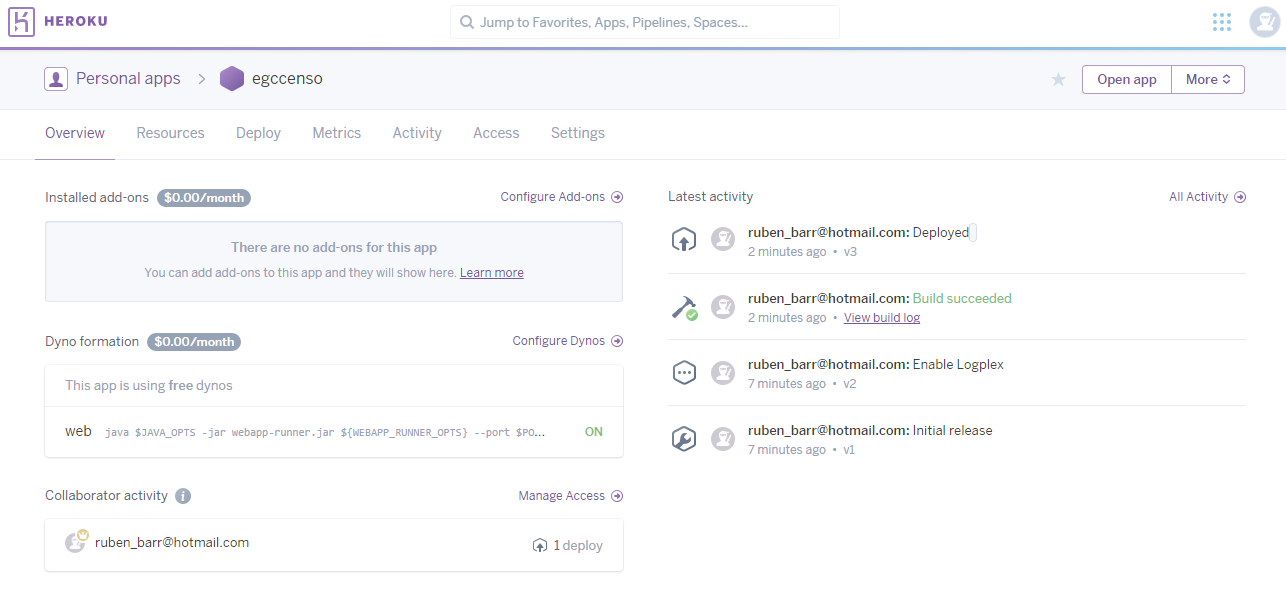


Ilustración Heroku panel de control

El problema que nos ha presentado heroku es que no soporta mysql.

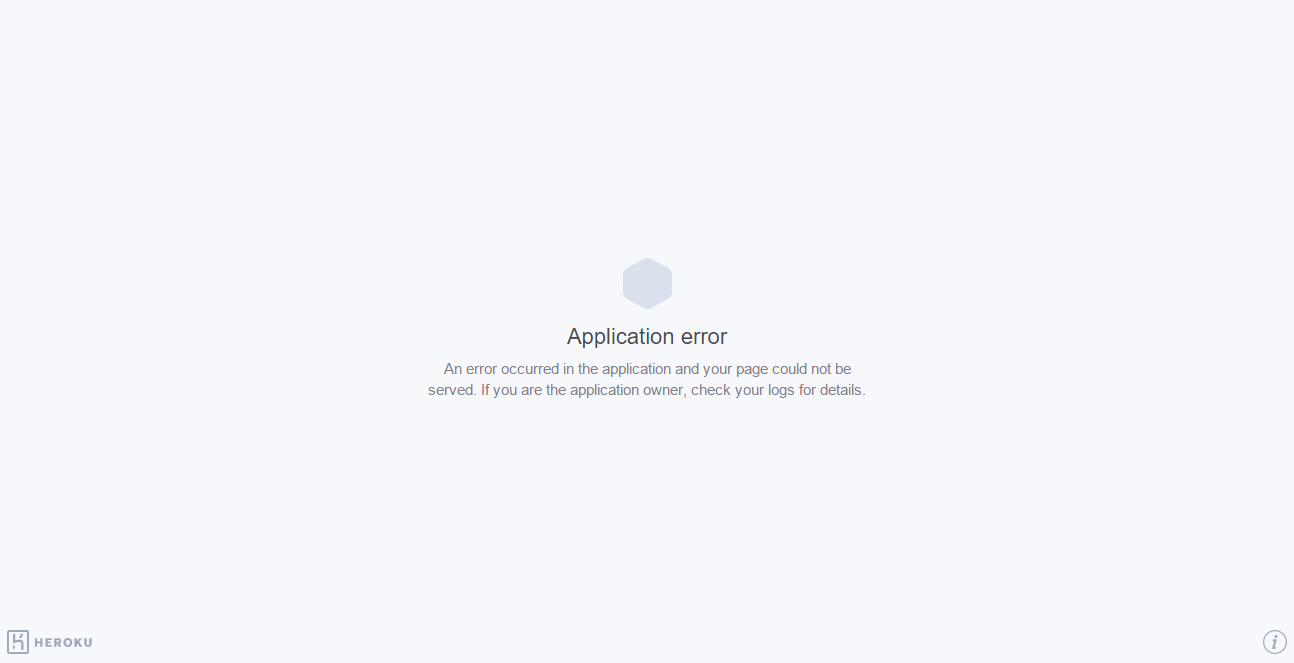


Ilustración Heroku application error

Heroku ofrece varios plugins para migrar de mysql a postgresql pero entonces pide que se introduzca una tarjeta de crédito para poder usarlo de forma gratuita, ningún miembro del grupo ha querido poner su tarjeta de crédito por lo que hemos optado por otras opciones.

La segunda aplicación en la que hemos intentado desplegar ha sido en Google App Engine, el problema es que para su uso también pide que se introduzca una tarjeta de crédito para asi asegurarse que no somos ningún bot. En el caso de no introducirla solo nos permite hacer un tutorial de como se despliega una aplicación de ejemplo.

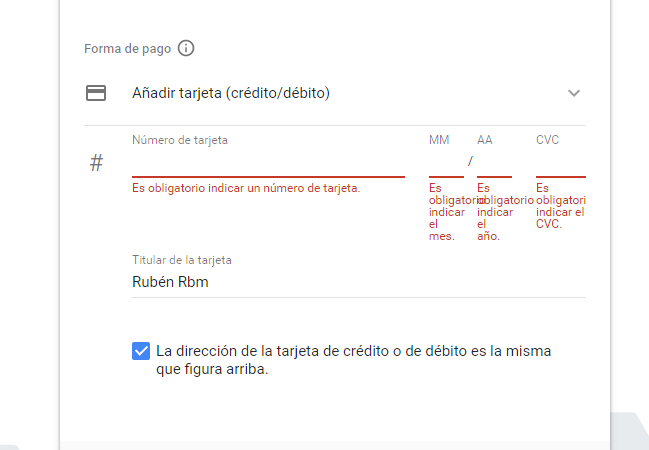


Ilustración App Engine

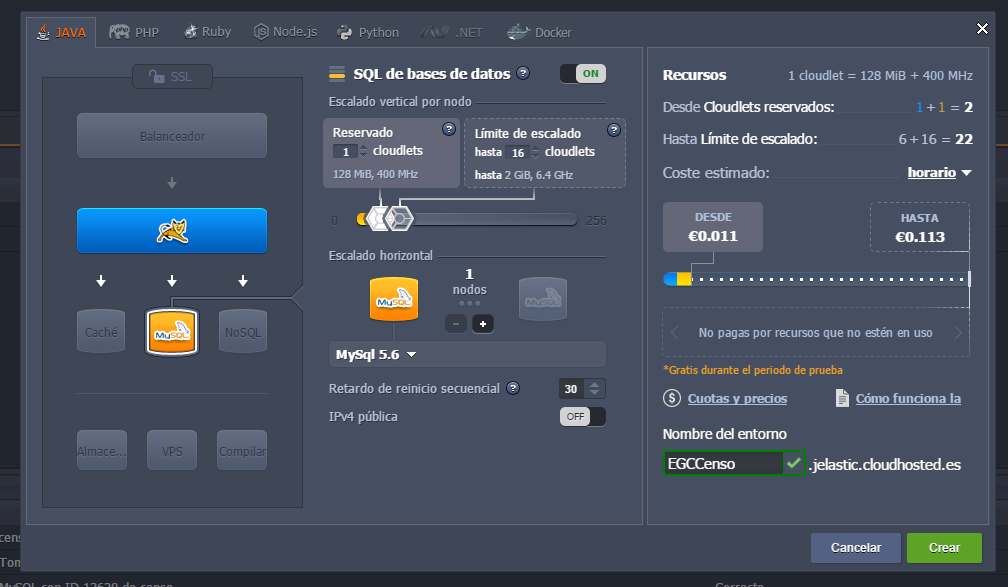
La tercera opción ha sido Jelastical, esta página nos ofrece un periodo de prueba. Esta página nos ofrece varias opciones a la hora de crear el entorno donde se va a realizar el despliegue. 

Ilustración Jelastical creación del entorno

Una vez hemos configurado el entorno debemos esperar un tiempo ya que la creación tarda varios minutos, entonces podremos ver en el panel de control nuestro entorno.

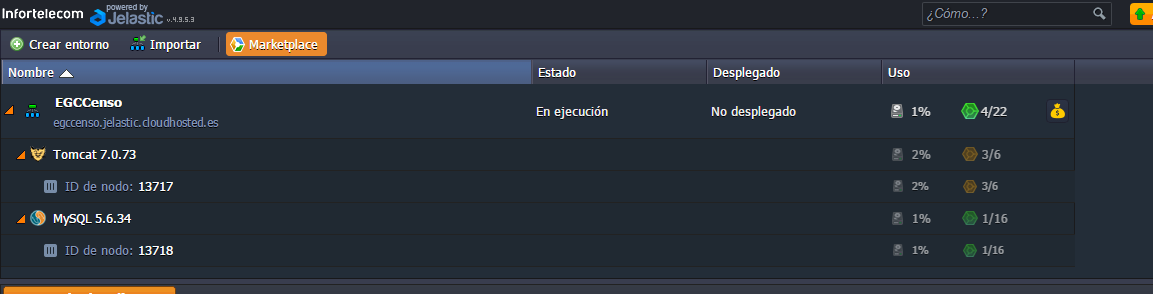


Ilustración Jelastical entorno creado

Podemos gestionar la base de datos mediante phpMyAdmin y así crear nuesta base de datos mediante un script que hemos generado.

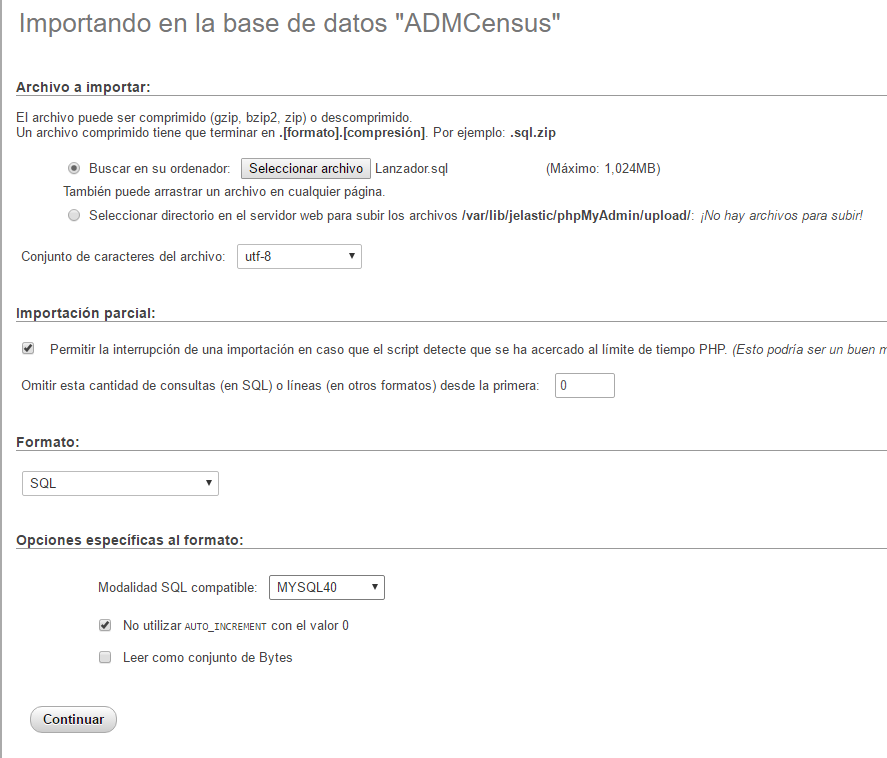


Ilustración Jelastical creación de la base de datos

Para desplegar el war en el entorno tenemos que introducir la ruta donde lo tenemos generado y el nombre que queremos que tenga.

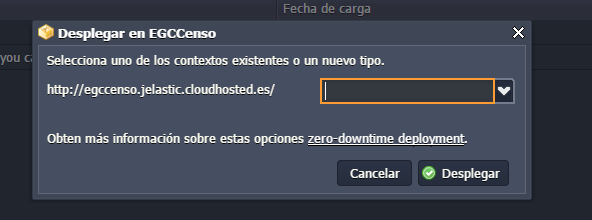


Ilustración Jelastical despliegue

A la hora de realizar la conexión entre la base de datos y el servidor de Tomcat nos ha dado problemas, por lo que finalmente hemos optado por configurar una maquina virtual como servidor e instalar todas las herramientas necesarias para poder desplegar de forma automática nuestro subsistema.

Hemos preparado una maquina virtual donde hemos instalado MySql Server, Apache Tomcat, java, maven, git y Jenkins.

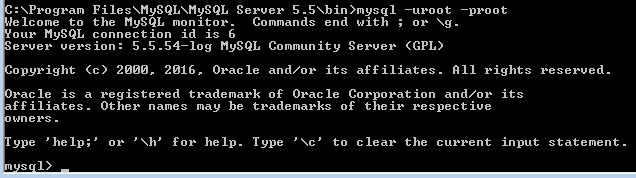


Ilustración Versión de MYSQL Server



Ilustración Versión de maven

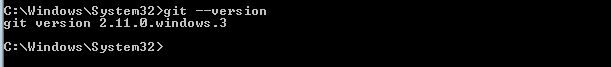


Ilustración Versión de git

Para crear nuestra base de datos vamos a introducir el comando mysql -uroot -proot < create-ADMCensus.sql, este script contiene todo lo necesario para generar las tablas y usuario de la base de datos con datos de ejemplo para comprobar su funcionalidad.

C:\Users\Simon\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Captura3.4.png

Ilustración Creación base de datos

Para desplegar de forma manual nuestro war debemos acceder a localhost/manager introducir el usuario y la pass que son root. A continuación, vamos a introducir la ruta del war como podemos ver en la siguiente imagen.

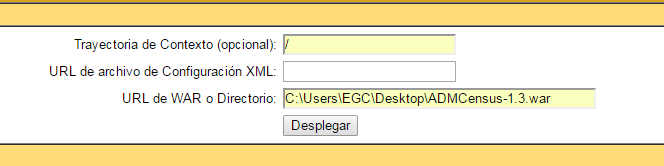


Ilustración Despliegue war

Para poder acceder desde nuestro pc cliente al servidor que hemos configurado debemos modificar como administrador el archivo hosts de nuestra maquina. Para ello vamos a la ruta C:\Windows\System32\drivers\etc\

C:\Users\Simon\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Captura3.7.png

Ilustración Archivo host

En este archivo debemos introducir la siguiente línea:

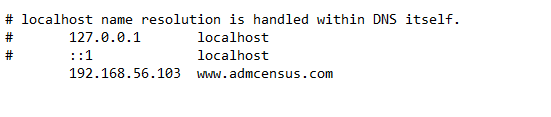


Ilustración Configuración archivo hosts

Una vez hechos estos cambios tenemos que vaciar los DNS para ello vamos a introducir el comando ipconfig /flushdns.

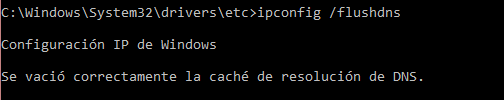


Ilustración Vaciado de DNS

Al acceder a www.admcensus.com podremos ver que se carga nuestro subsitema.



Ilustración ADMCensus

Para automatizar el despliegue hemos configurado en el servidor Jenkins con los siguientes plugins:

C:\Users\Simon\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Captura3.11.png

Ilustración Copy Artifact plugin

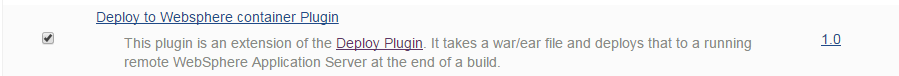


Ilustración Deploy to Websphere container plugin

Hemos creado dos tareas. La tarea CensoEGC se va a encargar de revisar el repositorio de forma automática cada hora y en el caso de que exista cambios, realizar los test y generar el war. En caso de que falle mandará un correo a todos los miembros del grupo.

Una vez finalice esta tarea, si no ha tenido ningún problema, se a llama de forma automática a la tarea censusWar, que se va a encargar de copiar el war generado y desplegarlo en tomcat.



Ilustración Jobs

## Mejora de la documentación

La documentación se ha vuelto a hacer prácticamente desde cero ya que faltaban demasiadas cosas.

La nueva documentación esta accesible desde la wiki del grupo en la siguiente url

<https://1984.lsi.us.es/wiki-egc/index.php/CreacionAdministracionCensos>