



---

# AGORA@US

---

Grupo 9: Creación y Administración de Censos



Grupo 2 (tarde)

Id Opera: 9

Pablo Romero Vázquez

Santiago Fraga Martín-Arroyo

Nicolás Lorenz Rosado

Simón Egea Guerrero

Rubén Barrientos Mohedano

## Enlaces de interés

Repositorio de Github del grupo “Creación y Administración de Censos”:

<https://github.com/Trabajo-EGC/Censo>

Repositorio de Github de incidencias común:

<https://github.com/Trabajo-EGC/Censo/issues>

Wiki de la asignatura:

[https://1984.lsi.us.es/wiki-egc/index.php/P%C3%A1gina\\_Principal](https://1984.lsi.us.es/wiki-egc/index.php/P%C3%A1gina_Principal)

Espacio del grupo “Creación y Administración de Censos” en la wiki:

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

Espacio del grupo “Creación y Administración de Censos” en Opera:

<http://opera.eii.us.es/egc/public/trabajo/ver/id/52>

Servidor ftp donde se puede descargar la máquina virtual utilizada (DP):

<ftp://postgrado.lsi.us.es/DT/>

## Integrantes

Santiago Fraga Martín-Arroyo Ingeniero Software	Pablo Romero Vázquez Ingeniero Software	Simón Egea Guerrero Coordinador
		
Nicolás Lorenz Rosado Ingeniero Software	Rubén Barrientos Mohedano Ingeniero Software	
		

## Miembros que participan en la mejora (Milestone 5)

Todos los integrantes del grupo participan en la entrega del milestone 5, dichos integrantes vienen recogidos a continuación:

- 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	Pablo Romero Vázquez	Esquema de la documentación y realización de la misma.	08/01/2017
1.1	Santiago Fraga Martín-Arroyo	Realización y corrección de la documentación.	10/01/2017
1.1	Santiago Fraga Martín-Arroyo	Revisión	10/01/2017
1.2	Santiago Fraga Martín-Arroyo	Reestructuración de la documentación y realización de la mejora	25/01/2017
1.2.1	Santiago Fraga Martín-Arroyo	Realización del apartado de cambios	26/01/2017
1.2.2	Santiago Fraga Martín-Arroyo	Realización del apartado planificación y tareas	26/01/2017
1.2.2.1	Santiago Fraga Martín-Arroyo	Corrección del apartado planificación y tareas	27/01/2017
1.2.3	Santiago Fraga Martín-Arroyo	Realización del apartado elementos de control	27/01/2017
1.2.4	Santiago Fraga Martín-Arroyo	Realización del apartado entorno de desarrollo	27/01/2017
1.2.5	Santiago Fraga Martín-Arroyo	Desarrollo del apartado gestión del código	27/01/2017
1.2.6	Pablo Romero Vazquez	Desarrollo del apartado gestión de ramas y roles en la gestión de código	28/01/2017
1.2.7	Santiago Fraga Martín-Arroyo	Desarrollo de los apartados políticas de nombres y estilos, aceptación de cambios y gestión de buenas prácticas en el código	28/01/2017
1.2.8	Pablo Romero Vázquez	Desarrollo de las lecciones aprendidas (gestión de las ramas) y del ejercicio de ramas.	28/01/2017

1.2.9	Santiago Fraga Martín-Arroyo	Desarrollo del apartado gestión de la construcción e integración continua	29/01/2017
1.2.10	Pablo Romero Vázquez	Desarrollo del apartado de gestión del cambio, incidencias y depuración.	29/01/2017
1.2.11	Santiago Fraga Martín-Arroyo	Correcciones menores en el documento y creación de los apartados mapa de herramientas, conclusión y trabajo futuro y referencias.	29/01/2017
1.2.12	Pablo Romero Vázquez	Eliminación de varios apartados del documento e inclusión de imágenes faltantes en los últimos apartados.	30/01/2017
1.2.12	Simón Egea Guerrero Rubén Barrientos Mohedano	Documentación del despliegue y de hostings probados.	31/01/2017
1.3	Santiago Fraga Martín-Arroyo	Revisión de la documentación.	01/02/2017

# Índice

Enlaces de interés .....	2
Integrantes .....	3
Miembros que participan en la mejora (Milestone 5) .....	4
Historial de versiones.....	5
Resumen.....	12
Introducción .....	13
Descripción del sistema.....	16
Componentes y relación con otros subsistemas .....	16
Cambios realizados al proyecto .....	19
Nuevas funcionalidades.....	19
<b>Mostrar en una tabla las últimas votaciones que se han realizado en el subsistema.</b> .....	19
<b>Obtención de los censos más participativos</b> .....	21
<b>Porcentaje de usuarios que no votan por censo</b> .....	23
<b>Mejora del css</b> .....	24
<b>Buscador de censos.</b> .....	24
Introducción dependencias maven.....	25
<b>Introducción de nuevas dependencias maven</b> .....	25
Introducción de nuevos plugins.....	26
<b>Introducción de nuevos plugins en maven</b> .....	26
Creación documento de ayuda .....	27
<b>Documento de ayuda</b> .....	27
Sistema de autenticación .....	27
<b>Sistema de autenticación</b> .....	27
Modificación base de datos .....	28
<b>Modificación censos almacenados</b> .....	28
Planificación y tareas.....	29
Elementos de control .....	31
Entorno de desarrollo .....	33
Máquina virtual.....	33
Instalación Git .....	37
Instalación Jenkins .....	38
Gestión del código fuente .....	40
Gestión de ramas .....	40
<b>Gestión interna</b> .....	40
<b>Creación de ramas</b> .....	40

Merge de una rama .....	40
Conflictos .....	41
Gestión externa.....	42
Roles en la gestión del código .....	42
Políticas de nombre y estilos .....	43
Aceptación de cambios .....	43
Gestión de buenas prácticas en el código .....	43
PMD.....	43
Instalación PMD .....	44
Ejercicio propuesto para el uso de PMD .....	45
Lecciones aprendidas .....	47
Gestión de la calidad del código fuente .....	47
Gestión de ramas internas.....	47
Ejercicio: Creación de una nueva rama y resolución de un conflicto. ....	47
Gestión de la construcción e integración continua .....	49
Gestión de la construcción .....	49
Ejercicio gestión de la construcción con maven .....	50
Lecciones aprendidas .....	51
Gestión de la integración continua .....	52
Ejercicio de integración continua con Jenkins .....	52
Lecciones aprendidas .....	59
Gestión del cambio, incidencias y depuración .....	60
Control del cambio.....	60
Ejercicio: Implementación adecuada para la gestión del cambio.....	61
Gestión de incidencias .....	64
Reportes de incidencias con GitHub .....	64
Política de gestión de incidencias .....	66
Lecciones aprendidas .....	66
Ejercicio: Resolución de una incidencia .....	66
Lecciones aprendidas .....	67
Ejercicio: Cómo gestionar una incidencia.....	67
Depuración con Eclipse .....	68
Gestión de liberaciones, despliegue y entregas .....	69
Mapa de herramientas.....	78
Conclusiones y trabajo futuro .....	80
Conclusiones.....	80

Trabajo futuro.....	81
Referencias.....	82

# Índice de figuras

Ilustración 1Diagrama de subsistemas.....	14
Ilustración 2Diagrama de subsistemas a integrar.....	18
Ilustración 3UltimasVotacionesRepo.....	20
Ilustración 4UltimasVotacionesController .....	20
Ilustración 5UltimasVotacionesVista .....	20
Ilustración 6UltimasVotacionesResultado .....	21
Ilustración 7CensosMasPopulares .....	22
Ilustración 8CensosMasPopularesTabla.....	22
Ilustración 9AbstencionMetodo .....	23
Ilustración 10AbstencionTabla .....	23
Ilustración 11BuscadorMetodo .....	24
Ilustración 12BuscadorQuery .....	24
Ilustración 13BuscadorBoton .....	24
Ilustración 14BuscadorTabla .....	25
Ilustración 15DependenciaWiremock.....	25
Ilustración 16PluginPMD .....	26
Ilustración 17PluginDoap .....	26
Ilustración 21Maquina virtual login .....	33
Ilustración 22Maquina virtual eclipse workspace .....	34
Ilustración 23Maquina virtual eclipse abierto.....	34
Ilustración 24Maquina virtual eclipse tomcat .....	35
Ilustración 25Maquina virtual eclipse importar .....	35
Ilustración 26Maquina virtual mysql .....	36
Ilustración 27Maquina virtual add tomcat .....	37
Ilustración 31pmdiInstalacion .....	44
Ilustración 32Maven test .....	44
Ilustración 33Bloc de notas pmd .....	45
Ilustración 34Linea de codigo erronea.....	46
Ilustración 35Reporte pmd.....	46
Ilustración 36 Creación de una nueva rama .....	47
Ilustración 37 Commit de la nueva funcionalidad .....	48
Ilustración 41Dependencia maven .....	50
Ilustración 42Archivos descargados por maven .....	51
Ilustración 43Login de jenkins .....	52
Ilustración 44Panel principal de jenkins.....	53
Ilustración 45Administracion de Jenkins.....	53
Ilustración 46Instalar plugin Jenkins .....	54
Ilustración 47Configurar plugin de maven Jenkins .....	54
Ilustración 49Crear tarea en Jenkins.....	55
Ilustración 50Configurar tarea en jenkins 1 .....	56
Ilustración 52Configurar disparadores en Jenkins.....	57
Ilustración 51Configurar origen del codigo en Jenkins .....	57
Ilustración 53Configurar pasos previos en Jenkins .....	58
Ilustración 54Configurar proyecto en Jenkins .....	58
Ilustración 55Ver tarea en jenkins .....	59
Ilustración 56 Mejoras registradas .....	61

Ilustración 57 Mejora #19 .....	62
Ilustración 58 Mejora #11 .....	62
Ilustración 60 Mejora #1 .....	63
Ilustración 61 Incidencia .....	64
Ilustración 62 Incidencia #23 .....	64
Ilustración 63 Labels de una incidencia .....	65
Ilustración 64 Bug .....	67
Ilustración 65 Heroku login .....	69
Ilustración 66 Heroku create .....	69
Ilustración 67 Heroku instalación plugin.....	69

## Resumen

En este proyecto, realizado para la asignatura “Evolución y Gestión de la Configuración”, se va trabajar con código heredado del curso anterior.

La finalidad de este proyecto es mostrar la dificultad que conlleva trabajar con código heredado y los problemas que trae cuando se intenta complementar con otros proyectos independientes.

El coordinador del equipo será Simón Egea Guerrero y el subsistema a desarrollar o mejorar es el de “Creación y Administración de Censos”. Dicho subsistema almacena la información correspondiente a los votantes pertenecientes a un censo asociado a una determinada votación. Para ello, el sistema debe complementarse con otros subsistemas como el de “Autenticación” o el de “Creación y Administración de Votaciones”.

Para el despliegue, los coordinadores de los diferentes equipos se reunieron y llegaron a un acuerdo en la forma en la que se iba a trabajar.

## Introducción

El proyecto a desarrollar es “Agora@us”, una herramienta software que gestiona las votaciones realizadas por internet. Este proyecto está dividido en varios subsistemas y cada uno de estos subsistemas será desarrollado por un grupo de alumnos con el objetivo de mejorarlo o arreglarlo. Los subsistemas son:

- **Almacenamiento de votos:** contiene los votos de una votación.
- **Autenticación:** registra a los usuarios del sistema.
- **Cabina de votación:** permite el voto de los usuarios.
- **Creación/Administración de censos:** asocia la votación con un censo.
- **Creación/Administración de votaciones:** crea una votación.
- **Deliberaciones:** Almacena comentarios emitidos por los votantes.
- **Recuento y modificación de resultados:** cuenta los votos de una votación.
- **Verificación:** verifica la autenticidad de los votos.
- **Visualización y Gestión de resultados.**

Hay que señalar que en el grupo de tarde no existen tantos grupos para cubrir todos los subsistemas, por lo que los subsistemas de “Autenticación” y “Deliberaciones” no han podido ser desarrollados o mejorados por ningún grupo de este año y se han tenido que usar los del año anterior.

La estructura del sistema Agora@us quedaría reflejada en la siguiente figura:

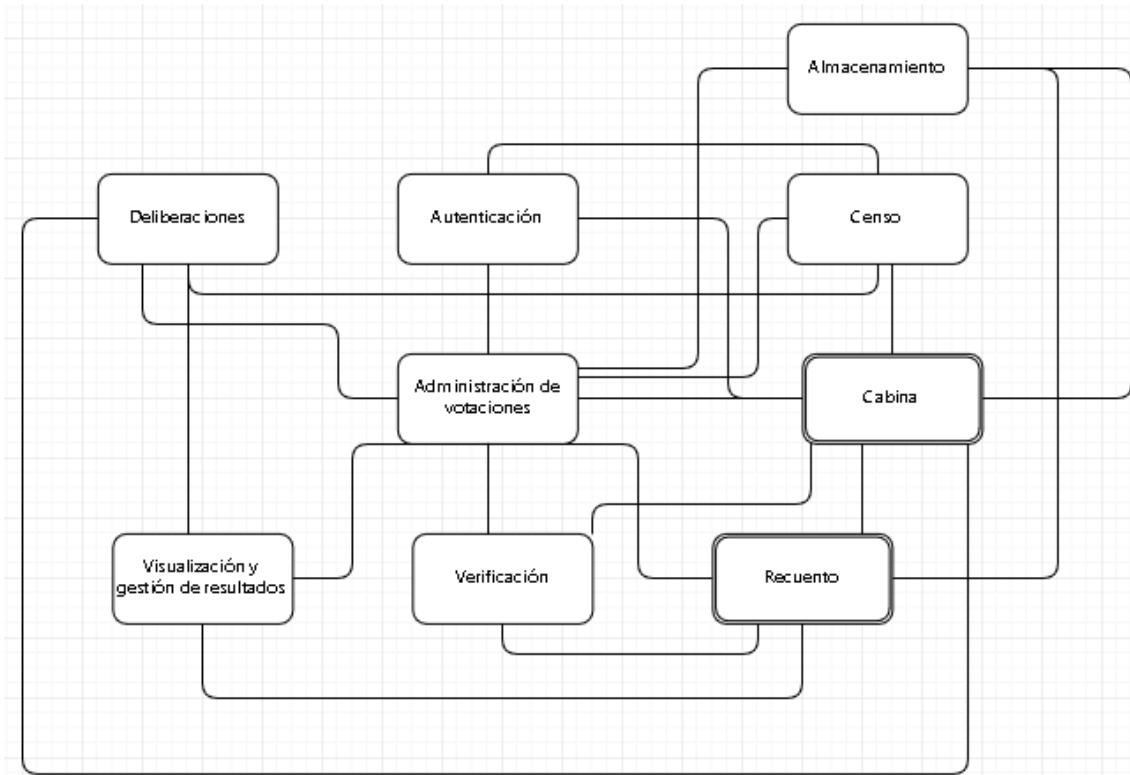


Ilustración 1 Diagrama de subsistemas

A nuestro equipo de proyecto se le ha asignado el subsistema de “**Creación y administración de censos**” y como se puede ver en el diagrama de arriba, nuestro subsistema va a poder integrarse con los subsistemas de “**Cabina**”, “**Deliberaciones**”, “**Administración de votaciones**” y “**Autenticación**”.

A grosor modo, nuestro sistema (Creación y administración de censos) se encargará de la creación de nuevos censos.

El código heredado que se nos ha proporcionado tiene las siguientes funcionalidades implementadas:

1. Envío de correos electrónicos a los usuarios que están registrados o eliminados en un censo.
2. Buscador que filtre usuarios por su nombre de usuario.
3. Exportar la información de un censo y los usuarios registrados en él en un archivo con formato .txt
4. Implementar votaciones en las que el censo sea abierto, es decir, donde se puede registrar cualquier usuario del sistema. [“1\(Documentación-censos-g2-1516-v2.00\)](#)

Alguna de las modificaciones que hemos realizado en el proyecto son las siguientes:

1. Mostrar en una tabla las últimas votaciones que se han realizado en el subsistema.
2. Se han realizado los servicios y las queries necesarias para que se puedan obtener los censos más participativos.

3. Se ha implementado un buscador de censos, el cuál buscara mediante una palabra clave todos los censos cuyo título coincida total o parcialmente con dicha palabra.
4. Se ha creado una nueva funcionalidad que permite conocer el porcentaje de usuarios que no votan para cada censo.

Estas son algunas de las funcionalidades que se han realizado, más adelante se podrá ver con más detalle todas las modificaciones realizadas.

# Descripción del sistema

En el siguiente punto se explicará el sistema desarrollado desde un punto de vista funcional.

## Componentes y relación con otros subsistemas

*"El subsistema de Creación y Administración de Censos tiene como funcionalidad principal recoger información de una serie de votaciones que nos llegarán a través del subsistema de Creación y Administración de Votaciones para poder asignarle a cada votación un determinado censo que recoja a los participantes/votantes de la misma.*

*Para conseguir el correcto funcionamiento, nuestro subsistema cuenta con varios métodos auxiliares que permiten a su vez enviar los censos creados a otros subsistemas que consumen nuestra información.*

*Nuestro proyecto está compuesto por una serie de paquetes y carpetas que se detallan a continuación:*

### 1. **src/main/java**

*Es una carpeta compuesta de una serie de paquetes que serán el foco de la programación, desde donde se creará un sistema de información de abajo a arriba.*

*Los paquetes que nos encontramos son los siguientes:*

- controllers, donde crearemos las clases destinadas a realizar las funciones CRUD definidas en el paquete Services y otros métodos o reglas de negocio.*
- converters, compuesto de clases donde especificaremos cómo transformar los objetos java a string y viceversa.*
- domain, donde definiremos nuestro modelo de dominio creando tantas clases como entidades tenga el mismo.*
- repositories, formado por clases en las cuales se definirán las distintas consultas a realizar a la Base de Datos de nuestro sistema.*
- security, donde definiremos las clases que gestionarán el login del sistema.*
- services, compuesto de clases donde se programarán los métodos CRUD y otros métodos definidos por las reglas de negocio oportunas haciendo uso del paquete Repositories y que serán llamados en las clases del paquete Controllers.*
- utilities, es un paquete formado por algunas clases cuya función es gestionar la conexión y la configuración con la Base de Datos.*
- utilities.intenal, ídem que el paquete definido arriba con alguna funcionalidad distinta.*

## 2. ***src/main/resources***

*Esta carpeta está compuesta de algunas subcarpetas y clases destinadas a la configuración de nuestro sistema.*

*Destacamos algunas de mayor importancia dado que deben ser gestionadas correctamente a la hora de crear nuestro subsistema:*

- *converters.xml, donde introduciremos un listado de todos los convertidores que creemos para que Spring los localice.*
- *i18n-l10n.xml, clase destinada a la internacionalización donde deberemos indicar la respectiva traducción de las entidades que creemos.*
- *security.xml, en esta clase especificaremos quién tiene permisos a los distintos recursos que creemos.*
- *tiles.xml, clase donde se especificará la existencia de enlaces en una determinada vista.*
- *PopulateDatabase.xml, documento destinado a la creación de entidades en un sistema y la relación entre las mismas, persistidas en la Base de Datos.*

## 3. ***src/test/java***

*Carpeta destinada a incluir clases con funciones para realizar tests unitarios a los métodos realizados dentro del paquete Service y comprobar su correcto funcionamiento.*

*También se incluyen librerías externas al proyecto. Estas librerías son un conjunto de clases con métodos y atributos que podremos utilizar en las distintas clases de nuestro proyecto. Además, también existen librerías, como Maven Dependencies, que se encargan de la gestión de las dependencias dentro del proyecto.*

## 4. ***bin***

*Carpeta que contiene las mismas carpetas definidas arriba (exceptuando las carpetas de tests) y además, una nueva carpeta llamada webapp donde especificaremos y crearemos las distintas imágenes, scripts, estilos, y vistas de nuestro proyecto.*

## 5. ***src***

*Comprende a las carpetas main y tests descritas anteriormente.*

## 6. ***target***

*Es una carpeta que contiene un archivo denominado MANIFEST.MF. Se usa para definir datos relativos a la extensión y al paquete.*

*Por último, encontramos el archivo pom.xml. Este fichero es la herramienta principal de un Proyecto Maven. Contiene información acerca del proyecto, de los plugins que instalamos, de versiones, de dependencias... Además, antes de ejecutarse el proyecto, siempre se visita este fichero. "1(Documentación-censos-g2-1516-v2.00)*

Los subsistemas de los cuales puede interactuar el nuestro son los descritos en el siguiente diagrama:



*Ilustración 2Diagrama de subsistemas a integrar*

Tal y como se puede observar en el diagrama de la “Ilustración 2” nuestro subsistema puede y debe interactuar con los subsistemas de “**Cabina**”, “**Autenticación**”, “**Administración de votaciones**” y “**Deliberaciones**”.

El subsistema de **cabina** accederá al nuestro para la obtención de todos los censos para, posteriormente, mostrárselo al cliente y que este pueda votar. Además, el subsistema de **deliberaciones** se encargará de almacenar los comentarios emitidos por los votantes, dichos comentarios contendrán información sobre el votante que los haya realizado.

Por otro lado, el subsistema de **autenticación** hará que el usuario pueda registrarse o autentificarse con su cuenta para así poder acceder a sus censos, por lo que será de vital importancia conseguir interactuar correctamente con este subsistema.

El último subsistema que nos encontramos es el de **administración de votaciones**, el cual se encargará de crear nuevas votaciones en función de las demandas exigidas por los clientes así como de administrar las votaciones creadas. Necesitamos integrarnos con este subsistema ya que necesitamos acceder a las votaciones de los usuarios.

## Cambios realizados al proyecto

El proyecto que hemos heredado se ha realizado en su totalidad con el IDE de Eclipse y las tecnologías que se han utilizado son las mismas que las de la asignatura Diseño y Pruebas (Spring, MySql, Maven, Hibernate y Tomcat 7). Además de estas tecnologías también han usado Jenkins y Git para la gestión del código fuente junto con Github.

A la hora de ponernos a trabajar sobre este proyecto hemos optado por usar la máquina virtual de la asignatura de Diseño y Pruebas (el grupo del año pasado no dejó ninguna máquina virtual por lo que hemos optado por esta opción) la cuál tiene todas las herramientas necesarias para comenzar a trabajar con el código heredado a excepción del Jenkins y del Git. Estas dos últimas se han instalado posteriormente ya que han sido las que hemos estado utilizando a lo largo de todo el desarrollo del proyecto.

Los cambios que se han introducido en el sub-sistema heredado de forma general son los siguientes:

- [Añadir nuevas funcionalidades](#)
- [Introducción de nuevas dependencias](#)
- [Introducción de nuevos plugins](#)
- [Automatización y empaquetado](#)
- [Creación de documento de ayuda](#)
- [Sistema de autenticación](#)
- [Modificación de la información guardada en la base de datos](#)

### Nuevas funcionalidades

Las nuevas funciones que se han añadido al subsistema heredado vienen recogidas en la siguiente lista:

*[Mostrar en una tabla las últimas votaciones que se han realizado en el subsistema.](#)*

Para la realización de esta funcionalidad se optó por crear una rama llamada “showLastVotes” en la cuál se realizaron todos los cambios necesarios.

Primero se realizo la query necesaria para obtener los censos de la base de datos ordenados por orden descendente de fecha tal que así:

```

@Repository
public interface CensusRepository extends JpaRepository<Census, Integer> {

    //Devuelve los censos por orden descendente de fecha
    @Query("select c from Census c order by c.fechaFinVotacion desc")
    Collection<Census> allCensus();
}

```

*Ilustración 3UltimasVotacionesRepo*

Una vez realizada la query correspondiente se pasa al desarrollo del controlador, el cuál obtendrá la información recuperada por la consulta anterior y la enviará a la vista correspondiente.

```

@RequestMapping(value="listAll", method = RequestMethod.GET)
public ModelAndView getListAll(){
    ModelAndView result = new ModelAndView("census/listAll");
    Collection<Census> cs;
    cs = censusService.findAll();
    result.addObject("census", cs);
    result.addObject("requestURI", "census/listAll.do");

    return result;
}

```

*Ilustración 4UltimasVotacionesController*

Para finalizar realizamos la vista en la cuál se mostrará el resultado. Para realizar la vista se usará una clase .jsp la cuál podemos encontrar en el directorio “webapp→views→census→listAll.jsp”

```

<!-- En esta tabla mostraremos los últimos censos que se han realizado -->
<display:table pagesize="5" class="census" name="census"
    requestURI="${requestURI}" id="census">

    <!-- Atributos de la tabla -->
    <spring:message code="census.username" var="username" />
    <display:column property="username" title="${username}" sortable="true" />

    <spring:message code="census.tituloVotacion" var="tituloVotacion" />
    <display:column property="tituloVotacion" title="${tituloVotacion}" sortable="true" />

    <spring:message code="census.tipoCenso" var="tipoCenso" />
    <display:column property="tipoCenso" title="${tipoCenso}" sortable="true" />

    <spring:message code="census.fechaInicioVotacion" var="fechaInicioVotacion" />
    <display:column property="fechaInicioVotacion" title="${fechaInicioVotacion}" sortable="true" />

    <spring:message code="census.fechaFinVotacion" var="fechaFinVotacion" />
    <display:column property="fechaFinVotacion" title="${fechaFinVotacion}" sortable="true" />

</display:table>

```

*Ilustración 5UltimasVotacionesVista*

Una vez realizada la funcionalidad el resultado final es el siguiente:

LIST ALL CENSUS				
13 items found, displaying 1 to 5. [First/Prev] <a href="#">1</a> , <a href="#">2</a> , <a href="#">3</a> [Next/Last]				
Name	Title	Type	Start date	End date
admin1	Censo abierto NO VACIO, ACTIVO	abierto	2015-10-07 00:00:00.0	2016-12-05 00:00:00.0
admin1	Censo cerrado VACIO, ACTIVO	cerrado	2015-10-01 00:00:00.0	2016-02-28 00:00:00.0
admin1	Elecciones	cerrado	2015-10-01 00:00:00.0	2015-12-02 00:00:00.0
admin1	Censo cerrado NO VACIO, NO ACTIVO	cerrado	2011-10-01 00:00:00.0	2012-12-02 00:00:00.0
admin1	Censo abierto NO VACIO, NO ACTIVO	abierto	2010-10-11 00:00:00.0	2011-12-03 00:00:00.0

[Back](#)

#### *Ilustración 6UltimasVotacionesResultado*

Una vez realizada la funcionalidad y después de comprobar que todo funciona correctamente se hizo un “merge” con la rama “master” y se procedió a subir todos los cambios al repositorio de GitHub.

#### *Obtención de los censos más participativos*

Se ha considerado que sería muy conveniente que el usuario de nuestro subsistema pueda saber cuales han sido los censos más participativos ya que aportaría mucha información acerca de cuales son las tendencias actuales de nuestros clientes.

Para la realización de esta nueva funcionalidad se creó una nueva rama llamada “censoMayorParticipación” ya que es conveniente separar el código que se encuentra en línea base con el que está en proceso de desarrollo. Una vez que se ha comprobado que la funcionalidad funciona correctamente se unió con la rama “master”.

El servicio que ofrece esta funcionalidad es el mostrado en la siguiente figura:

```

public Map<Census, Integer> mostPopularCensus() {
    Collection<Census>censos=censusRepository.findAll();
    Map<Census, Integer>res= new HashMap<Census, Integer>();
    Map<Census, Integer>result= new HashMap<Census, Integer>();
    for(Census c:censos){
        Integer cont=0;
        for(String s:c.getVotoPorUsuario().keySet()){
            if(c.getVotoPorUsuario().get(s)==true){
                cont=cont+1;
            }
        }
        res.put(c, cont);
    }
    res = sortByValue(res);
    int cont = 0;
    for(Census c : res.keySet()){
        if(cont<10){
            result.put((Census)c, res.get(c));
        }else{
            break;
        }
        cont++;
    }

    return result;
}

```

*Ilustración 7CensosMasPopulares*

Una vez completada la funcionalidad el resultado es una tabla en la cuál se muestran los 10 censos que más participativos (populares):

LIST MOST POPULAR CENSUSES					
10 items found, displaying 1 to 5.[First/Prev] 1, 2 [Next/Last]					
	Owner	Name of vote	Start date	Finish date	
	admin1	Elecciones	01/10/2015	02/12/2015	1
	admin1	Censo cerrado VACIO, ACTIVO	01/10/2015	28/02/2016	2
	admin1	Censo cerrado NO VACIO, NO ACTIVO	01/10/2011	02/12/2012	1
	admin1	Censo abierto NO VACIO, ACTIVO	07/10/2015	05/12/2016	1
	admin1	Censo abierto NO VACIO, NO ACTIVO	11/10/2010	03/12/2011	1

[Back](#)

*Ilustración 8CensosMasPopularesTabla*

Tal y como se puede observar en la tabla, salen los 10 censos más populares que nuestro subsistema tiene almacenados en su base de datos.

### *Porcentaje de usuarios que no votan por censo*

La siguiente funcionalidad que se ha implementado es la de mostrar el porcentaje de usuarios que no han votado por cada censo. Dicha funcionalidad fue desarrollada en la rama “porcentajeAbstencion” y posteriormente, una vez comprobado que funcionaba correctamente se pasó a hacer un “merge” con la rama “master”.

El grosor de la funcionalidad es el método que calcula dicho porcentaje de abstención y que podemos ver en la siguiente figura:

```
public Map<Census, Double> abstentionPercentage(){
    Collection<Census> censos=censusRepository.findAll();
    Map<Census,Double>res= new HashMap<Census,Double>();
    for(Census c:censos){
        Integer cont=0;
        for(String s:c.getVotoPorUsuario().keySet()){
            if(c.getVotoPorUsuario().get(s)==false){
                cont=cont+1;
            }
        }
        res.put(c, (double) cont/c.getVotoPorUsuario().keySet().size()*100);
    }
    return res;
}
```

*Ilustración 9AbstencionMetodo*

Posteriormente, este método enviará un “Map” a la vista, la cual mostrará los censos con sus respectivos porcentajes de abstención tal y como se puede observar en la siguiente figura:

LIST ABSTENTION PERCENTAGE					
13 items found, displaying 1 to 5.[First/Prev] 1, 2, 3 [Next/Last]					
	Owner	Name of vote	Start date	Finish date	
	admin1	Elecciones	01/10/2015	02/12/2015	50.0
	admin1	Censo cerrado VACIO, ACTIVO	01/10/2015	28/02/2016	0.0
	admin1	Censo cerrado NO VACIO, NO ACTIVO	01/10/2011	02/12/2012	0.0
	admin1	Censo abierto NO VACIO, ACTIVO	07/10/2015	05/12/2016	50.0
	admin1	Censo abierto NO VACIO, NO ACTIVO	11/10/2010	03/12/2011	0.0

Back

*Ilustración 10AbstencionTabla*

Podemos observar que al final de cada tupla de la tabla aparece el porcentaje de abstención de dicho censo.

### *Mejora del css*

Dicha mejora ha sido solo una leve modificación del CSS a nivel de colores. Para la realización de dicho cambio se creó una rama llamada “cssmod” en la cuál se realizó íntegramente el cambio. Lamentablemente, dicho cambio no se llegó a introducir en la rama “master”.

### *Buscador de censos*

Se ha implementado un buscador de censos, el cuál buscara mediante una palabra clave todos los censos cuyo título coincida total o parcialmente con dicha palabra. Para la realización de esta nueva funcionalidad se creó una rama llamada “buscador” que posteriormente una vez comprobado que funcionaba correctamente se unió a la rama “master” haciendo uso del comando “merge”.

El método que realiza esta función es el mostrado en la siguiente figura:

```
public Collection<Census> findByKey(String key) {
    return censusRepository.findByRecipeKeyWord(key);
}
```

*Ilustración 11BuscadorMetodo*

Dicho método accederá al repositorio para hacer uso de una query, la cuál devolverá todos los censos cuyo título contenga la palabra clave introducida por el usuario.

```
//Devuelve los censos que contengan un título o parte de un título dado
@Query("select c from Census c where c.tituloVotacion like concat('%', ?1, '%')")
Collection<Census> findByRecipeKeyWord(String key);
```

*Ilustración 12BuscadorQuery*

El usuario introducirá una palabra clave en el recuadro correspondiente y posteriormente dará click en “Search” para realizar la búsqueda.



*Ilustración 13BuscadorBoton*

Finalmente, los censos que sean recuperados de la base de datos serán mostrados en una tabla tal y como se muestra en la siguiente ilustración:

LIST OF CENSUS					
6 items found, displaying 1 to 5 [First/Prev] <a href="#">1</a> , <a href="#">2</a> [Next/Last]					
Owner	Name of vote	Start date	Finish date	Number of registered	
 admin1	Censo cerrado VACIO, ACTIVO	01/10/2015	28/02/2016	2	
 admin1	Censo cerrado NO VACIO, NO ACTIVO	01/10/2011	02/12/2012	1	
 admin1	Censo cerrado vacio activo	01/01/2016	03/12/2016	0	
 admin1	Censo cerrado vacio NO activo	11/10/2011	03/12/2012	0	
 admin1	Censo cerrado vacio NO activo	11/08/2016	03/09/2016	0	

[Back](#)

Ilustración 14BuscadorTabla

## Introducción dependencias maven

### *Introducción de nuevas dependencias maven*

Estos cambios en las dependencias de nuestro proyecto maven se realizaron en una una nueva rama llamada “mvnDependency”.

La nueva dependencia que se ha optado por introducir en nuestro fichero “pom.xml” ha sido la de **WireMock**, la cuál es un simulador de API basado en HTTP. Se introdujo esta dependencia ya que se consideró interesante debido a que había que interactuar con otros substistemas y sería muy posible que se usase HTTP para las comunicaciones. En la siguiente ilustración se muestra el “xml” necesario la introducir esta dependencia.

```
<dependency>
    <groupId>com.github.tomakehurst</groupId>
    <artifactId>wiremock</artifactId>
    <version>2.4.1</version>
</dependency>
```

Ilustración 15DependenciaWiremock

Posteriormente, cuando se comprobó que todo funcionaba correctamente se hizo un “merge” con la rama “master” para consolidar los cambios.

Si se necesita más información acerca de esta dependencia se puede encontrar en su sitio web (<http://wiremock.org/>)

## Introducción de nuevos plugins

### *Introducción de nuevos plugins en maven*

A la hora de introducir los nuevos plugins se creó una nueva rama para separar dichos cambios de la rama principal, esta nueva rama se llamó “mvnDependency”.

Uno de los plugins que se ha elegido para introducirlo en nuestro proyecto ha sido el de **PMD** el cuál nos dará consejos de buenas prácticas y de como podemos mejorar nuestro código haciendo uso de estas. Este plugin se ejecutará junto con la fase de test y te generará automáticamente un archivo con los consejos a poner en práctica. En la siguiente figura podemos ver como quedaría dicho plugin en el archivo “pom.xml”

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-pmd-plugin</artifactId>
<version>3.7</version>
<executions>
    <execution>
        <id>Before testing</id>
        <phase>test</phase>
        <goals>
            <goal>pmd</goal>
        </goals>
    </execution>
</executions>
</plugin>
```

Ilustración 16PluginPMD

Si se quiere conocer más acerca de este plugin se puede acceder a la siguiente dirección web <https://maven.apache.org/plugins/maven-pmd-plugin/>

El siguiente plugin que hemos introducido ha sido **Doap**, este curioso plugin te genera a partir del archivo “pom.xml” un documento para que sea más fácil su comprensión. La introducción de este plugin se ha realizado de la siguiente manera.

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-doap-plugin</artifactId>
<version>1.2</version>
</plugin>
```

Ilustración 17PluginDoap

Si se quiere investigar un poco más acerca de este plugin se puede encontrar más información en la siguiente dirección <https://maven.apache.org/plugins/maven-doap-plugin/>

Una vez realizado todas estas modificaciones y comprobado que todo funcionaba correctamente se paso ha hacer un “merge” con la rama “master”.

## Creación documento de ayuda

### *Documento de ayuda*

Se ha realizado un documento en el cual se explica detalladamente las distintas funciones que realizo nuestro subsistema para de esta forma ayudar a otros grupos a que se integren con nosotros. Dicho documento se puede encontrar en nuestro repositorio de Github cuyo enlace esta disponible al principio del documento.

## Sistema de autenticación

### *Sistema de autenticación*

Dado que el subsistema de autenticación no funciona correctamente hemos decidido implementar nosotros mismo un pequeño sistema de logueo que supla este vacio.

Para la realización de esta mejora se ha optado por crear una rama llamada “censoLogin”, que posteriormente se ha unido a la rama “master” con el comando “merge” de git, una vez comprobado su correcto funcionamiento.

Dado que el sistema heredado contaba con un pequeño sistema de logueo se ha optado por mantenerlo e introducir manualmente las cuentas de usuarios en la base de datos. Esto se hace añadiendo nuevos elementos en el “PopulateDatabase.xml” tal y como se puede ver en la figura siguiente:

```
<bean id="userAccount2" class="security.UserAccount">
    <property name="username" value="user1" />
    <property name="password" value="24c9e15e52afc47c225b757e7bee1f9d" />
    <property name="authorities">
        <list>
            <bean class="security.Authority">
                <property name="authority" value="CUSTOMER" />
            </bean>
        </list>
    </property>
</bean>
```

*Ilustración 18LoginPopulate*

Una vez almacenado en nuestro subsistema el usuario, lo único que debemos de hacer es loguearnos con este haciendo uso de la interfaz de usuario que proporcionamos para este propósito.



Ilustración 19 Login Interfaz

Gracias a esto un usuario se puede loguear con su cuenta y acceder a sus censos, así como a otras funciones del subsistema.

## Modificación base de datos

### *Modificación censos almacenados*

Dado que hemos heredado este subsistema del grupo del año pasado las fechas de inicio y fin de los censos estaban anticuadas y hemos tenido que modificar el "PopulateDatabase.xml" para que a la hora de guardar los censos dichas fechas fuesen modificadas.

Debido a que es relativamente sencillo realizar esta acción se optó por realizar el cambio directamente en la rama "master". La modificación es la mostrada en la figura siguiente:

```
<bean id="census13" class="domain.Census">
    <property name="username" value="admin1" />
    <property name="idVotacion" value="35" />
    <property name="tituloVotacion" value="Censo abierto" />
    <property name="tipoCenso" value="abierto" />
    <property name="votoPorUsuario">
        <map>
        </map>
    </property>

    <property name="fechaInicioVotacion" value="10/01/2016 00:00" />
    <property name="fechaFinVotacion" value="10/28/2016 00:00" />
</bean>
```

Ilustración 20 Modificación Censos

## Planificación y tareas

En la siguiente tabla se muestra la planificación que hemos seguido y las tareas que se han realizado durante el proyecto, así como las asignaciones de los distintos miembros del grupo a cada una de ellas.

Tabla 1 Planificación y tareas

Tarea	Descripción	Asignado
Últimas votaciones	El sistema mostrará al usuario las últimas votaciones por orden descendente de fecha	Santiago Fraga Martín-Arroyo
Censos con más participación	El sistema mostrará al usuario los censos con más participación	Rubén Barrientos Mohedano
Porcentajes de usuarios que no votan	El sistema mostrará el porcentaje de usuarios que no votan en cada uno de los censos.	Simón Egea Guerrero
Mejorar CSS	Se ha mejorado el CSS ya que en un principio se consideró que era pobre	Nicolás Lorenz Rosado
Buscador de censos	El sistema implementará un buscador el cuál devolverá los censos cuyo título coincida con la palabra clave	Pablo Romero Vázquez, Santiago Fraga Martín-Arroyo, Rubén Barrientos Mohedano
Dependencias maven	Se ha introducido una dependencia nueva con "WireMock"	Santiago Fraga Martín-Arroyo
Plugins maven	Se ha introducido los plugins de PMD y Doap	Santiago Fraga Martín-Arroyo
Automatización y empaquetado	Se ha modificado el "pom.xml" para que se pueda realizar tanto el empaquetado como el deploy	Simón Egea Guerrero
Documento de ayuda	Dicho documento contiene las descripciones de los distintos métodos que tiene nuestro subsistema	Simón Egea Guerrero
Sistema autenticación	Se ha implementado un sistema de autenticación en nuestro subsistema	Rubén Barrientos Mohedano

Modificación de la base de datos	Se ha modificado las fechas de inicio y de fin de los censos.	Rubén Barrientos Mohedano
Wiki del grupo	Se ha realizado la wiki del grupo	Santiago Fraga Martín-Arroyo
Documentación del proyecto	Realización de la documentación del proyecto	Santiago Fraga Martín-Arroyo, Pablo Romero Vázquez
Realización de tests	Realizar los tests que faltan para algunos métodos	Pablo Romero Vázquez
Diario de grupo	Realizar el diario del grupo	Nicolás Lorenz Rosado
Maquina virtual	Preparar máquina virtual para su posterior descarga	Santiago Fraga Martín-Arroyo
Estudio de Git, Github, Jenkins y Maven	Aprender sobre el uso de estas herramientas	Grupo
Uso de Jenkins en nuestro proyecto	Realizar las tareas correspondientes con la herramienta Jenkins	Simón Egea Guerrero

En esta otra tabla se muestra la planificación y las distintas tareas que se han ido realizando a lo largo de la mejora del proyecto (Milestone 5):

Tabla 2 Planificación y tareas para mejora

Tarea	Descripción	Asignado
Creación del “.gitignore” y limpieza del repositorio	Se debe de evitar que los archivos generados durante la ejecución del proyecto se suban como son el “.class”, “classpath” y el “.project”	Pablo Romero Vázquez
Reacer documentación	La documentación debe de reacerse debido a que no cumple con los requisitos	Santiago Fraga Martín-Arroyo y Pablo Romero Vázquez
Creación de tests	Algunas funciones carecen de un test que compruebe su funcionamiento por lo que se deben de crear dichos tests	Pablo Romero Vázquez
Despliegue automatico	La aplicación se debe de desplegar automáticamente.	Simon Egea Guerrero y Ruben Barrientos Mohedano
Diario de grupo	Creación del diario de grupo.	Nicolas Lorenz Rosado

Integración con otro subsistema	Nuestro sistema se debe de integrar con otro adecuadamente.	Nicolas Lorenz Rosado
---------------------------------	---	-----------------------

## Elementos de control

Los elementos de control (Configuration Items, CI) serán este documento, el diario de grupo, las incidencias, la máquina virtual que hemos preparado, así como el mismo código de la aplicación.

Las documentaciones están alojadas en la wiki para que fuesen accesibles a todos. Las incidencias se han gestionado en GitHub y la política que hemos seguido se encuentra detallada en el apartado “Gestión del cambio, incidencias y depuración”. Las herramientas que se han usado a lo largo de todo el proyecto son las recogidas en la siguiente tabla:

Tabla 3 Elemento de control

Nombre	Desarrollador	Versión	Plataforma (SO)
Acrobat Reader DC	Adobe Systems	15.023.20056	Windows 10 64 bits
Microsoft Word	Microsoft	16.0.6965.2117	Windows 10 64 bits
Skype	División de Skype de Microsoft	7.31.104	Windows 10 64 bits
Whatsapp	Whatsapp Inc, Facebook	2.17.24	Android
Eclipse	Eclipse Foundation	Eclipse Indigo EE SR2	Windows xp 64 bits
Mysql Workbench	Oracle Corporation	5.2 CE	Windows xp 64 bits
Git	Linus Torvalds	2.10.1	Windows 10 64 bits
Jenkins	Jenkins CI	0.2.37	Windows 10 64 bits
Virtual Box	Oracle Corporation	5.1.6	Windows 10 64 bits
Spring	Pivotal Software	4.0.0	Windows xp 64 bits
Tomcat	Apache	7.0	Windows xp 64 bits
7zip	Igor Pavlov	9.20	Windows 10 64 bits
Google Chrome	Google	55.0.2883.87	Windows 10 y xp 64 bits
Microsoft Windows	Microsoft	10 y xp	-

- **Adobe Acrobat Reader:**  
Se ha utilizado para poder visualizar documentos en formato PDF.
- **Microsoft Word:**  
Usado para realizar la documentación y otros documentos necesarios para la realización de proyecto.
- **Skype:**  
Usado para comunicarnos entre los distintos miembros del grupo.
- **Whatsapp:**  
Usado para comunicarnos entre distintos miembros del grupo.
- **Eclipse:**  
IDE usado para el desarrollo del proyecto.
- **MySql Workbench:**  
Herramienta de administración de BB.DD utilizada para el desarrollo de nuestro código.
- **Git:**  
Usado para controlar las versiones de nuestro código.
- **Jenkins:**  
Utilizada para la gestión de la integración del proyecto.
- **VirtualBox:**  
Máquina virtual utilizada para el desarrollo del código de nuestro subsistema.
- **Microsoft Windows:**  
Sistema operativo usado para el desarrollo del proyecto en su totalidad.
- **Spring:**  
Framework usado para el desarrollo del proyecto.
- **Tomcat:**  
Usado para el despliegue local de nuestro proyecto.
- **7zip:**  
Usado para comprimir la máquina virtual y el entregable.
- **Google chrome:**  
Navegador utilizado para probar nuestro subsistema.

## Entorno de desarrollo

La máquina virtual usada por el grupo es la misma que se usó para cursar la asignatura de Diseño y Pruebas. Dicha máquina virtual la podemos encontrar disponible en la wiki del grupo, en la sección “Máquina virtual”. La máquina virtual que ofrecemos tiene instalado el siguiente software:

- Java 7
- MySQL
- Spring
- Tomcat 7
- Hibernate
- Eclipse

## Máquina virtual

Para configurar adecuadamente la máquina virtual que proporcionamos debemos de seguir los pasos descritos a continuación:

1. Lo primero que debemos de hacer es descargarnos un software de virtualización, nosotros recomendamos el Virtual Box ya que es el que hemos usado para la realización del proyecto.  
Podemos descargarnos dicho software de la siguiente dirección:  
<https://www.virtualbox.org/>
2. Una vez descargado el software de virtualización procederemos a ejecutar la máquina virtual y nos aparecerá la siguiente pantalla:

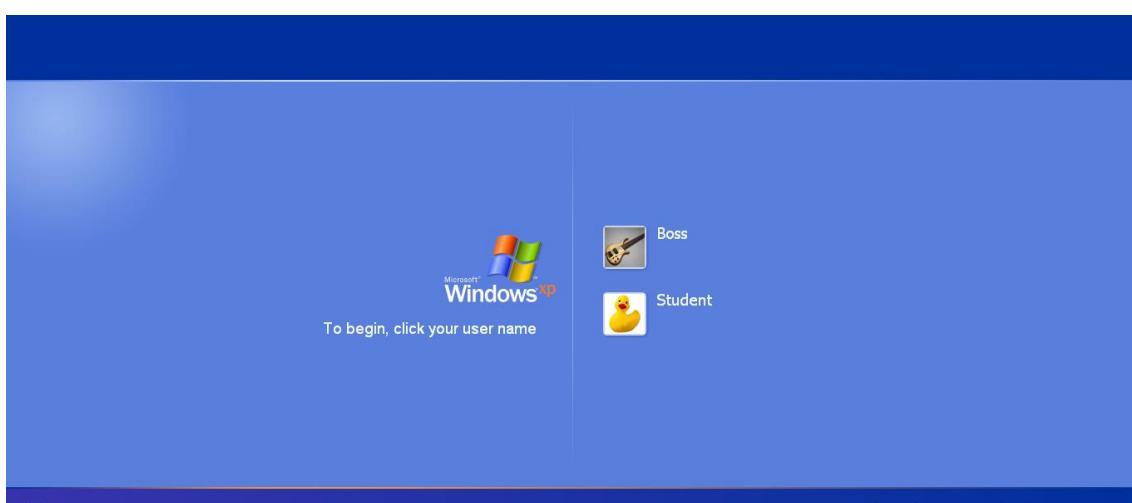


Ilustración 181 Maquina virtual login

Tal y como se muestra en la imagen hay dos tipos de usuario, nosotros elegiremos el usuario llamado "Student". Nos pedirá una contraseña, dicha contraseña viene en un bloc de notas juntos con el .zip de la máquina virtual.

3. Ahora procederemos a descargarnos un workspace pre configurado el cuál podemos encontrar en la wiki del grupo en la sección de "Máquina virtual".
4. Una vez realizado los pasos anteriores procederemos a ejecutar el eclipse de nuestra máquina virtual y seleccionaremos el workspace que acabamos de descargarnos.

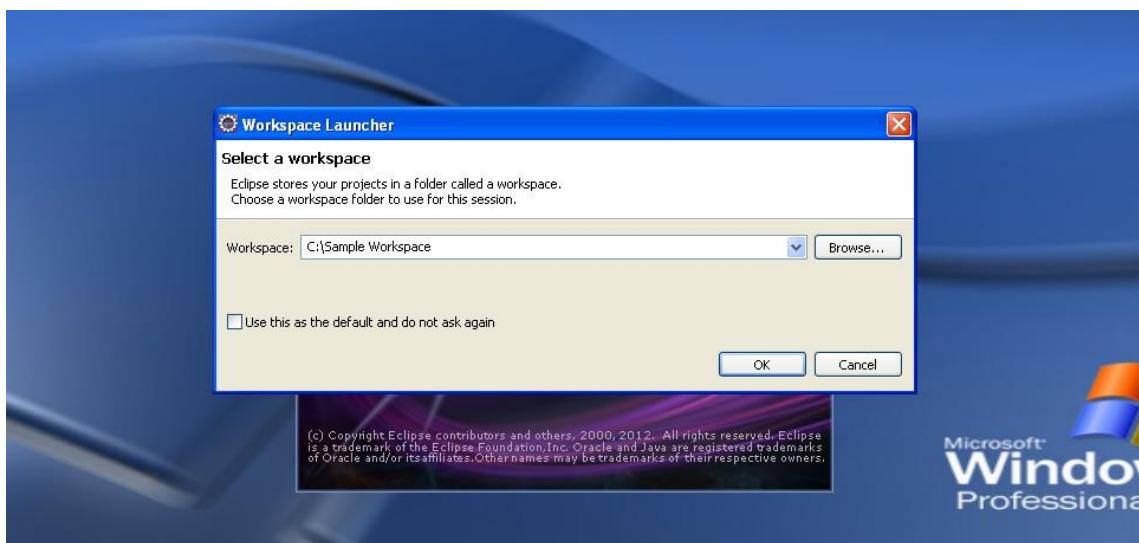


Ilustración 192 Maquina virtual eclipse workspace

5. Una vez abierto el Eclipse nos tiene que salir una ventana parecida a la mostrada a continuación:

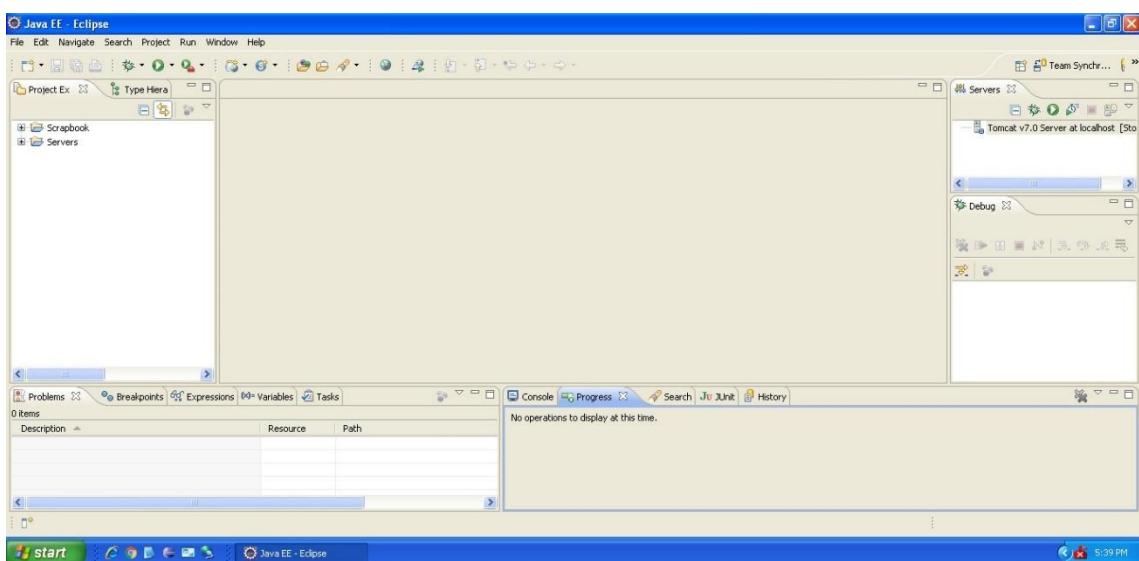


Ilustración 203 Maquina virtual eclipse abierto

6. Ahora debemos de configurar el puerto en el cuál el servidor de Tomcat se ejecutará. Para configurarlo debemos de darle doble click a “Tomcat v7.0 Server” que esta situado en la pestaña de “Servers” y nos saldrá la siguiente ventana:

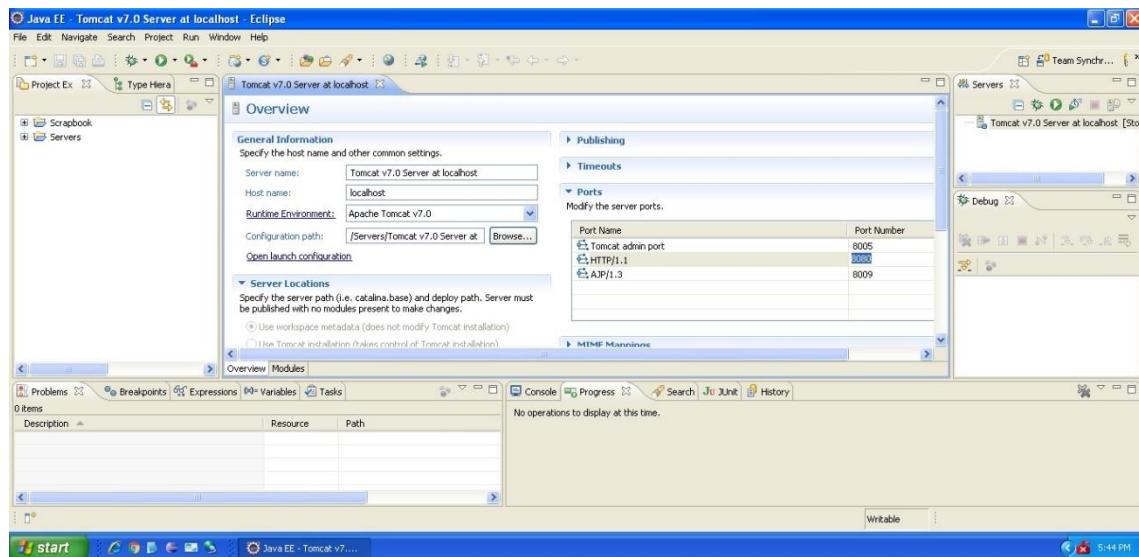


Ilustración 214 Maquina virtual eclipse tomcat

Lo que debemos de hacer es modificar el valor de “HTTP/1.1” a 8080 tal y como esta en la imagen y posteriormente guardar los cambios.

7. El siguiente paso a realizar es importar el proyecto, para ello nos iremos a la ventana de “File→Import→Maven→Existing Maven Projects” como muestra la imagen siguiente:

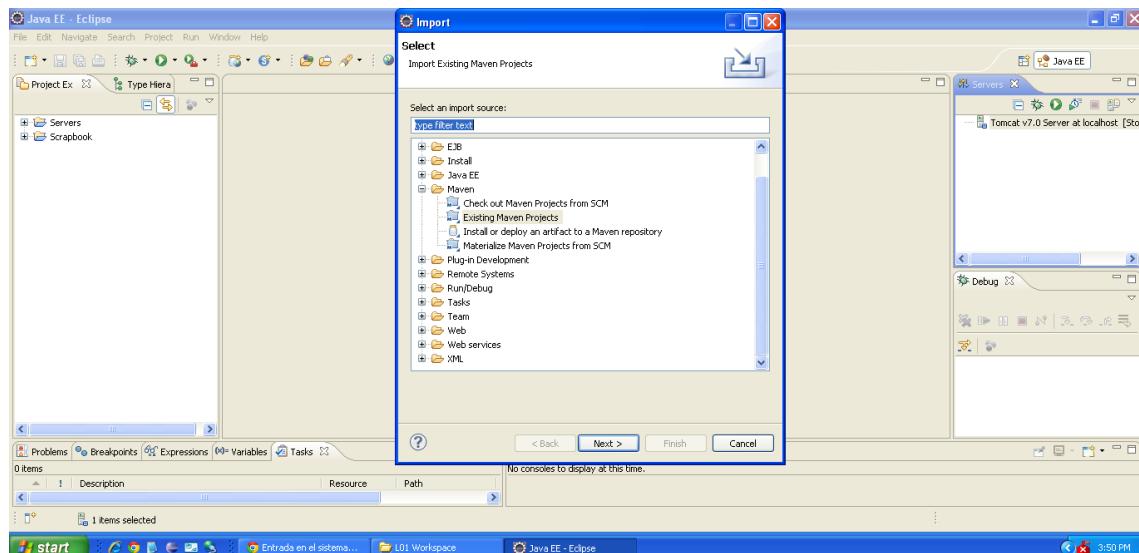


Ilustración 225 Maquina virtual eclipse importar

Una vez realizado este paso, el proyecto se importará. Puede tardar entre 15-20 minutos la importación ya que maven se tendrá que descargar todas las dependencias y plugins que aparecen en el archivo “pom.xml”.

- El siguiente paso es ejecutar el MySQL Workbench con el usuario “Boss” cuya contraseña se encuentra en el bloc que notas que viene junto con la máquina virtual.

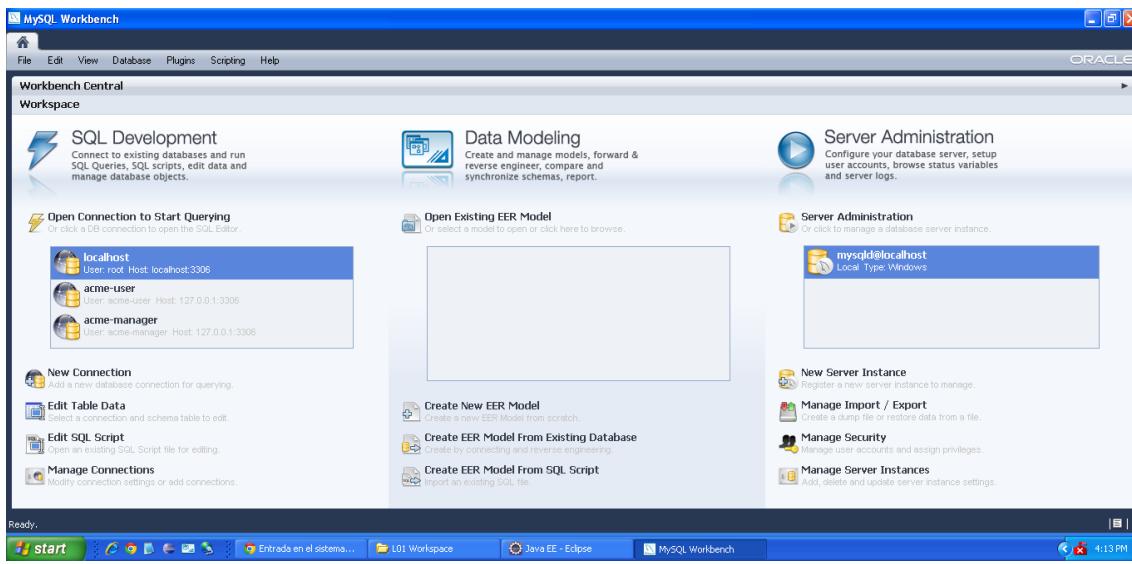


Ilustración 236Maquina virtual mysql

- Ahora procederemos a crear los permisos de los usuarios. Para ello debemos de ejecutar el siguiente script:

```
drop database if exists `ADM Census`;
create database `ADM Census`;
```

```
grant select, insert, update, delete
on `ADM Census`.* to 'acme-user'@'%';
```

```
grant select, insert, update, delete, create, drop, references, index, alter,
create temporary tables, lock tables, create view, create routine,
alter routine, execute, trigger, show view
on `ADM Census`.* to 'acme-manager'@'%';
```

- Ahora ejecutaremos la clase “PopulateDatabases.java” que se encuentra dentro de nuestro proyecto para popular la base de datos.

- Una vez realiado este paso debemos de añadir nuestro proyectoa tomcat haciendo click derecho sobre “Tomcat v7.0 Server” y “Add and Remove...” tal y como se muestra en la siguiente figura:

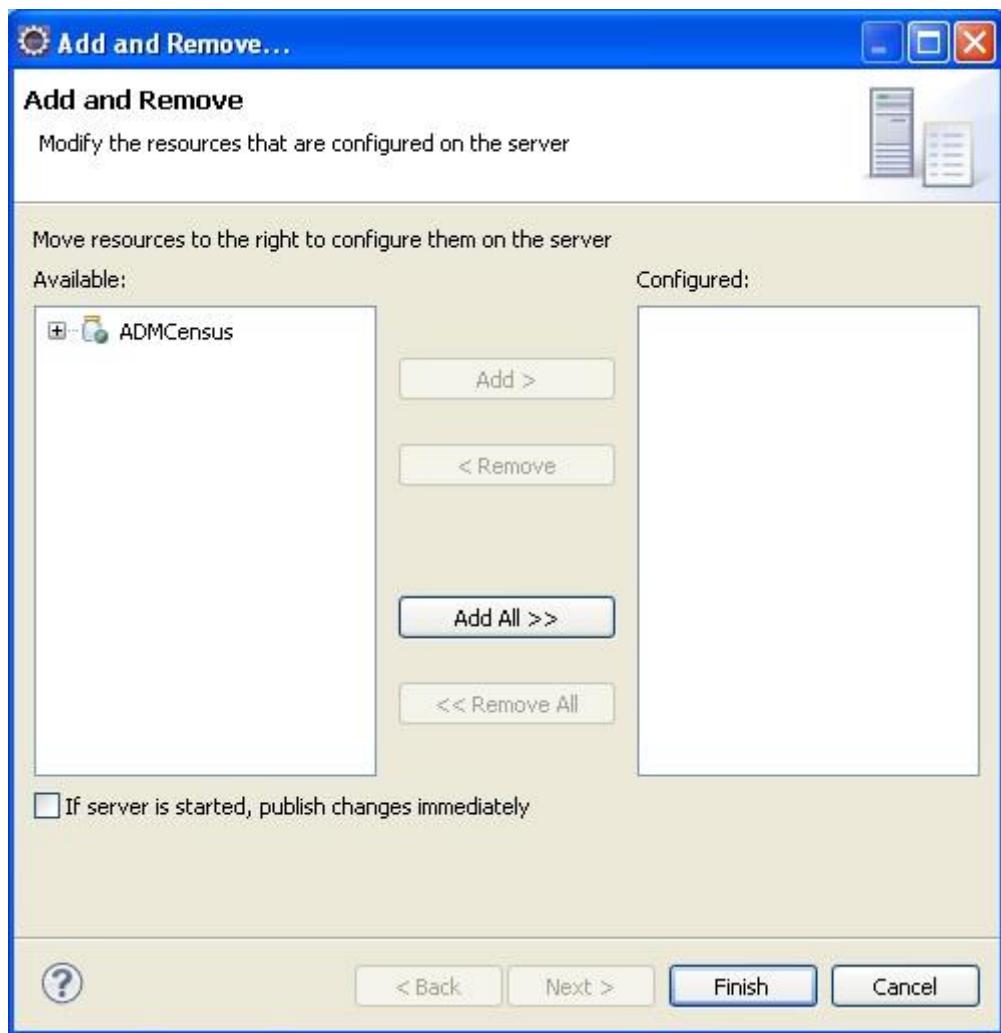


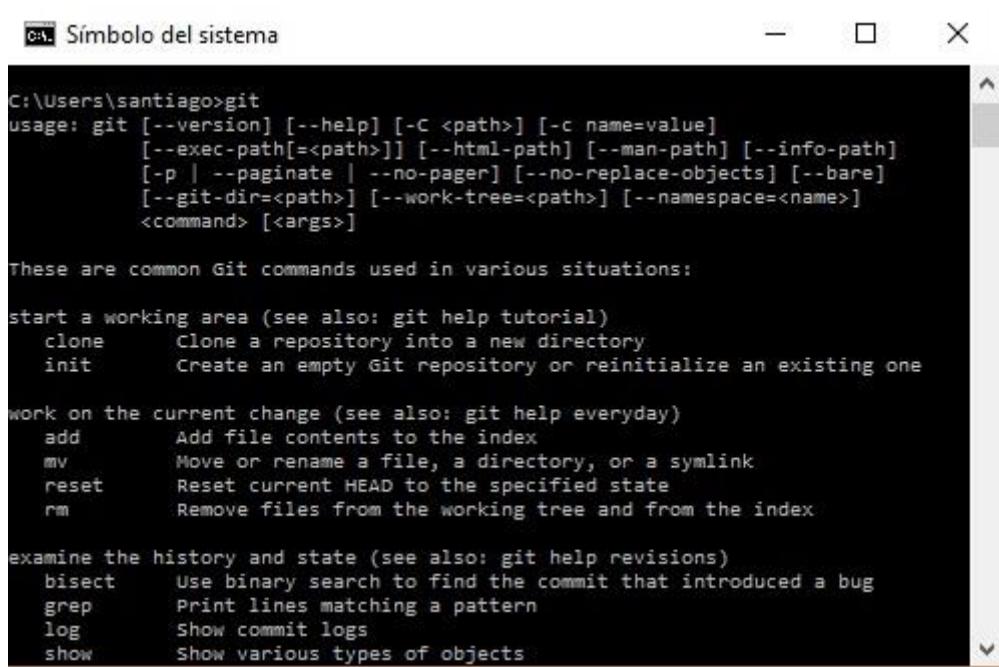
Ilustración 247Maquina virtual add tomcat

12. Por último, debemos de ejecutar el servidor de Tomcat y acceder a la dirección url “localhost:8080/ADMCensus” para ver el subsistema de censo en funcionamiento.

## Instalación Git

Para la instalación de Git lo más fácil es acceder a la dirección siguiente para descargarlo <https://git-scm.com/downloads> , una vez descargado lo instalaremos como cualquier otro programa en Windows y comprobaremos que se ha instalado correctamente introduciendo en la consola de comandos “git”.

La respuesta que nos tiene que dar es algo parecida a la siguiente:



```
C:\Users\santiago>git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset     Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types of objects
```

Ilustración 28Consola git

## Instalación Jenkins

Jenkins nos proporciona un instalador para Windows, en el cuál lo único que tenemos que hacer es darle a siguiente para que se nos instale automáticamente. Dicho instalador lo podemos encontrar en su web oficial <https://jenkins.io/>

Una vez instalado se nos creará un proceso que mantendrá Jenkins ejecutándose en nuestra máquina. Para acceder a Jenkins lo único que debemos de hacer será introducir la dirección “localhost:8080”.

Tras registrarnos con nuestro usuario ya podremos usar Jenkins.

**Jenkins**

Jenkins >

Nueva Tarea Personas Historial de trabajos Relacion entre proyectos Comprobar firma de archivos Administrar Jenkins Mis vistas Credentials

**Todo** + S W Nombre ↓ Último Éxito Último Fallo Última Duración

S	W	Nombre ↓	Último Éxito	Último Fallo	Última Duración
●	●	Censo	2 días 6 Hor - #33	16 días - #30	16 Min
●	●	Tarea1	1 Mes 7 días - #2	N/D	1,9 Seg
●	●	TareaMaven	1 Mes 6 días - #3	1 Mes 6 días - #2	3 Min 0 Seg

Icono: S M L Guía de iconos RSS para todos RSS para fallas RSS para los más recientes

Trabajos en la cola

No hay trabajos en la cola

Estado del ejecutor de construcciones

1 Inactivo  
2 Inactivo

Página generada: 27-ene-2017 18:32:29 CET REST API Jenkins ver. 2.37

Ilustración 29Jenkins

## Gestión del código fuente

Para gestionar el código y las versiones se ha usado Git junto con el repositorio de Github, al ser la herramienta sobre la que se asienta gran parte de la asignatura. El repositorio de nuestro proyecto es el siguiente: <https://github.com/Trabajo-EGC/Censo>

Hemos decidido usar Git bajo la consola de comandos ya que se vió de esta forma en las prácticas de la asignatura y creemos que de esta manera aprenderemos más en el uso de esta herramienta.

## Gestión de ramas

El uso de ramas es inevitable en el transcurso del proyecto ya que ayuda enormemente a separar el código que se encuentra en línea base del que no lo está.

### Gestión interna

#### *Creación de ramas*

En nuestro proyecto, una rama será creada cada vez que se vaya a implementar una nueva funcionalidad o se solucione un bug encontrado en el código. A la hora de crear una rama debemos de tener en cuenta el nombre que le vamos a dar, ya que este no puede estar repetido en el repositorio de Github y además debe ser representativo de la función que va a tener dicha rama.

#### *Merge de una rama*

Una vez que el código de la rama secundaria esté adecuadamente testeado y se compruebe que efectivamente funciona correctamente esa rama secundaria se unirá con la master mediante un merge.

Hay que tener especial cuidado en este paso ya que si no se ha comprobado adecuadamente que no existen bugs y se hace un merge con la rama master este error será trasladado a la rama principal y eso sería inaceptable.

## *Conflictos*

Cuando aparece un conflicto entre dos ramas el método a seguir es el descrito a continuación:

1. Localizar en que línea de código da conflicto.
2. Intentar arreglar el conflicto. Nunca debemos quedarnos con una de las dos posibilidades siempre tenemos que fusionar ambas.
3. Si no conseguimos arreglar el conflicto debemos de ponernos en contacto con el miembro del equipo que realizó ese código para que conjuntamente se vaya revisando lo realizado y lograr solucionar el conflicto.
4. Una vez que el conflicto esté solucionado procederemos a realizar el merge con la rama master.

Una imagen que ilustraría el uso de las ramas en nuestro proyecto sería la siguiente:

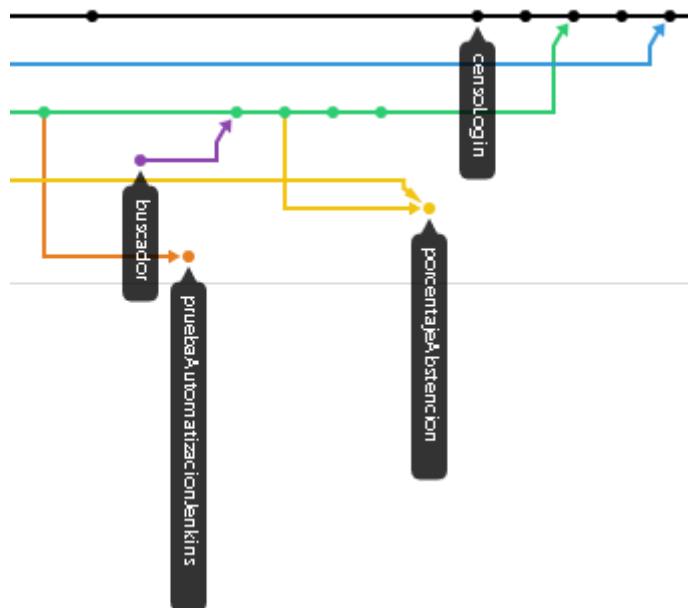


Ilustración 30Ramas de nuestro repo

Tal y como se puede observar, las ramas convergen en la rama master (en negrita) una vez que se ha implementado y testeado una nueva funcionalidad.

## Gestión externa

La gestión externa de las ramas la llevaron a cabo los distintos coordinadores de cada equipo de proyecto y se utilizó el siguiente repositorio:

<https://github.com/AgoraUS-G2-1617/G9>

## Roles en la gestión del código

Para la gestión interna del proyecto, todos los miembros del equipo cumplimos el mismo rol, que es el de desarrollador. Debido a esto, se establece una relación de igualdad en la que ningún miembro posee más autoridad que el resto.

Las ramas serán creadas por algún miembro que necesite hacer uso de ella para implementar una funcionalidad y no será borrada. Una vez que la rama cumpla con el objetivo que tenía al ser creada, se le aplicará un “merge” con la rama master. Previamente la funcionalidad debe haber sido testeada para asegurar su correcto funcionamiento y evitar problemas.

Para la gestión externa, sin embargo, si que hemos establecido una jerarquía, teniendo el rol de coordinador uno de los miembros. Dicho coordinador será el encargado de mantener actualizado el repositorio común de los diferentes grupos con la última versión de nuestro proyecto.

## Políticas de nombre y estilos

*“En este apartado detallaremos la sintaxis que se ha seguido a la hora de desarrollar el código para hacerlo lo más homogéneo posible.*

*Seguiremos los estándares de Java, usando el estilo “UpperCamelCase” para los nombres de las clases, y “lowerCamelCase” para los nombres de los métodos y las variables.*

*Por último, los nombres de los repositorios, servicios y controladores seguirán el patrón “NombreEntidad[Repository | Service | Controller]” respectivamente.”<sup>1</sup>*

(Documentación-censos-g2-1516-v2.00)

## Aceptación de cambios

Cuando un miembro del equipo quiera realizar un cambio en el proyecto deberá de seguir una serie de pasos que se detallan a continuación:

1. El miembro que quiere realizar un cambio deberá de crear un “Issues” en el repositorio de GitHub con el “Label” llamado “Propuesta”. En dicho “Issues” tendrá que detallar cual es el cambio que quiere realizar y el motivo por el cual piensa que es necesario para el desarrollo del proyecto.
2. Posteriormente, los miembros del grupo debatirán sobre el cambio y si es o no necesario.
3. Una vez tomada la decisión, si se ha rechazado el cambio el coordinador del proyecto dejará un comentario con el motivo y procederá a cerrar el “Issue”. Por el contrario, si se ha aceptado el coordinador dejará un comentario del motivo por el cuál se acepta y posteriormente creará un nuevo “Issue” para la realización de dicho cambio.

## Gestión de buenas prácticas en el código

### PMD

Para controlar que en nuestro proyecto se realizan buenas prácticas a la hora del desarrollo de nuevo código se ha optado por el uso del plugin PMD, el cuál esta configurado para que cada vez que se realice la fase de test revise todo el código del proyecto en busca de malas prácticas.

Cuando PMD descubre una mala práctica las reporta en un bloc de notas que generará en dentro de la carpeta “target” con el nombre de pmd.

## Instalación PMD

Debido a que se trata de un plugin lo único que debemos de hacer para su correcta “instalación” es añadir un pequeño fragmento de código xml en nuestro archivo “pom.xml” para que de esta manera maven lo descargue y sea capaz de utilizarlo.

El código que debemos de usar es el mostrado en la siguiente ilustración:

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-pmd-plugin</artifactId>
<version>3.7</version>
<executions>
    <execution>
        <id>Before testing</id>
        <phase>test</phase>
        <goals>
            <goal>pmd</goal>
        </goals>
    </execution>
</executions>
</plugin>
```

Ilustración 251pmdInstalacion

Una vez introducido el siguiente código el plugin de PMD se ejecutará en la fase de test tal y como hemos introducido en la línea “<phase>”.

Para ejecutarlo lo único que debemos de hacer es ejecutar la fase test de maven, para ello le daremos con el botón derecho al proyecto →Run as... → Maven test

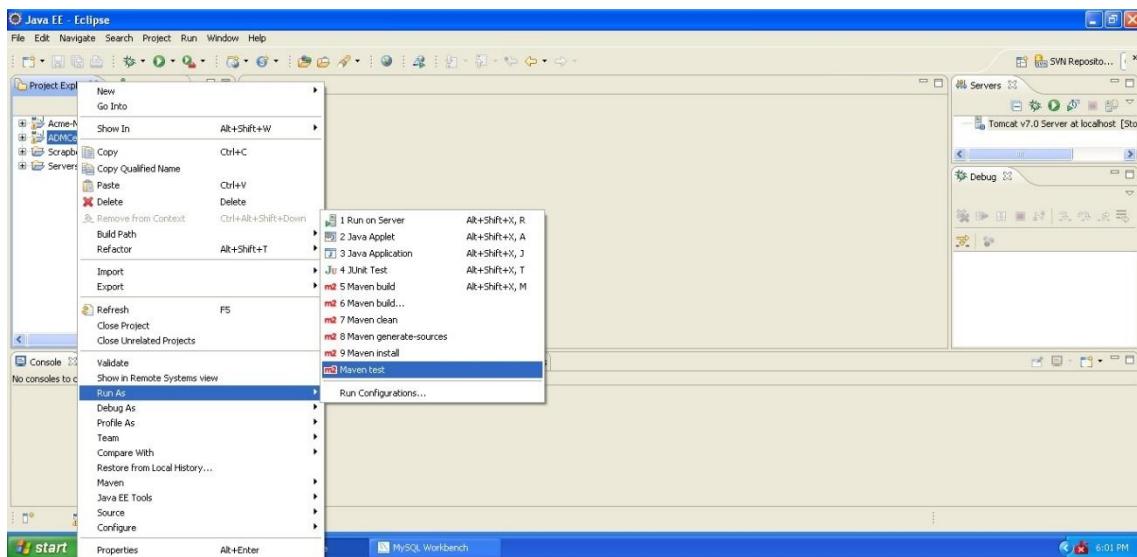


Ilustración 262Maven test

Una vez realizada esta acción maven comenzará a ejecutar los test y dado que PMD se ejecutará en esta fase dicho plugin analizará el código del proyecto.

Una vez completado los test nos iremos a “target” ya que pmd nos habrá creado un bloc de notas llamado “pmd” con todas las correcciones que se deben de realizar en el código.



```
<?xml version="1.0" encoding="UTF-8"?>
<pmd version="5.5.1" timestamp="2017-01-28T17:38:37.529">
<file name="Z:\EGC\Censo\src\main\java\controllers\CensusController.java">
<violation beginline="28" endline="28" begincolumn="1" endcolumn="47" rule="UnusedImports" ruleset="Import Statements" package="controllers" externalInfoUrl="https://p
Avoid unused imports such as 'org.eclipse.jetty.security.LoginService'
</violation>
</file>
<file name="Z:\EGC\Censo\src\main\java\domain\DomainEntity.java">
<violation beginline="74" endline="74" begincolumn="35" endcolumn="65" rule="UselessParentheses" ruleset="Unnecessary" package="domain" class="DomainEntity" method="eq
Useless parentheses.
</violation>
<violation beginline="78" endline="78" begincolumn="35" endcolumn="80" rule="UselessParentheses" ruleset="Unnecessary" package="domain" class="DomainEntity" method="eq
Useless parentheses.
</violation>
</file>
<file name="Z:\EGC\Censo\src\main\java\repositories\CensusRepository.java">
<violation beginline="20" endline="20" begincolumn="16" endcolumn="55" rule="UnusedModifier" ruleset="Unused Code" package="repositories" class="CensusRepository" meth
Avoid modifiers which are implied by the context
</violation>
<violation beginline="25" endline="25" begincolumn="16" endcolumn="71" rule="UnusedModifier" ruleset="Unused Code" package="repositories" class="CensusRepository" meth
Avoid modifiers which are implied by the context
</violation>
<violation beginline="30" endline="30" begincolumn="16" endcolumn="58" rule="UnusedModifier" ruleset="Unused Code" package="repositories" class="CensusRepository" meth
Avoid modifiers which are implied by the context
</violation>
<violation beginline="34" endline="34" begincolumn="16" endcolumn="66" rule="UnusedModifier" ruleset="Unused Code" package="repositories" class="CensusRepository" meth
Avoid modifiers which are implied by the context
</violation>
</file>
<file name="Z:\EGC\Censo\src\main\java\security\Authority.java">
<violation beginline="91" endline="91" begincolumn="35" endcolumn="96" rule="UselessParentheses" ruleset="Unnecessary" package="security" class="Authority" method="equ
Useless parentheses.
</violation>
</file>
<file name="Z:\EGC\Censo\src\main\java\services\CensusService.java">
<violation beginline="237" endline="239" begincolumn="33" endcolumn="33" rule="CollapsibleIfStatements" ruleset="Basic" package="services" class="CensusService" method

```

Ilustración 273Bloc de notas pmd

Como podemos observar pmd nos dice en que archivo se ha detectado el problema, en que línea ha sido y el tipo de regla que no se ha cumplido.

## Ejercicio propuesto para el uso de PMD

En este ejercicio vamos a corregir una línea de código que PMD nos dice que no cumple con alguna regla.

La línea de código que hemos introducido es una importación que no es usada, dicha línea de código se encuentra en la clase “WelcomeController.java”.

The screenshot shows the Eclipse IDE interface with a Java file named 'WelcomeController.java' open. The code is as follows:

```
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import domain.Census;
import domain.DomainEntity;

import services.CensusService;

@Controller
@RequestMapping("/welcome")
public class WelcomeController extends AbstractController {

    // Constructors -----
    public WelcomeController() {
        super();
    }

    // Constructors -----
    @Autowired
    private CensusService censusService;
}
```

A red squiggly underline is under the import statement 'import domain.DomainEntity;'. The status bar at the bottom indicates 'Writable'.

Ilustración 284 Línea de código errónea

Ahora lo que debemos de hacer es ejecutar los test con ayuda de maven tal y como se hizo en la sección anterior.

Tras la ejecución pmd nos creará un bloc de notas en el cuál aparecerá dicha línea de código.

The screenshot shows a Microsoft Word document titled 'pmd: Bloc de notas'. The menu bar includes Archivo, Edición, Formato, Ver, Ayuda. The content of the document is a PMD XML report:

```
<?xml version="1.0" encoding="UTF-8"?>
<pmd version="5.5.1" timestamp="2017-01-28T18:21:21.591">
<file name="Z:\EGC\Censo\src\main\java\controllers>WelcomeController.java">
<violation beginline="15" endline="15" begincolumn="1" endcolumn="27" rule="UnusedImports" rule="Avoid unused imports such as 'domain.DomainEntity'">
</violation>
</file>
</pmd>
```

Ilustración 295 Reporte pmd

Tal y como podemos observar en el bloc de notas pmd nos dice que esa línea de código puede ser borrada ya que nos da la regla “rule=”UnusuedImports” ”

Una vez eliminada esa línea de código, si ejecutamos de nuevo los tests pmd nos seguirá generando el bloc de notas, pero esta vez no habrá ninguna violació de ninguna regla.

## Lecciones aprendidas

### *Gestión de la calidad del código fuente*

Gracias al uso de este plugin el código desarrollado por el equipo es mucho mas limpio y será más fácil de comprender en un futuro.

Hemos aprendido a usar una herramienta de análisis de código lo cuál ayudará enormemente en futuros desarrollos y hará que nuestro código sea mas legible y fácil de comprender para otras personas.

### *Gestión de ramas internas*

Como se dijo anteriormente, el uso de las ramas tiene como obejtivo la creación y testeo de nuevas funcionalidades de forma que una vez incorporadas con la rama master, no existan problemas. Sin embargo, hay casos en los que no se ha hecho uso de las ramas y se ha incorporado un determinado cambio directamente a la rama master, como por ejemplo comentar algún método o borrar un importe no usado.

## Ejericio: Creación de una nueva rama y resolución de un conflicto.

En este ejercicio vamos a crear una rama en la que haremos una pequeña modificación de nuestro código para posteriormente hacer un merge con la rama master y corregir el conflicto

Para ello, debemos crear primero una nueva rama llamada nuevoCambio. Para crear una rama y pasarnos de master a dicha rama, usaremos el comando “git checkout -b nuevoCambio”

```
C:\Temp\UM\Censo>git checkout -b nuevoCambio
Switched to a new branch 'nuevoCambio'
```

Ilustración 306 Creación de una nueva rama

Una vez hecho esto, procedemos a implementar la funcionalidad para la que estuviera destinada dicha rama y a testear que funcione. Hay que tener en cuenta que cualquiera de los otros miembros puede estar implementando también otra funcionalidad, por lo que, si combina su rama con la rama master y modifica alguna parte del código que influya en el nuestro, nos provocará un conflicto al hacer nosotros el merge con la rama master.

Si la nueva funcionalidad cumple el test, pasaremos dicha funcionalidad a la rama master mediante un merge. Para ello, debemos hacer uso primero del comando “git add \*”) seguido del comando “git commit -m “nueva funcionalidad (ejerc. Pract.)”.

```
C:\Temp\UM\Censo>git add *
C:\Temp\UM\Censo>git status
On branch nuevoCambio
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   src/test/java/services/CensusServiceTest.java

C:\Temp\UM\Censo>git commit -m "nueva funcionalidad (ejerc. pract.)
[nuevoCambio d1d130] nueva funcionalidad (ejerc. pract.)
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Ilustración 317 Commit de la nueva funcionalidad

Una vez hecho esto, volvemos a la rama master y ejecutamos el comando “git merge nuevoCambio”.

```
C:\Temp\UM\Censo>git merge nuevoCambio
Auto-merging src/test/java/services/CensusServiceTest.java
CONFLICT (content): Merge conflict in src/test/java/services/CensusServiceTest.java
Automatic merge failed; fix conflicts and then commit the result.
C:\Temp\UM\Censo>
```

Ilustración 38 Conflicto en el merge

Como se observa en la ilustración, se ha producido un conflicto debido a que un miembro del equipo modificó algún parte del código que nosotros también usábamos de la rama master. Para solucionar este conflicto, abrirímos el archivo que lo provoca y git nos señalará en donde se encuentra el error

```

    @Test
    public void mostPopularCensus(){

<<<< HEAD
        /* Inicializamos una variable como tipo Map<Census, Integer> y almacenamos en ella
           el resultado del método mostPopularCensus(), que debe dar 10. */
=====

        /* Inicializamos una variable como tipo Census y almacenamos en ella
           el resultado del método mostPopularCensus(), que debe dar 10. */
>>>> nuevoCambio

        Map<Census, Integer> result;

        result = censusService.mostPopularCensus();
        Assert.isTrue(result.size()==10);
    }

```

*Ilustración 39 Razón del conflicto*

Una vez que resolvamos dicho conflicto y comprobemos el correcto funcionamiento del sistema, volveremos a usar el comando merge seguido de un push a nuestro repositorio.

## Gestión de la construcción e integración continua

### Gestión de la construcción

Para realizar la construcción de toda la estructura de nuestro proyecto hemos usado el IDE de Eclipse con un plugin de Maven, el cuál nos facilita mucho las cosas a la hora de construir la estructura nuestro subsistema.

*“Maven es una herramienta de software para la gestión y construcción de proyectos Java, además Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.”<sup>2</sup>(Wikipedia)*

A la hora de guardar los datos necesarios usamos una base de datos de MySQL, ya que nos es de pago y es la que el grupo del año pasado eligió. Además, a la hora de interactuar con la base de datos usamos MySQL Workbench, que nos ofrece una interfaz gráfica con la cuál podremos crear nuestra base de datos sin mucha complicación.

## Ejercicio gestión de la construcción con maven

En este ejercicio veremos como añadir una nueva dependencia a maven y comprobar que dicha dependencia se ha instalado correctamente.

La dependencia que hemos optado por introducir en nuestro proyecto será la de “Wiremock” y lo haremos de la siguiente forma:

Lo primero que debemos de hacer es irnos al archivo que maven utiliza para describir sus dependencias, dicho archivo es el “pom.xml” y lo podemos encontrar en la raíz de nuestro proyecto.

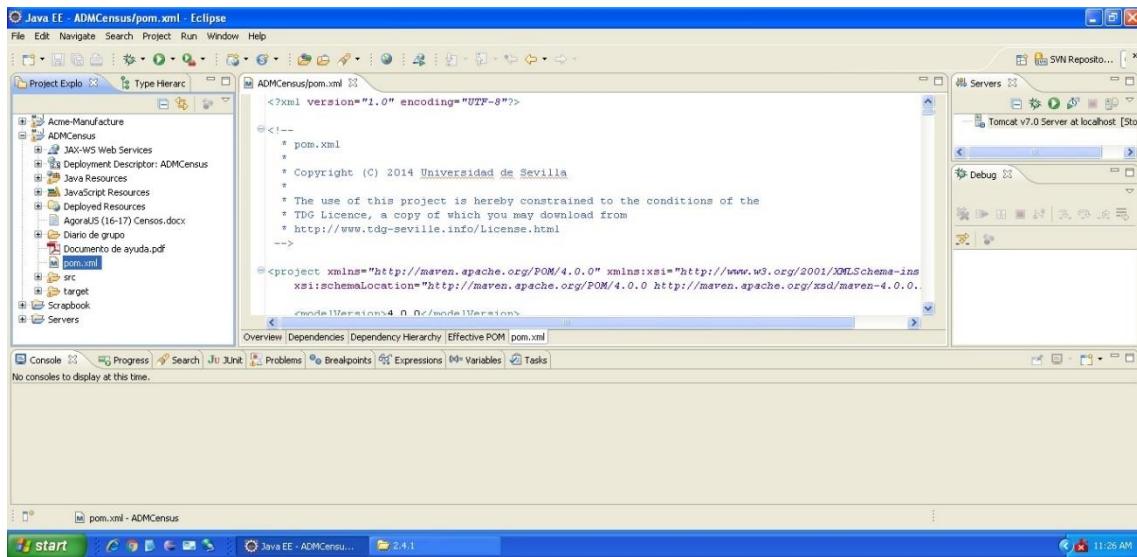


Ilustración 40Pom del maven

Una vez localizado donde se encuentra nuestro archivo “pom.xml” lo abriremos y nos encontraremos con un xml en el cuál podemos encontrar desde las dependencias hasta los repositorios que queremos que maven use.

Ahora procederemos a introducir nuestra dependencia introduciendo el código que se muestra en la figura siguiente en nuestro archivo “pom.xml”

```
<!-- Dependencias 2016-2017 -->
<!-- Wiremock -->
<dependency>
    <groupId>com.github.tomakehurst</groupId>
    <artifactId>wiremock</artifactId>
    <version>2.4.1</version>
</dependency>
```

Ilustración 321Dependencia maven

Con el siguiente código xml le decimo a maven que nos descargue la dependencia de wiremock cuya versión sea la especificada entro los tags “<versión></versión>” que en este caso en particular es la 2.4.1.

Una vez realizado este paso, procederemos a guardar los cambios y maven comenzará a construir de nuevo el proyecto añadiendo la nueva dependencia que le hemos introducido en su archivo “pom.xml”.

Para comprobar que se ha descargado correctamente debemos de irnos a la siguiente ruta “C:\Documents and Settings\Student\.m2\repository\com\github\tomakehurst\wiremock\2.4.1”, en dicha ruta encontraremos los archivos que maven se ha descargado tal y como podemos observar en la siguiente imagen.

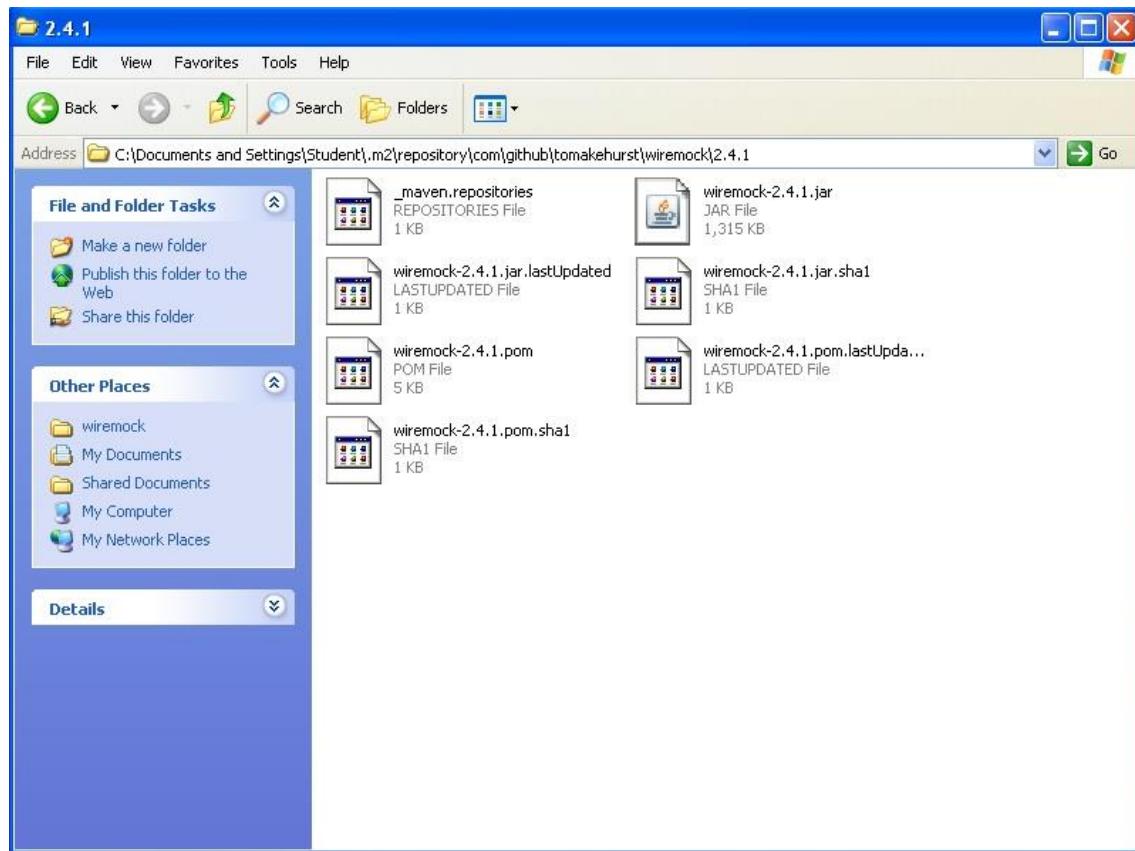


Ilustración 33 Archivos descargados por maven

Se debe de aclarar que maven no se descargará todas las dependencias en el mismo lugar, ya que dependiendo de la dependencia esta será descargada en un sitio u otro, pero siempre dentro de la carpeta “repository”.

## Lecciones aprendidas

Con la realización de este sencillo ejercicio hemos aprendido a introducir una nueva dependencia en Maven, a pesar de que solo hemos visto el caso de introducir una

dependencia si queremos introducir un nuevo plugin por ejemplo se haría de la misma forma.

Todos los cambios que queramos hacer sobre la configuración que tiene maven se puede realizar en el archivo “pom.xml” ya sean dependencias, plugins, deploy etc.

## Gestión de la integración continua

Para nuestro caso, para la realización de la integración continua hemos hecho uso de la herramienta Jenkins, ya que es sencilla de usar y nos da muchas opciones para la realización de diversas tareas.

La instalación de Jenkins viene recogida en el apartado “[Instalación Jenkins](#)” por lo que procederemos directamente a ver el ejercicio propuesto para su configuración.

### Ejercicio de integración continua con Jenkins

En este ejercicio vamos a ver como configurar adecuadamente Jenkins para que se ejecuten las pruebas de nuestro proyecto.

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.

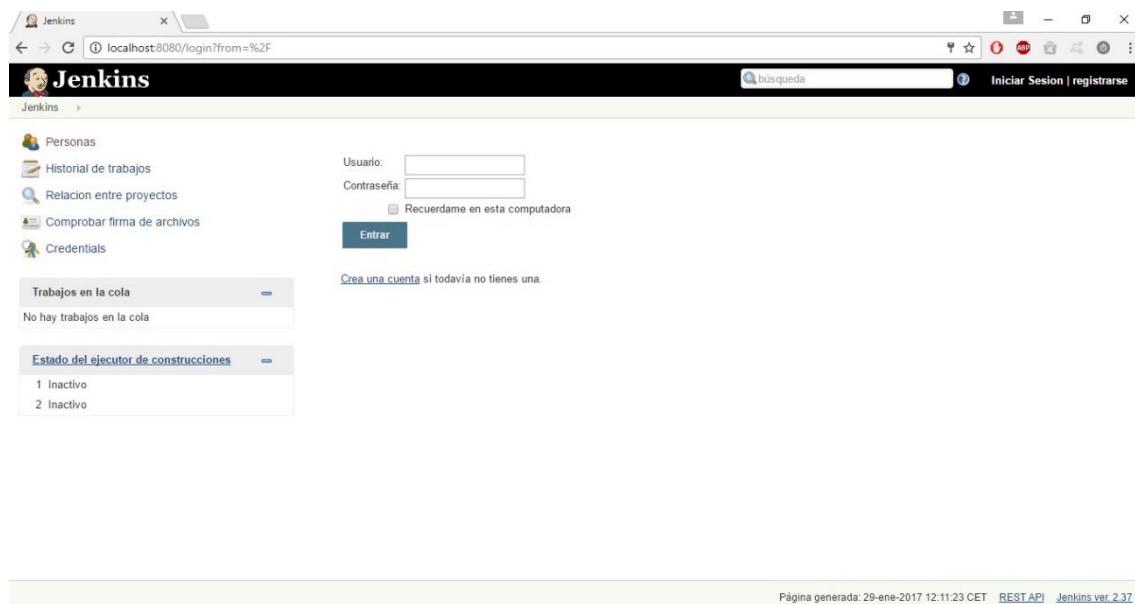


Ilustración 343Login de jenkins

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.

The screenshot shows the Jenkins main dashboard. On the left, there's a sidebar with links like 'Nueva Tarea', 'Personas', 'Historial de trabajos', etc. The main area has a table titled 'Todos' with columns: S, W, Nombre, Último Éxito, Último Fallo, and Última Duración. Three projects are listed:

S	W	Nombre	Último Éxito	Último Fallo	Última Duración
		Censo	15 Hor - #36	18 días - #30	19 Min
		Tarea1	1 Mes 8 días - #2	N/D	1.9 Seg
		TareaMaven	1 Mes 8 días - #3	1 Mes 8 días - #2	3 Min 0 Seg

Below the table, there are links for 'Guía de íconos', 'RSS para todos', 'RSS para fallas', and 'RSS para los más recientes'. At the bottom right, it says 'Página generada: 29-ene-2017 12:17:52 CET REST API Jenkins ver.2.37'.

Ilustración 354Panel principal de jenkins

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

The screenshot shows the Jenkins administration interface. On the left, there's a sidebar with links like 'Nueva Tarea', 'Personas', 'Historial de trabajos', etc. The main area is titled 'Administrar Jenkins' and lists several configuration options:

- Hay una nueva versión de Jenkins disponible (2.42). [descargar \(listado de cambios\)](#). [O actualizar automáticamente](#)
- Configurar el Sistema**: Configurar variables globales y rutas.
- Configuración global de la seguridad**: Seguridad en Jenkins. Define quién tiene acceso al sistema (autenticación) y qué puede hacer (autorización).
- Configure Credentials**: Configure the credential providers and types.
- Global Tool Configuration**: Configure tools, their locations and automatic installers.
- Actualizar configuración desde el disco duro**: Descartar todos los datos cargados en memoria y actualizar todo nuevamente desde los ficheros del sistema. Útil cuando se modifican ficheros de configuración directamente en el disco duro.
- Administrador Plugins**: Añadir, borrar, desactivar y activar plugins que extienden la funcionalidad de Jenkins. [Existen actualizaciones disponibles](#).
- Información del sistema**: Muestra información del entorno que puedan ayudar a la solución de problemas.
- System Log**: El log del sistema captura la salida de la clase java.util.logging en todo lo relacionado con Jenkins.
- Estadísticas de Carga**: Comprobar la utilización de los recursos y comprobar si es necesario añadir nuevos nodos para la ejecución de tareas.

Ilustración 365Administracion de Jenkins

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

Activados	Nombre	Versión	Versión previamente instalada.	marcado	Desinstalar
<input checked="" type="checkbox"/>	bouncycastle API Plugin	2.16.0			<button>Desinstalar</button>
<input checked="" type="checkbox"/>	Javadoc Plugin	1.4			<button>Desinstalar</button>
<input checked="" type="checkbox"/>	JUnit Plugin	1.19			<button>Desinstalar</button>
<input checked="" type="checkbox"/>	Mailer Plugin	1.18			<button>Desinstalar</button>
<input checked="" type="checkbox"/>	Maven Integration plugin	2.14			<button>Desinstalar</button>
<input checked="" type="checkbox"/>	Token Macro Plugin	2.0			<button>Desinstalar</button>

Página generada: 29-ene-2017 12:31:01 CET [REST API](#) [Jenkins ver. 2.37](#)

Ilustración 376Instalar plugin Jenkins

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.

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 cual configuraremos tal y como se muestra en la figura siguiente:

Nombre	Instalar automáticamente	Instalar desde Apache
Maven	<input checked="" type="checkbox"/>	Versión 3.3.9

Añadir Maven      Borrar Maven

Página generada: 29-ene-2017 12:36:49 CET [Jenkins ver. 2.37](#)

Ilustración 387Configurar 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 esta 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:

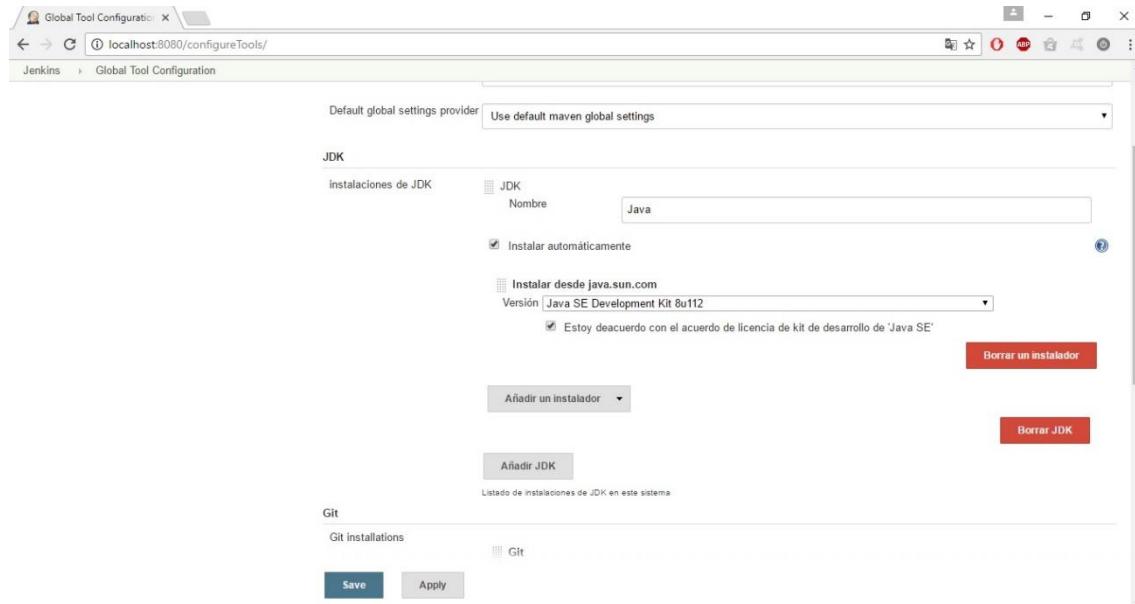


Ilustración 48Configurar jdk en jenkins

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.

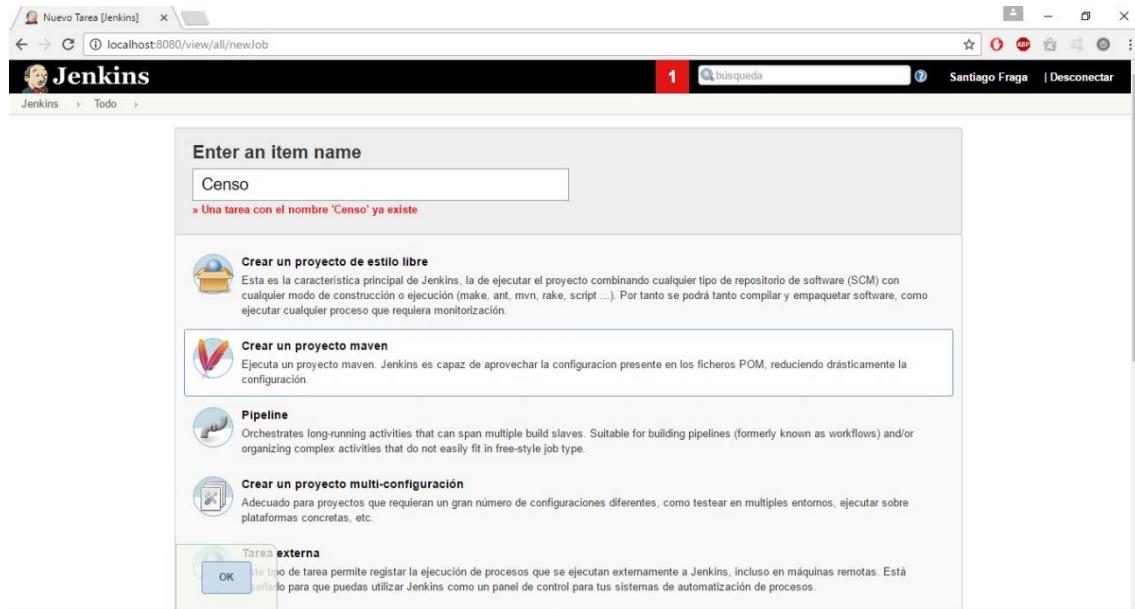


Ilustración 4939Crear tarea en Jenkins

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:

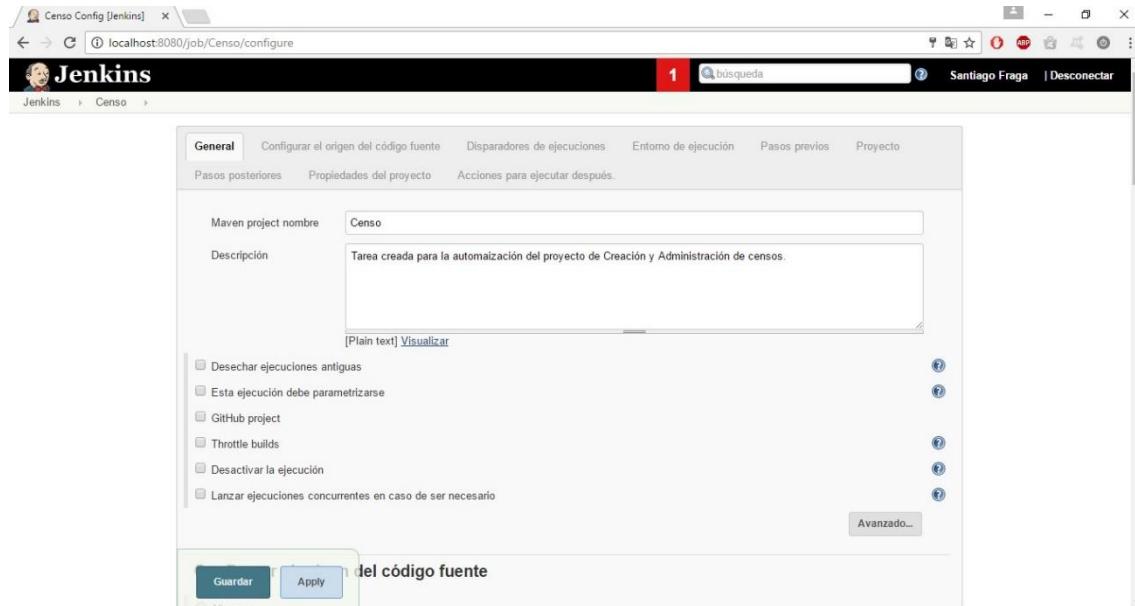


Ilustración 5040Configurar tarea en jenkins 1

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.

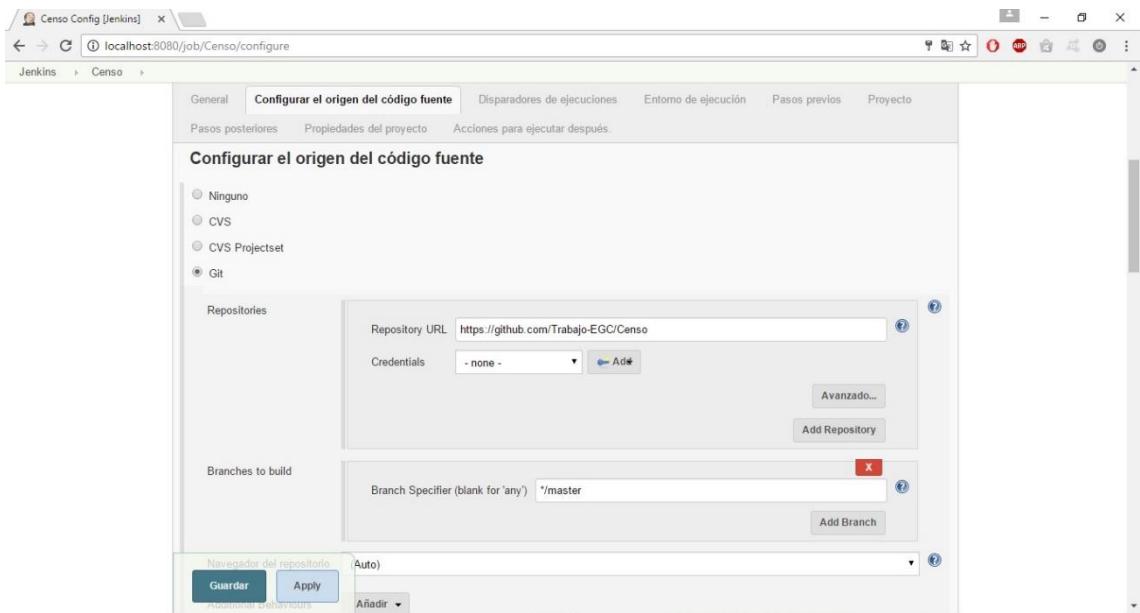


Ilustración 421Configurar origen del código en Jenkins

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.

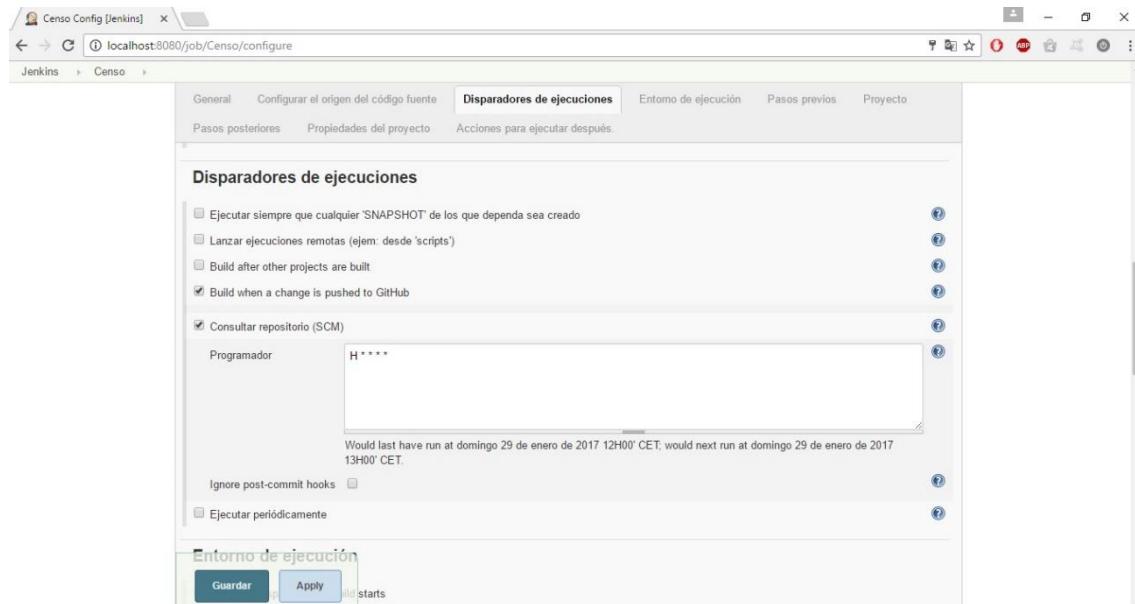


Ilustración 412Configurar disparadores en Jenkins

Despues 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:

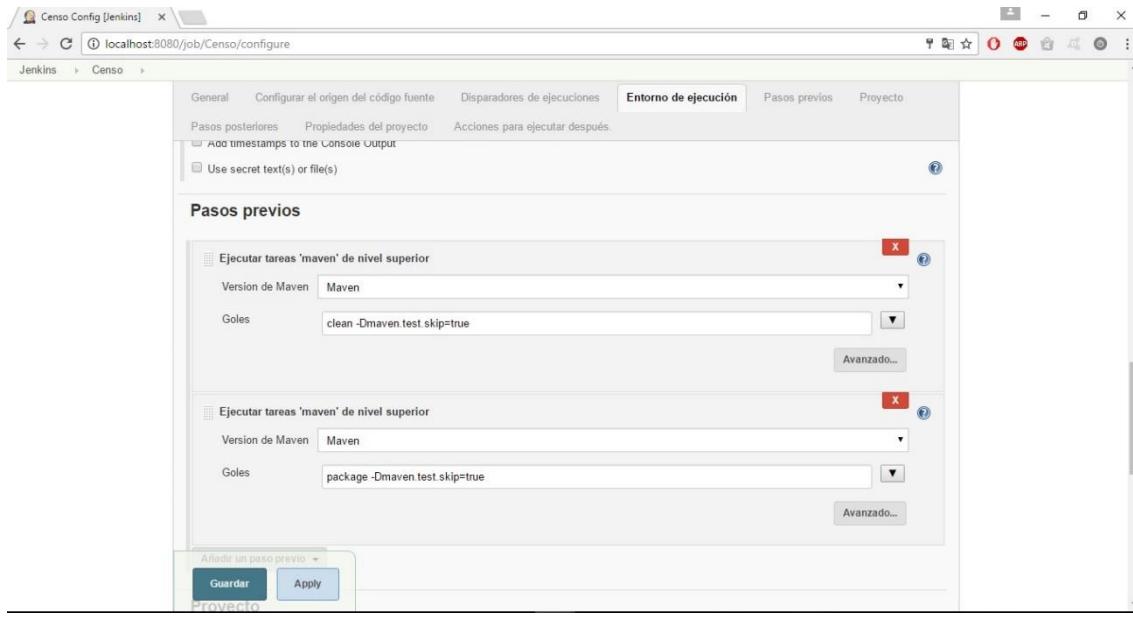


Ilustración 433Configurar pasos previos en Jenkins

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:

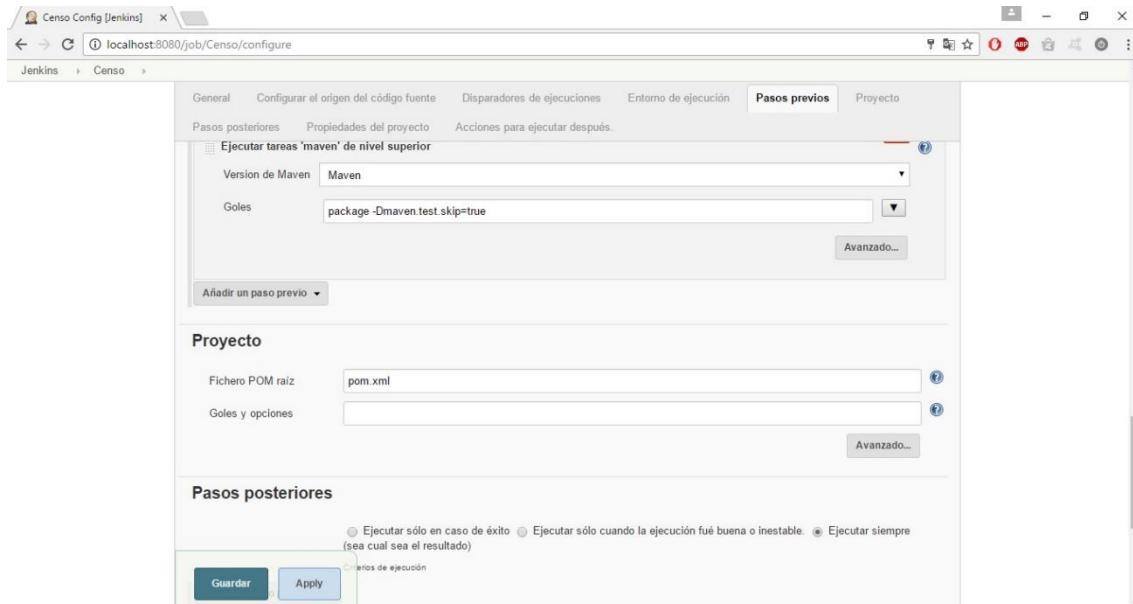


Ilustración 444Configurar proyecto en Jenkins

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

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

## Lecciones aprendidas

Gracias al uso de Jenkins hemos podido aprender un poco más sobre la integración continua, así como automatizar ciertas tareas que si las realizásemos cada vez que se hiciera un cambio serían bastante tedioso.

# Gestión del cambio, incidencias y depuración

## Control del cambio

*“Los cambios provocan un aumento del grado de confusión entre los ingenieros de software que trabajan en un proyecto. La confusión surge cuando:*

- *No se analiza el impacto de llevar a cabo un cambio antes de realizarlo.*
- *No se registran los cambios antes de implementarlos.*
- *No se informa a quienes deben conocer dichos cambios.*
- *No se aplica un control sobre los cambios con el fin de que mejore la calidad del proyecto y se reduzcan sus errores.”<sup>1</sup>(Documentación-censos-g2-1516-v2.00)*

A la hora de implementar nuevas funcionalidades o hacer modificaciones en el código heredado, el equipo de proyecto analizó dicho código heredado y posteriormente se acordaron cuales serian las nuevas funcionalidades o cambios a implementar. De esta forma se evitarían las situaciones enunciadas anteriormente. Las nuevas funcionalidades implementadas o cambios mas signifacitivos (detalladas en el apartado “nuevas funcionalidades”) y sus responsables son las siguientes:

- Buscador de censo por título (Pablo Romero Vázquez).
- Obtención de censos más participativos (Ruben Barrientos Mohedano).
- Obtención de los censos por orden descendiente de fecha fin (Santiago Fraga Martín-Arroyo).
- Mejora del css (Nicolás Lorenz Rosado)
- Porcentaje de abstención en un censo (Simon Egea Guerrero).

Es posible que durante o tras el desarrollo de una funcionalidad surjan nuevas ideas de mejora o bien modificaciones a dicha funcionalidad para adaptarla a otros objetivos, por lo que es conveniente usar una herramienta para comunicar dichos cambios a todo el equipo. En nuestro caso hemos usado el apartado “issues” que nos proporciona GitHub.

La decisión de usar GitHub es porque dicha herramienta notifica a los usuarios por correo cada vez que se crea o modifica una issue, permitiendo a los integrantes del equipo de proyecto estar siempre informados sobre este aspecto. Además, tanto nuestro subsistema como los subsistemas del resto de grupos usan como repositorio GitHub, por lo que al usar el apartado de issues, permitimos seguir los cambios introducidos de una forma más cómoda.

Las incidencias seguirán el siguiente esquema:

- Un título. Dicho título describirá el problema.
- Una descripción.
- Una etiqueta (label).

De esta forma, si se quiere realizar un cambio en el subsistema, el procedimiento a seguir por el miembro que propone dicho cambio será crear un issue en el que se plantee al resto del equipo implementar dicho cambio en el subsistema y lo asignará a un miembro del equipo para que lo realice. Una vez hecho esto, el responsable al que se le designó dicho trabajo será el que lo implemente siempre y cuando se pruebe que dicha implementación es factible. En el momento en el que se empiece a trabajar en la implementación de la incidencia, se deberá etiquetar dicha incidencia como “En curso”. Cuando se solucione dicha incidencia, se deberá de etiquetar como “Revisar” para que un miembro distinto al que la implementó, la revise y en caso estar correctamente implementada, pasaría a tener la etiqueta “closed”.

La evolución de la implementación se puede llevar a cabo mediante los comentarios en la misma issue. Una vez que la issue ha sido resuelta, se procederá a subir los cambios al repositorio mediante una rama diferente a la de master, para posteriormente unificarla con la rama master.

## Ejercicio: Implementación adecuada para la gestión del cambio.

Las mejoras registradas se muestran en la siguiente imagen:

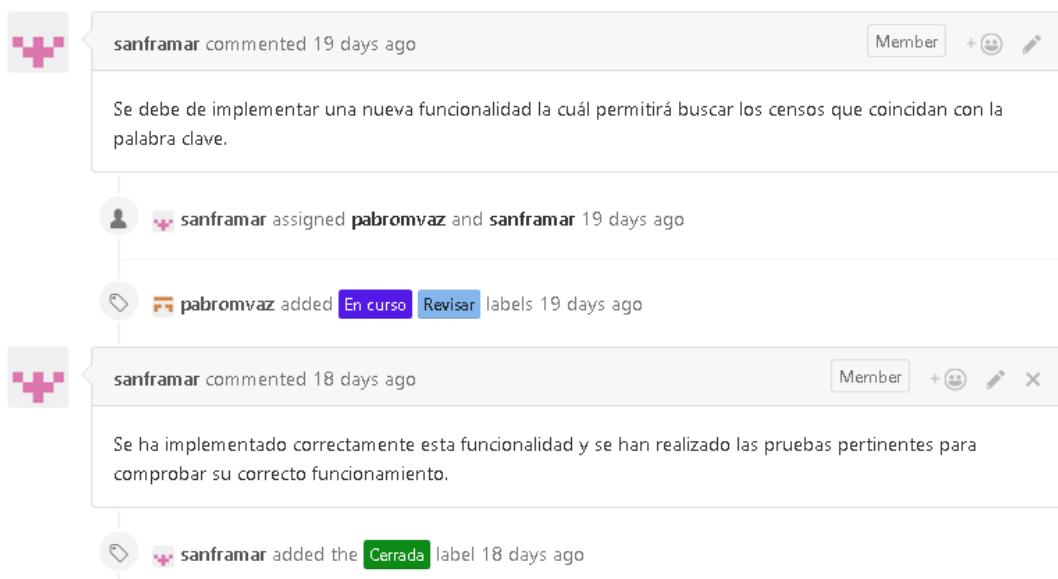
<input type="checkbox"/> Clear current search query, filters, and sorts				
	Author	Labels	Milestones	Assignee
Sort ▾				
<input type="checkbox"/> ① 0 Open <input checked="" type="checkbox"/> 4 Closed				
<input type="checkbox"/> ⓘ Buscador de censos por título <span style="background-color: #28a745; color: white; padding: 2px 5px;">Cerrada</span> Mejora #19 by sanframmar was closed 18 days ago				<span>3</span>
<input type="checkbox"/> ⓘ Mostrar los censos con mas participación <span style="background-color: #28a745; color: white; padding: 2px 5px;">Cerrada</span> Mejora #11 by simegegue was closed 18 days ago				<span>1</span>
<input type="checkbox"/> ⓘ Mostrar el porcentaje de abstenciones en un censo. <span style="background-color: #28a745; color: white; padding: 2px 5px;">Cerrada</span> Mejora #10 by simegegue was closed 18 days ago				<span>1</span>
<input type="checkbox"/> ⓘ Mostrar ultimas votaciones <span style="background-color: #28a745; color: white; padding: 2px 5px;">Cerrada</span> Mejora #1 by sanframmar was closed on 4 Dec 2016				<span>1</span>

Ilustración 466 Mejoras registradas

## MEJORA #19: Buscador de censos por título

### Buscador de censos por título #19

 **Closed** sanframar opened this issue 19 days ago · 3 comments



sanframar commented 19 days ago

Se debe de implementar una nueva funcionalidad la cuál permitirá buscar los censos que coincidan con la palabra clave.

Member + 

sanframar assigned pabromvaz and sanframar 19 days ago

pabromvaz added **En curso** **Revisar** labels 19 days ago

sanframar commented 18 days ago

Se ha implementado correctamente esta funcionalidad y se han realizado las pruebas pertinentes para comprobar su correcto funcionamiento.

Member +  

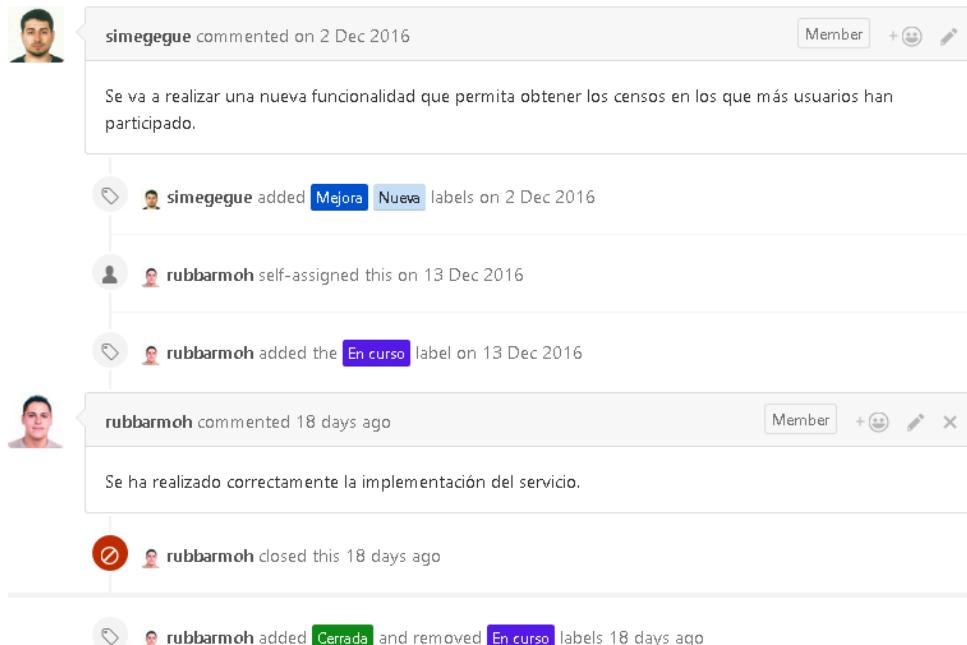
sanframar added the **Cerrada** label 18 days ago

*Ilustración 477 Mejora #19*

## MEJORA #11: Mostrar censos con más participación.

### Mostrar los censos con mas participación #11

 **Closed** simegue opened this issue on 2 Dec 2016 · 1 comment



simegue commented on 2 Dec 2016

Se va a realizar una nueva funcionalidad que permita obtener los censos en los que más usuarios han participado.

Member + 

simegue added **Mejora** **Nueva** labels on 2 Dec 2016

rubbarmoh self-assigned this on 13 Dec 2016

rubbarmoh added the **En curso** label on 13 Dec 2016

rubbarmoh commented 18 days ago

Se ha realizado correctamente la implementación del servicio.

Member +  

rubbarmoh closed this 18 days ago

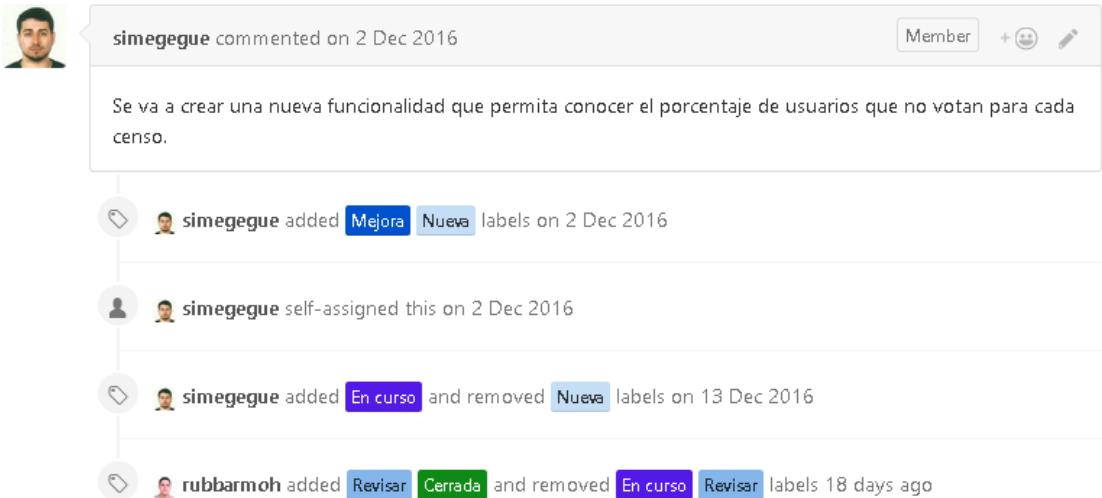
rubbarmoh added **Cerrada** and removed **En curso** labels 18 days ago

*Ilustración 5848 Mejora #11*

MEJORA # 10: Mostrar porcentaje de abstenciones.

## Mostrar el porcentaje de abstenciones en un censo. #10

 **Closed** simegegue opened this issue on 2 Dec 2016 · 1 comment



simegegue commented on 2 Dec 2016

Se va a crear una nueva funcionalidad que permita conocer el porcentaje de usuarios que no votan para cada censo.

simegegue added **Mejora** **Nueva** labels on 2 Dec 2016

simegegue self-assigned this on 2 Dec 2016

simegegue added **En curso** and removed **Nueva** labels on 13 Dec 2016

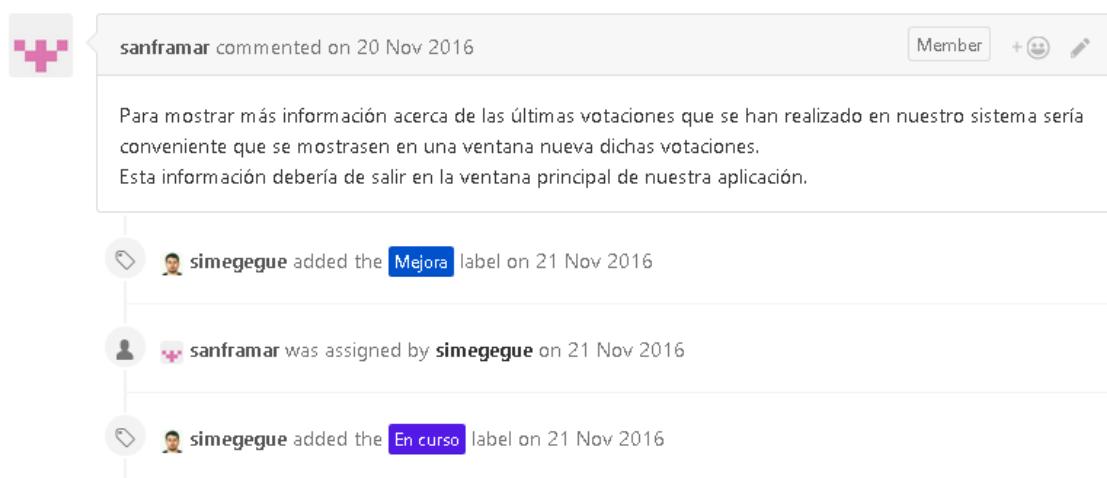
rubbarmoh added **Revisar** **Cerrada** and removed **En curso** **Revisar** labels 18 days ago

Ilustración 59 Mejora #10

MEJORA #1: Mostrar últimas votaciones. Se encuentra en curso

## Mostrar ultimas votaciones #1

 **Closed** sanframar opened this issue on 20 Nov 2016 · 1 comment



sanframar commented on 20 Nov 2016

Para mostrar más información acerca de las últimas votaciones que se han realizado en nuestro sistema sería conveniente que se mostrasen en una ventana nueva dichas votaciones.  
Esta información debería de salir en la ventana principal de nuestra aplicación.

simegegue added the **Mejora** label on 21 Nov 2016

sanframar was assigned by simegegue on 21 Nov 2016

simegegue added the **En curso** label on 21 Nov 2016

Ilustración 6049 Mejora #1

La secuencia que hay que seguir en el etiquetado se exponndrá en el siguiente punto.

## Gestión de incidencias

### Reportes de incidencias con GitHub

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:

The screenshot shows the GitHub interface for the repository 'Trabajo-EGC / Censo'. At the top, there are buttons for 'Code', 'Issues 4', 'Pull requests 0', 'Projects 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. Below this is a navigation bar with 'Filters', a search bar containing 'is:issue is:open', and buttons for 'Labels' and 'Milestones'. A green 'New issue' button is on the right. The main area displays a list of issues:

- 4 Open ✓ 21 Closed
- ① Despliegue de aplicación En curso #25 opened a day ago by rubbarmoh
- ① Mejora de la documentación En curso #23 opened 4 days ago by sanframmar
- ① Creación de tests Cerrada #22 opened 4 days ago by sanframmar
- ① Incidencias Incidencia #2 opened on 21 Nov 2016 by sanframmar

Ilustración 501 Incidencia

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”.

### Mejora de la documentación #23

① Open sanframmar opened this issue 4 days ago · 0 comments

The screenshot shows the GitHub issue page for issue #23, titled 'Mejora de la documentación'. The issue was opened by 'sanframmar' 4 days ago. A comment from 'sanframmar' is displayed:

sanframmar commented 4 days ago

Se debe de mejorar la documentación ya que es demasiado pobre. Para ello se deberá de seguir el esquema que el profesor nos ha facilitado. [https://1984.lsi.us.es/wiki-egc/index.php/Comentarios\\_para\\_mejorar\\_los\\_trabajos\\_16-17](https://1984.lsi.us.es/wiki-egc/index.php/Comentarios_para_mejorar_los_trabajos_16-17)

Below the comment, there are several actions taken by 'sanframmar':

- Added the 'Nueva' label 4 days ago
- Assigned 'pabromvaz' and 'sanframmar' 4 days ago

Ilustración 512 Incidencia #23

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

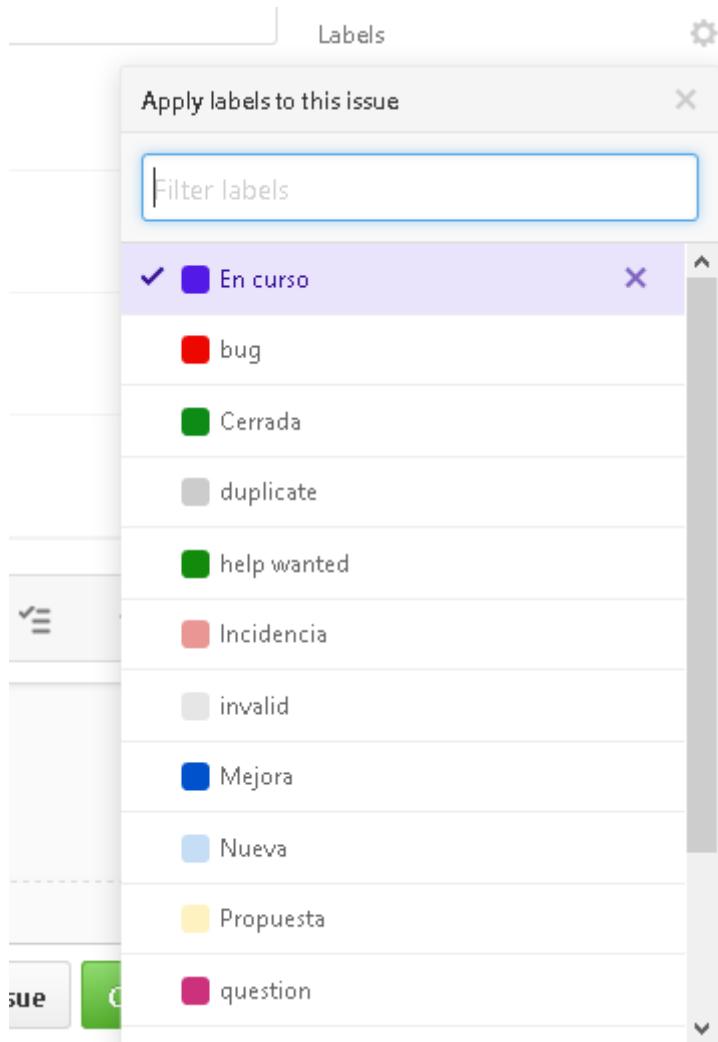


Ilustración 523 Labels de una incidencia

En el apartado “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 sería 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.

## Política de gestión de incidencias

Hemos acordado que todos los miembros del equipo podrán crear y asignar incidencias. La persona a la que se le asigne una incidencia será la encargada de implementarla, el resto podrán revisar su correcta implementación.

## Lecciones aprendidas

En el tema de las incidencias, se puede observar que, a la hora de asignar etiquetas, algunas veces no se ha seguido el esquema, por ejemplo, se pasa de “en curso” a “cerrada” saltándose el estado de “revisada”. Esto se debe a que generalmente una vez implementado el cambio, este se testeaba antes de dejarlo indicado en GitHub y solo se informaba de su correcto funcionamiento al etiquetarla como “cerrada”. Una posible solución es haber asignado a un miembro como encargado de las issues.

## Ejercicio: Resolución de una incidencia

En la incidencia #1 recoge un ejemplo de un bug encontrado al implementar una nueva funcionalidad. Dicho bug provocaba que se mmostrasen las fechas de inicio como las fechas de fin de votación y viceversa. Este bug lo detectó Simón Egea tras etiquetar Santiago esta incidencia con la etiqueta “Revisar”. Al detectar dicho bug, Simón etiquetó la incidencia como “bug” y explicó dicho bug. Una vez resuelto el bug, la incidencia fue etiquetada como “cerrada”, cerrándose así dicha incidencia.

The screenshot shows a GitHub issue thread for a project named 'Trabajo-EGC/Censo'. The issue number is 1. The main message discusses a bug related to date display. It includes several comments from users 'simegue' and 'sanframar' and adds labels like 'Mejora', 'En curso', 'Revisar', and 'bug'. The issue was closed by 'sanframar' on December 4, 2016. The right sidebar displays project details, milestones, notifications, and participant information.

Ilustración 534 Bug

## Lecciones aprendidas

Dado que el bug anterior fue detectado antes de realizar un merge con la rama master, el bug no se propagó a la rama principal, quedando el proyecto principal intacto.

## Ejercicio: Cómo gestionar una incidencia

Durante la realización del proyecto, se detectó también otro bug que sí que se propagó a la rama master al implementar la funcionalidad del buscador de censos tras hacer el merge, siendo detectado el error al mandar mensajes de error Jenkins y no Git. El error pudo ser resuelto quedando constancia en el apartado issue de GitHub.

## Depuración con Eclipse

*"Es el proceso de encontrar e intentar reducir y solventar los errores o defectos que el software pueda contener. Se llevará a cabo en la fase "**Empezada**" de la incidencia donde se hará el diagnóstico de la misma.*

*¿Por qué la depuración?*

*Muchas veces el programa no funcionará tal y como se espera. Por ello, será de utilidad herramientas de depuración potentes que nos permita ejecutar operaciones tales como realizar una ejecución paso a paso, modificar el valor de algunos campos, parámetros de métodos y variables locales.*

*Para poder realizar la depuración, en primer lugar, tendremos que colocar dos breakpoints, o puntos de ruptura, los cuales indican el comienzo y el fin del método a depurar. Acto seguido, haremos clic sobre el botón de depuración (.). Tras esto, comenzaremos a realizar operaciones en nuestro sistema de información, independientemente de Eclipse, de manera que, cuando intentemos realizar la acción que nos da fallo, Eclipse parpadeará con un color anaranjado y se nos abrirá la vista de depuración. Una vez en ella, podremos utilizar las teclas F5, F6 y F7, para ir revisando el método línea a línea, condición a condición o método a método, si entre los breakpoints hubiera más de un método implicado, consecutivamente.*

*Con esto llegaremos a un punto en el que al comprobar la variable que nos da error, puede ser porque se encuentre nula, o directamente nuestro método no hace que esa variable obtenga un valor esperado, por lo que podremos dar con el fallo, solucionándolo y corrigiéndolo."<sup>1</sup>(Documentación-censos-g2-1516-v2.00)*

## Gestión de liberaciones, despliegue y entregas

Para realizar el despliegue de nuestro subsistema 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.

```
C:\>heroku login  
Enter your Heroku credentials.  
Email: ruben_barr@hotmail.com  
Password (typing will be hidden):  
Logged in as ruben_barr@hotmail.com
```

Ilustración 545 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”.

```
C:\>heroku create egccenso  
Creating egccenso... done  
https://egccenso.herokuapp.com/ | https://git.heroku.com/egccenso.git
```

Ilustración 556 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:\>heroku plugins:install heroku-cli-deploy  
Installing plugin heroku-cli-deploy... done
```

Ilustración 567 Heroku instalación plugin

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

```
C:\>heroku war:deploy C:\Censo.war --app egccenso
Uploading Censo.war
----> Packaging application...
      - app: egccenso
      - including: Users/Ruru/AppData/Local/Temp/heroku-deploy313658417536402135/heroku/app/webapp-runner.jar
      - including: Censo.war
----> Creating build...
      - file: Users/Ruru/AppData/Local/Temp/heroku-deploy313658417536402135/heroku/slug.tgz
      - size: 40MB
----> Uploading build...
      - success
----> Deploying...
remote:
remote: ----> heroku-deploy app detected
remote: ----> Installing OpenJDK 1.8... done
remote: ----> Discovering process types
remote:          Procfile declares types -> web
remote:
remote: ----> Compressing...
remote:          Done: 88.8M
remote: ----> Launching...
remote:          Released v3
remote:          https://egccenso.herokuapp.com/ deployed to Heroku
remote:
----> Done
```

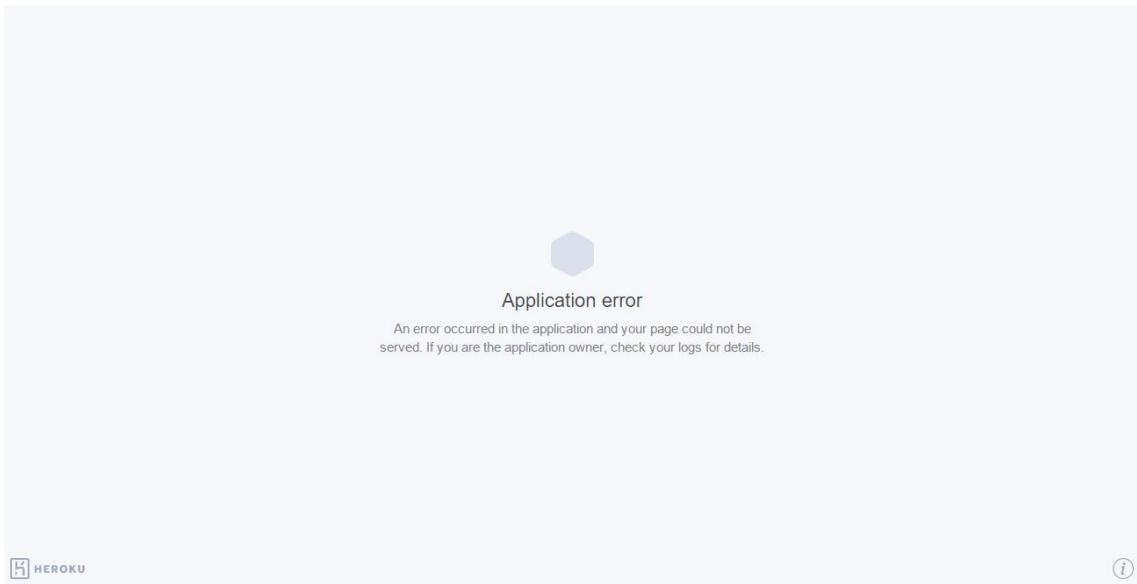
Ilustración 68 Heroku deploy

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

The screenshot shows the Heroku dashboard for the 'egccenso' application. At the top, there are tabs for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. The Overview tab is selected. Below the tabs, there are sections for Installed add-ons (\$0.00/month) and Dyno formation (\$0.00/month). The Dyno formation section indicates that the app is using free dynos. On the right side, there is a 'Latest activity' feed showing deployment logs from ruben\_barr@hotmail.com. The feed includes entries for deployment, build success, enabling Logplex, and an initial release. At the bottom, there is a 'Collaborator activity' section showing ruben\_barr@hotmail.com has 1 deploy.

Ilustración 69 Heroku panel de control

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



*Ilustración 70 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 así 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.

The screenshot shows a payment form titled "Forma de pago". It includes fields for "Número de tarjeta" (Card number), "Titular de la tarjeta" (Cardholder name), and "MM / AA / CVC". Validation messages in red indicate that the card number is required, and the month and year must be indicated. A checkbox at the bottom states: "La dirección de la tarjeta de crédito o de débito es la misma que figura arriba." (The address of the credit or debit card is the same as the one above).

Ilustración 71 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.

The screenshot shows the Jelastical interface for creating a MySQL environment. On the left, there's a diagram of the architecture: a "Balanceador" (Load Balancer) connects to a "Caché" (Cache) and two "MySQL" databases. On the right, configuration options are shown: "SQL de bases de datos" (ON), "Escalado vertical por nodo" (Vertical scaling by node), "Reservado: 1 clouddlets" (Allocated: 1 cloudlets), "Límite de escalado: hasta 16 clouddlets" (Scaling limit: up to 16 cloudlets), "Escalado horizontal" (Horizontal scaling), "1 nodos" (1 nodes), "Retardo de reinicio secuencial" (Sequential restart delay: 30 seconds), "IPv4 pública" (Public IPv4: OFF), and "Recursos" (Resources). The cost is listed as "DESDE €0.011" (From €0.011) and "HASTA €0.113" (Up to €0.113). The environment is named "EGCCenso" and the URL is ".jelastical.cloudhosted.es". Buttons for "Cancelar" (Cancel) and "Crear" (Create) are at the bottom.

Ilustración 72 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.

Nombre	Estado	Desplegado	Uso
EGCCenso egccenso.jelastic.cloudhosted.es	En ejecución	No desplegado	1% 4/22
Tomcat 7.0.73 ID de nodo: 13717			2% 3/6
MySQL 5.6.34 ID de nodo: 13718			1% 1/16

Ilustración 73 Jelastical entorno creado

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

### Importando en la base de datos "ADMCensus"

#### Archivo a importar:

El archivo puede ser comprimido (gzip, bz2, zip) o descomprimido.  
Un archivo comprimido tiene que terminar en .[formato].[compresión]. Por ejemplo: .sql.zip

- Buscar en su ordenador:  Lanzador.sql (Máximo: 1,024MB)  
También puede arrastrar un archivo en cualquier página.
- Seleccionar directorio en el servidor web para subir los archivos /var/lib/jelastic/phpMyAdmin/upload/: ¡No hay archivos para subir!

Conjunto de caracteres del archivo:

#### Importación parcial:

- Permitir la interrupción de una importación en caso que el script detecte que se ha acercado al límite de tiempo PHP. (Esto podría ser un buen recurso)
- Omitir esta cantidad de consultas (en SQL) o líneas (en otros formatos) desde la primera:

#### Formato:

#### Opciones específicas al formato:

- Modalidad SQL compatible:
- No utilizar AUTO\_INCREMENT con el valor 0
  - Leer como conjunto de Bytes

Ilustración 74 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.

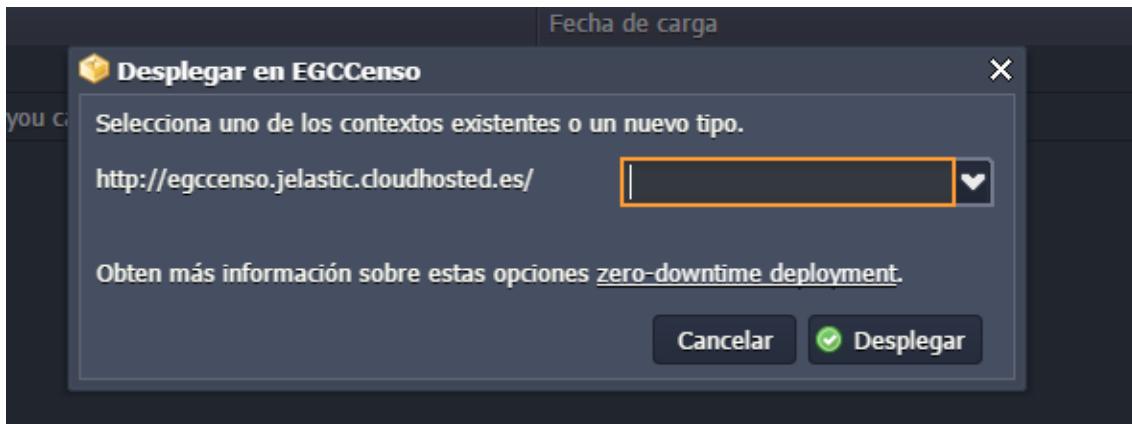


Ilustración 75 Jelastical despliegue

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.

```
C:\Program Files\MySQL\MySQL Server 5.5\bin>mysql -uroot -proot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.5.54-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Ilustración 76 Versión de MYSQL Server

```
C:\Windows\system32>mvn -v
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T17:41:4
7+01:00)
Maven home: C:\Maven\bin..
Java version: 1.8.0_121, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_121\jre
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"
```

Ilustración 77 Versión de maven

```
C:\Windows\System32>git --version  
git version 2.11.0.windows.3  
C:\Windows\System32>
```

Ilustración 78 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:\Program Files\MySQL\MySQL Server 5.5\bin>mysql -uroot -proot < create-ADMCensus.sql
```

Ilustración 79 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.

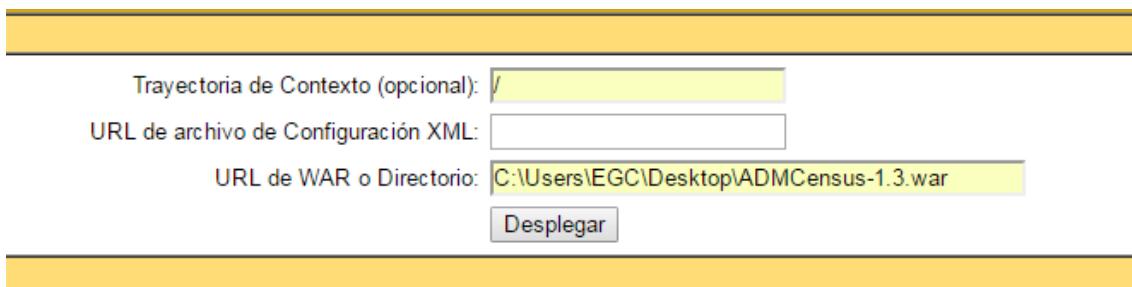


Ilustración 80 Despliegue war

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:\Windows\System32\drivers\etc>notepad hosts
```

Ilustración 81 Archivo host

En este archivo debemos introducir la siguiente línea:

```
# localhost name resolution is handled within DNS itself.  
#      127.0.0.1      localhost  
#      ::1            localhost  
192.168.56.103  www.admcensus.com
```

Ilustración 82 Configuración archivo hosts

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

```
C:\Windows\System32\drivers\etc>ipconfig /flushdns

Configuración IP de Windows

Se vació correctamente la caché de resolución de DNS.
```

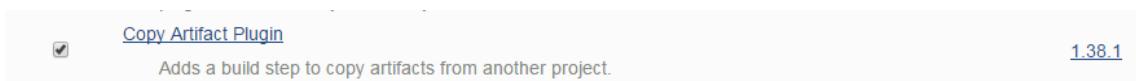
*Ilustración 83 Vaciado de DNS*

Al acceder a [www.admcensus.com](http://www.admcensus.com) podremos ver que se carga nuestro subsistema.



*Ilustración 84 ADMCensus*

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



*Ilustración 85 Copy Artifact plugin*



*Ilustración 86 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 llama de forma automática a la tarea censusWar, que se va a encargar de copiar el war generado y desplegarlo en tomcat.

Todo		+		
S	W	Nombre ↓	Último Éxito	
		<a href="#">CensoEGC</a>	5 Min 6 Seg - #31	
		<a href="#">censusWar</a>	3 Min 38 Seg - #44	

Icono: [S](#) [M](#) [L](#)

*Ilustración 87 Jobs*

## Mapa de herramientas

Todas las herramientas junto con una breve descripción se encuentran en el apartado de “Elementos de control” por lo que procederemos directamente a la visualización del diagrama con las interacciones de las distintas herramientas usadas en el proyecto.

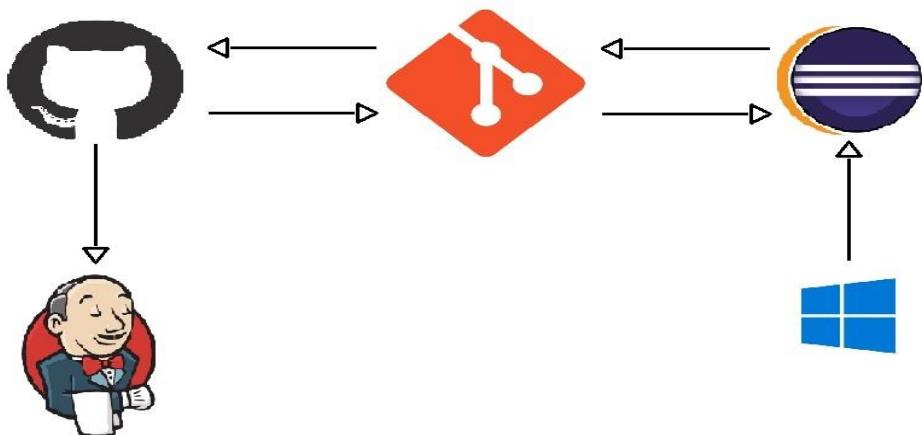


Ilustración 88 Diagrama de Herramientas

Tal y como podemos observar en el diagrama, todo el desarrollo del subsistema se hará bajo el sistema operativo **Windows**, ya que es con el que estamos más familiarizado y hay multitud de herramientas que se pueden usar en él.

El IDE que hemos usado ha sido el de **Eclipse**, ya que el código que hemos heredado estaba realizado en dicho IDE. Este IDE estará fuertemente relacionado con el repositorio local de **Git**, ya que será donde tengamos el código en nuestra máquina local.

Git a su vez estará conectado con el repositorio de **Github** que será donde los miembros del equipo tengan todo el código correspondiente al subsistema. Dicha interacción se llevará a cabo principalmente con los comandos “git pull”, “git clone”, “git push” entre otros.

Por otra parte, **Jenkins** tomará el código de la rama master del repositorio de Github cada vez que se haga un “git push” a dicho repositorio o se fuerce desde Jenkins su ejecución.

# Conclusiones y trabajo futuro

## Conclusiones

Gracias a esta asignatura hemos obtenido unos conocimientos considerables sobre algunas tecnologías bastante interesantes como pueden ser git, github, maven, Jenkins, etc.

Esto nos será muy útil en nuestro futuro profesional ya que son herramientas que actualmente están en la cresta de la ola y saber usarlas es un punto a nuestro favor.

Además de la obtención de estos conocimientos hemos obtenido más soltura a la hora de trabajar con un código heredado ya que en ninguna otra asignatura de la carrera podemos practicar ante esta situación. No es algo que nos haya gustado demasiado porque es un tanto difícil entender el código que otra persona ha realizado el año anterior, pero es algo que tarde o temprano ocurriría y es mejor que ocurra en una asignatura de la carrera donde todo está más o menos controlado que en una empresa donde por un fallo puedes acarrear graves consecuencias.

Una de las cosas que más nos ha costado ha sido la integración con los demás subsistemas ya que al tratarse de sistemas que en muchos casos no funcionaban correctamente es complicado saber si el error es culpa de tu sistema o del otro. No obstante, ha sido una buena experiencia interactuar con los demás compañeros para llevar a cabo un proyecto de este tamaño.

Para concluir, todos los miembros del equipo estamos satisfechos con la asignatura y gracias a ella hemos aprendido un poco más de todas estas tecnologías y de cómo realizar una integración continua de un proyecto.

## Trabajo futuro

Algunos de los cambios que los alumnos del año que viene deberían realizar son los citados a continuación:

- Arreglar la integración con el sistema de autenticación, ya que conseguimos conectarnos a su subsistema, pero no loguearnos correctamente.
- Integrar nuestro subsistema con algún otro, ya sea Cabina o Deliberaciones.

## Referencias

<sup>1</sup>Documentación-censos-g2-1516-v2.00 consultado en 2017 de  
“[https://1984.lsi.us.es/wiki-egc/index.php/Creaci%C3%B3n/Administraci%C3%B3n\\_de\\_censos\\_Grupo\\_2\\_\(Curso\\_2015-2016\)](https://1984.lsi.us.es/wiki-egc/index.php/Creaci%C3%B3n/Administraci%C3%B3n_de_censos_Grupo_2_(Curso_2015-2016))”

<sup>2</sup>Wikipedia consultado en 2017 de “<https://es.wikipedia.org/wiki/Maven>”