DOCUMENTO DEL PROYECTO

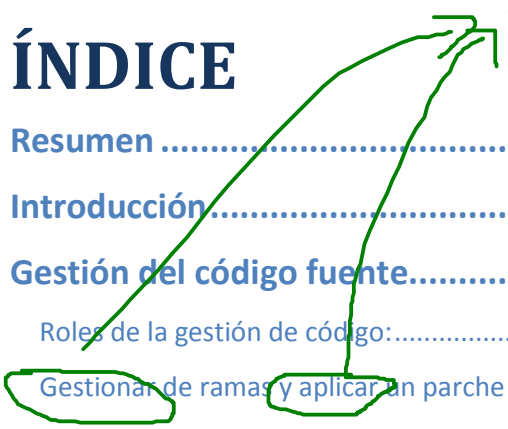
ACME Certifications

Realizado por:
Ana Isabel Espiñeira Carmona
Diego Galocha Florindo
Luisa María Reyes Suárez

CONTROL DEL DOCUMENTO

REGISTROS DE CAMBIOS EN EL DOCUMENTO		
VERSIÓN	FECHA	MOTIVO
1.0	03/12/2013	No finalizado por cambio de proyecto.
2.0	24/12/2013	Final del trabajo ACME Certifications.

ÍNDICE



Resumen

Acme Certifications se especializa en certificaciones personales. Estas certificaciones están relacionadas con las tecnologías, herramientas, metodologías, mejores prácticas, idiomas, etc.

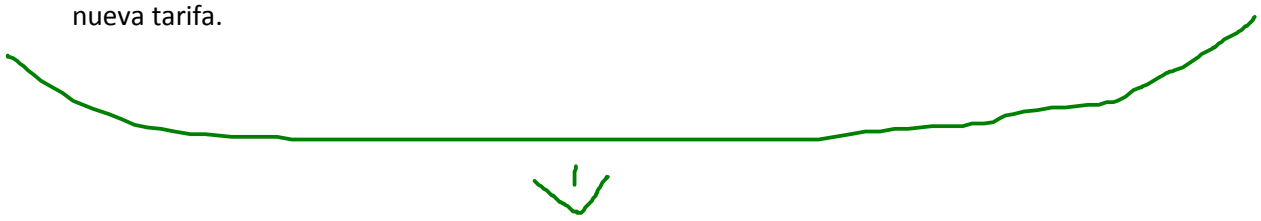
Acme actúa como intermediario entre las compañías que emiten las certificaciones y los clientes que quieren obtenerlos. Esta empresa es la responsable de garantizar que los clientes que obtienen una certificación de haber aprobado un examen que está diseñado por la empresa emisora.

La frecuencia con la que se anuncia un examen depende del número de clientes que quieran obtener esa certificación y estos exámenes consisten en una serie de preguntas de tipo test o de respuesta abierta. Las preguntas de tipo test tienen una serie de respuestas predefinidas mientras que las de respuesta abierta requieren que un revisor de Acme Certifications las revise.

Los exámenes son enviados a Acme Certifications por la empresa emisora en distintos formatos y lo normal es que cada certificación tenga un número diferentes exámenes para evitar que la gente se aprenda las respuestas de memoria.

Para obtener la certificación, los clientes se deben registrar para un examen y pagar una cuota. Luego, cuando se anuncie el examen deben ir a una oficina Acme. Deben realizar un examen en papel y esperar a que sea revisado para ver su resultado.

Si un cliente pasa un examen sus datos personales y de contacto se transfieren a la empresa emisora pero, si no pasa, tiene que volver a realizar el proceso incluyendo el pago de una nueva tarifa.



No está bien planteado el resume pues parece un resumen de la aplicación pero debe ser un resumen de vuestro proyecto, es decir, lo que habéis hecho y presentáis en el documento entregable del proyecto.

Cuándo veis una marca y no tenga explicación es que se trata de algo de forma que tenéis que corregir

Introducción

Este documento surge a raíz la asignatura de cuarto curso de Evolución y Gestión de la configuración, del Grado en Ingeniería del Software. El objetivo dicho trabajo es usar un proyecto real para poner en práctica los conocimientos adquiridos en la asignatura y poder profundizar en ellos.

El proyecto usado ha sido elegido de la asignatura de tercero de Diseño y Pruebas, este proyecto consiste en una aplicación cuyas funcionalidades han sido explicadas en el anterior resumen.

A lo largo del documento se explicarán diferentes secciones relacionadas con la gestión de un proyecto, en cada una de las cuales se incluirá, aparte de la información correspondiente a los procesos y técnicas usadas, el enunciado de un ejemplo que ayude a entender el contenido explicado en la sección.

Entre estas secciones se incluirán los procesos, técnicas y herramientas usadas para la gestión del código del proyecto y la gestión de la construcción. También se profundizará en la gestión de los entregables y se explicará que mecanismos se han usado para la gestión del despliegue, de las incidencias y depuración, para la gestión de la variabilidad y la integración y despliegue. Como última sección, se dará un esquema de cómo se conectan las herramientas que se usan en el proyecto y sus relaciones.

¿dos veces lo mismo?

Gestión del código fuente

Roles de la gestión de código:

Los roles parecen elegidos de manera solo intuitiva y no hay una explicación convincente de por qué estos roles y no otros o un análisis crítico.

Para la gestión del código se diferenciarán básicamente dos roles:

- *Equipo de desarrollo:* Será el encargado de crear las nuevas líneas de código y de desarrollar las nuevas funcionalidades que se requieran para la aplicación.
- *Equipo de gestión de cambios:* Su función es la de comprobar que los cambios realizados por el equipo de desarrollo pasan todas las pruebas necesarias que confirmen que la nueva funcionalidad aportada no tiene errores y puede ser integrada en la aplicación principal de forma que no se produzca ningún error en su ejecución.
- *Equipo de gestión de cambios:* No realiza ningún cambio en la gestión del código pero es el que decide las nuevas funcionalidades que se le añaden a la aplicación.

Gestionar de ramas y aplicar un parche (patch):

Las herramientas aquí no son lo importante. Lo importante es cómo, cuándo por qué se decide o no hacer una rama

Para la gestión de ramas nos ayudamos de un plug-in de eclipse llamado Subclipse, donde podemos trabajar con los archivos del repositorio SVN que nos ha proporcionado projetsii, (la funcionalidad de estas herramientas será explicada con mayor detalle en el apartado de Gestión de la Construcción). Hemos elegido el uso de este plug-in por facilitarnos bastante el trabajo al funcionar directamente desde Eclipse.

Al proyecto inicial, el cual es completamente funcionable, será conocido como la baseline, que será de donde partimos inicialmente y del que se creará el trunk, que será la base desde la que partirán todas las demás ramas que se realicen. Esta será la línea principal de desarrollo, donde se llevan a cabo los cambios menos complejos del día a día. Idealmente debería poder compilarse y pasar todas las pruebas en todo momento.

En general, las cosas que sean en inglés, deben ir en cursivas

Cuando queramos crear una nueva funcionalidad a la aplicación se deberá crear una rama, a la cual la llamaremos branch. Esta branch tendrá el objetivo de crear una funcionalidad concreta. Esta branches una copia del trunk donde trabajará el equipo de desarrollo y donde irán poniendo en común los cambios realizados haciendo commit de los archivos modificados mediante el SVN. Cuando haya una versión terminada puede crearse otra nueva rama, que será lo que llamaremos parche (patch).

Muy probremente justificado y argumentado, ¿qué bibiografía habéis consultado? ¿qué modelo habéis seguido?

Si en la realización de un commit hay algún problema de compilación en la aplicación de nuevas líneas de código, se realizará un diff para elegir la versión del código que queremos preservar.

Aprobación de cambios:

¿no debería ir esto en gestión de cambios?

Cuando haya una versión terminada de una rama, es decir cuando se haya realizado un parche (patch), el equipo de gestión de cambios, que se encargará de comprobar que los cambios realizados funcionan correctamente para todos los imprevistos que puedan suceder. Si

detectan algún problema con la nueva funcionalidad se lo comunicará a los miembros del equipo de desarrollo encargado de desarrollar esa funcionalidad y le expondrá el problema detectado para que sea solucionado. Si el equipo de gestión de cambios no detecta ningún error en la funcionalidad se realiza un merge hacia el trunk, donde se integrará la nueva funcionalidad.

Una vez comprobado se avisará al jefe de proyecto, que comprobará que la funcionalidad implementada se corresponde con la requerida por el cliente. Si el jefe del proyecto acepta la funcionalidad de la aplicación implementada se realizará el último merge al baseline, cambiando la versión del código fuente.

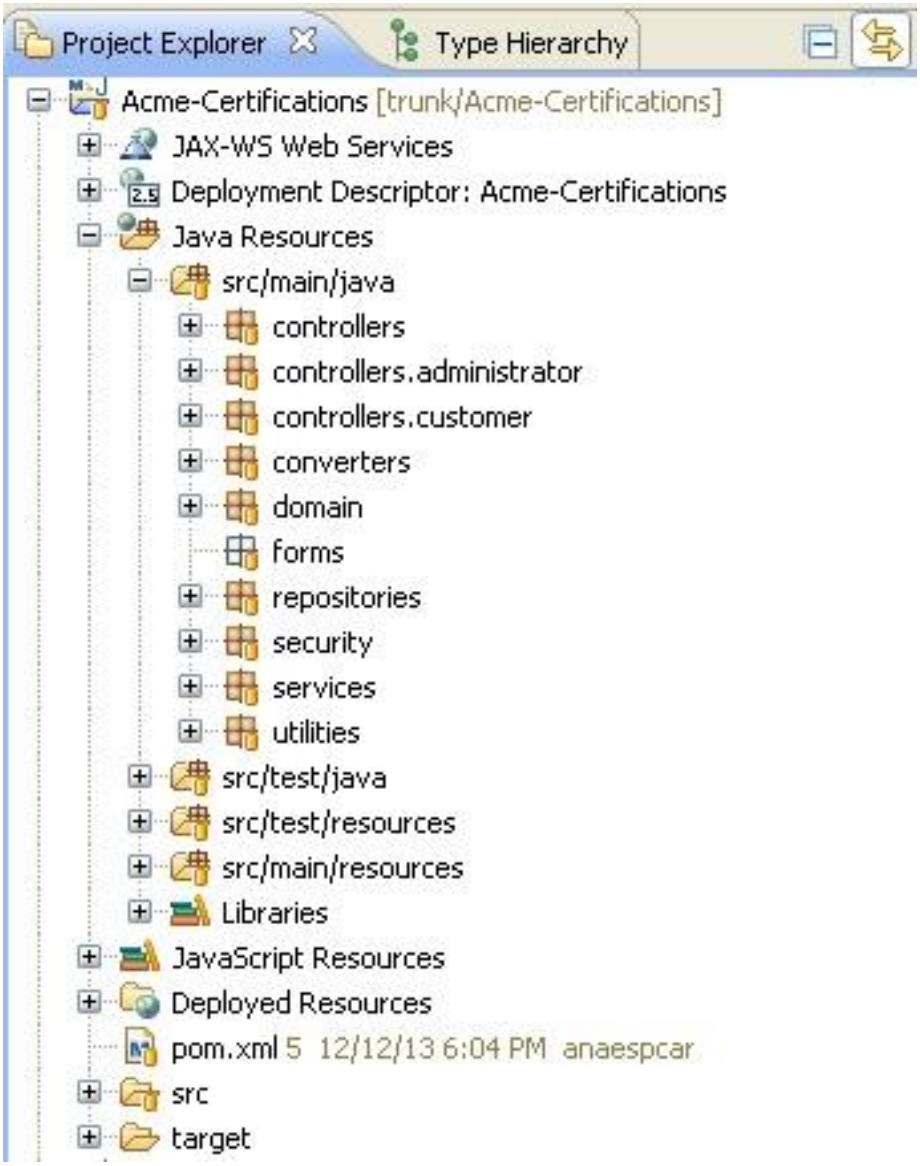
Lo mismo, esta subsección sólo habla de generalidades
pero no trasmite algo que pudiera ser útil en un proyecto real

Políticas y nombres de estilo del código fuente:

La estructura del código fuente de la aplicación está dividida en una serie de carpetas y paquetes, cada una con una funcionalidad concreta, y en cada una de las cuales se agrupan las clases correspondientes a tal funcionalidad. Este código fuente de encuentra dentro de una carpeta llamada src, en la que se pueden distinguir, por un lado las clases las clases java y por otro los recursos. A continuación se muestra la estructura principal de paquetes donde se encuentras las clases java de nuestra aplicación:

- *Controllers:* Están todas las clases de los controladores que se encargan de manejar y decidir qué vistas se van a mostrar y qué datos pedirle a los servicios para enviarle a las vistas.
- *Converters:* Son clases que se encargan de transformar las clases de dominio en String y viceversa.
- *Domain:* Es donde se encuentran las clases del dominio de la aplicación, cada una con sus atributos y sus métodos getter y setter.
- *Repositories:* Son clases (una por cada clase de dominio) encargadas de manejar los datos de la base de datos.
- *Security:* Están todas las clases necesarias para la seguridad de la aplicación y el login. En él se incluyen todas las clases incluidas el controlador, converter, domain y service, pero sólo de lo relacionado con la seguridad,
- *Services:* Son las clases del servicio (una por cada clase del paquete repositories), encargada de gestionar los datos que se obtienen de ese repositorio y relacionarlo con otros servicios. Aquí se incluyen los métodos correspondiente al guardado, borrado, etc. De los elementos del dominio.
- *Utilities:* Clases java necesarias para la conexión con la base de datos y la inclusión de datos de prueba necesarios para el manejo de objetos.

Aporta poco con respecto a lo que se espera de esta sección

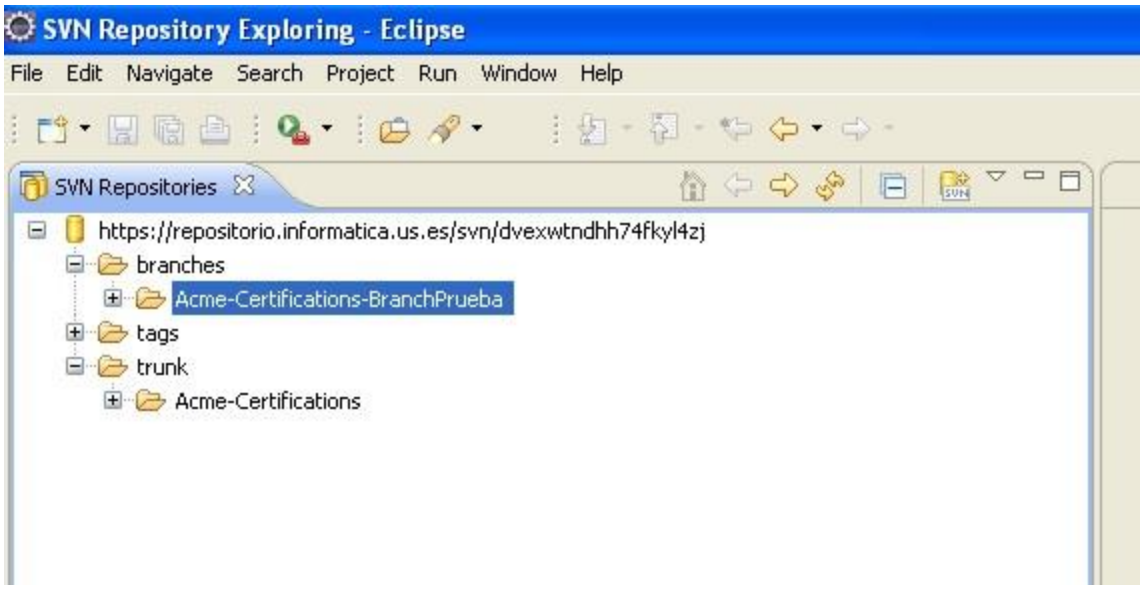


EJERCICIO:

Vamos a añadir una nueva funcionalidad a nuestro proyecto donde se podrán crear nuevos usuarios, por lo que será necesario crear una nueva rama de la baseline para trabajar sobre ella, y ya cuando verifiquemos que todo funciona correctamente pasaremos la información mediante un merge a la rama principal o baseline.

El ejercicio consiste en añadir la funcionalidad de poder registrarse en el sistema. La nueva funcionalidad consiste en dar la posibilidad de poder registrar un nuevo usuario en el sistema.

SOLUCIÓN: Para ello hemos partido del trunk para crear una nueva rama (branch) y poder trabajar sobre ella. La siguiente imagen muestra cómo quedaría la estructura:



Desarrollamos la nueva funcionalidad en la rama. Una vez finalizada, debe pasar las pruebas para comprobar que funciona correctamente y no hay errores. Pasadas las pruebas realizamos un merge para unificar los cambios realizados en el branch hacia el trunk.

¿y qué pasaría si hay problemas de merge? ¿dónde está analizado cómo se gestionarían los conflictos sintácticos y semánticos?

En general esta sección está muy pobre y muy poco elaborado. No parece que hayáis consultado nada de bibliografía de apoyo o soporte y parece que habéis llegado a una serie de propuestas más bien genéricas y difíciles de argumentar y justificar que no tienen un análisis crítico de por qué se han tomado las decisiones y por qué se decide hacerlo así.

Gestión de la construcción

Requisitos:

No soy capaz de entender este párrafo.

Para la construcción del proyecto que estamos trabajando se van a necesitar una serie de herramientas que nos aportan en la documentación de éste, y que es recomendada igualmente para muchos tipos de los principales sistemas operativos (Windows, Linux y OS X) siempre y cuando sean en su versión de 64 bits correspondientes. Nosotros, dada nuestra experiencia y conocimientos, usaremos una máquina virtual con sistema operativo Windows, en el que desplegaremos el proyecto indicado.

Además, se disponen de una serie de requisitos que nos provienen para que el funcionamiento de la aplicación sea el más óptimo posible. Estos requisitos recomendados son los siguientes:

- Al menos, una pantalla de 22'.
- 8 GiB de RAM (aunque 16 sea la mejor de las recomendaciones).
- Un disco duro de 500 GiB de memoria con soporte de 7500 RPM.
- Un procesador Intel i7.
- Un puerto Ethernet de 100 Mbps.
- Una buena tarjeta gráfica.

¿por qué esto y qué tiene que ver esto con la gestión de la construcción?

Sin embargo, no se descarta que la aplicación pueda funcionar igualmente de una manera correcta con los siguientes requisitos mínimos:

- Una pantalla de 15'.
- 2 GiB de RAM.
- Un disco duro de 150 GiB de memoria con soporte de 5700 RPM.
- Un procesador Intel i5.
- Una decente tarjeta WIFI.
- Una decente tarjeta gráfica.

Esta terminología no es propia de un documento profersional

A parte de estos requisitos anteriormente mostrados, serán necesarias también una serie de herramientas que habrá que integrar en nuestro equipo para el correcto funcionamiento de la aplicación. Uno de ellos será el Java Development Kit (JDK), del que necesitaremos instalar la versión 1.7.

Además, necesitaremos Eclipse Indigo EE SR2 como entorno de desarrollo integrado. En él instalaremos una serie de plugins y librerías que nos facilitarán la ejecución y trabajo de nuestra aplicación. Entre estos plugins se encuentra Maven, una herramienta con la que administraremos los proyectos, su compilación, ejecución de prueba, etc.,y Subclipse, con el que trabajaremos para el control de versiones de la aplicación.

Herramientas:

Usaremos la herramienta proporcionada por la empresa Oracle, en concreto, MySQL Server versión 5.5, y sus correspondientes conectores que nos ayudarán a la perfecta ejecución de la aplicación en el servidor. Se trata de un gestor de base de datos bastante robusto e intuitivo a la hora de trabajar con él, además de estar perfectamente integrado con el entorno de desarrollo que vamos a usar.

¿qué tiene que ver con gestión de la construcción?

Como servidor de aplicaciones usaremos la herramienta proporcionada por la famosa empresa Apache, concretamente la de Tomcat versión 7.0, que al igual que el gestor de base de datos, está bastante bien integrado en nuestro entorno de desarrollo. Utilizaremos tanto la versión de desarrollador como la de servicio, siendo esta última la pensada para la ejecución y producción de la aplicación.

Serán necesarias una serie de pequeñas herramientas cuya funcionalidad será bastante grande en nuestra aplicación. Entre ellas, se encuentra la integración del Framework de Spring, y los componentes de herramientas muy importantes como Hibernate, Javax, Apache, JQuery… que nos proporcionarán una serie de librerías útiles para el correcto funcionamiento del proyecto.

Por último comentar que usaremos la herramienta Maven que proporciona una interfaz de línea de comandos para administrar nuestro proyecto, es decir, limpiarlo, compilarlo, ejecutar pruebas, etc. La estructura de Maven incluye en nuestro proyecto la carpeta test usada únicamente para realizar tests sobre el proyecto. Como hemos desarrollado un software en Java hemos usado el framework JUnit para las pruebas unitarias del proyecto. Maven también automatiza la gestión de las dependencias encontradas en pom.xml.

No hemos utilizado la herramienta Maven en sí, la hemos utilizado en el entorno de desarrollo Eclipse descargando el plugin de Maven (The Maven Integration for Eclipse).

¿no hay ningún inconveniente con esta decisión? algo de estos hemos comentado en clases

Construcción:

La construcción del proyecto solo se realizará cuando se de algunas de las tres ocasiones que se explican a continuación. Uno de ellas será cuando se necesite un nuevo equipo para el desarrollo de la aplicación. Otro aspecto que requiere la construcción será cuando en alguno de los equipos ya existentes se produzca una avería que haga que se pierdan datos relevantes. También se requerirá una construcción del proyecto cuando se modifiquen las herramientas usadas, por ejemplo, si cambiáramos la base de datos actual de MySQL.

No entiendo este párrafo que se supone que es el que da información real de vuestro proyecto.

EJERCICIO:

Usar las herramientas descritas anteriormente para construir un nuevo proyecto.

Mal enunciado. Debe ser un ejercicio autocontenido.

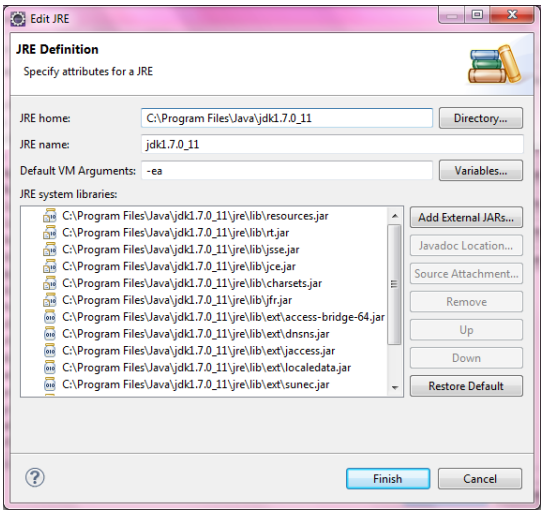
SOLUCIÓN:

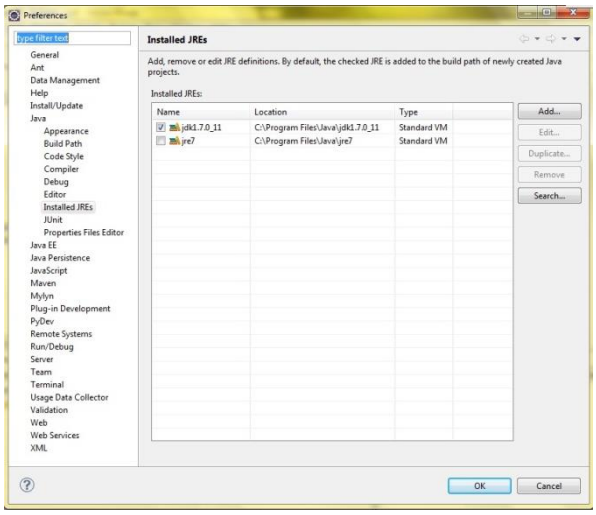
En primer lugar se procede a instalar todas las herramientas necesarias para trabajar, nombradas anteriormente.

Después de tener todo instalado, se procede a configurar MySQL. Se instala una instancia llamada “localhost” y se crea una cuenta administrador con usuario “root” y contraseña “V3rY=$tR0nG=P@$$w0rd$”. A continuación usar la cuenta administrador para crear una cuenta llamada “acme-user” con contraseña “*ACME-Us3r-P@ssw0rd*” y una última cuenta llamada “acme-manager” con contraseña “*ACME-M@n@ger-6874*”.

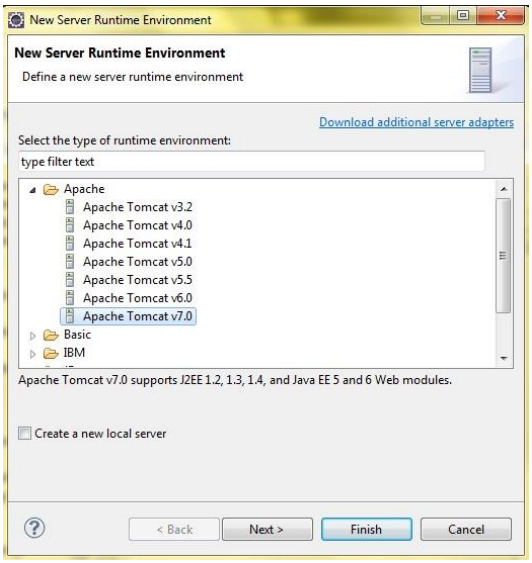
Tras la instalación de MySQL se procede con la instalación de Tomcat, en la que sólo se debe configurarel puerto (80) y el usuario (admin) con su contraseña (T0mc@t=Adm1n$trat0r).

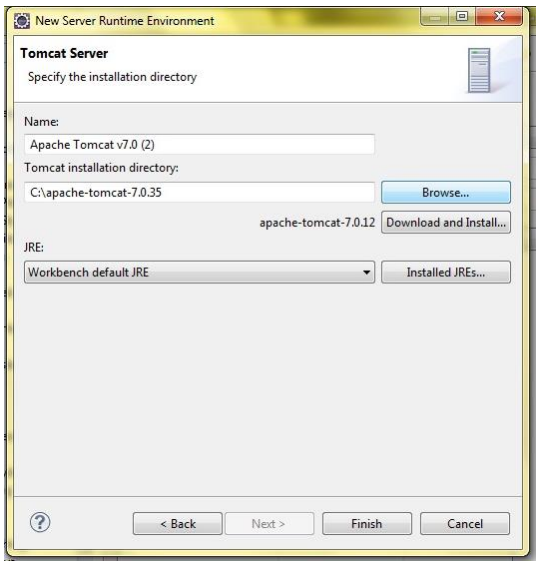
A continuación, se ejecuta Eclipse Indigo y se configura el entorno de Java en Eclipse:



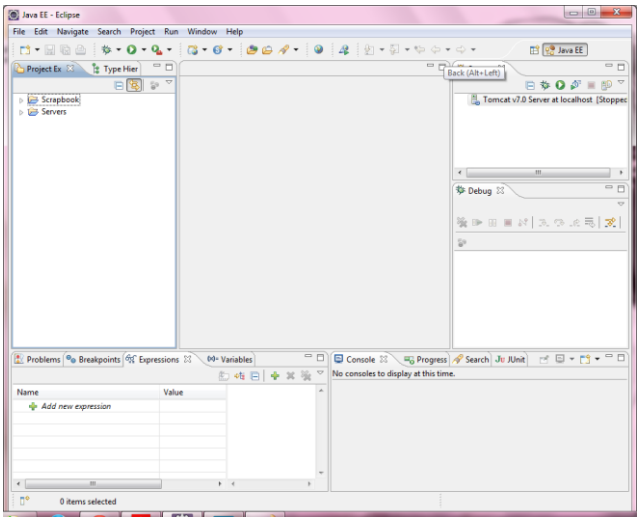


Una vez configurado el JRE, hemos pasado a configurar Tomcat Server, para ello se debe borrar el Server existente en la vista de servidores y agregar el requerido:

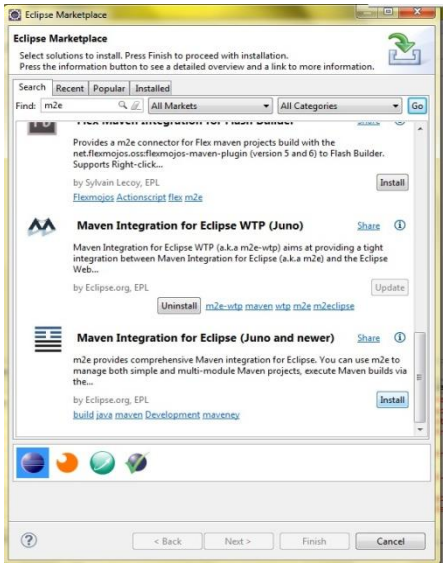




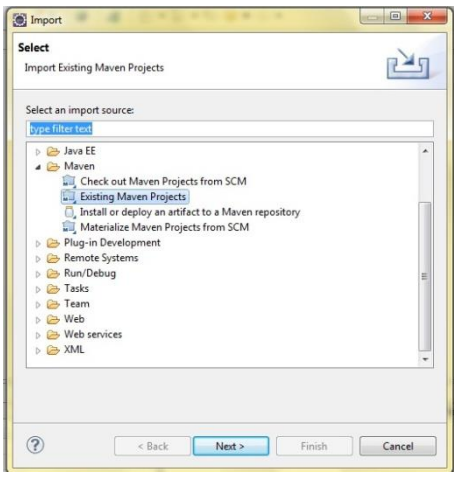
El resultado quedaría así:



Antes de importar el proyecto se configura Eclipse con Maven.



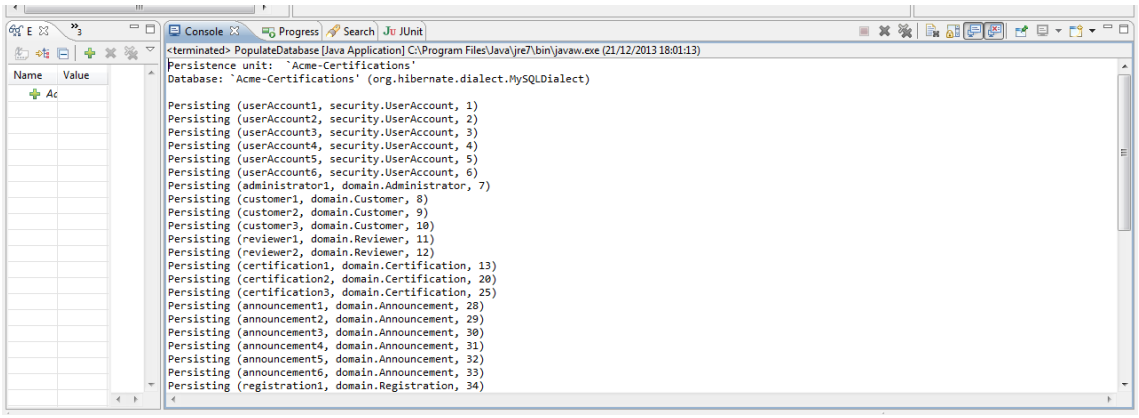
Cuando está configurado Maven, se importa el proyecto en Eclipse como proyecto Maven: File→Import→Maven→Existing Maven Project



A continuación se crea la base de datos y se configura para los usuarios que van a acceder a ella y sus privilegios. Se configura de la siguiente forma:

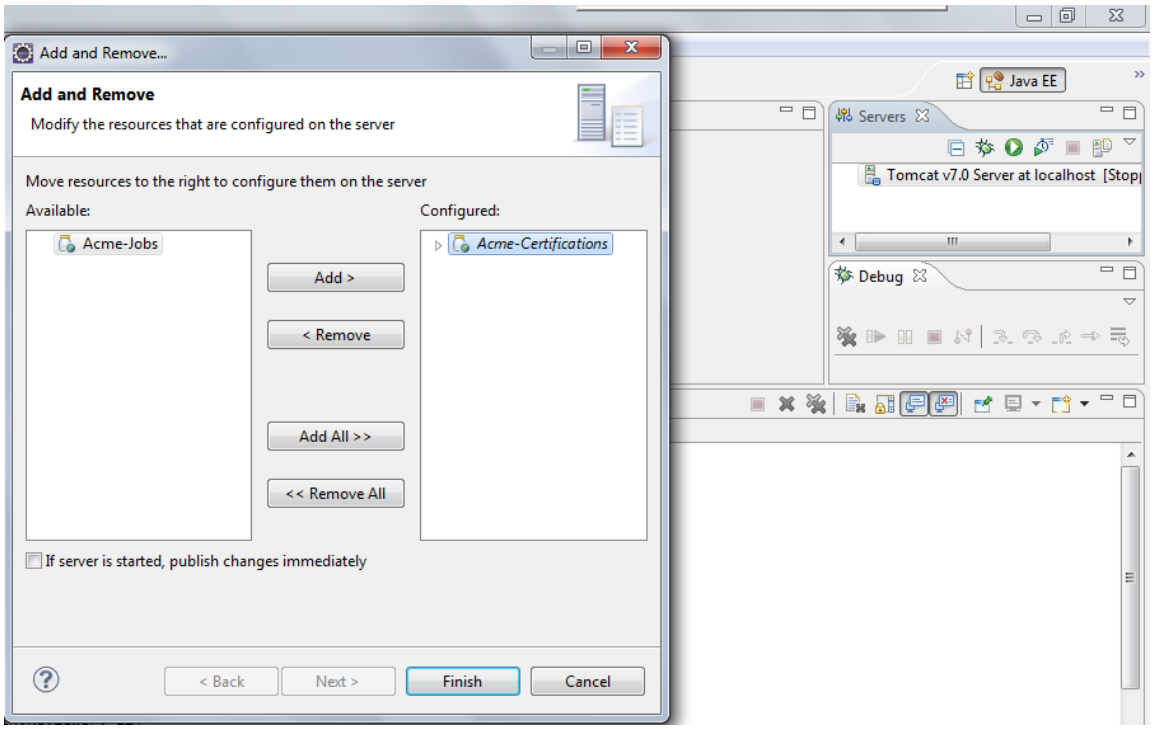
```
drop database if exists ` Acme-Certifications `;
create database ` Acme-Certifications `;

grant select, insert, update, delete
on `Acme-Certifications`.* 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 ` Acme-Certifications `.* to 'acme-manager'@'%';
```

Después de crear la base de datos, se inserta los datos ejecutando la clase del proyecto "PopulateDatabase.java" que se encuentra en la carpeta src→main→java→utilities. En la consola de Eclipse debe aparecer esto:

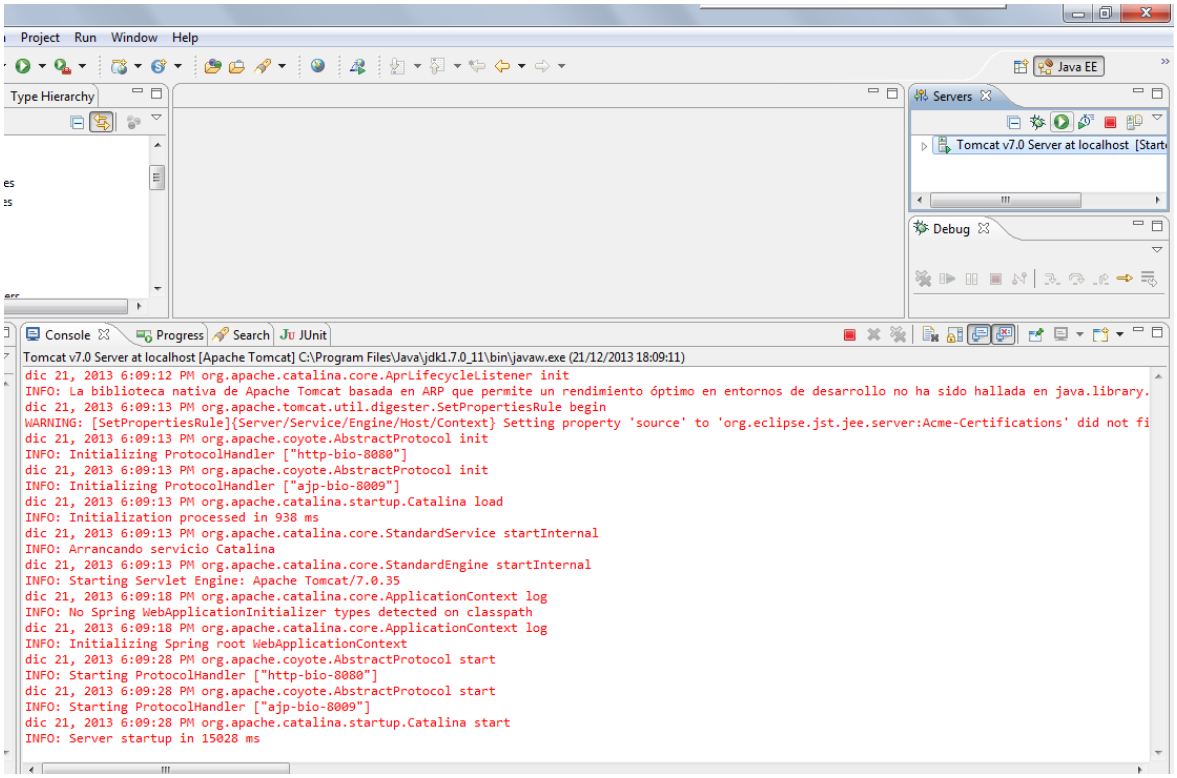


Finalmente se ejecuta el Tomcat de la siguiente manera:

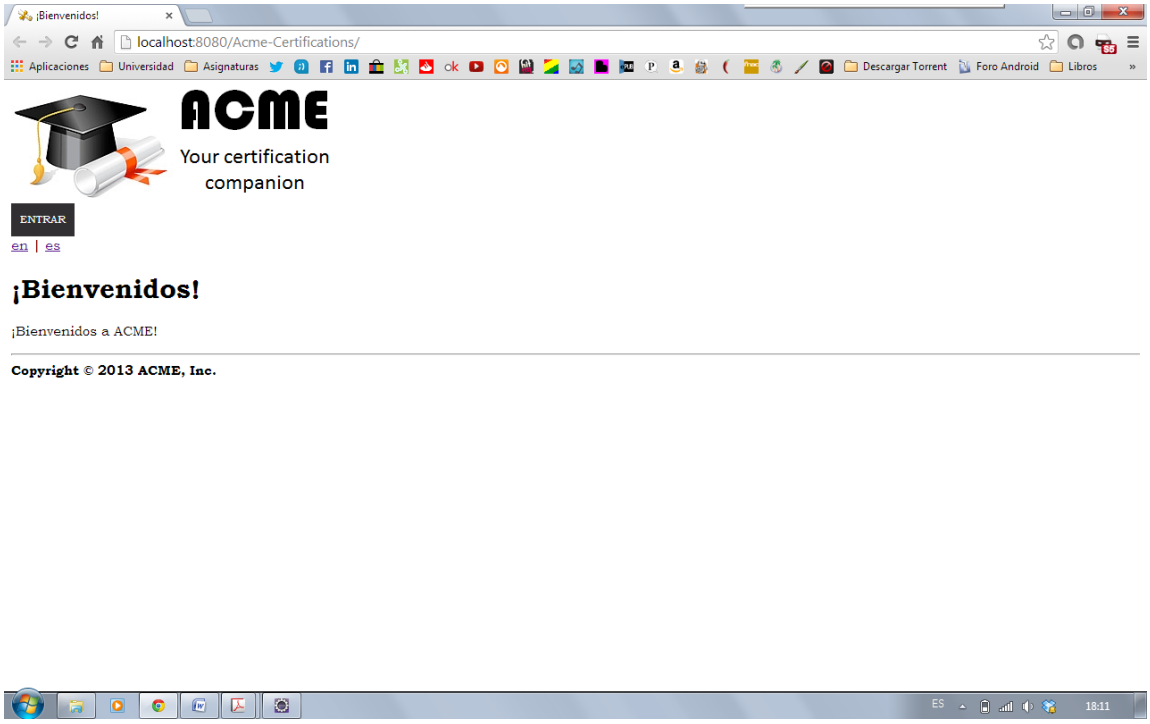
1. Botón derecho en "Tomcat v7.0 Server at localhost", localizado en "Servers" de Eclipse (arriba a la derecha) y se selecciona "Add and remove" y se agrega en la parte de "Configured" el proyecto "Acme-Certifications" y Finish:



2. Pulsamos en la parte de “Servers” de Eclipse el símbolo de “start” y debe aparecer lo siguiente en consola:



Y ya está el proyecto listo. En nuestro navegador ponemos la dirección: http://localhost:8080/Acme-Certifications



Gestión de entregables

"Un entregable es cualquier producto medible y verificable que se elabora para completar un proyecto o parte de un proyecto. Si el proyecto fuese una fábrica, los entregables son lo que produce esa fábrica. Existen entregables intermedios (internos), que se utilizan para producir los entregables finales que validará el cliente del proyecto. Los entregables ayudan a definir el alcance del proyecto y el avance del trabajo en el proyecto debe ser medido monitoreando el avance en los entregables." [i]

Las citas literales deben ir en cursiva además de entre comilladas.
¿por qué cogéis esta definición? ¿para qué?

Un entregable del proyecto se identifica por aportar nueva funcionalidad. En este proyecto se diferenciarán dos tipos de entregables, uno de ellos será el entregable final que se entrega al cliente. Este entregable será producido por el equipo de gestión de cambios y debe haber pasado las pruebas necesarias para verificar que la funcionalidad añadida es correcta y no produce fallos. El equipo de gestión de cambios le proporcionará el entregable al jefe del proyecto. El otro tipo de entregable son los entregables intermedios, estos serán producidos por el equipo de desarrollo hacia el equipo de gestión de cambios que debe probar y buscar fallos (si los hubiera). Luego este entregable intermedio, después de pasar las pruebas de validación, se convertirá en un entregable final que el equipo de gestión de cambios le entregará al jefe del proyecto, el cual se lo facilitará al cliente.

Los roles para la gestión de entregables serán los mismos que para la gestión del código fuente. Los entregables son encargados por el jefe de proyecto que es el responsable de coordinar y aceptar que cambios realizar a la aplicación. Y una vez que el equipo de desarrollo produce ese cambio, es el equipo de gestión de cambios el que se encarga de revisar la correcta producción del entregable y ejecutar su publicación. La publicación de un nuevo entregable se producirá por el equipo de gestión de cambios con la realización de un path mediante Subclipse que es la herramienta usada para la gestión del código fuente.

??

Realmente esto no es así

Aunque los párrafos aquí expuestos son más o menos coherentes con la sección, la sección en general adolece de ser MUY pobre.

EJERCICIO:

Definir un patrón para la entrega de los entregables y poner un ejemplo de este patrón en relación al proyecto.

SOLUCIÓN:

Esto no debe ser planteado como un ejercicio, debe ser planteado como una parte de la sección

Existe un patrón en la producción de entregables con relación al número de versión del proyecto. Cada entregable tendrá siempre el mismo nombre y únicamente variará el número de versión que irá avanzando progresivamente según se refiera a un entregable intermedio o un entregable final.

Recordemos que un entregable intermedio es aquel que es proporcionado por el equipo de desarrollo al equipo de gestión de cambios para que este haga las pruebas oportunas en la búsqueda y detección de errores. Por tanto esta es una versión bastante inestable y es por ello que su número de versión debe terminar en un número impar. Que termine en un número impar significará que la versión es inestable.

Debe ser explicado mucho mejor en todo caso.

En cambio, los entregables que produce el equipo de gestión de cambios para el jefe del proyecto ya ha pasado las pruebas de validación y, por tanto, es un producto estable y completo que no tiene ningún fallo en su desarrollo y que podría ser usado por el cliente sin ningún tipo de problema. A este tipo de entregables se le asignará una numeración par en su versión del entregable. El término de un entregable con una numeración par significa que este entregable tiene una gran estabilidad.

Es por ello que si nos en una entrega inicial llamada, por ejemplo, ACME-V1.0, el equipo de desarrollo proporcionará una nueva funcionalidad del proyecto y realizará un entregable del proyecto de nombre ACME-V1.1, este llega al equipo de gestión de cambios y si detecta fallos es el equipo de desarrollo quien resuelve estos datos y proporcionará otro entregable de nombre ACME-V1.3. Como se puede comprobar, este entregable sigue siendo inestable por lo que debe conservar la numeración impar. Si el entregable pasa las pruebas de validación necesarias, al jefe de proyecto le llegará un entregable del proyecto de nombre ACME-V1.4, la cual es estable y cumple con una nueva funcionalidad completa en comparación con la versión inicial de ACME-V1.0.

Gestión del despliegue

"El procedimiento Gestión de Despliegues en Entornos de Producción *tiene como objetivo asegurar la validez, calidad, seguridad y operatividad de las actividades a desarrollar para efectuar la puesta en entornos (preproducción, producción y formación) de los sistemas, a fin de minimizar el riesgo de alteración de los mismos."* [ii]

Es un caso particular pero hay app que no corren sobre "servidores" por lo que no se puede decir que sea una def general.

Otra definición más personal sobre el despliegue, dice que desplegar un sistema significa instalarlo en un servidor. El primer punto de comienzo sobre el despliegue de un sistema es la configuración de pre-producción. Se entiende, obviamente, que previos a estos pasos se han desarrollado las fases de desarrollo del código, y evaluación de éste mediante test, lo que llamamos la fase testing. En esta fase de pre-producción el desarrollador puede darse cuenta de errores que haya en la aplicación previos a la entrega final de esta, ahorrando mucho en costes en caso de corregir bugs.

No procede. Debéis haber puesto también algo sobre pruebas pero en una sección aparte.

El primer paso para el despliegue será proceder a la creación de un artefacto war (archivo que contendrá las imágenes, los scripts, los estilos, las vistas, las clases, los archivos de configuración...es decir, todo excepto el código fuente de nuestras clases Java).

Una vez tenemos listo el artefacto war, tenemos que desplegarlo en un servidor. En nuestro caso, el despliegue se realiza sobre un servidor de Tomcat, que provee de una aplicación para dirigir nuestro sistema. Además, lo podemos configurar para crear un dominio web en local, que será en nuestro caso la url www.acme.com.

Muy pobre la descripción, la propuesta, los detalles, las justificaciones,...

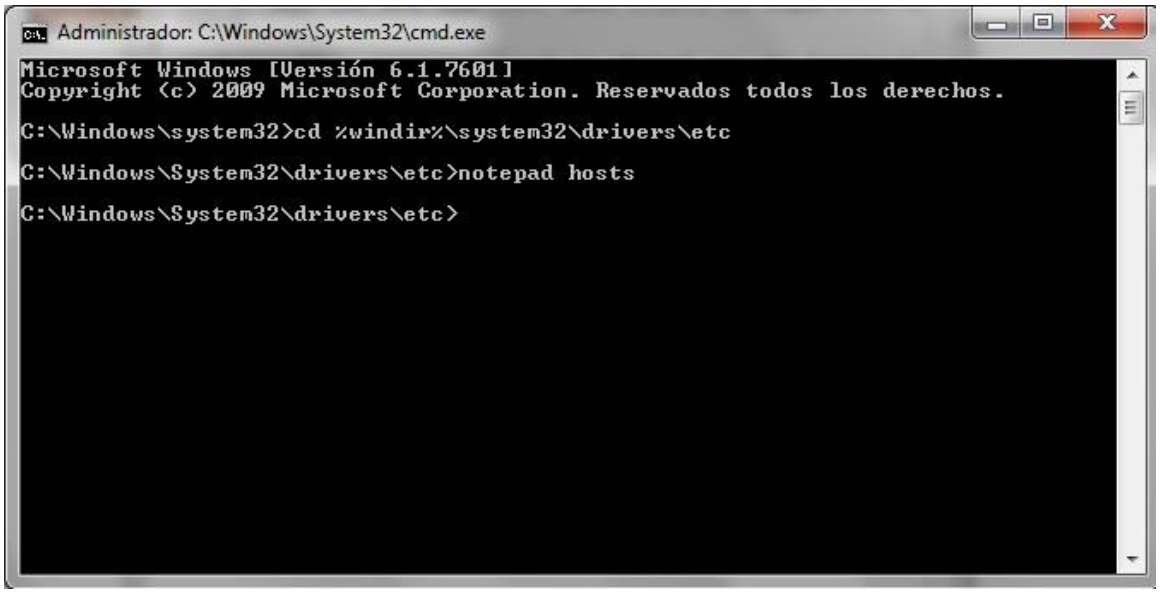
EJERCICIO:

Realizar un despliegue a partir de un artefacto war de la aplicación.

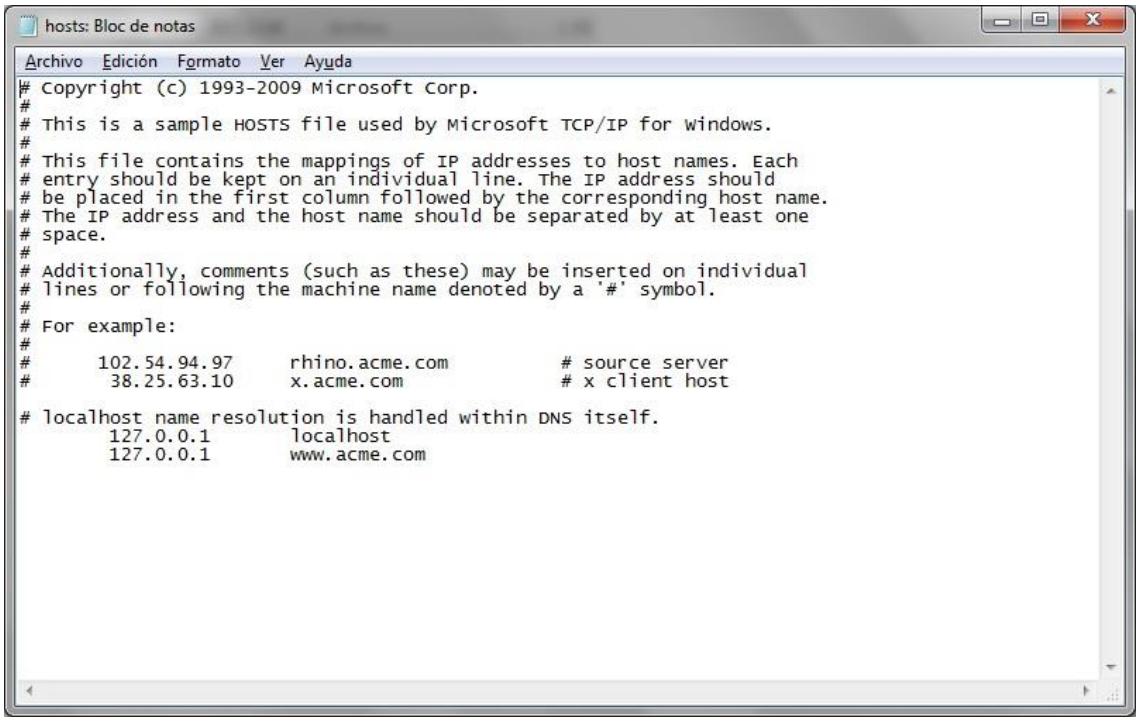
SOLUCIÓN:

Para la realización del ejercicio será necesario tener instalados en la máquina servidores tanto de base de datos (MySQL en nuestro caso) y de aplicaciones (como hemos dicho anteriormente, en nuestro caso usaremos un servidor de Tomcat).

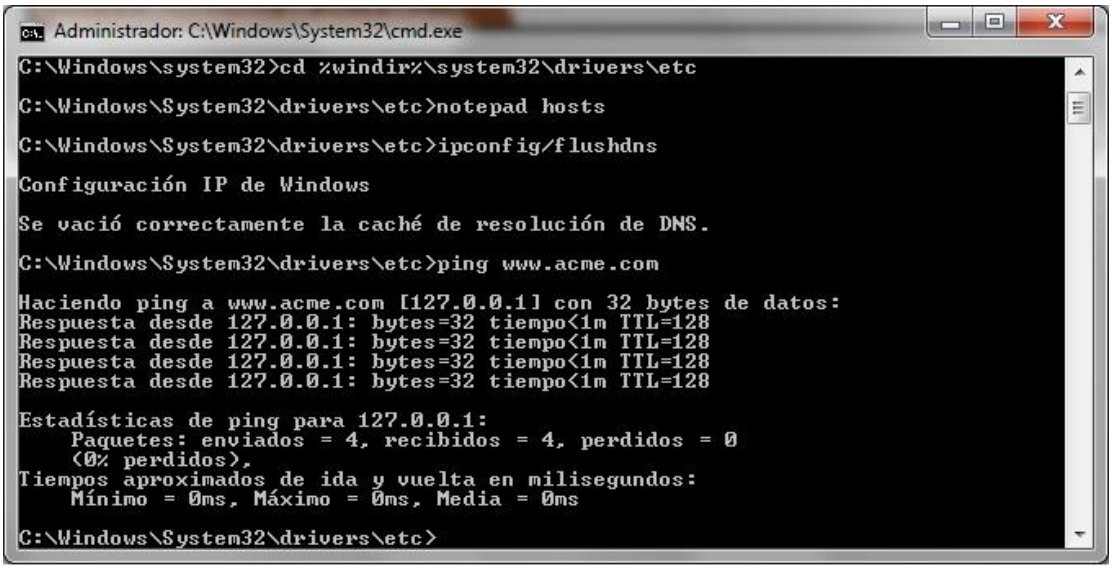
El primer paso será "hackear" nuestro sistema, ya que lógicamente no vamos a comprar un dominio para probar el despliegue. Tendremos que modificar el archivo "hosts" donde existe una traducción local de los DNS. Para ello, abrimos un terminal como administrador para poder editar este archivo de configuración, y escribimos las líneas que podemos ver en la imagen:



Posteriormente, pasamos a editar ese archivo de configuración, y editamos las líneas finales añadiendo 127.0.0.1 localhost y 127.0.0.1 www.acme.com en él. Como podemos observar, se trata de un archivo "vacío", pues a pesar de tener multitud de escrituras, son todas comentadas, por lo que pasa a tener funcionalidad desde el momento en el que añadimos esas líneas. Se puede ver como quedaría en la siguiente imagen:

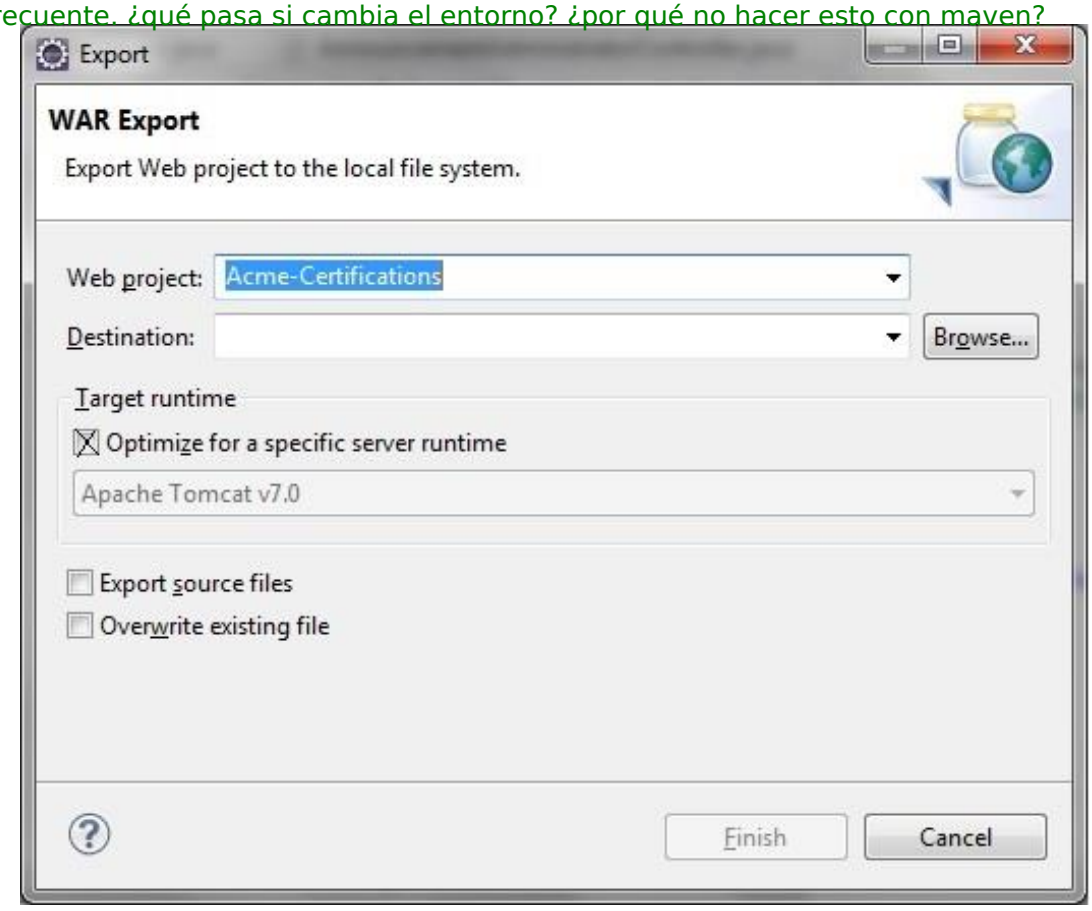


El siguiente paso será el de refrescar nuestra caché de DNS para que así nuestro sistema operativo recargue el archivo hosts. Lo haremos escribiendo en la misma terminal el comando: >ipconfig /flushdns. De paso, podemos comprobar si todo funciona correctamente haciendo ping a nuestra dirección. Todo ello, se puede ver en la siguiente captura:



Podemos decir que nuestra configuración de pre-producción está completa, por lo que pasamos entonces a la creación del artefacto war para seguir con el procese de despliegue. En nuestro caso, mediante el entorno de desarrollo eclipse será muy sencillo crear este artefacto, ya que sólo clickando en *Export*, podemos indicar la creación, teniendo en cuenta de marcar la opción de *Optimizar* para nuestro servidor Tomcat, como podemos ver en la captura:

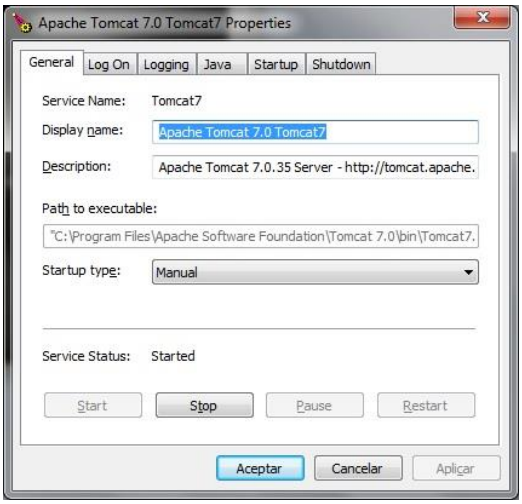
Fijaros que siempre vuestros procesos están atados a los entornos de desarrollo concretos lo cual es un error y un antipatrón frecuente. ¿qué pasa si cambia el entorno? ¿por qué no hacer esto con maven?



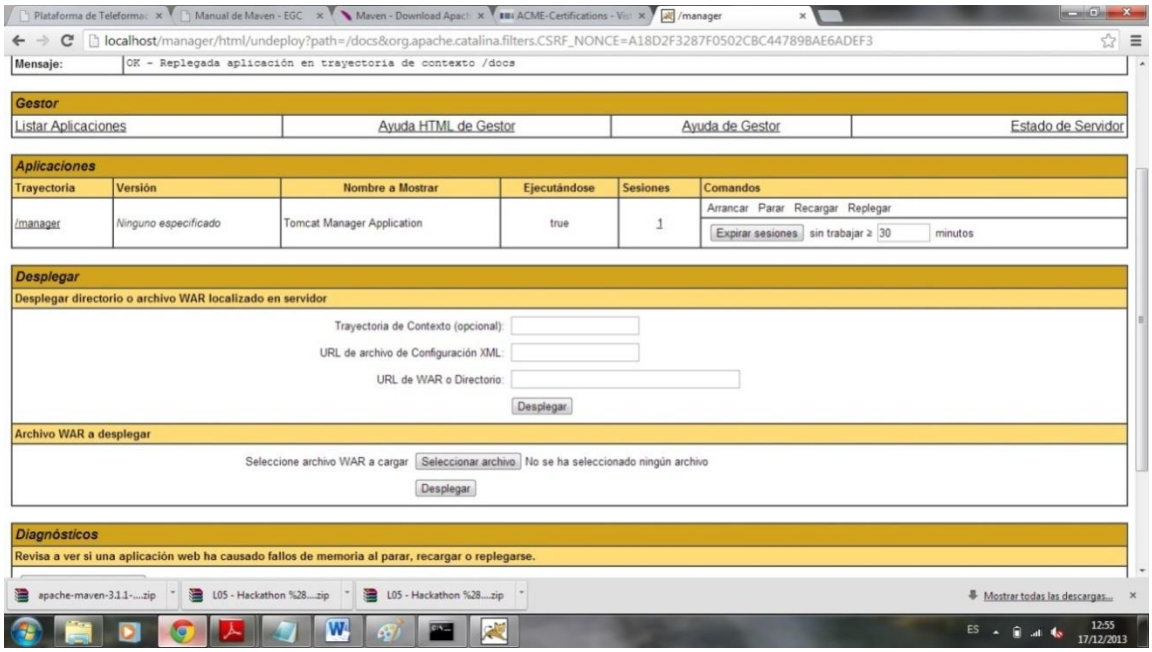
Una vez creado, es hora ya de pasar al despliegue de nuestra aplicación en el servidor web. El primer paso será arrancar nuestro servicio Tomcat desde la consola de comandos que teníamos abierta como administrador, y ejecutar los siguientes comandos:

```
> cd %programfiles%
> cd "Apache Software Foundation\Tomcat 7.0\bin"
> Tomcat7w.exe
```

Tras ello, presionamos *Start*y automáticamente arrancará el servicio.



Para comenzar a configurar el servidor, abrimos un navegador y entramos en la dirección http://localhost/manager Se abrirá una simple aplicación de Tomcat que nos ayudará a hacer el despliegue de nuestra aplicación. Indicamos nuestras credenciales de acceso de administración, y presionamos en log in. Posteriormente, replegamos los contenidos que no nos sean útiles, y dejamos solo el que pone manager, como se ve en la captura siguiente:



Antes de desplegar nuestro artefacto, será necesario que iniciemos nuestro servidor de base de datos MySQL previamente, puesto que será necesario que esté en ejecución antes del despliegue.

Estos pasos de despliegue manual van en contra de las recomendaciones de un "continuos delivery" que se han comentado en clases y referenciados en la bibliografía. Debiera conocerse y comentarse en la doc del proyecto

Y por último, el paso final, indicar a la aplicación de Tomcat nuestro proyecto a desplegar. Para ello añadiremos en “Trayectoria de Contexto” el símbolo ”/” por temas de configuración, y en “URL de WAR” la dirección del artefacto War que exportamos del entorno de desarrollo. Pulsamos en desplegar, y tendremos nuestra aplicación óptima para visualizar a través de los enlaces http://localhost o www.acme.com.

Gestión de incidencias y depuración

Gestión de incidencias

En primer lugar se debe identificar la incidencia para saber por qué ha ocurrido y que cambios hay que implementar para solucionar dicha incidencia, ésta pasa a manos del equipo de control de cambios que la analiza y decide si se admite dicha incidencia y cuándo se va a realizar los cambios para subsanar dicha incidencia o no se admite porque sea un problema ?? eléctrico o de entorno software (distinto sistema operativo, falta de herramientas instaladas…).

Cuando una incidencia es admitida, el equipo de gestión de cambios ordena dichas incidencias por prioridad (de mayor a menor) y se le asigna la incidencia de mayor prioridad primero al equipo de desarrollo o específicamente a las personas o persona que programó dicha parte del código. Finalmente dicha persona o equipo se encarga de solventar la incidencia (depuración).

Muy poquito trabajo en esta sección. No hay ni siquiera lo que hemos visto en teoría

Depuración

La depuración es realizada por el equipo de desarrollo.

Estos reciben un informe detallado del equipo de gestión de cambios en el que explican detalladamente los motivos de la incidencia y por qué se ha producido.

En primer lugar el equipo de desarrollo comprueban en su archivos de incidencias (que quedan registradas todas las incidencias anteriores) si ya se produjo dicha incidencia o parecida, si se realizó dicha incidencia siguen los pasos y cambios que se realizó en aquella vez comprobando más tarde que funciona y pasándole el cambio al equipo de gestión de cambios, para que comprueben que el cambio se ha realizado correctamente.

Si dicha incidencia nunca ha sido tratada se monta la aplicación en una máquina virtual, con el mismo sistema operativo y herramientas en el que ocurrió la incidencia.

A continuación intentan diagnosticar y detectar de donde viene el error, para ello al usar Eclipse como entorno de programación usan la herramienta DEBUG, en la que van paso a paso en cada método del código viendo cómo se comporta el programa, realizando un breakpoint y viendo el valor de todas las variables en ese punto.

Realizan hipótesis de las cosas que habría que cambiar para que la incidencia se solucione. Se programa en el código cada una de las hipótesis, individualmente y se va anotando los errores que realizan dichas hipótesis, si hay errores. Comprueban si esa hipótesis es acertada para solucionar la incidencia y si lo es realizan el cambio.

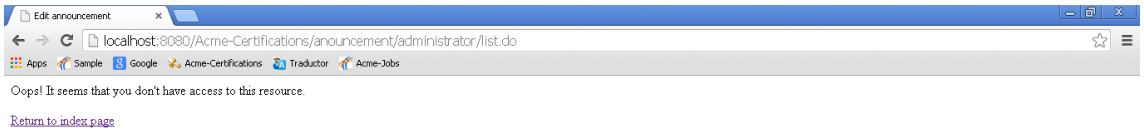
Al realizar el cambio comprueban que no da ningún error y el código pasa a manos del equipo de gestión de cambios que realiza los mismos pasos que se realizó anteriormente y que provocó la incidencia que han tratado, si vuelve a dar error se repite el proceso anterior, pero si el error queda solucionado se aceptan los cambios y archivan el cambio realizado como ayuda para futuras incidencias.

EJERCICIO:

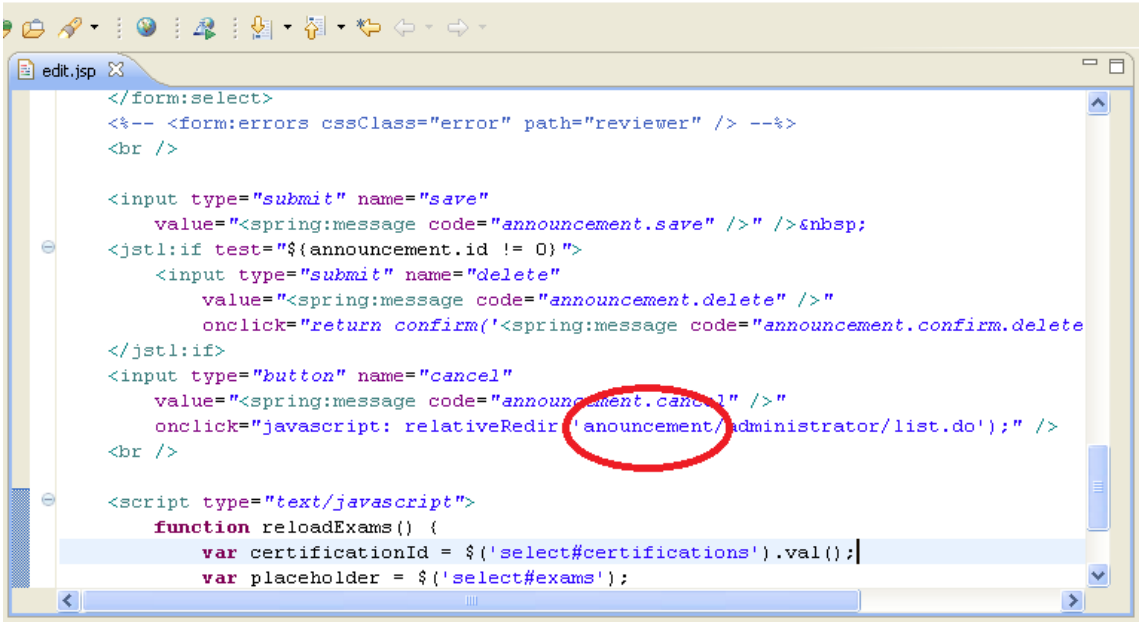
Poner el ejemplo de una incidencia y depurarla para encontrar solución.

SOLUCIÓN:

Un ejemplo sería que el equipo de desarrollo le pasara una un entregable a equipo de gestión de cambios donde, en uno de los nuevos formularios implementados, el botón de *cancelar*redirecciona a una ruta incorrecta donde la página jsp que se indica en la ruta del botón cancelar del formulario no existiera. Este es un error que se les ha pasado al equipo de desarrollo y que observa el equipo de gestión de cambios al realizar sus pruebas. Esta sería la vista que se obtendría a dicho error:



El equipo de gestión de cambios le pasa al equipo de desarrollo un informe detallado del problema y es el equipo de desarrollo el encargado, por medio del depurador, de ver qué datos fluyen bien y cuáles no, para obtener la raíz del problema y solucionarlo.



El problema era simple, un error de sintaxis donde a la palabra announcement, le faltaba una n. El equipo de desarrollo soluciona el problema y le envía un nuevo entregable al equipo de gestión de cambios.

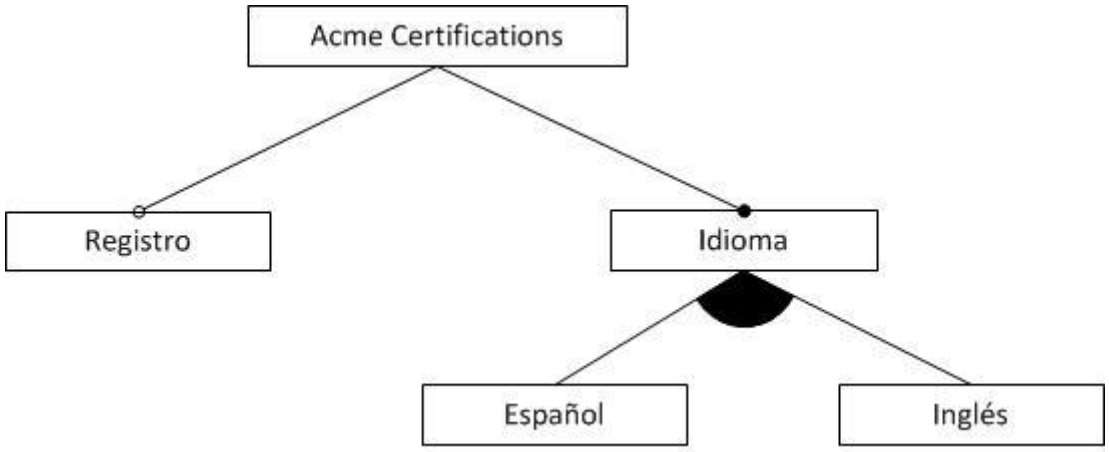
Gestión de la variabilidad

Este proyecto no se caracteriza principalmente por su variabilidad, pero tendría una pequeña parte variable debido a su funcionalidad. Por ejemplo, se ha creado una tabla donde se indica cómo sería un ejemplo de ello en nuestro proyecto, con su correspondiente modelo de características:

Tabla de variabilidad:

	AC1.0	AC2.0	AC3.0
Registro en el sistema		•	•
Sistema en español			•
Sistema en inglés	•	•	•

Modelo de características:



Como se observa el producto AC3.0 tiene variabilidad positiva respecto a AC1.0 y el producto AC2.0 tiene una variabilidad negativa respecto a AC3.0.

Aunque este apartado está muy pobre es cierto que en teoría se ha expresado de manera más abstracta y por lo tanto es más difícil de concretar que las demás secciones. En todo caso, tampoco luce por su profundidad esta sección.

Integración/despliegue continuos

"La integración continua (continuous integration en inglés) es un modelo informático propuesto inicialmente por Martin Fowler que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes. Entendemos por integración la compilación y ejecución de tests de todo un proyecto. El proceso suele ser, cada cierto tiempo (horas), descargarse las fuentes desde el gestor de versiones (por ejemplo CVS, Git, Subversion, Mercurial o Microsoft Visual SourceSafe o Team Foundation Source Control) compilarlo, ejecutar tests y generar informes."[iii]

¿por qué no ponéis la entrada concreta?

Como podemos leer en el párrafo anterior, la integración continua es un modelo informático ¿qué es esto? con el que se busca la detección de fallos lo más temprana posible para así evitar costes innecesarios en el ciclo del software. Consiste, entre otras cosas en descargar el proyecto cada cierto tiempo mediante el control de versiones, para luego compilarlo, ejecutar tests y generar informes que nos validarán o no la integración.

Se os olvida que esto debería ser un documento profesional y en un documento de este tipo no se hacen referencias a "lo dicho en clase" o a "los apuntes"

Otra definición que encontramos en los apuntes de la asignatura entiende la integración como la combinación de dos o más herramientas de software.

Aquí no se trata de lo que se "podría" hacer sino de lo que proponéis hacer y hacerlo

En el caso del proyecto que abarcamos en nuestro trabajo, no existen mecanismos de integración continua, pero dado el entorno con el que trabajamos, y el repositorio del que disponemos, podríamos crear un mecanismo de integración continua, usando una herramienta vista en clase como es "Jenkins". Con esta herramienta podríamos construir diariamente nuestro proyecto integrando todos los componentes, lanzar pruebas automáticas previamente definidas, y obtener informes del resultado de éstas para un correcto seguimiento del mecanismo. Además, siendo en nuestro caso un proyecto Maven, la integración mediante la herramienta de Jenkins sería aún más útil y eficiente si cabe.

En la línea de las demás secciones, este apartado está muy pobre, muy poco elaborado, sin horas invertidas, sin concretar las cosas.

EJERCICIO:

Integración del proyecto en la herramienta web de Jenkins.

SOLUCIÓN:

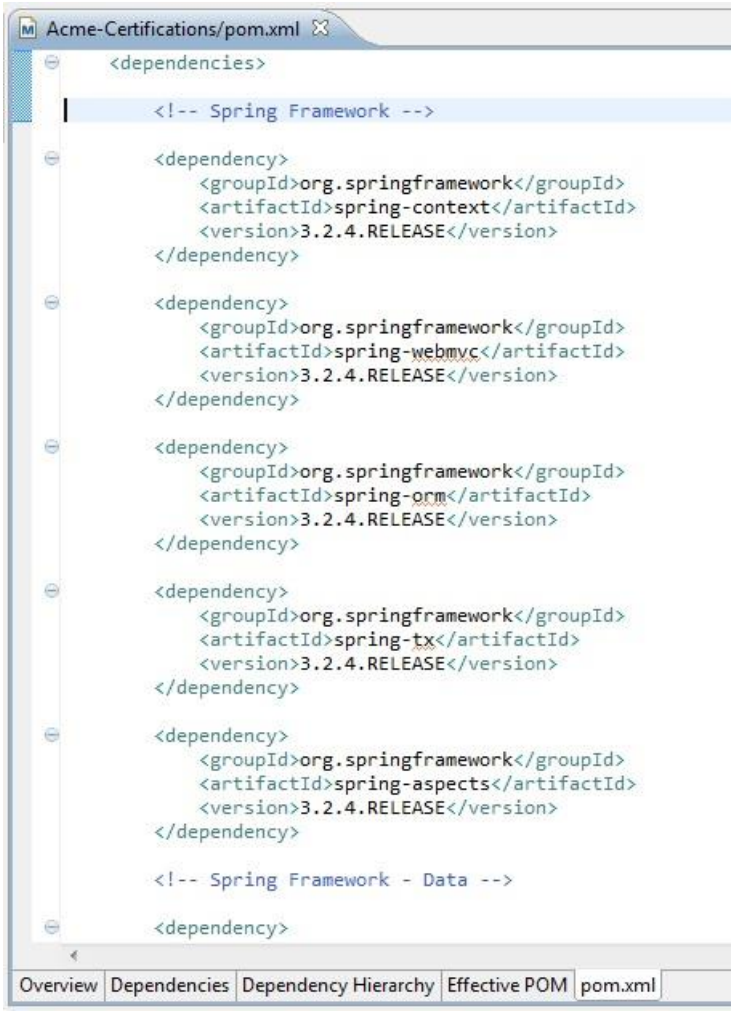
Para ello, lo primero es descargar el paquete de instalación para Windows versión 1.538, para evitar problemas de compatibilidad con, por ejemplo, sistemas operativos de Windows 8.

El primer paso será configurar Jenkins, y dado que el acceso a él es público, deberíamos crearnos unas credenciales de acceso antes de crear nuestro proyecto. A continuación, crearemos una tarea en Jenkins, de modo que soporte nuestro proyecto Maven, por lo que al crearla nos aseguramos de que es tal cuál nuestro proyecto.

A la hora de crearlo necesitaremos indicar la ruta del código fuente de nuestro proyecto (integrado en un repositorio local de subversión, para poder realizar la integración continua), además de indicar que se hace con el código y qué hacer cuando acabe. De este modo, Jenkins acudirá directamente a la última versión válida de nuestro proyecto. Posteriormente, habrá que indicar en el disparador de ejecución cada cuanto tiempo queremos que Jenkins construya el proyecto, lo cual, tras lo visto en la asignatura debería ser una vez al día, para así optimizar el rendimiento y la consecución de errores de nuestro trabajo.

Hay que tener en cuenta que al hacer el checkout de nuestro proyecto en la herramienta de Jenkins, se sube a un repositorio central. Jenkins se programará para que cada cierto tiempo se traiga el código del programa y lo pruebe. Hay opciones de traerlo desde cero siempre, o de actualizar los cambios.

Una vez configurado todo, Jenkins se encargará de construir el proyecto integrando todos los componentes según se indiquen en el pom.xml (véase captura1) y del lanzamiento de una serie de pruebas que han sido previamente programadas (véase captura2). Automáticamente se obtendrán una serie de informes que permitan hacer el seguimiento del proyecto y si han sido o no satisfactorias las pruebas realizadas.



Captura1

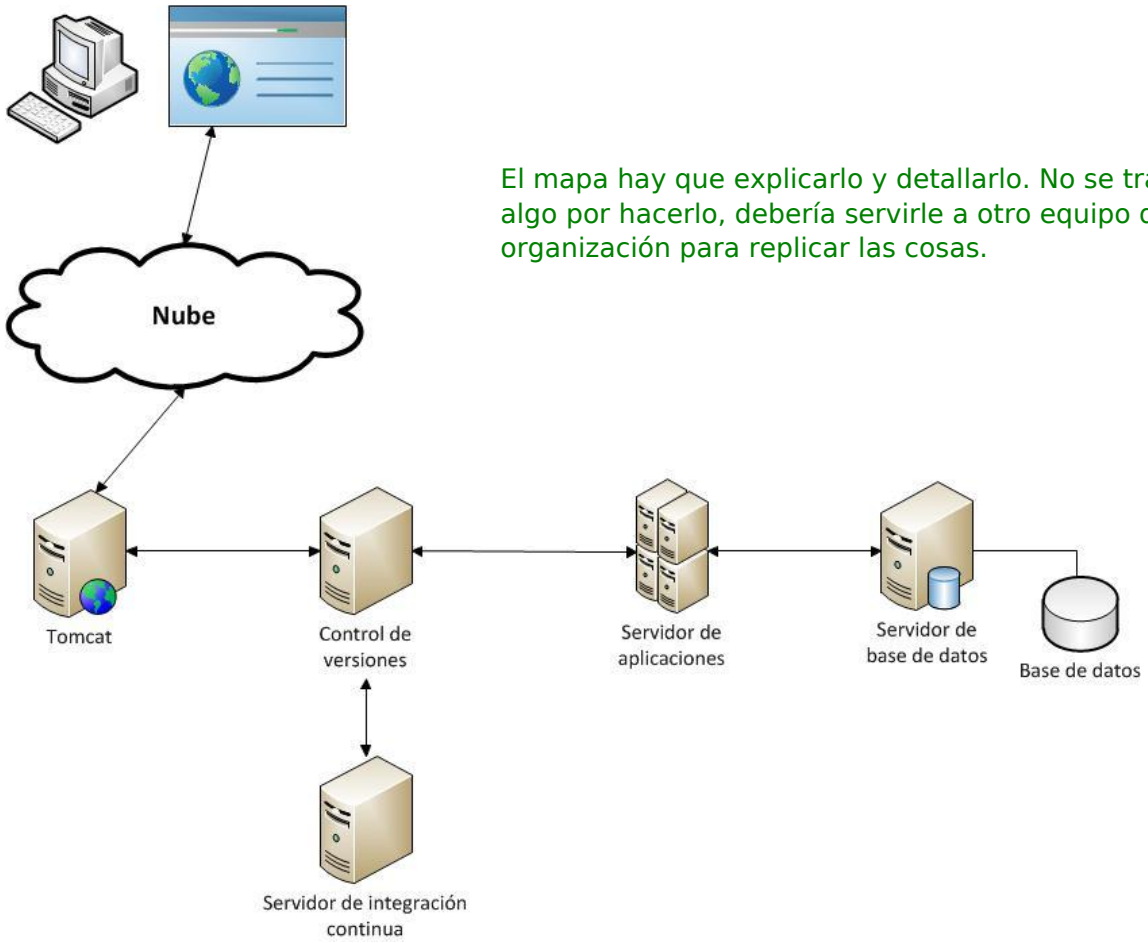
Captura2

En la realización del ejercicio pueden surgir una serie de problemas que se podrán resolver según se indica a continuación:

- Pueden aparecer problemas referentes al puerto donde se inicie Jenkins, por lo que tendremos que abrir la consola de comandos y ejecutar el siguiente script para cambiar el puerto de salida:
    - C:\Users\practica\jenkins\ → java -jar jenkins.war --httpPort=8082
- Puede darnos problemas a la hora de indicar el repositorio de subversión. Lo único que hay que hacer es añadir los credenciales de acceso de projetsii, para que así este error de acceso desaparezca.
- Habrá que instalar el plugin de maven en la aplicación web de Jenkins, por lo que vamos al apartado de administrar Jenkins, y pulsamos en configurar sistema. Ahí debe aparecer Maven, por lo que debemos añadir un instalador, que será el que muestra Apache 3.1.1. Debemos parar y volver a inicializar Jenkins tras este paso, para que se reconozca el nuevo plugin instalado.
- Recordar siempre que la ejecución es siempre en el servidor desde el que procede, y no en nuestro ordenador.

Esta parte está mejor y algo referís a que habéis usado las herramientas pero creo que, en general, en el documento proponéis en los ejercicios cosas que deberían ir en el núcleo de la sección.

Mapa de herramientas



El mapa hay que explicarlo y detallarlo. No se trata de hacer algo por hacerlo, debería servirle a otro equipo de vuestra organización para replicar las cosas.

Nuestra aplicación está directamente conectada a un repositorio de control de versiones, pero una relación que podría añadirse para mejorar el sistema en general, es introducir un **servidor de integración continua** entre ambos.

Conclusiones

Pero eso no debería haber sido un problema

Nos hemos enfrentado a un proyecto que, la verdad, poseía bastante poco contenido del visto en las clases de teoría y prácticas, por lo que nos ha sido imposible el reconocer muchos de los apartados que teníamos que incluir en este documento. A pesar de ello, creemos que ha sido una buena práctica pues hemos afrontado detalles del mundo del software que no tienen la visibilidad de por ejemplo, el código, pero en cambio son tan importantes como él, pues debemos presentar unos proyectos con las herramientas suficientes para garantizar de él un trabajo robusto y eficiente, por lo que el conocimiento y utilización de éstas técnicas es muy importante en lo que al fin y al cabo se requiere, un software eficiente y de calidad.

Respecto a las técnicas utilizadas, hemos aprendido que el uso de repositorios de control de versiones es bastante útil, ya no solo para el salvado y compartimento de éste respecto al trabajo que realicemos sobre él, sino también en el concepto de poder integrar herramientas como la de Jenkins, que nos lo analice periódicamente en busca de fallos para una correcta ejecución. Además, otras herramientas como Maven nos permiten una integración de diferentes frameworks y plugins sólo mediante un archivo de configuración bien implementado, cosa harto difícil pues esto de integrar determinados frameworks en un mismo entorno de desarrollo viene a dar muchos problemas de compatibilidades, y se encuentra en Maven una buena solución. Por otro lado, el uso de herramientas como la proporcionada por Projetsii nos ha permitido una buena organización respecto a las tareas a determinar, estudio de tiempos dedicados a cada tarea… es decir, organización, algo muy útil y que apenas se aprecia su importancia en el tema de desarrollo del software-

En fin, ha sido un bonito proyecto donde quizás no hayamos elegido un buen proyecto inicial puesto que faltaban muchas de las herramientas que teníamos que estudiar y hemos tenido que integrarlas casi todas nosotros, pero en cambio, éste hecho también puede habernos favorecido puesto que ha sido necesario el total estudio de la herramienta previo paso a integrarlo en nuestro proyecto. La clave en futuros softwares ha quedado muy clara, y es, además de una buena implementación de código, una buena documentación de éste con sus respectivas herramientas que integren todo lo visto para que se obtenga lo que todo ingeniero del software debe buscar, es decir, un software de calidad.

Glosario de términos

Baseline: Es la línea base del proyecto del que se parte. Cuando existan cambios importantes al proyecto, actualizaremos la línea de base. No mezclar definiciones con procedimientos

Branch: Es una nueva rama de desarrollo que consiste en una copia del código o la rama de la que deriva. Se utiliza cuando se va a añadir funcionalidad o modificar una parte del código, para no molestar ni interferir en los cambios que otros puedan estar haciendo al mismo tiempo.

Diff: Se llama diff a la acción de elegir qué cambios se añaden y cuáles no, cuando se produce un conflicto dentro del código fuente.

Merge: Mecanismo por el cual integramos los cambios realizados desde el trunk a un branch o desde un branch concreto al trunk.

Path: Se llama path a una versión terminada de algún branch, es decir, realizamos un path .cuando tenemos una versión útil de una rama o branch.

No entiendo esta definición, ¿de dónde sale? ¿dónde la habéis visto?

Bibliografía:

[i] http://iaap.wordpress.com/ ¿dónde está la entrada concreta de la que se ha sacado?

[ii] http://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/12

[iii] http://es.wikipedia.org