Máster Oficial en Software Libre
Ingeniería del Software en Entornos de Software Libre
PEC 2
22/04/16

PEC 2

Índice

Presentación

La actividad **PEC 2** consiste en la segunda prueba de evaluación continuada del curso. Tal como se describió en el plan docente, el curso tiene 3 PECs. Esta segunda PEC corresponde a los módulos 3 - “Control de Calidad y Pruebas” , 4 - “Construcción de software en entorno GNU” y 5 - “Control de versiones – CVS”.

Objetivos

El objetivo de esta PEC es poner en práctica los conocimientos adquiridos después de realizar los módulos 3, 4 y 5 del curso. En concreto, mediante la PEC el estudiante:

- Familiarizarse con la terminología relacionada con el control de calidad y pruebas que es de uso común en la ingeniería del software.
- Conocer las principales técnicas de comprobación manual de software usadas en la ingeniería del software y en entornos libres.
- Conocer los sistemas de comprobación de unidades de software y entender su funcionamiento.
- Conocer los sistemas de gestión de errores, la anatomía de un error, su ciclo de vida, y familiarizarse con el sistema de gestión de errores libre Bugzilla.
- Familiarizarse con la parte cliente de las herramientas de control de versiones para poder seguir la evolución de un proyecto de software donde participen diversos desarrolladores.

Enunciado

PARTE A
Cada pregunta contestada correctamente tiene una puntuación de 0,5 puntos. Cada pregunta contestada incorrectamemte resta 0,25 puntos. Se debe obtener al menos 2,5 puntos para poder promediar con la Parte B.

1. ¿Qué afirmación es falsa?
   a. Durante el mantenimiento de una aplicación, deberé hacer pruebas de stress para comprobar que la funcionalidad implantada es correcta a nivel funcional
   b. Durante el mantenimiento de una aplicación, deberé hacer pruebas de usabilidad para conocer su la funcionalidad implantada es correcta a nivel de usuario
   c. Durante el mantenimiento de una aplicación, deberé hacer pruebas de regresión si quiero saber si la funcionalidad implantada previamente es correcta

Justifica la respuesta

a. Estas pruebas no tienen como objetivo el comprobar la funcionalidad sino medir el rendimiento de un sistema bajo cargas elevadas de trabajo.

2. Si utilizo Selenium para realizar mis pruebas. ¿Qué afirmación es cierta?
a. Estoy obteniendo el code coverage de la aplicación en su parte visual
b. Estoy haciendo black-box testing
c. Estoy haciendo withe-box testing
d. Estoy haciendo una mezcla entre black-box testing y white-box testing

3. Dispongo de un sistema que mediante un archivo de configuración podrá utilizarse para reservar vuelos con la compañía A o con la compañía B. La única diferencia estriba en que el precio del billete a emitir será un 20% más caro con la compañía B que con la compañía A. Por economía lo mejor es considerar el cálculo del precio como una única unidad de comprobación. ¿Verdadero o Falso?. Justifica la respuesta.

FALSO, dado las características de la unidad de comprobación que debe cumplir; ser desatendida, universal y atómica y como en este ejemplo no basarse en alternativas para configurarla.

4. Si liberamos la version 1.0 de un programa, y queremos comenzar a desarrollar la versión 1.1, mientrás corregimos los errores de la versión 1.0. Debemos:
a. Crear una etiqueta con los archivos que componen la versión 1.0 y otra para los archivos que componen la versión 1.1
b. Crear la rama de la versión 1.1 a partir de la rama principal que contiene la versión 1.0
c. Congelar la versión 1.0 y únicamente modificar la versión 1.1
d. La respuesta b y c son correctas

Justifica la respuesta.

Lo más recomendable es la creación de una rama a partir de la principal que contiene la versión 1.0 y así continuar con una versión en desarrollo derivada.

5. En GNU Automake, si encontramos una instrucción como esta “load_CUSTOM = 1” debemos suponer que:
a. la notación no es correcta
b. se trata de una variable derivada
c. se trata de una variable primaria
d. Ninguna de las anteriores

Justifica la respuesta.

La variable primaria CUSTOM no existe, lo que hace imposible tener una variable derivada

6. Necesito modificar un archivo de un programa existente, por lo cual importo del CVS dicho archivo a mi directorio local. Mi cambió durará un mes y lo haré de manera incremental, por lo cual es posible que durante ese periodo otros miembro del equipo tengan que modificarlo. La solución para que todos los miembros del equipo tengan siempre la última versión de mis modificaciones es hacer un update diario de mi directorio local. ¿Verdadero o Falso?. Justifica la respuesta.

FALSO, para cumplir esto es necesario hacer commit desde el directorio local para modificar el repositorio y que todos los miembros del equipo hagan update y ver estos cambios.

7. He acabado de modificar el archivo de la pregunta 6, pero he tenido que crear un nuevo archivo para completar la funcionalidad. Para poder añadirlo todo en CVS lo que debo hacer es ejecutar el comando "$cvs add archivo2.txt". ¿Verdadero o Falso?. Justifica la respuesta.

FALSO, así solo se añadiría, pero para que también la modificación del archivo se vea reflejada hay que ejecutar un $csv commit luego.

8. Los sistemas de control de versiones se basan siempre en disponer de un repositorio en un lugar centralizado. Los directorios locales solo son lugares de trabajo. ¿Verdadero o Falso?. Justifica la respuesta.

FALSO, puede darse el caso que se constituyan como una copia del repositorio

PARTE B

Eres el analista de sistemas de una empresa que ha sido elegida para desarrollar una aplicación de una compañía aérea low-cost.

Un programador de tu empresa ha desarrollado una de las pantallas (Reserva de Vuelos)

a) Haz un esbozo de como debería ser esa pantalla y adjuntálo como imagen (0,5 puntos)

b) Como Analista deberás realizar 5 guiones de prueba de la pantalla para que un testeador las lleve a cabo (1 punto).

Ver archivo adjunto Guiones de Prueba.pdf

c) Ahora ponte en el rol de testeador (te cuento un secreto, siempre se cometen errores) imagina 5 de ellos y realiza el informe correspondiente para cada uno. (1 punto)

Ver archivo adjunto Informes de errores.pdf

d) Instala GIT. Inicializa un repositorio en un directorio que hayas creado previamente y que contanga un archivo pruebaGIT.txt con el texto “Esto es una prueba” en su contenido.

Este procedimiento se realizó bajo la distribución Ubuntu 15.10. Primero se realiza actualización de los repositorios con `osboxes@osboxes:~$ sudo apt-get update`, posteriormente se instala GIT `osboxes@osboxes:~$ sudo apt-get install git` . Con esto queda instalado el servidor, ahora se procede a verificar la versión

```
osboxes@osboxes:~$ git --version
git version 2.5.0
```

y a configurar el servicio configurando variables

```
osboxes@osboxes:~$ git config --global user.name "Andres Arteaga"
osboxes@osboxes:~$ git config --global user.mail "earteaga@uoc.edu"
```

Se crea el directorio para el repositorio, dentro de éste se crea el archivo pruebaGIT.txt, con el texto “Esto es una prueba”.

```
osboxes@osboxes:~$ git config --global user.name "Andres Arteaga"
osboxes@osboxes:~$ git config --global user.mail "earteaga@uoc.edu"
osboxes@osboxes:~$ mkdir git
osboxes@osboxes:~$ cd git
osboxes@osboxes:~/git$ echo "Esto es una prueba" > pruebaGIT.txt
osboxes@osboxes:~/git$ more pruebaGIT.txt
Esto es una prueba
```

Se inicializa el repositorio

```
root@osboxes:/home/osboxes/git# git init
Initialised empty Git repository in /home/osboxes/git/.git/
```

Se agrega el archivo creado al repositorio con

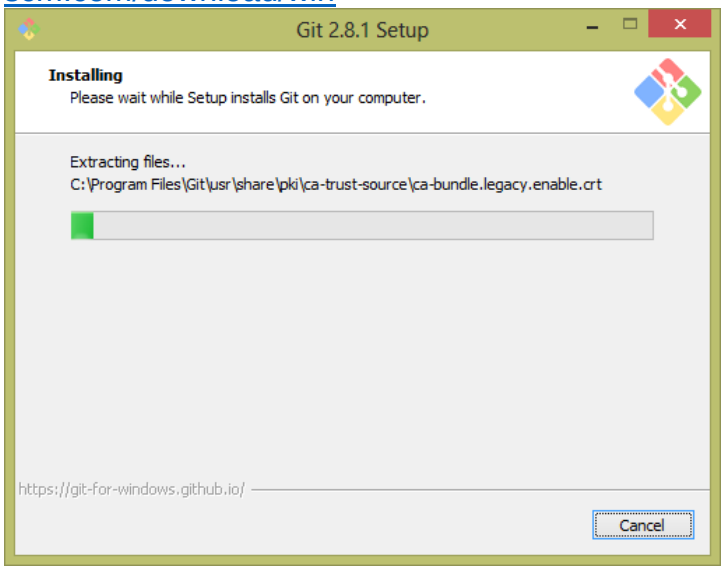
```
root@osboxes:/home/osboxes/git# git add pruebaGIT.txt
```

Resta ejecutar ***commit*** para efectuar los cambios en el repositorio

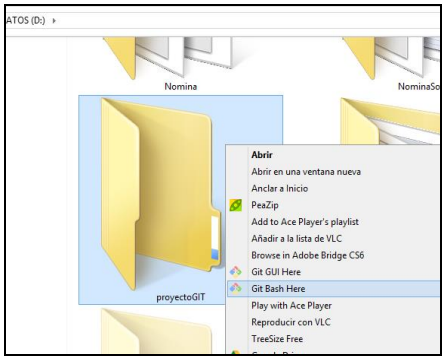
```
osboxes@osboxes:~/git$ sudo git commit -m "version 1"
[master (root-commit) 90afe66] version 1
 1 file changed, 1 insertion(+)
 create mode 100644 pruebaGIT.txt
```

e) Crea un directorio local donde harás tus modificaciones y sigue el archivo pruebaGIT.txt

En un equipo distinto, en este caso con Windows se instala GIT desde https://git-scm.com/download/win



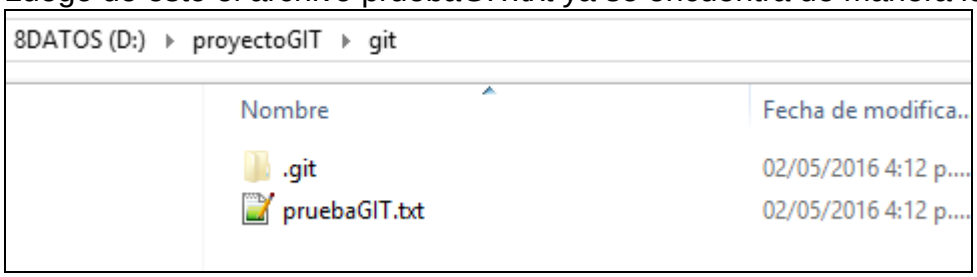
Se crea un directorio en el equipo llamado proyectoGIT y luego se despliega el menú contextual de la carpeta para seleccionar la opción **Git Bash Here**



Esta acción arranca una consola en la que se digita el comando para hacer el seguimiento al proyecto creado previamente en el servidor: ***git clone osboxes@172.16.142.134:/home/osboxes/git***

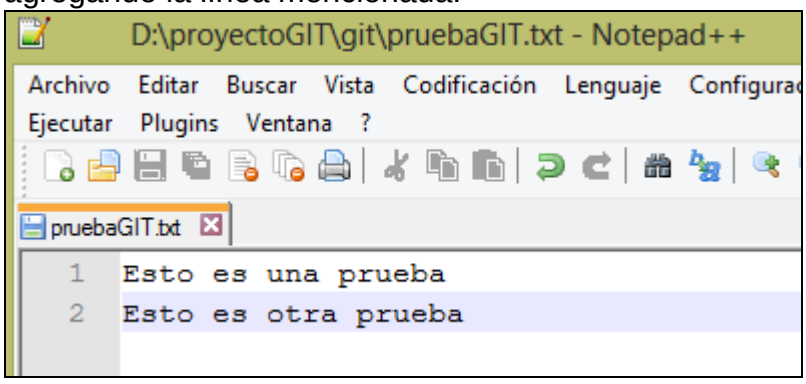
```
Andrés@QuadCore MINGW64 /d/proyectoGIT
$ git clone osboxes@172.16.142.134:/home/osboxes/git
Cloning into 'git'...
The authenticity of host '172.16.142.134 (172.16.142.134)' can't be establis
hed.
ECDSA key fingerprint is SHA256:ehYTBwac78qHM9VsbCb8FUczOtZeKFI2sTMH5ivkBw4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.142.134' (ECDSA) to the list of known hos
ts.
osboxes@172.16.142.134's password:
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.
Checking connectivity... done.
```

Luego de esto el archivo pruebaGIT.txt ya se encuentra de manera local.



f) Modifica el archivo y añade “Esto es otra prueba”. Importa el contenido de este directorio, al repositorio que has creado.

Se modifica el archivo de manera local en el equipo Windows con cualquier editor de texto agregando la línea mencionada.



Una vez guardados los cambios se adicionan al repositorio

```
Andrés@QuadCore MINGW64 /d/proyectoGIT/git (master)
$ git push origin master
osboxes@172.16.142.134's password:
Everything up-to-date
```

Con el comando ***git log*** se comprueba que los cambios se han importado al repositorio creado

```
Andrés@QuadCore MINGW64 /d/proyectoGIT/git (master)
$ git log
commit bb2b8a9fa81fbbcdcb1a075fb3236efb27fb47ab
Author: Andrés Arteaga Castillo <Andrés Arteaga Castillo>
Date:   Mon May 2 16:43:37 2016 -0500

    actualizacion pruebaGIT.txt

commit 90afe66beb5446ef96f04893a2c4c0c07153ae6a
Author: Andres Arteaga <earteaga@uoc.edu>
Date:   Mon May 2 21:17:33 2016 +0100

    version 1

Andrés@QuadCore MINGW64 /d/proyectoGIT/git (master)
```

g) Haz un commit del archivo (d,e, f y g 2 Puntos)

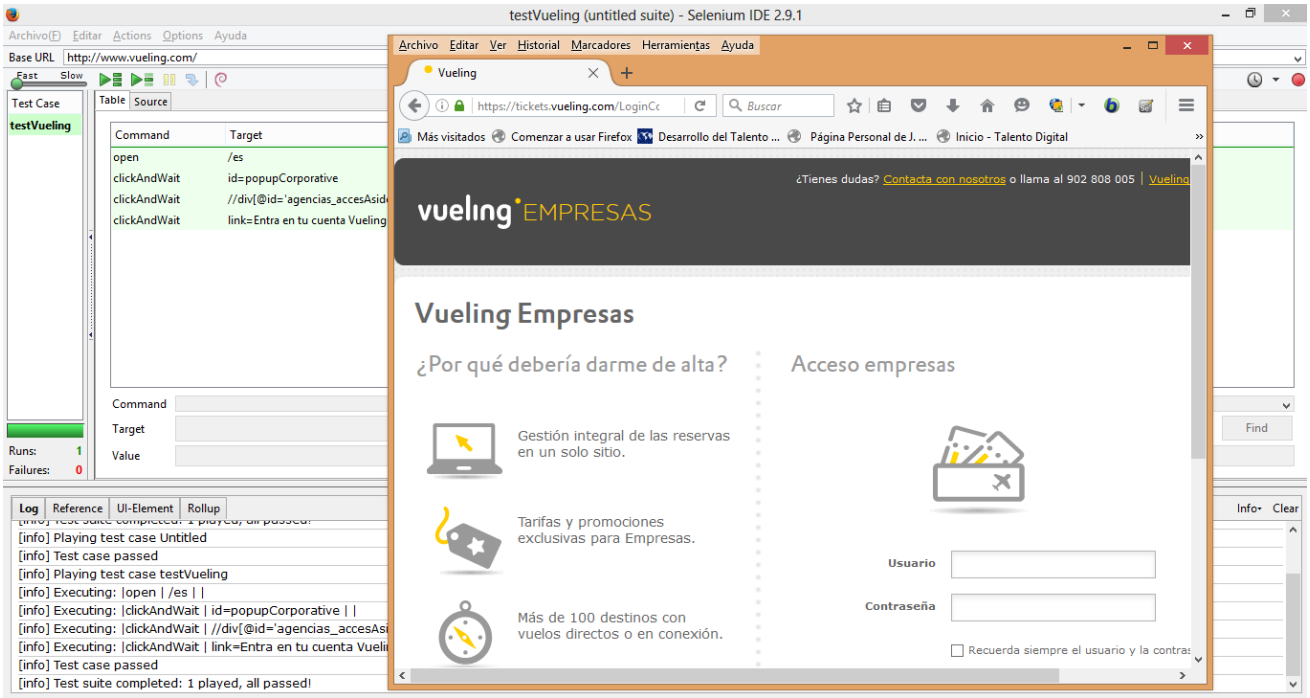
Se agregan los cambios el directorio local

```
Andrés@QuadCore MINGW64 /d/proyectoGIT/git (master)
$ git add pruebaGIT.txt

Andrés@QuadCore MINGW64 /d/proyectoGIT/git (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   pruebaGIT.txt
```

Se ejecuta el ***commit*** para los cambios hechos

```
Andrés@QuadCore MINGW64 /d/proyectoGIT/git (master)
$ git commit -m "actualizacion pruebaGIT.txt"
[master bb2b8a9] actualizacion pruebaGIT.txt
 1 file changed, 1 insertion(+)
```

h) Utilizando Selenium IDE, debes automatizar la siguiente prueba:
Entrar en www.vueling.com
Seguir el enlace “Empresas” de la parte superior
Seguir el enlace “Quiero darme de alta”
Seguir el enlace “Entra en tu cuenta Vueling Empresas” (1,5 puntos)



Ver archivo adjunto testVueling.html

Formato de entrega

La PEC a entregar tendrá las siguientes características para la PARTE B:

- Para el punto a) entrega al menos, una imagen insertada al documento de la solución con la pantalla diseñada
- Para el punto b) entrega un fichero con los 5 guiones de prueba
- Para el punto c) entrega un fichero con los 5 informes de errores
- Para el punto d,e, f y g) tienes que explicar paso a paso todo lo que vas haciendo e ilustrarlo con los "pantallazos". Se debe comprender perfectamente con explicaciones los comandos que ejecutas y su resultado correspondiente..
- Para el punto h) entrega el archivo que genera Selenium para la automatización.
- Una vez que tengas todos estos ficheros, comprimelos en un fichero con extensión **.zip**

El documento se entregará a través del apartado Evaluación -> Entrega y registro de EC

La fecha de entrega máxima de la PEC es el día 3 de Mayo de 2016.