

RubyDT

Manejo del Ambiente Controlado

Versión 1.0

Historia de revisiones

Fecha	Versión	Descripción	Autor
30/08/2014	1.0	Creación del documento	Líber Azambuya
14/08/2014	2.0	Se agrega manejo del pull request	Líber Azambuya

Contenido

1.DEFINICIÓN DEL AMBIENTE CONTROLADO.....	3
2.DESCRIPCIÓN DEL USO DEL AMBIENTE CONTROLADO.....	3
2.1. ESTRUCTURA:.....	3
2.2. USO DEL REPOSITORIO.....	4
2.3. MANEJO DE HISTORIAS.....	4
3.RESPONSABILIDADES.....	5

1. Definición del Ambiente Controlado

Para el ambiente controlado se utilizara el sistema de control de versiones GIT ya que al ser un sistema distribuido nos permite mayor flexibilidad en el momento del desarrollo.

Por otro lado como repositorio central del proyecto se decide utilizar la plataforma GitHub debido a las prestaciones que de gran utilidad que ofrece para el trabajo en equipo.

2. Descripción del uso del Ambiente Controlado

Inicialmente todos los integrantes del grupo deben descargar el sistema de control de versiones GIT e instalarlo en su computador ¹. Luego de tener el repositorio instalado, cada integrante tiene que crearse una cuenta en

GitHub ² e indicarle al responsable de SCM el usuario creado así éste lo agrega como colaborador en el repositorio del grupo creado en dicha herramienta.

Dentro del repositorio creado en GitHub, todos los miembros tendrán permisos de administrador, es decir, podrán realizar lectura y escritura de código en el repositorio remoto. Esta decisión es tomada debido a la corta duración del proyecto para proporcionar mayor agilidad en el desarrollo. De todas formas mas adelante en este documento se describirán las pautas a tener en cuenta cada vez que un miembro del grupo quiera subir alguna modificación del proyecto al repositorio remoto de GitHub.

2.1. Estructura

Inicialmente se tendrán dos ramas en el repositorio Definición del Ambiente Controlado, la rama principal (master) donde se encontrara la linea base del proyecto, y una rama para el desarrollo. Es importante tener en cuenta que **NO se trabajara directamente sobre la rama principal**, de lo contrario, los cambios se realizan en la rama de desarrollo y cuando estos estén aprobados y verificados serán mergeados a la rama principal bajo el control del responsable de SCM.

Eventualmente se pueden crear nuevas ramas para atender algún error especifico o si la situación lo justifica.

La rama principal de la cual va a partir la rama de desarrollo, estará estructurada de la siguiente forma:

1. Una carpeta que contendrá el aplicativo en desarrollo.

¹ <http://git-scm.com/downloads>

² <https://github.com/>

2.2. Uso del Repositorio

Para el uso del repositorio hay que tener en cuenta las siguiente pautas:

1. Debido a que Git es un sistema distribuido, cada miembro tendrá un repositorio local, el cual es un clon del repositorio remoto. Entonces para mantener mayor claridad, se define la ruta en la cual los miembros deben clonar el repositorio remoto:
primario maestro\pis\nombreRepositorioRemoto
ejemplo: C:\pis\RubyDTRepo
2. Antes de comenzar a trabajar en el repositorio local, se debe hacer un **pull** del repositorio remoto, para asegurarse que se esta trabajando sobre las ultimas versiones subidas por el equipo de desarrollo.
3. El proceso a seguir a la hora de comenzar a desarrollar una historia es el siguiente:
 - 3.1. Crear una branch que parta de la branch de desarrollo, con la nomenclatura estipulada, para el desarrollo de la historia.
El objetivo de esta branch es que al finalizar la historia se realizará un **pull request** donde se van a comparar la branch de desarrollo del proyecto con la branch perteneciente a la historia y se podrán visualizar los cambios realizados por la persona para la revisión de código.
 - 3.2. Cada vez que se realice un merge a la branch de desarrollo, todos los integrantes del grupo de implementación deben integrar estos cambios a su branch de trabajo realizando un merge desde la branch de desarrollo a la branch de trabajo.
4. El proceso a seguir a la hora de subir modificaciones al repositorio remoto consta del siguiente:
 - 4.1. Solicitar la revisión de código al equipo de desarrollo mediante un **pull request**.
 - 4.2. Dos integrantes del equipo serán elegidos al comienzo de cada sprint como encargados de realizar la revisión de código y decidir si se deben hacer modificaciones del mismo con el objetivo de mejorar la codificación del proyecto.
 - 4.3. Luego de la aprobación de los dos integrantes se debe hacer un merge a la branch de desarrollo correspondiente al sprint.
5. Al final de cada sprint cuando el código ya fue aprobado y verificado, el responsable de SCM con soporte del equipo de desarrollo se encargara de realizar el merge de la rama de desarrollo a la rama principal.

6. El responsable de SCM deberá etiquetar la rama principal y crear una nueva rama de desarrollo que parta de la etiqueta realizada para continuar codificando en el siguiente sprint.

2.3. Manejo de Historias

Para el manejo de historias se utilizara el servicio de **issues** que proporciona GitHub.

2.4. Manejo de Bugs

Para manejar los bugs también se va a utilizar el servicio de issues proporcionado por github de la siguiente forma:

Los responsable y asistente de verificación realizaran pruebas de sistema y reportaran los bugs encontrados asignándolo a algún desarrollador. Dicho desarrollador corregirá el error reportado y luego lo asigna al verificador que encontró el error para que el mismo pueda corroborar la corrección del mismo, de ser así se cierra la issue correspondiente al bug.

3. Responsabilidades

Como se menciona anteriormente, todos los miembros del grupo tendrán permiso de escritura y lectura del repositorio, por lo cual la responsabilidad de mantener el ambiente controlado consistente es de todos los integrantes.

De todas formas el responsable de SCM es el encargado de verificar que las pautas mencionadas en el punto anterior sean cumplidas y deberá dar soporte a los integrantes del grupo ante cualquier cuestionable que surja.