

# Challenge Accepted

## Manejo del Ambiente Controlado

### Versión 2.1

#### Historia de revisiones

Fecha	Versión	Descripción	Autor
30/08/2014	1.0	Creación del Documento.	Joaquín Gatica.
31/08/2014	1.1	Revision final de SQA.	Federico Blumetto.
31/08/2014	1.2	Verificación de Documento.	Gonzalo Sintas.
13/09/2014	2.0	Revisión para Iteración 2, Fase 1.	Joaquín Gatica
14/09/2014	2.1	Revisión del documento.	Gonzalo Sintas

**Contenido:**

- [1. Definición del Ambiente Controlado](#)
  - [1.1. Ambiente Controlado para Documentos](#)
  - [1.2. Ambiente Controlado para Código Fuente](#)
- [2. Descripción del uso del Ambiente Controlado](#)
  - [2.1. Cliente](#)
  - [2.2. Autenticación](#)
  - [2.3. Funcionamiento](#)
    - [2.3.1. Configuración inicial](#)
    - [2.3.2. Seguimiento](#)
  - [2.3. Entornos de desarrollo](#)
  - [2.4. Respaldos](#)
- [3. Responsabilidades](#)

# 1. Definición del Ambiente Controlado

El ambiente controlado se dividirá en dos partes:

1. Ambiente controlado para documentos
2. Ambiente controlado para código fuente.

Se decidió esta separación, ya que las herramientas de gestión de código no permiten el nivel de versionado que poseen las herramientas de edición colaborativa de documentos, y vice versa.

## 1.1. Ambiente Controlado para Documentos

Este ambiente se definió exclusivamente mediante el uso de Google Drive como herramienta de almacenamiento, edición y versionado de documentos. Se evaluó que el uso de esta herramienta no requiere de capacitación, ya que todos están familiarizados con editores de texto y hojas de cálculo, y todos han usado el versionado que viene incorporado con esta herramienta. Lo que se realizó fue un consenso sobre la estructura de carpetas y nombres de archivo.

## 1.2. Ambiente Controlado para Código Fuente

Para el código fuente, sin embargo, se creó el ambiente controlado sobre un repositorio Git, cuyo servidor remoto se alojará en SourceForge.net, que garantiza disponibilidad casi del 100% del tiempo. De todas maneras, siendo Git un sistema distribuido, en caso que el servidor no esté activo, el versionado se puede realizar igual.

Dada la naturaleza del proyecto, se han creado tres repositorios distintos, uno para cada plataforma:

1. Server: para versionar el código de la aplicación del servidor Azure.
2. Android: para versionar el código de la aplicación móvil para dispositivos Android.
3. Windows Phone: para versionar el código de la aplicación móvil para dispositivos Windows Phone.

## 2. Descripción del uso del Ambiente Controlado

### 2.1. Cliente

Se utilizarán 2 clientes principales para el versionado con Git:

1. SourceTree: como gestor de repositorios a través de GUI
2. Git Bash: como gestor de repositorios a través de la línea de comandos

A la vez, se utilizarán las funcionalidades de versionado de los IDEs a usar, por ejemplo, a la hora de indicar los archivos que han sufrido cambios que no pertenecen a ningún commit.

### 2.2. Autenticación

Cada miembro del equipo se ha creado un usuario en SourceForge.net, y el SCMR les ha dado acceso completo de R/W a los repositorios correspondientes.

### 2.3. Funcionamiento

#### 2.3.1. Configuración inicial

##### Clone

Cada usuario se clonará el repositorio remoto una única vez al comienzo de la implementación. Esta acción le creará una copia del repositorio en SourceForge para que pueda trabajar localmente con Git.

#### 2.3.2. Seguimiento

##### Pull

Al inicio de cada día de desarrollo, y con cierta frecuencia durante el día, el usuario obtendrá todos los cambios realizados desde el último *pull* utilizando este comando. De esta manera obtendrá la última versión del código, sobre el cual podrá desarrollar.

Siempre se utilizará el flag *rebase* al hacer *pull*, para mantener la consistencia entre todos los usuarios del repositorio, y obtener un historial de cambios ordenado.

##### Resolución de conflictos

En caso de ocurrir conflictos al hacer *pull*, se deben resolver para poder finalizar el *rebase*. Una vez resueltos los mismos, el *pull* finaliza con éxito y se puede continuar.

## Trabajar con los archivos

Trabajar normalmente continuando el desarrollo en el Working Copy.

### Add/remove

Git provee una funcionalidad de Staging Area para los commits, es decir, que se deben agregar los archivos que van a un nuevo commit a esta sección, y luego el mismo se realiza sobre toda esa área. al finalizar una parte del desarrollo o una nueva funcionalidad, agregar al Staging Area los archivos a *commit*ear.

Esto es útil cuando tenemos cambios en el código por varios cambios o funcionalidades, pero se agregan a esta área solo los que van en un *commit*: una funcionalidad o un cambio individual.

### Commit

Sobre los archivos presentes en el Staging Area, realizar el commit, siempre con un mensaje descriptivo de lo que se está haciendo. Se deben separar los commits por funcionalidad o cambio realizado, y evitar un *commit* con varios de ellos juntos.

### Push

Previo a enviar los cambios al repositorio de origen, realizar un *pull* nuevamente para obtener los cambios que se pudieron haber hecho desde otras copias del repositorio. Una vez resueltos los conflictos, si los hubo, realizar un *push* al servidor.

Esta acción envía todos los commits realizados localmente al repositorio compartido, para hacerlos accesibles al resto de los usuarios.

## 2.3. Entornos de desarrollo

Debido a las distintas actividades que se realizan sobre el mismo código, y aprovechando la facilidad que proporciona Git para el trabajo en branches, se utilizarán las siguientes en cada repositorio:

- **"master" branch:** tendrá la versión entregada en cada iteración, y por ende únicamente el código aceptado en la Línea Base.
  - Única actividad: SCM.
- **"dev" branch:** tendrá el código sobre el que trabajarán los implementadores día a día. De esta rama se podrán derivar otras según se necesite.
  - Única actividad: Implementación.
- **"sqa" branch:** tendrá el código sobre el cual trabajarán los responsables de SQA.
  - Única actividad: SQA y Verificación.

## 2.4. RespalDOS

Dada la confiabilidad de SourceForge, y el hecho de que Git es un sistema distribuido, no se realizarán respaldos diarios del código. Sin embargo, una copia semanal del repositorio se guardará en la línea base, y podrá ser utilizada para recuperación del código.

## 3. Responsabilidades

Todos los usuarios del repositorio tienen las siguientes responsabilidades:

- Trabajar en la rama correspondiente a la actividad que realiza.
- Utilizar *rebase* en cada *pull*.
- Resolver los posibles conflictos sin sobrescribir el código de otros.
- Proveer mensajes descriptivos en los *commits*.
- Realizar un *commit* por cada cambio/parte de funcionalidad, evitando el uso de un solo *commit* grande.

El SCMR será responsable de:

- Mantener la branch "master" o línea base, con los tags correspondientes para indicar versiones.
- Capacitar al resto del equipo en el uso de las herramientas.
- Recuperar, o ayudar a hacerlo en caso de asignar la tarea a quien corresponda, si llega a ocurrir pérdida de código o información.
- Proveer los instaladores de las distintas herramientas, o indicar que versiones se utilizarán de cada una.