******

Plan de Gestión de Configuración

Versión 1.1

**Integrantes:**

* Angeles Rojas, Jorge Alexander.
* Arizola Yánac, Carlos.
* Arotuma Martinez, Victor Miguel.
* Arteaga Quico, Alisson Diane.
* Barreto Trujillo, Larry Steve.
* Castillo Chávez, Luigi Jair.

**CONSULTORA:** Programsy

**PROFESORA:** Lenis Rossi Wong Portillo

**CURSO:** GESTIÓN DE LA CONFIGURACIÓN Y MANTENIMIENTO



**Historial de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 14/09/2018 | 1.0 | Creación del documento | Alisson Arteaga |
| 14/09/1028 | 1.1 | Agregado la introducción | Equipo Programsy |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tabla de contenido

[1. Introducción 4](#_Toc525252369)

[2. Introducción 5](#_Toc525252370)

[2.1. Introducción 5](#_Toc525252371)

1. Introducción

-Problemática de la empresa

-Propósito

-Finalidad del Plan

Actualmente nuestra consultora cuenta con 5 proyectos de los cuales 4 en producción y uno de ellos en desarrollo que es SGVIF. Nuestra consultora cuenta con diversos productos de software los cuales durante su desarrollo surgieron inconvenientes en el manejo de las solicitudes de cambios que han generado pérdida de tiempo por parte del equipo al querer realizar cambios de versiones y dificultad el querer restaurar versiones funcionales, nuestro equipo identifica esto como una problemática originado por no contar con una guía o algún documento que permita llevar a cabo una correcta gestión de versionamiento. Debido a ello estamos realizando la siguiente propuesta para contar con un Plan de Gestión de la Configuración.

Este plan se espera que sea aplicado a todos los proyectos de nuestra consultora, sean pequeños, medianos o grandes. La gestión de la configuración del software es un proceso que incluye aplicar una serie de actividades a lo largo del ciclo de vida del software y describe un conjunto de actividades que deben ser llevados a cabo durante su desarrollo. Este conjunto de actividades que incluyen la identificación, control, estado, auditorias y revisiones de la configuración hacen que la Gestión de la Configuración sea calificada como una garantía de calidad.

La finalidad de este plan es garantizar que no se realicen cambios incontrolados y que todos los participantes en el desarrollo del software dispongan de una versión adecuada de los productos que manejan durante el ciclo de vida de desarrollo del software y de este modo facilitar el mantenimiento. Este plan pretende llevar un control, obtener informe de estado de desarrollo en el que se encuentra un software que permita aumentar la calidad del producto, logrando de esta forma una mayor satisfacción del cliente y de este modo también la mejora de nuestra consultora.

1. Introducción

## HERRAMIENTAS, ENTORNO E INFRAESTRUCTURA

### HERRAMIENTAS

#### GIT

Es un sistema de control de versiones distribuido (VCS) escrito en C originalmente para albergar el código de Linux. Git permite la creación de una historia para una colección de archivos e incluye la funcionalidad para revertir la colección de archivos a otro estado.

**CARACTERÍSTICAS**

* El desarrollo de la aplicación será únicamente nuestro, pudiendo decidir qué parte de nuestro proyecto compartimos y con quién.
* El control de versiones se puede realizar dentro de la propia red con una mayor velocidad de acceso y escritura, así como eliminando el requisito de contar con una conexión a internet obligatoria.
* Se puede crear diferentes ramas sobre las que aplicar nuestras modificaciones en entornos aislados de la línea principal de desarrollo.
* Si en la creación de una rama del proyecto encontramos que uno de los cambios incluidos se integra sin presentar conflictos con las diferentes partes de nuestra aplicación, podremos hacer converger dicha ramificación con el desarrollo principal de forma sencilla y segura.
* Las ramificaciones nos preparan un entorno aislado de pruebas sobre el desarrollo de la línea central de nuestra aplicación.
* Se admite una multitud de configuraciones que nos permitirán dentro de su estructura organizar el trabajo tal y como deseemos.
* Se hace uso de sistemas de árbol SHA1, lo que asegura que hasta que no se realice la comprobación del cifrado o firma, los cambios no se escribirán en el servidor.
* Es un sistema de control de versiones libre de código abierto.

**LIMITES**

* No permite indexar directorios.
* Teniendo una gran curva de aprendizaje, con solo 152 comandos por aprender, la documentación de muchos de estos comandos es obsoleta.

En la figura 03 se muestra el flujo que se utilizará.



**Figura 01.** Flujo de Git

Para realizar el seguimiento de nuestro trabajo, se define los distintos estados en que pueden estar nuestros ficheros en Git.

**Sin seguimiento**: Se encuentran en ese estado todos los ficheros que han sido creados fuera de Git, y nunca los hemos incorporado al gestor de versiones.

**Sin modificar**: Se trata de la situación base en la que se encuentran todos los ficheros sujetos al gestor de versiones, y sobre los que nunca se ha hecho nada.

**Modificado**: Todos aquellos ficheros que están bajo el control de versiones, y sobre los que hemos realizado alguna modificación.

**Preparados**: Son archivos que hemos modificado, y consideramos que ya están listos para entregar, por lo que los asignamos a una entrega.

**Liberado**: Se trata de la entrega de los archivos al repositorio en el que se van almacenando las versiones.

#### GITHUB

Github es una plataforma creada para facilitar el desarrollo colaborativo de software, nos permite alojar proyectos como repositorios en la web gratuitamente, por lo general de forma pública, aunque podemos alojar los proyectos de modo privado, si pagamos una pequeña suscripción mensual.

**CARACTERÍSTICAS**

* Guardar en determinado momento los cambios efectuados a un archivo o conjunto de archivos, con la oportunidad que tener acceso a ese historial de cambios, ya sea para regresar a una de esas versiones o para hacer comparaciones entre ellas.
* Como muchos desarrolladores tienen de forma pública sus proyectos en Github es posible acceder a su código, leerlo, estudiarlo y aprender de él, e incluso podrías hacer cambios y experimentar sin afectar el código original.
* Si luego de copiar un proyecto (hacer Fork) haces ajustes que arreglan bugs o introducen una nueva funcionalidad, puedes proponerle al dueño del proyecto que integre tus cambios en su código a través de un Pull Request.
* Permite trabajar conjuntamente en una idea con un amigo o colega, además se puede invitar a otros usuarios como colaboradores, permitiendo así que estos lean y escriban directamente sobre el repositorio.
* Posee un estupendo visor de código mediante el cual, a través del navegador, podremos consultar en cualquier instante el contenido de un archivo determinado.
* Tiene un sistema de notificaciones con el cual se puede estar al tanto de las actividades alrededor de un repositorio en el cual estas participando o uno en el cual estés interesado.
* Cada proyecto creado en Github incluye un sistema de seguimiento de problemas, del estilo sistema de tickets.
* Es una plataforma web, por tanto, es independiente del sistema operativo que utilices, y además Git que es la herramienta que si requiere instalación es compatible con todos los sistemas; Linux, OSX y Windows.
* Cuenta con una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero.
* Es completamente gratis e ilimitado para proyectos públicos.

**LIMITES**

* Github tiene dos modalidades: una gratuita y otra de pago. En la versión gratuita, se podrán crear “n” repositorios de acceso público; pero para el uso de repositorios privados tendrá un coste.
* Limitación de versionamiento para elementos de texto plano.

### ENTORNO

El ambiente de trabajo estará compuesto por:

**Desarrolladores:** Las personas involucradas en el proyecto, tienen acceso para poder modificar los documentos del repositorio.

**Administrador:** Se encarga de verificar los cambios de los documentos, y revisar que se trabaja en el branch establecido. Así mismo debe dar los permisos necesarios a los desarrolladores para realizar el desarrollo.

**Repositorio Remoto de producción:** Github.

**Repositorio Remoto de pruebas:** Github.

### INFRAESTRUCTURA

Se manejará 2 tipos de ramas o branch, que nos va servir para controlar mejor los commit, se va desarrollar en la rama de desarrollo (branch development) que corresponderá 1 branch para cada miembro del equipo y la rama maestra (branch master) la cual alojará las versiones cuyos cambios hayan sido aprobados y estén listos para su respectivo release.

* **Branch master**: Esta rama será la principal, donde se pondrá los cambios aprobados por el administrador.
* **Branch Development**: Esta rama será para los desarrolladores, donde podrán hacer sus cambios previa aprobación, sin alterar la rama principal.

1. Calendario

|  |  |  |
| --- | --- | --- |
| **ACTIVIDAD** | **TIEMPO(días)** | **ROL** |
| **1.Planificación de la SCM** | **12 días** | **Gestor de la configuración** |
| Identificar la problemática de la empresa. | 2 días | Gestor de la configuración |
| Definir el propósito y finalidad del plan. | 1 día | Gestor de la configuración |
| Identificar roles y responsabilidades. | 2 días | Gestor de la configuración |
| Identificar políticas y responsabilidades. | 2 días | Gestor de la configuración |
| Analizar herramientas. | 3 días | Gestor de la configuración |
| Elaborar el calendario de PGC | 2 días | Gestor de la configuración |
| **2.Identificación de la SCM** | **8 días** | **Gestor de la configuración** |
| Identificar ítems de configuración. | 3 días | Gestor de la configuración |
| Clasificar ítems de configuración | 2 días | Gestor de la configuración |
| Definir la nomenclatura de los elementos. | 2 días | Gestor de la configuración |
| Listar ítems con la nomenclatura. | 1 día | Gestor de la configuración |