# **UNIVERSIDAD NACIONAL MAYOR DE**

# **SAN MARCOS**

# (Universidad del Perú, Decana de América)

# **E.A.P. INGENIERÍA DE SOFTWARE**

# **Resultado de imagen para logo unmsm**

# **PROYECTO: “WhalE Software”**

# **Gestión de la configuración del Software**

# 

# **DOCENTE**: Lenis Wong Portillo

# **NOMBRE DE LA EMPRESA**: Soluciones Disruptivas Perú

# **INTEGRANTES: CÓDIGO:**

# AGUILAR BURGA, Piero 16200203

# AVILA CHUNGA, Emmanuel 16200206

# CRUZ HUAMAN, Carlos 16200211

# ESCOBAR BENDEZU, Alvaro 16200216

# TUEROS MONTES, Rolando 16200243

# **E.A.P. Ingeniería de Software**

# **Semestre 2022 - I**

# **Introducción**

* 1. **Propósito del plan**

El plan de gestión de la configuración se realizó para planificar y ejecutar las actividades de identificación y control correspondiente a la configuración del proyecto Whale, así como también con la intención de evaluar y proponer un esquema de disposición de roles con sus respectivas responsabilidades y las herramientas de soporte a utilizar para este fin.

* 1. **Situación de la empresa**

Somos una empresa especializada en brindar soluciones digitales mediante software de calidad para emprendedores y diferentes empresas nacionales.

Como equipo nos encontramos en búsqueda de retos que nos hagan evolucionar en el área de desarrollo de software, siempre enfocándonos en dar las soluciones acertadas a cada cliente. Nos estamos especializando en asesoramiento y auditoría, además actualmente contamos con un vasto portafolio de proyectos de código abierto, entre los que contamos con tres proyectos de procesamiento de datos, IA, respuestas en tiempo real, serverless y desarrollo cloud; sin embargo, estamos experimentando algunas deficiencias en torno a la calidad y mantenibilidad del software, debido al pésimo gestionamiento de cambios y a la evolución continua de la informática, como las herramientas y parches de seguridad.

1. **Gestión de la SCM**
   1. **Roles o responsabilidades**

En la tabla mostrada a continuación se presentan los roles, sus responsabilidades y la cantidad de miembros del equipo que ocupan dichos roles.

|  |  |  |
| --- | --- | --- |
| Nombre | Responsabilidad | Persona Asignada |
| Gestor de la Gestión de la Configuración | Persona encargada de la creación y asignación de roles a los miembros del equipo. Implementa, mantiene y mejora la gestión de configuración. Se responsabiliza también de la utilización de herramientas durante la gestión de configuración. | * PIERO AGUILAR |
| Comité de Control de Cambios | Evaluación de registro de eventos. Gestionan las solicitudes de cambios relevantes.    Da seguimiento al registro de eventos y solicitudes de cambios a través de sus ciclos de vida.    Proveer apoyo a la persona que registró el evento.    Coordinar con otros Comité de Control de Cambios y con la gestión de proyecto | * ALVARO ESCOBAR * ROLANDO TUEROS * CARLOS CRUZ |
| Bibliotecario | Define y da mantenimiento a las bibliotecas del repositorio, que son usadas durante  la gestión de configuración.    Es el encargado de asegurarse que los aspectos prácticos de la gestión de configuración trabajen entre sí adecuadamente. | * CARLOS CRUZ |
| Auditor | Verifica que los elementos de la configuración de los que es responsable están registrados en la base de datos de configuración con el estado y datos de configuración apropiados. | * EMMANUEL ÁVILA |

Tabla 1. Roles de Gestión de la Configuración

* 1. **Herramientas (Benchmarking)**

Se consideran las siguientes herramientas de control de versiones:

* Git/GitHub
* Concurrent Versions System (CVS)
* Apache Subversion (SVN)
* Azure DevOps Server (antiguamente Team Foundation Server)

**Comparación de Características**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Git/GitHub** | **CVS** | **SVN** | **Azure DevOps Server** |
| **Licencia** | GLP | GLP | Licencia Apache | Comercial |
| **Colaborativa** | Si | Si | Si | Si |
| **Control de Versiones** | Distribuido | Centralizado | Centralizado | Centralizado |
| **Repositorio** | Copias locales | Central: Se generan copias de trabajo | Central: Se generan copias de trabajo | Central: Se generan copias de trabajo |
| **Curva de aprendizaje** | Moderada | Moderada | Baja | Moderada |
| **Seguimiento** | Basado en contenido | Basado en archivos | Basado en archivos | Basado en contenido |
| **UI Estables** | Si | No | Si | Si |
| **Arquitectura** | Arq. tres árboles | Cliente - Servidor | Cliente - Servidor | Multi-tier |
| **Dificultad de trabajo con**  **ramas** | Sencilla | Complicada | Complicada | Moderada |
| **Conectividad** | Solo para sincronizar | - | Con cada acceso | Con cada acceso |
| **Historial** | Repositorio y copias de trabajo | Solo repositorio completo | Solo repositorio completo | Solo repositorio completo |
| **Archivos binarios** | Soportado | Limitado | Soportado | Soportado |

**Comparación de Precios**

Las herramientas Current Versions System (CVS) y Apache Subversion (SVN) son gratuitas, por lo que solo comparamos GitHub y Azure DevOps Server.

|  |  |  |
| --- | --- | --- |
|  | **GitHub** | **Azure DevOps Server** |
| **Plan Gratuito** | Si: para individuos y organizaciones | No posee versión gratuita |
| **Plan Básico** | $3.6/mes/usuario | $6/mes/usuario + $45/mes/usuario de suscrip. a VS profesional |
| **Plan Avanzado** | $19.25/mes/usuario | $52/mes/usuario + $45/mes/usuario de suscrip. a VS profesional |

Se consideran las siguientes herramientas para administrar el Cliente de GIT:

* GitKraken
* GitHub Desktop
* Sourcetree
* Git Cola

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **GitKraken** | **GitHub Desktop** | **Sourcetree** | **Git Cola** |
| **1 Licencia para instalación en Windows, Mac, y Linux** | Si | No | No | Si |
| **Conectar a cualquier repositorio Git incluido GitHub, GitLab, Bitbucket y Azure DevOps** | Si | No | No | No |
| **Gráfico visual de Commits** | Si | Si | Si | Si |
| **Deshacer/rehacer con 1 clic** | Si | No | No | No |
| **Conectar repositorios a**  **Jira, Trello, GitKraken Boards, GitHub Issues y GitLab Issues** | Si | Si | No | No |
| **Funciones de colaboración en equipo** | Si | Si | No | No |
| **Herramienta para Merge conflict** | Si | No | Si | No |
| **Editor de texto incorporado** | Si | No | Si | Si |
| **Manejo de pull request dentro de la aplicacion** | Si | No | Si | No |

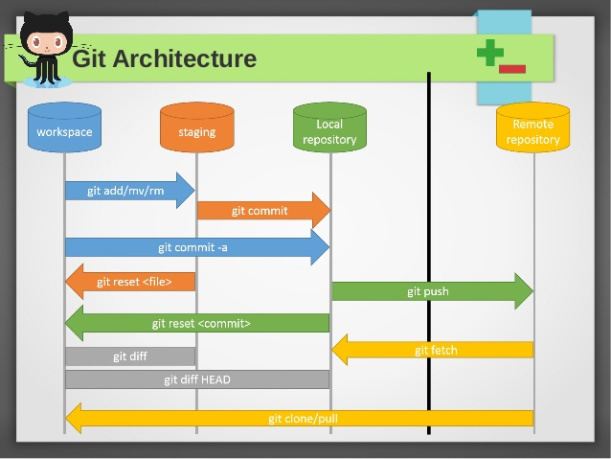
Se consideran las siguientes herramientas para Control de Versiones propuestos:

* Monotone
* Mercurial
* Bazaar
* Plastic SCM
* Team Fundation Server

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Monotone** | **Mercurial** | **Bazaar** | **Plastic SCM** | **Team Fundation Server  (Azure DevOps)** |
| **Licencia** | GPL | GPL | GPL | Comercial | Comercial |
| **Colaborativa** | Si | Si | Si | Si | Si |
| **Control de Versiones** | Distribuido | Distribuido | Distribuido, Cliente-Servidor | Distribuido, Cliente-Servidor | Distribuido |
| **Repositorio** | Central: Se generan copias | Central: Se generan copias de trabajo | Central: Se generan copias de trabajo | Central: Se generan copias de trabajo | Central: Se generan copias de trabajo |
| **Herramienta de Merge automático personalizada** | Si | Si | Si | Si | Si |
| **Estado de desarrollo** | Activo | Activo | Activo | Activo | Activo |
| **Plataformas soportadas** | Unix-like, Windows, macOS | Unix-like, Windows, macOS | Unix-like, Windows, macOS | Unix-like, Windows, macOS | Windows, cross-platform via Azure DevOps Services |
| **Costo** | Gratis | Gratis | Gratis | Gratis hasta 3 usuarios, después costando 7$ por usuario mensualmente, o $ 23 por usuario por mes para la edición local. Sin licencias perpetuas. | Gratis para hasta 5 usuarios en Azure DevOps Services o para proyectos de código abierto; de lo contrario, al costo, con licencia a través de suscripción a MSDN o compra directa. |

* 1. **Diagrama de arquitectura de la herramienta elegida**

**Git: Arquitectura de tres árboles**



# 

# 

A diferencia de otros sistemas de control de versiones que poseen una arquitectura de dos árboles, (entendemos por un árbol a una estructura de archivos y carpetas), Git posee tres. Estos son:

* **Workspace:** Se crea cuando se inicializa un proyecto Git en la máquina local, permite editar el código fuente copiado.
* **Staging:** El código se prepara en el área de preparación aplicando el comando, git add. Esto muestra una vista previa de la siguiente etapa. En caso de que se realicen más modificaciones en el directorio de trabajo, las instantáneas de estas dos capas serán diferentes. Sin embargo, estos se pueden sincronizar usando el mismo comando 'git add'.
* **Repositorio Local:** Si no se necesitan más ediciones, se continúa aplicando el comando git commit. Esto replica las últimas instantáneas en las tres etapas, lo que las sincroniza entre sí.