EMPRESA

**Plan de Gestión de la Configuración del Software**

# Integrantes

* Sarmiento Quispe, Sofía Fernanda
* Chipana Arellano, Franz Esteban
* Contreras Quispe, Harumi Nayeli
* Aranda Vega, Joar Jesús
* Flores Gómez José Benjamín
* Parrales Torres, José Luis
* Palacios Aguilar Adrian Alessandro

# Versión 1.0

## Control de versión

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Autor** | **Descripción** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. Introducción

1.1 Situación de la empresa y problemática **Situación de la Empresa:**

MelodyLabs es una empresa en crecimiento que se especializa en la creación y desarrollo de software innovador en el ámbito de la tecnología musical. A lo largo de los años, ha destacado por la creación de soluciones personalizadas, tanto para empresas como para usuarios individuales, enfocadas en la producción, organización y distribución de contenido musical. Actualmente, la empresa ha expandido su línea de productos y se encuentra desarrollando un ambicioso proyecto denominado TheVault, un sistema avanzado de reproducción de música diseñado para almacenar, gestionar y ofrecer acceso a grandes catálogos de música de manera eficiente y fluida.

**Problemática:**

## 1. Sobreescritura de partes del código

Actualmente, varios desarrolladores trabajan de manera simultánea en diferentes módulos de TheVault, y debido a la falta de un sistema de control de versiones robusto y centralizado, se han presentado numerosos casos de sobrescritura involuntaria de código.

## 2. Falta de actualizaciones consistentes entre los equipos

Otro grave problema es la falta de sincronización entre los equipos de desarrollo. Algunos equipos o individuos no cuentan con la versión más actualizada del código fuente al comenzar su trabajo, lo que provoca que implementen sus cambios sobre una versión antigua o incompleta del software.

## 3. Ausencia de historial claro de cambios

La falta de un sistema formalizado para la gestión de versiones ha provocado que el equipo de desarrollo no tenga acceso a un historial claro de los cambios realizados. Esto genera dificultades para rastrear quién introdujo una modificación específica o cuándo y por qué se hizo, lo que complica la identificación de errores recientes y la toma de decisiones informadas.

**1.2 El objetivo del plan**

Se debe implementar una herramienta que organice y gestione bien las versiones de código ( como git) para evitar que el equipo de desarrolladores sobreescriben código, asegurarse que todos trabajen con la versión más actualizada y llevar un registro de todos los cambios.

2. Gestión de Configuración de Software

* 1. Roles y responsabilidades

|  |  |
| --- | --- |
| Project Manager | - Establecer las metas generales del proyecto y asegurarse de que el equipo siga las reglas del control de versiones. |
| Gestor de configuración | - Definir cómo los desarrolladores deben crear, fusionar y revisar las ramas de código. |

Inspector de aseguramiento de calidad - Trabajar con el gestor de configuración para

chequear el historial de cambios y que todo esté en orden.

Miembros del equipo de proyecto - Seguir las reglas para el control de versiones,

actualizando siempre la rama antes de empezar a trabajar.

* 1. Herramientas, entorno e Infraestructura

[**Benchmarking** (Min 3 herramientas), Ponderar y selección ] Descripción de cada herramienta:

## Herramienta 1: Git

Es un sistema de control de versiones distribuido ampliamente utilizado en la industria del software, conocido por su flexibilidad y eficiencia. Permite tanto trabajo local como colaboración distribuida.

## Herramienta 2: Mercurial

Similar a Git en ser distribuido, Mercurial es conocido por su facilidad de uso y menor curva de aprendizaje, aunque no es tan flexible como Git en algunos aspectos avanzados.

## Herramienta 3: Subversion (SVN)

A diferencia de Git y Mercurial, SVN es un sistema de control de versiones centralizado. Es más fácil de entender para los equipos que prefieren un enfoque de servidor único, pero con menos flexibilidad en cuanto al trabajo distribuido.

|  |  |  |  |
| --- | --- | --- | --- |
| **CRITERIOS** | **HERRAMIENTA 1** | **HERRAMIENTA 2** | **HERRAMIENTA 3** |
| Gratuito | Sí | Sí | Sí |
| Curva de  aprendizaje | Moderada | Baja | Alta |
| Centralizado | No | No | Sí |
| Distribuido | Sí | Sí | No |

## Herramienta seleccionada: Git

Los beneficios que hemos podido destacar para seleccionar esta herramienta:

* Distribución y Flexibilidad
* Gestión Eficiente de Ramas
* Fusión y Colaboración
* Desempeño
* Amplia Adopción y Soporte
* Control Fino y Registro Detallado
* Portabilidad y Compatibilidad

Reportes de estado

Reporte de contribución

Interfaz de usuario gráfica

Descripción generada automáticamente

Se puede apreciar la cantidad de commits realizados en el proyecto, tanto en el primer, segundo y tercer hito ( hito en curso) , en este se puede apreciar una gran cantidad de aportes en el primer hito dado que hubieron varias correcciones y carga de ítems, también cabe resaltar que el 1er hito por desconocimiento no hubo ramas por cada contribuidor, tema que fue subsanado a partir del 2do hito donde cada contribuidor realizo su aporte en su rama y posteriormente un merge para llevarlo al main

Interfaz de usuario gráfica

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Se puede apreciar que todas las ramas trabajaron en el 2do hito, donde comenzó el desarrollo del código del proyecto y se trabajo tanto el backend como el frontend, cabe resaltar que la base de datos es trabajada en SUPABASE y los cambios no se ven reflejados en el repositorio.

Interfaz de usuario gráfica

Descripción generada automáticamente

Actualmente en el 3er hito hemos realizado contribuciones con respecto a terminar el codigo del proyecto (contenido de páginas, llenado de información en la base de datos, y el desarrollo del plan de prueba)