**1.Introducción**

Innovatech fue fundada en 2015 y desde entonces ha estado en constante crecimiento y expansión. Innovatech se especializa en el desarrollo de sistemas y software a medida para empresas de diversos sectores, incluyendo aplicaciones web, móviles y soluciones de software empresarial.

La empresa se destaca por su experiencia en el desarrollo de sistemas de gestión empresarial (ERP), sistemas de gestión de relaciones con clientes (CRM), aplicaciones móviles y soluciones de comercio electrónico.

Innovatech cuenta con una amplia cartera de clientes que abarca desde pequeñas empresas locales hasta grandes corporaciones internacionales en diversos sectores como la salud, educación, ambiente.

**Problemáticas:**

Durante la trayectoria de la empresa ha tenido diversas dificultades durante el desarrollo de cada producto que aún siguen siendo persistentes a lo largo de los años. Algunos de estos son:

* **Cambios de alcance durante el desarrollo:** En varios proyectos, los clientes han solicitado cambios significativos en el alcance del proyecto una vez que el desarrollo ha comenzado, lo que ha resultado en retrasos y aumentos en los costos.
* **Integración de sistemas heredados:** Algunos clientes han enfrentado dificultades para integrar los nuevos sistemas desarrollados por Innovatech con sus sistemas heredados, lo que ha generado problemas de compatibilidad y funcionamiento.
* **Escalabilidad y rendimiento:** En proyectos de gran envergadura, se han presentado desafíos relacionados con la escalabilidad y el rendimiento de los sistemas desarrollados, especialmente cuando se trata de manejar grandes volúmenes de datos o de usuarios concurrentes.
* **Seguridad de la información:** La seguridad de la información ha sido una preocupación constante para algunos clientes, especialmente en sectores altamente regulados como la salud y las finanzas. Garantizar la seguridad y la privacidad de los datos ha sido un desafío importante en varios proyectos.
* **Capacitación y adopción del usuario final:** En varios casos, la capacitación y la adopción por parte del usuario final han sido un desafío, ya que algunos usuarios pueden resistirse al cambio o encontrar dificultades para adaptarse a los nuevos sistemas y procesos.

**Objetivo**:

Finalidad de aplicar las buenas prácticas de gestión de la configuración

**2.Gestión de Documentos:**

1. **Definir los Roles de la GCS**

**Gestor de la configuración:** Mattos Hilario, Yayir Flabio;

**Bibliotecario:** Castillo Layme, Sebastian Castillo

**Auditor:** Lorenzo Ramos, Daniel Lorenzo

**CSS:** Bellido Medina, Kelly Milene

1. **Lista de 6 herramientas**
   1. **Git:**

Git es un sistema de control de versiones distribuido que permite registrar los cambios que se hacen en los archivos y volver a versiones anteriores si algo sale mal. Fue diseñado por Linus Torvalds para garantizar la eficiencia y confiabilidad del mantenimiento de versiones de aplicaciones que tienen un gran número de archivos de código fuente.

**Fuente:** [**https://git-scm.com**](https://git-scm.com/)

**Características:**

* Distribuido y Descentralizado: A diferencia de los sistemas de control de versiones centralizados, en los que se almacena una única copia del repositorio, Git es distribuido y como desarrolladores tenemos una copia completa del historial de cambios.
* Rastreo Preciso de Cambios: Git realiza un seguimiento preciso de los cambios realizados en cada archivo a lo largo del tiempo. Esto nos permite como desarrollador ver el historial completo de cambios, incluidas las diferencias entre versiones y la posibilidad de regresar a versiones anteriores en caso de problema.
* Ramificación Eficiente: Git hace que la creación y gestión de ramas (branches) sea sencillo. Como desarrollador podemos crear ramas para trabajar con nuevas características o arreglos sin afectar la rama principal. Esto fomenta el desarrollo paralelo y facilita la colaboración en equipos grandes.
* Fusiones Simplificadas: La capacidad de fusionar cambios entre diferentes ramas es una característica clave de Git. Como desarrollador podemos combinar cambios de manera eficiente, lo que facilita la integración continua y evita conflictos de código.
  1. **Gitlab**

Es una suite completa que permite gestionar, administrar, crear y conectar los repositorios con diferentes aplicaciones y hacer todo tipo de integraciones con ellas, ofreciendo un ambiente y una plataforma en la cual se puede realizar las varias etapas de su SDLC/ADLC y DevOps

**Fuente:** <https://about.gitlab.com/platform/>

**Características:**

* Gestión de repositorios: GitLab permite administrar grupos, personas y los permisos que quieras que tengan los usuarios dentro de los grupos o proyectos a los que pertenezcan.
* Seguimiento del estado actual y del histórico de los proyectos: Permite ver todos los cambios y modificaciones producidas en el tiempo de desarrollo.
* Software libre: GitLab es un software libre que puedes descargar e instalar en cualquier servidor.
* Construido sobre Git: GitLab está construido sobre Git, proporcionando el código para generar un servidor y gestionar los clientes, sus operaciones y los servicios que ofrece.
  1. **Source Off Site (C/S)**

Es una herramienta de control de versiones y DevOps basada en SourceSafe. Fue desarrollada por SourceGear y permite acceder a los datos de SourceSafe de manera rápida y segura, ya sea localmente o a través de Internet.

**Fuente:** <https://www.sourcegear.com/sos/>

**Características:**

* Interfaz completamente nueva: La interfaz de cliente de SourceOffSite 5 ha sido reescrita desde cero. Se ve, se siente y se comporta como un cliente de control de versiones moderno y potente.
* Trabajo más inteligente: El rediseño no es solo un “re-skin”, está todo orientado a ayudarte a hacer tu trabajo más rápido y fácilmente. La vista de Checkins Pendientes de SourceOffSite 5 te muestra todos tus trabajos en progreso, en todas las carpetas, en un solo lugar.
* Más rápido y más confiable: SourceOffSite 5 es la versión más rápida de SourceOffSite hasta ahora, a menudo 10 veces más rápida, o más, con mejoras serias en términos de estabilidad y robustez.
  1. **Subversion SVN (Servidor)**

Es un sistema de control de versiones centralizado. Fue desarrollado por SourceGear y permite a los desarrolladores administrar los cambios en archivos y directorios a lo largo del tiempo. Aquí te dejo algunas de sus características principales.

**Fuente:** <https://subversion.apache.org/>

**Características:**

* Seguimiento de directorios: Al igual que los archivos, las operaciones como copias, eliminaciones y renombrados de los archivos son versionadas.
* Almacenamiento de metadatos: Permite almacenar información arbitraria sobre los archivos o directorios del proyecto.
* Envíos atómicos: Una colección cualquiera de modificaciones o bien entra por completo al repositorio, o bien no lo hace en absoluto.
* Elección de las capas de red: Subversion tiene una noción abstracta del acceso al repositorio, facilitando a las personas implementar nuevos mecanismos de red.
* Manipulación consistente de datos: Subversion expresa las diferencias del fichero usando un algoritmo de diferenciación binario, que funciona idénticamente con ficheros de texto (legibles para humanos) y ficheros binarios (ilegibles para humanos).
  1. **Tortoises SVN (Cliente)**

Es un cliente de Subversion, implementado como una extensión del shell de Windows. Es un software libre liberado bajo la licencia GNU GPL.

**Fuente:** <https://tortoisesvn.net/>

**Características:**

* Integración con Windows: TortoiseSVN se integra perfectamente con Windows, por ejemplo, en el Explorador de archivos.
* Iconos superpuestos: El estado de cada carpeta y archivo versionado se indica por pequeños iconos superpuestos.
* Interfaz Gráfica de Usuario: Cuando listas los cambios realizados a un archivo o carpeta, puedes hacer clic en una revisión para ver los comentarios de esa confirmación.
* Fácil acceso a los comandos de Subversion: Todos los comandos de Subversion están disponibles desde el menú contextual del explorador.
* Versionado de carpetas: Subversion implementa un sistema “virtual” de archivos versionados que toma en cuenta los cambios en todos los árboles de directorios en el tiempo.
  1. **Rational ClearCase - IBM**

Es un sistema de gestión de configuración de nivel empresarial que proporciona acceso controlado a activos de software. Fue desarrollado por IBM y permite a los desarrolladores administrar los cambios en archivos y directorios a lo largo del tiempo.

**Fuente:** https://www.ibm.com/es-es/products/rational-clearcase

**Características:**

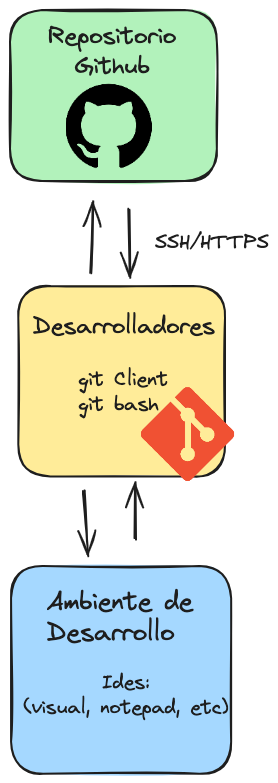
* Control de versiones: Permite eliminar versiones anteriores, crear y eliminar ramas, enumerar historiales de versiones y comparar y fusionar versiones.
* Trabajo rápido e independiente: Proporciona modelos de desarrollo e integración, espacios de trabajo privados y áreas de integración pública.
* Cumplimiento de los requisitos de conformidad: Incluye autenticación de usuarios y registros de auditoría para ayudar a cumplir con los requisitos de cumplimiento con mínimas complicaciones administrativas.
* Confianza en las versiones: Gestiona y controla prácticamente cualquier artefacto de proyecto que se pueda representar como contenido digital.
* Control de la actividad de desarrollo: Permite controlar áreas de trabajo personales y proporciona acceso a las versiones de archivos y directorios que necesite

1. **Herramienta Elegida (Diagrama de arquitectura)**

**Repositorio GitHub:** Este es el repositorio central donde se almacena el código fuente y los archivos de configuración.

**Desarrolladores:** Utilizan clientes de Git para interactuar con el repositorio de GitHub.

**Ambiente de Desarrollo:** Los desarrolladores trabajan en sus entornos de desarrollo local utilizando IDEs y herramientas de desarrollo estándar.



**3.Actividades**

1. **Identificación:**
   1. **Clasificación de ítems**

| **Tipo** | **Item** | **Ext** | **Proyecto** |
| --- | --- | --- | --- |
| E | Project Charter | .DOCX | UW |
| E | Doc. de Arquitectura de Software | .DOCX | UW |
| E | Doc. de Requisitos | .DOCX | UW |
| E | Doc. de Casos de Uso | .DOCX | UW |
| E | Doc. de Especificación de Interfaz de Usuario | .DOCX | UW |
| E | Doc. de Guia de estilos | .DOCX | UW |
| E | Doc. de la Especificación de la Base de Datos | .PDF | UW |
| E | Doc. de Negocio | .DOCX | UW |
| E | Cronograma del proyecto | .XLSX | UW |
| F | Mod. de la Página Principal | .JSX | UW |
| F | Mod. de Autenticación de Usuario | .JSX | UW |
| F | Mod. de Realizar Alerta | .JSX | UW |
| E | Doc. de lista de Historias de Usuarios | .DOCX | UW |
| E | Doc. Manual de Usuario | .DOCX | UW |
| E | Doc. de pruebas de Software | .DOCX | UW |
| F | Acta de cierre del proyecto | .DOCX | UW |
| E | Main | .JS | UW |
| S | Variables del Entorno | .ENV | UW |
| S | node-package | .JSON | UW |

* 1. **Definición de nomenclatura:**

**Regla 1** = “Acrónimo del proyecto” + “-” + “Acrónimo del ítem”

**Regla 2** = En caso de tener elementos con nombres repetidos, procedemos con agregarle “(dup)”, como el siguiente ejemplo:

“Acrónimo del proyecto” + “-” + “Acronimo de item” + “(dup)”

**Regla 3** = Enumerar los documentos que sean del mismo tipo

**Regla 4** = Los ítems que no son específicos a un PY único, son identificados únicamente por su acrónimo.

**Regla 5** = Los ítems que son específicos de un PY pero no estan asociados con un componente del PY utilizan un identificador de dos partes:

“Acrónimo del proyecto” + “-” + “Acronimo de derivado del tipo de artefacto”

**Regla 6** = Los ítems en evolución que son específicos de un PY y están asociados con un componente específico, utilizan un identificador de tres partes:

“Acrónimo del proyecto” + “-” + ”Acrónimo del componente ” + ” - ” + “Acronimo de derivado del tipo de artefacto”

**Regla 7** = El nivel de versión de cada ítem se mantiene como un identificador separado.

**Regla 8** = El nivel de versión se mantiene como un identificador numérico con dos componentes:

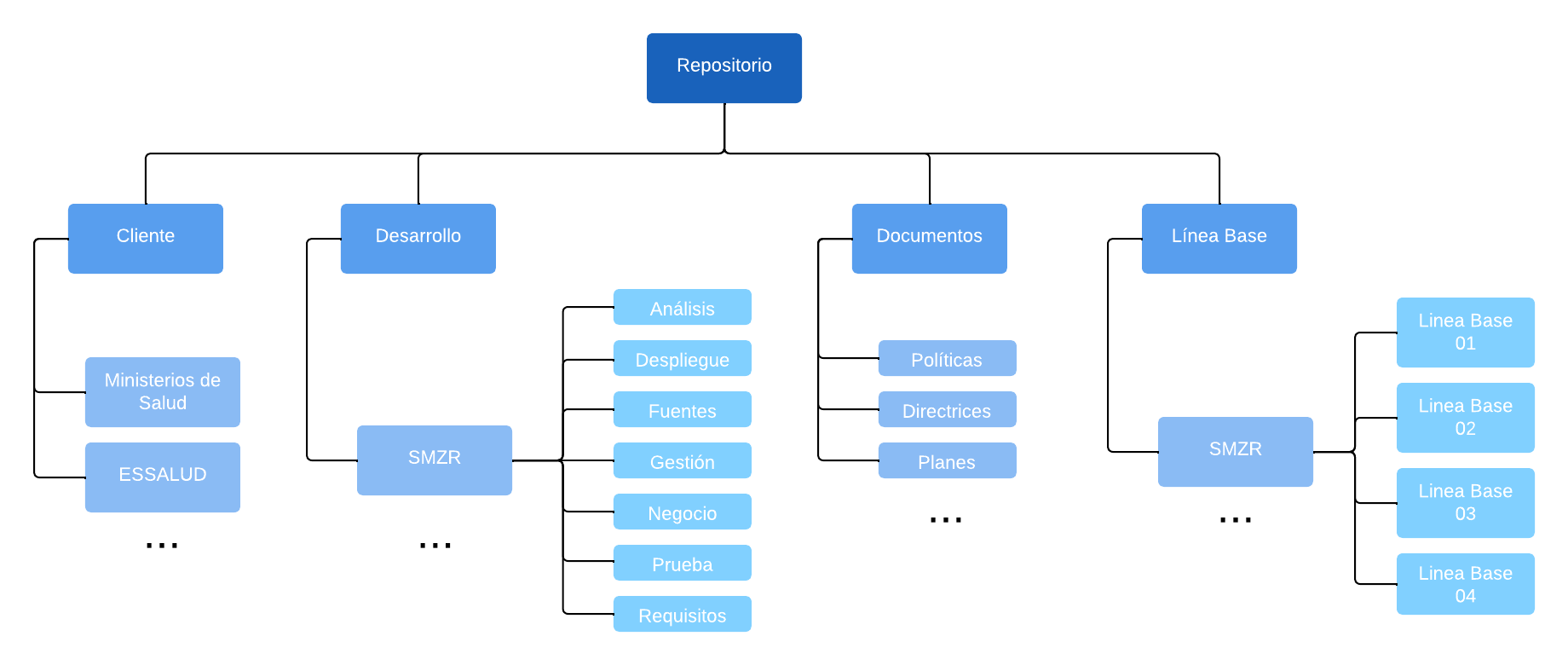
Versión.Revisión.

**Regla 9** = El número de versión cambiará cuando la arquitectura principal del ítem ha cambiado, o cuando el ítem es completamente reconstruido, con cambios internos sustanciales.

**Regla 10** = El número de revisión cambiará cuando el contenido ha cambiado, pero la estructura principal y el flujo del ítem se mantiene igual.

* 1. **Diseño del Repositorio:**

El diseño que vamos a usar para nuestro repositorio es el siguiente, el diagrama solo es algo referencial, después se explicará a mayor detalle cada punto.



El repositorio debe de tener el nombre de la empresa, en nuestro caso sería Innovatech.

En la carpeta clientes debemos de poner las diferentes entidades que solicitaron algún software a la empresa.

En la carpeta desarrollo debemos de colocar carpetas con el nombre de los diferentes proyectos que haya realizado la empresa y a su vez cada carpeta de proyecto debe de tener las siguientes carpetas: análisis, despliegue, fuentes, gestión, negocio, prueba y requisitos. De esta manera se organiza de mejor manera cada proyecto.

En la carpeta documentos se colocan las carpetas políticas, directrices y planes, estas carpetas sirven para que los clientes conozcan la forma en la que trabaja la empresa.

En la carpeta línea base se crea una carpeta por cada proyecto que haya realizado la empresa y dentro de estas deben de la misma cantidad de carpetas que hitos marcados en los cronogramas de cada proyecto, estas carpetas creadas dentro de cada proyecto se llamarán línea base y estarán numeradas. Servirán para tener una copia de seguridad del proyecto hasta el momento en el que acaba un hito.

* 1. **Línea Base**

| **Línea Base** | **Hito** | **Artefactos** |
| --- | --- | --- |
| LB1 | 28/04/2024 | * Project Charter * Cronograma de proyecto * Documento de Negocio * Documento de planificación de la gestión de la configuración * Documento de Requisitos Funcionales y no Funcionales * Documento de Especificación de Casos de Uso * Documento de Especificación de UI * Documento de Guía de Estilos * Documento de Especificación de la BD * Documento de Arquitectura del Software * Reporte del Desarrollo del Software * Reporte del Primer Sprint |
| LB2 | 23/05/2024 | * Documento de Especificación de Requisitos * Documento de Especificación de UI * Documento de Especificación de la BD * Módulo Página Principal * Módulo de Autenticación de Usuario * Módulo de Realizar Alerta * Reporte del Desarrollo del Software * Reporte del Segundo Sprint |
| LB3 | 13/06/2024 | * Módulo de Moderador * Documento de Especificación de Requisitos * Documento de Especificación de UI * Documento de Especificación de la BD * Documento de Arquitectura del Software * Lista de Historias de Usuario * Documento de Guía de Estilos * Manual de usuario * Documento de Pruebas del Software * Reporte del Desarrollo del Software * Reporte del Tercer Sprint * Acta de cierre del proyecto |