EMPRESA:

CODECRAFTERS

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

Integrantes:

* Barrantes Navarro Joshúa André
* Toro Vargas, Karlo Andrés
* Gamarra Chavez Diego Valentino
* Serna Quiroz, Andrew Gabriel
* Calderon Zuñiga, Rodrigo Joaquin
* Ridoutt Quiñones Jhon Franco

Versión 1.1

**Control de versión**

| **Fecha** | **Autores** | **Descripción** |
| --- | --- | --- |
| 19/09/24 | Andre Barrantes  Karlo Toro  Diego Gamarra  Andrew Serna  Rodrigo Calderón  Jhon Ridoutt | Primera versión del documento |
| 26/09/2024 | Andre Barrantes  Karlo Toro  Diego Gamarra  Andrew Serna  Rodrigo Calderón  Jhon Ridoutt | Identificación |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **Introducción**
   1. Situación de la empresa y problemática
2. Empresa:

CodeCrafters es una empresa tecnológica dedicada a la creación de aplicaciones integradas a la inteligencia artificial, llevamos más de 5 años en el mercado, trabajando junto a grandes empresas e instituciones como Google, Meta y la UNMSM.

Nos enfocamos en crear soluciones que optimicen procesos a través de IA, mejorando la eficiencia y automatización de tareas complejas. Algunos de nuestros proyectos fueron un asistente virtual para un e-commerce y un sistema de reconocimiento de voz para la identificación de requisitos funcionales y no funcionales, y actualmente en un videojuego aplicando las IA sobre controlar y gestionar los recursos de una universidad..

1. Problemática:

Actualmente, la empresa no cuenta con un control preciso de las versiones del código, ya que cada equipo utiliza un sistema de control de versiones diferentes, esto genera desorganización en el desarrollo, que generado duplicación de esfuerzos donde varios equipos trabajan en funcionalidades similares a la vez sin saberlo, esto genera el desperdicio de recursos tanto del personal y del presupuesto.

La falta de un control preciso de versiones, provoca que los equipos no pasen sus avances por un proceso de revisión y aprobación de los cambios realizados, lo que hace que al subir directamente sus avances al código principal, esto puede provocar incompatibilidades con lo agregado por otros equipos, causando errores, fallas o bugs, retrasando el cronograma y en ocasiones no siendo posible restaurar el estado del proyecto a una versión anterior.

Adicionalmente, no hay una buena gestión de la documentación para cada funcionalidad del código, lo que dificulta la integración de nuevo personal a los proyectos, y mala gestión de configuraciones especificadas pedidas por los clientes. Todos estos errores han llevado a diversos bugs o demoras en las entregas.

* 1. El objetivo del plan

El objetivo de este plan es estandarizar un sistema de control de versiones en CodeCrafters y mejorar la gestión de la documentación para cada funcionalidad del código, teniendo en cuenta las versiones y el control de cambios de manera precisa. Esto permitirá unificar los procesos de desarrollo, asegurando que cada equipo utilice un único sistema para gestionar las versiones de código, reduciendo errores e incompatibilidades. Con un sistema centralizado de control de versiones y un registro detallado de los cambios, se busca optimizar la eficiencia en la gestión de proyectos basados en inteligencia artificial, minimizar los bugs, evitar demoras en las entregas y mejorar la calidad de los productos y servicios ofrecidos a los clientes.

1. **Gestión de Configuración de Software**
   1. Roles y responsabilidades

| Nombre del Rol | Responsabilidades | Niveles de Autoridad |
| --- | --- | --- |
| Project Manager | Supervisar el funcionamiento de la Gestión de la Configuración y la implementación de IA en los proyectos. | Toda autoridad sobre la gestión del software, IA, y el proyecto en general. |
| Gestor de Configuración | Ejecutar las tareas de configuración del software, asegurando la integridad y consistencia. | Autoridad para operar y gestionar todas las funciones de la Configuración del Software. |
| Inspector de Aseguramiento de Calidad | Auditar la configuración y asegurar que el software cumple con los estándares de calidad. | Auditar la configuración del software siguiendo las directrices del Project Manager. |
| Desarrollador Principal | Implementar las configuraciones y coordinar los aspectos técnicos del desarrollo de software. | Responsable de las decisiones técnicas en el desarrollo, reportando al Project Manager. |
| Bibliotecario | Definir y mantener las bibliotecas usadas en la gestión de configuración. Asegurarse de que los aspectos prácticos de la gestión de configuración funcionen adecuadamente entre sí. | Responsable de la organización y mantenimiento de las bibliotecas de configuración. |
| Comité de Control de Cambios (CCB) | Evaluar registros de eventos, gestionar solicitudes de cambios relevantes y hacer seguimiento de su ciclo de vida. Proveer apoyo y coordinar con otros comités de control de cambios. | Autoridad sobre la gestión de cambios y coordinación con otras áreas del proyecto. |

* 1. Herramientas, entorno e Infraestructura

| **CRITERIOS** | **GITHUB** | **GITLAB** | **MERCURIAL** | **Apache Subversion** |
| --- | --- | --- | --- | --- |
| Gratuito | GitHub ofrece planes gratuitos para proyectos públicos y privados con funcionalidades básicas como repositorios ilimitados, colaboración en código, y revisiones de código. Las versiones pagas incluyen características adicionales como CI/CD (Integración y Entrega Continua) más avanzadas y acceso a soporte técnico. | GitLab también tiene una versión gratuita bastante robusta que ofrece repositorios ilimitados y herramientas de CI/CD integradas desde la versión gratuita. Las versiones de pago incluyen mayor capacidad para personalizar el flujo de trabajo, características avanzadas de seguridad y monitoreo. | Es una herramienta de código libre distribuida bajo licencia GNU 2. | Es una herramienta de código libre bajo licencia Apache/BSD. |
| Curva de aprendizaje | La curva de aprendizaje de GitHub es relativamente suave para desarrolladores principiantes, ya que tiene una interfaz intuitiva y fácil de usar. Existen abundantes tutoriales y recursos, pero aprender conceptos más avanzados como flujo de trabajo de Git, pull requests, y automatización puede requerir más tiempo. | GitLab tiene una curva de aprendizaje un poco más empinada que GitHub, especialmente en la configuración de su CI/CD integrado, pero también proporciona muchos recursos educativos. Para principiantes, puede ser menos intuitivo, pero es muy potente para usuarios avanzados. | Tiene una curva de aprendizaje mucho mas accesible que Git, ya que maneja solo 2 tipos de ramas (permanentes y anónimas) y además de esto tiene restringido el acceso accidental a herramientas potencialmente agresivas o peligrosas, las cuales se tienen que habilitar previamente. | El manejo de Subversion tanto en sus características básicas como opciones más avanzadas, por ejemplo el trabajo con ramas y etiquetas (funcionan de forma análoga a un backup del repo en un momento dado solo que este es almacenado dentro del servidor centralizado a diferencia de un Bck normal que se encuentra en una unidad de almacenamiento diferente), resulta ser más simple que en otros sistemas como Gitt. |
| Centralizado | **GitHub (la plataforma)**: Centralizada en términos de que los repositorios están alojados en sus servidores. Los usuarios interactúan con esta plataforma centralizada para realizar colaboraciones, gestionar repositorios, y otras tareas.  **Git (el sistema de control de versiones)**: Es **distribuido**, lo que significa que cada desarrollador tiene una copia completa del repositorio en su máquina local. Pueden hacer cambios, commits y trabajar de forma independiente sin estar conectados a la plataforma central hasta que necesiten sincronizar con el repositorio principal. | GitLab puede ser centralizado si se usa la versión de GitLab.com, donde los repositorios y datos se alojan en los servidores de GitLab. | No es centralizado. | Es un sistema de control de versiones centralizado. Aquí, hay un único repositorio central donde se almacenan todos los cambios, y los desarrolladores trabajan haciendo "check-out" y "commit" a ese repositorio. |
| Distribuido | Aunque GitHub es centralizado, Git en sí es una herramienta distribuida. Los desarrolladores pueden clonar el repositorio a sus máquinas locales, trabajar de manera offline y hacer commits localmente antes de sincronizar los cambios con el repositorio principal. | GitLab ofrece una opción de instalación **on-premises**, lo que permite que la organización aloje su propia Instancia de GitLab y distribuya el control del código dentro de su infraestructura local, haciéndolo completamente distribuido. | Es distribuido ya que cada desarrollador tiene una copia del repositorio y a sí mismo del historial de cambios. | No es distribuido. |
| Características Especiales | GitHub Actions para CI/CD.  GitHub Copilot para asistencia con código mediante IA.  Integraciones con herramientas externas como Jira, Slack, y Azure.  GitHub Pages para alojar sitios web estáticos directamente desde los repositorios. | CI/CD nativo con pipelines que se pueden personalizar en profundidad.  Integración con Kubernetes para despliegue automatizado.  Mayor enfoque en DevSecOps con características de escaneo de seguridad integradas.  Gráficas de burndown y otras herramientas avanzadas de gestión de proyectos.  GitLab Pages para alojar sitios web estáticos. | Desarrollo offline: ya que es descentralizado, permite realizar commits locales antes de sincronizarse con el servidor principal.  Historial de cambios predecible: Permite ver el flujo y evolución de las ramas de una forma más sencilla que con git ya que por ejemplo las ramas son inmutables de manera que siempre serán visibles desde el historial, cosa que no ocurre con git, ya que tenemos opciones para eliminarlas. | Commits atómicos: A diferencia de otras herramientas en subversion los commits se aplican completamente o no se hacen en absoluto, lo que facilita el control de la integridad del repositorio.  Soporta el trabajo parcial: Permite que el desarrollador solo necesite descargar la información relevante si así gusta, para poder trabajar en sus funcionalidades, esto agiliza el flujo en grandes y complejos proyectos.  Permite el bloqueo de archivos: como las imágenes o bin que no son fácilmente modificables. |

**Conclusión:** Se eligió Github como herramienta, ya que el equipo de desarrollo se encuentra familiarizado con sus características y además ofrece una gran cantidad de funcionalidades avanzadas que permiten un flujo de trabajo que se adapta mejor a las necesidades del equipo.



* 1. Definición de nomenclaturas de ítems

Caso 1: Para ítems que pertenecen a un proyecto

Caso 2: Si los nombres son iguales

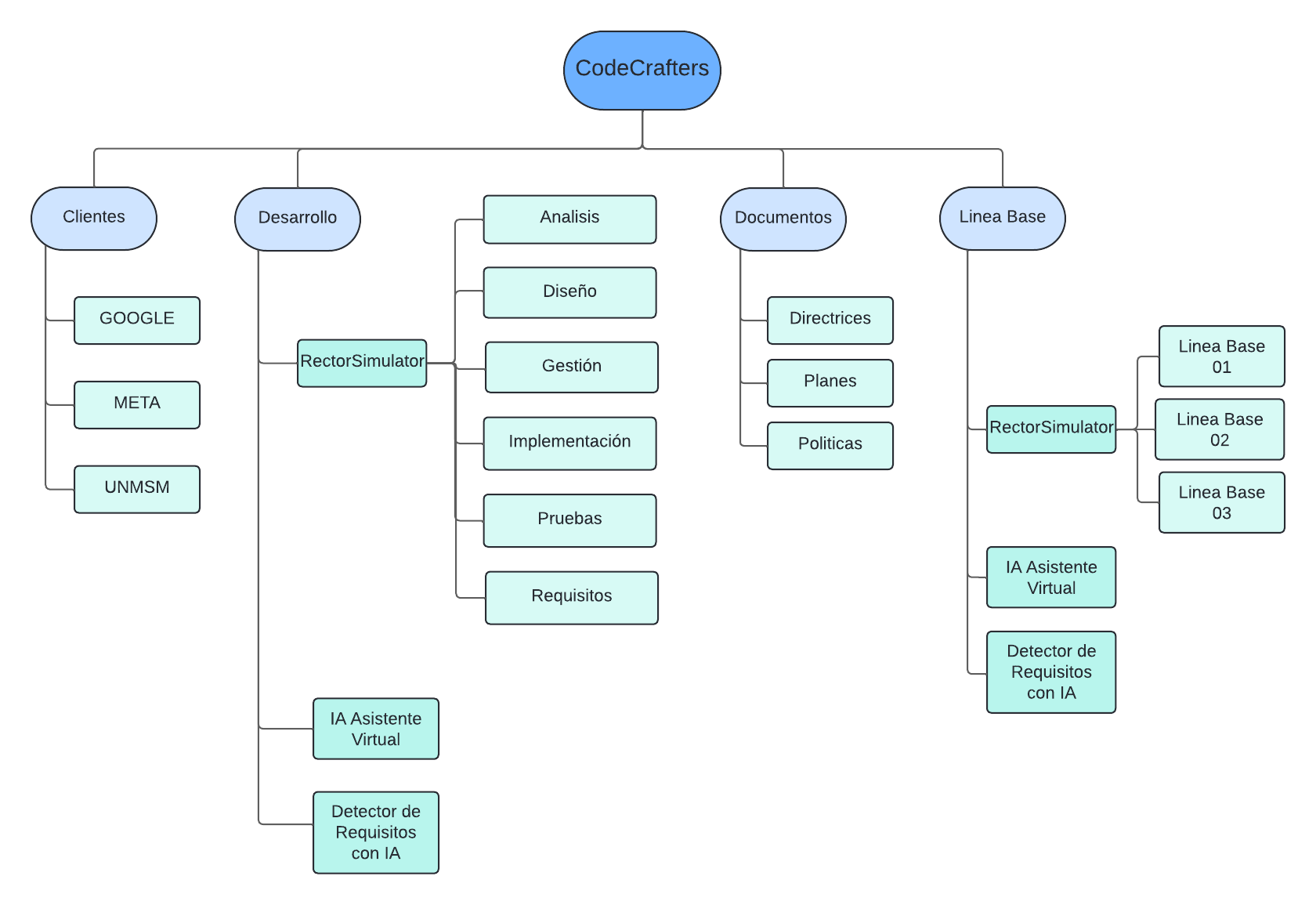
Caso 3:Si los documentos son del mismo tipo



* 1. Clasificación de los ítems

| **Tipo**  (E= Evolución  F= Fuente  S= Soporte) | **Nombre del Item** | **Nomenclatura** | **Extensión** | **Proyecto** |
| --- | --- | --- | --- | --- |
| E | Project Charter | RS-PC | docx | RS |
| E | Cronograma del proyecto | RS-CP | xlsx | RS |
| E | Documento de historia de Usuario 01 | RS-DHU01 | docx | RS |
| E | Documento de historia de Usuario 02 | RS-DHU02 | docx | RS |
| E | Documento de historia de Usuario 03 | RS-DHU03 | docx | RS |
| E | Documento de historia de Usuario 04 | RS-DHU04 | docx | RS |
| E | Documento de historia de Usuario 05 | RS-DHU05 | docx | RS |
| E | Documento de historia de Usuario 06 | RS-DHU06 | docx | RS |
| E | Documento de lista de epicas | RS-DLE | docx | RS |
| E | Documento de análisis | RS-DA | docx | RS |
| E | Jira | - | - | RS |
| E | Documento de estado del proyecto | RS-DEP | docx | RS |
| E | Documento de arquitectura | RS-DEBD | docx | RS |
| E | Documento de especificacion de UI | RS-DEUI | docx | RS |
| E | Documento de especificacion de DB | RS-DEDB | docx | RS |
| F | Funcionalidad de historia de usuario 1 | //Front, //Back | - | RS |
| F | Funcionalidad de historia de usuario 2 | //Front, //Back | - | RS |
| F | Funcionalidad de historia de usuario 3 | //Front, //Back | - | RS |
| F | Funcionalidad de historia de usuario 4 | //Front, //Back | - | RS |
| F | Funcionalidad de historia de usuario 5 | //Front, //Back | - | RS |
| F | Funcionalidad de historia de usuario 6 | //Front, //Back | - | RS |
| S | Python 3.12.3 | - | .py | RS |
| S | React 19 | - | .jsx | RS |
| S | Django 5.1.1 | - | .py | RS |
| S | MySQL 8.0.34 | - | - | - |

* 1. Diseño de la estructura del repositorio



* 1. Definición de Líneas Base

Proyecto Rector Simulator

| **Hito** | **Nombre de línea base** | **Items (nomenclatura)** |
| --- | --- | --- |
| Hito 1: 20/09/24 | Línea Base 1 (Análisis) | Repositorio de Github (CodeCrafters)  Project Charter (RS-PC.docx)  Cronograma del proyecto (RS-CP.xlsx)  Documento de historia de Usuario 01 (RS-HU01.docx)  Documento de historia de Usuario 02 (RS-HU02.docx)  Documento de historia de Usuario 03 (RS-HU03.docx)  Documento de historia de Usuario 04 (RS-HU04.docx)  Documento de historia de Usuario 05 (RS-HU05.docx)  Documento de historia de Usuario 06 (RS-HU06.docx)  Documento de lista de épicas (RS-DLE.docx)  Documento de análisis (RS-DA.docx)  Entorno de gestión (RS-EG.docx)  Documento de estado del proyecto (RS-DEP.docx) |
| Hito 2: 12/10/24 | Línea Base 2 (Diseño) | Documento de arquitectura (RS-DEBD.docx)  Documento de especificacion de UI (RS-DEUI.docx)  Documento de especificacion de DB (RS-DEDB.docx)  Documento de estado del proyecto (RS-DEP.docx) |
| Hilo 3: 14/11/24 | Línea Base 3 (Desarrollo y despliegue) | Funcionalidad de historia de usuario 1 (//Front, //Back)  Funcionalidad de historia de usuario 2 (//Front, //Back)  Funcionalidad de historia de usuario 3 (//Front, //Back)  Funcionalidad de historia de usuario 4 (//Front, //Back)  Funcionalidad de historia de usuario 5 (//Front, //Back)  Funcionalidad de historia de usuario 6 (//Front, //Back)  Documento de pruebas de software (RS-DPS.docx)  Aplicación desplegada (RS-AD)  Documento de estado del proyecto (RS-DEP.docx) |