ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

“Desarrollo de un repositorio de versionamiento mediante la herramienta gitlab para almacenar de código fuente de los sistemas de software en la dirección de tecnologías de la información y comunicación del ejército ecuatoriano”.

Soria Sailema, José Luis

Departamento de Electrónica y Computación

Carrera de Tecnología en Computación

Monografía Previa a la Obtención del Título de Tecnólogo en Computación

Ing. Bastidas Bravo, William Robert

14 de agosto de 2021

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE TECNOLOGÍAS Y COMPUTACIÓN

CERTIFICACIÓN

Certifico que la monografía, **"DESARROLLO DE UN REPOSITORIO DE VERSIONAMIENTO MEDIANTE LA HERRAMIENTA GITLAB PARA ALMACENAR DE CÓDIGO FUENTE DE LOS SISTEMAS DE SOFTWARE EN LA DIRECCIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN DEL EJÉRCITO ECUATORIANO",** fue realizado por el señor **Soria Sailema, José Luis**, la cual ha sido revisada y analizado en su totalidad por la herramientas de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPEL, razón por la cual me permito acreditar y autorizar para lo que sustente públicamente.

Latacunga, 14 de agosto de 2021

Firma:

WILLIAM ROBERT BASTIDAS BRAVO

Ing. Bastidas Bravo, William Robert

C.C.:0501908636

Curiginal

Document Information

Analyzed document	MonografiaSoriaV2.docx (D110778071)
Submitted	7/27/2021 12:50:00 AM
Submitted by	
Submitter email	jrcaiza@espe.edu.ec
Similarity	9%
Analysis address	jrcaiza.espe@analysis.urkund.com

Sources included in the report

SA	**Universidad de las Fuerzas Armadas ESPE / Tesis Josè Luis Soria - GitLab Lista.pdf** Document Tesis Josè Luis Soria - GitLab Lista.pdf (D110220159) Submitted by: jrcaiza@espe.edu.ec Receiver: jrcaiza.espe@analysis.urkund.com	28
W	URL: https://gitlab.sideralis.co/community/documentacion/-/wikis/C%C3%B3mo-funciona-Gitlab Fetched: 7/27/2021 12:51:00 AM	1

Firma:

Firmado electrónicamente por:
WILLIAM ROBERT
BASTIDAS BRAVO

Ing. Bastidas Bravo, William Robert

C.C.:0501908636

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE TECNOLOGÍAS Y COMPUTACIÓN

RESPONSABILIDAD DE AUTORÍA

Yo, **SORIA SAILEMA, JOSÉ LUIS,** declaro que toda la investigación, criterios e ideas de esta monografía **"DESARROLLO DE UN REPOSITORIO DE VERSIONAMIENTO MEDIANTE LA HERRAMIENTA GITLAB PARA ALMACENAR DE CÓDIGO FUENTE DE LOS SISTEMAS DE SOFTWARE EN LA DIRECCIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN DEL EJÉRCITO ECUATORIANO",** es de mi total autoría, responsabilidad y dando cumpliendo a todos los requisitos legales teorías científica, metodológicas y técnicas establecidos por la universidad de las Fuerzas Armadas ESPEL, siendo consientes y respetando todos los derechos intelectuales de terceras personas y referenciado las respectivas citas bibliográficas.

Latacunga, 14 de agosto 2021

Soria Sailema, José Luis
C.C.: 0502914781

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE TECNOLOGÍAS Y COMPUTACIÓN

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **SORIA SAILEMA, JOSÉ LUIS,** autorizo a la Universidad de las Fuerzas Armadas ESPEL publicar la presente monografía **"DESARROLLO DE UN REPOSITORIO DE VERSIONAMIENTO MEDIANTE LA HERRAMIENTA GITLAB PARA ALMACENAR DE CÓDIGO FUENTE DE LOS SISTEMAS DE SOFTWARE EN LA DIRECCIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN DEL EJÉRCITO ECUATORIANO",** en el repositorio Institucional, cuyo trabajo investigativo, criterios e ideas son de mi responsabilidad.

Latacunga, 14 de agosto de 2021

Soria Sailema, José Luis
C.C.: 0502914781

DEDICATORIA

El presente trabajo de titulación en primer lugar está dedicado a nuestro Padre Celestial, por la vida que nos otorga día a día para culminar con éxito este trabajo de investigación y seguir cumpliendo nuestros objetivos planteados a lo largo de nuestra vida personal y profesional.

A mi esposa Nataly que ha sido el pilar incondicional en este largo caminar, que comprendió mi ausencia y siempre estuvo dándome palabras de aliento para no desmayar, un agradecimiento profundo a mis hermosos hijos Kerly y Jeinner por siempre recibirme con los brazos abiertos, con una sonrisa en sus rostros que me alentaban todos los días y de esta manera ser un ejemplo para ellos.

Como no hacerles partícipe a mis compañeros y a los ilustres docentes que impartieron sus experiencias y su conocimiento durante este proceso de aprendizaje para dar fin a un propósito planteado cuando iniciamos esta linda carrera.

Para concluir un agradecimiento especial a mi madre que desde el cielo cuida de su hijo que cada vez cumple los objetivos planteados y dispuestos con ella; a mi padre que sigue apoyándome con una voz de aliento además de ser un ejemplo en mi vida.

Soria Sailema, José Luis

AGRADECIMIENTO

Un agradecimiento profundo a todos los que me apoyaron que son mis familiares, docentes, amigos, que durante este largo tiempo han aportado con un granito de arena un consejo, palabras alentadoras para poder llegar a su fin con éxito, siempre con la constancia la paciencia y sobre todo con la dedicación, que como estudiantes nos caracteriza, en la vida existen muchas circunstancias positivas y negativas de las cuales debemos elegir con madurez el camino correcto para seguir adelante, y nunca desmayar en el primer tropiezo, es por ello que ahora están y seguirán siendo parte de mi vida profesional y con una amistad llena de sinceridad.

Soria Sailema, José Luis

Tabla de contenidos

Índice de figuras

Resumen

El presente trabajo investigativo tiene como finalidad la elaboración de un repositorio de almacenamiento de código fuente con la herramienta GitLab, el mismo que tiene como objetivo de archivar los sistemas de software que realizan los desarrolladores de la Dirección de Tecnologías de la Información y Comunicaciones del Ejército, la codificación o líneas de programación reposan en el repositorio, estando disponible para el desarrollador cuando requiera efectuar una continuidad de versión, cambio o a su vez revertir a una versión anterior del sistema de software que lo necesite, este repositorio GitLab ayudará a mantener un debido control de versionamiento de todos los sistemas de software que elaboren en ésta Dirección, además que fue diseñada para minimizar la carga laboral entre los profesional que se dedican al desarrollo de software con una herramienta de fácil interacción con el usuario, cuenta con muchas características y facilidades al momento de realizar los proyectos, su ventaja principal está diseñada para el trabajo distribuido esto quiere decir que tiene un repositorio central y una serie de repositorios locales así fomentar el trabajo en equipo y ser eficientes a la hora de entregar el producto terminado, es por ello que se ha optado por trabajar con esta herramienta, cabe recordar que la Dirección es el ente proveedor de sistemas para la fuerza terrestre en todos los aspectos tecnológicos mostrando un alto comprometimiento con nuestra institución

Palabras Clave:

- **REPOSITORIO GITLAB**
- **LENGUAJE DE PROGRAMACIÓN**
- **INGENIERÍA DE SOFTWARE**

Abstract

The purpose of this research work is the development of a source code storage repository with the GitLab tool, which has the objective of archiving the software systems developed by the developers of the Information Technology and Communications Directorate of the Army, The coding or programming lines rest in the repository, being available to the developer when required to make a version continuity, change or revert to a previous version of the software system that needs it, this GitLab repository will help to maintain a proper versioning control of all software systems developed in this Directorate, Besides it was designed to minimize the workload among professionals who are dedicated to software development with a tool for easy interaction with the user, it has many features and facilities at the time of making projects, Its main advantage is designed for distributed work, this means that it has a central repository and a series of local repositories to encourage teamwork and be efficient when delivering the finished product, which is why we have chosen to work with this tool, remember that the Directorate is the supplier of systems for the land force in all technological aspects showing a high commitment to our institution.

Key Words:

- **GITLAB REPOSITORY**
- **SOURCE CODE ARCHIVER**
- **VERSIONS CONTROL**

CAPÍTULO I

1. Tema

Desarrollo de un repositorio de versionamiento mediante la herramienta GitLab para almacenar de código fuente de los sistemas de software en la dirección de tecnologías de la información y comunicación del ejército ecuatoriano**.**

1.1 Antecedentes

En la actualidad la calidad del software se ha convertido en un eje primordial dentro del entorno social que utilizan los procesos para el desarrollo del mismo; en el cual se está invirtiendo grandes esfuerzos para lograr obtener productos de alta calidad. Actualmente el análisis y desarrollo de repositorios que se encuentra bajo diferentes normas, certificaciones de procesos, además; prácticas y herramientas que brindan mejoras en análisis y diseño de los repositorios de versionamiento.

Por la relevancia del tema se han analizado varios documentos digitales mismos que abordan las siguientes investigaciones las cuales tienen relación con el tema de estudio.

En la ciudad de Guayaquil en la Universidad Politécnica los autores Chavarría B. & Gudiño E. (Chavarría Neira, 2017) en su tesis titulado “Implementación de un repositorio web y un diseño de una página utilizando herramientas de software libre” destaca que: una de las formas de hacerlo es mediante el uso del software libre; que se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software de modo más preciso.

Sobre este mismo tema, en la ciudad de México en la Universidad Autónoma de Tamaulipas los autores Tello E. & Sosa C. (R. E. T., 2012) en la tesis que lleva como título “Revisión de los sistemas de control de versiones utilizados en el

desarrollo de software” argumenta que: los sistemas de control de versiones son aplicaciones que ayudan al proceso de desarrollo del software facilitando la gestión del control de versiones de archivos de código fuente generado por los desarrolladores además, el proyecto permitió recuperar archivos generados previamente, los cuales pueden ser utilizados para solucionar errores del sistema.

Por lo tanto, los resultados de las investigaciones sirvieron como base para recomendar que se dispongan de un mecanismo que permita coordinar las actividades y resultados de todos los desarrolladores involucrados, la Dirección de Tecnología de la Información y Comunicación del Ejército Ecuatoriano deberían disponer de los SCV permiten que un proyecto pueda avanzar con varias versiones al mismo tiempo y generar informes que muestren los cambios entre las etapas del proyecto.

1.2 Planteamiento del Problema

El Ejército ecuatoriano va con la mano con la gesta imperecedera del 10 de agosto de 1809, cuando al albor de la libertad, nace el Ejército ecuatoriano, cuya labor en más de dos siglos ha contribuido indiscutiblemente a la edificación del Ecuador democrático y soberana. Particularmente en el Departamento de la Dirección de Tecnologías de la Información y Comunicación tiene como propósito de realizar sistemas de software para el adelanto institucional y dar cumplimiento a todas las necesidades que emana el mando militar en sus subunidades bajo su mando, mismo que cuenta con un repositorio actualizado y con mejores herramientas que controle sus versiones en los sistemas de software; en la actualidad disponen de un controlador de versiones demasiado básico y casi obsoleto (subversión).

- Proyectos difíciles de gestionar y liderar.
- Riesgos a sobrescribir con mi código el avance formal del equipo

- La centralización y poca probabilidad de trabajar remotamente.
- No generar modificaciones en el sistema de software
- Contar con información de la persona que lo hizo y en las líneas de código que fueran afectadas.
- No tener control de los sistemas de software.

De no solucionar seguirá con el problema de no tener un control exhaustivo de todos los sistemas de software y se corre el riesgo incluso de perder tiempo de trabajo

Por lo expuesto es fundamental que el Departamento de la Dirección de Tecnología de la Información y Comunicación, cuente con software para el almacenamiento de código fuente, así como tener un mayor control de los sistemas de software; con el siguiente fortalecimiento del desempeño militar y coadyuvar el cumplimiento de las misiones encomendadas por el escalafón superior del Ejército Ecuatoriano.

1.3 Justificación

En la actualidad la industria del software cumple un papel cada vez más indispensable para la economía mundial, los sistemas de software han trasformado los procesos de control de la mayoría de las actividades de las cuales dependemos cada día; es así como surgen más y mejores tecnologías de esta manera se facilita el convivir diario del ser humano y su inserción en el mundo cibernético. Se considera factible el presente proyecto porque está relacionado al ámbito tecnológico, permitiendo a través de esta investigación dar una mayor facilidad a los programadores y ofrecerles mejores herramientas actualizadas para trabajar en el desarrollo de los sistemas de software, los mismo que dispondrán de un versionamiento cronológico y único por cada sistema de software que exista en la

Dirección de Tecnologías de la Información y Comunicación. Esta investigación es de gran utilidad por los aportes teóricos, prácticos y metodológicos en el desarrollo de los sistemas de software centralizando el código fuente y disponer del control de versiones cuando se necesite y revertir una modificación anterior, además se cuenta con profesionales comprometidos con la calidad de su trabajo y el desarrollo del país.

Se busca solucionar el problema de una manera automatizada que genere una actualización diaria de todas los sistemas de software en las que trabaja la Dirección de Tecnologías de la Información y Comunicación y llevar un control minucioso y documentado de los sistemas mediante un repositorio que cumpla con todos los estándares y normas de seguridad, los resultados se verán reflejados una vez que el proyecto haya cursado todas sus fases de manera exitosa, la cual debe estar lista para su implementación en su entorno de trabajo, además proporcionará un informe periódico de las versiones que desde ese momento se efectúen.

Los beneficiarios para la ejecución de este proyecto serán directamente la Dirección de Tecnologías de la Información y Comunicación por ende el Ejército ecuatoriano, lo que permite a sus desarrolladores brindarles esta herramienta de trabajo actualizada, una manera fácil y estable para conocer la versión exacta en la que se encuentra cada sistema de software. Los resultados se verán reflejados una vez que el proyecto haya cursado todas sus fases de manera exitosa, la cual debe estar lista para su implementación en su entorno de trabajo, además proporcionará un informe periódico de las versiones que desde ese momento se efectúen.

Por lo expuesto es necesario adquirir esta herramienta de control de versionamiento será de mucha ayuda para los desarrolladores de sistemas de software que laboran el departamento de DTIC.

1.4 Objetivos

1.4.1 Objetivo General:

Desarrollar un repositorio de versionamiento mediante la herramienta GitLab para almacenar el código fuente de los sistemas de software en la Dirección de Tecnologías de la Información y Comunicación del Ejército ecuatoriano.

1.4.2 Específicos:

- Determinar los fundamentos científicos de la herramienta GitLab para el repositorio de control de versionamiento.
- Examinar la situación actual del repositorio de control de versionamiento (subversión) que dispone la Dirección de Tecnologías de la Información y Comunicación.
- Desarrollar un repositorio de control de versionamiento para el almacenamiento del código fuente de los sistemas software de la DTIC.

1.5 Alcance

El presente trabajo de investigación tiene como alcance la representación y el desarrollo de un repositorio de control de versionamiento y su conciliación a la infraestructura como código fuente, enfocándose en la gestión de las aplicaciones tecnológicas existentes en la Dirección de Tecnologías de la Información y Comunicación.

Como parte del proyecto se incluye algunos de los desafíos que enfrenta la Dirección de Tecnologías de la Información y Comunicación tales como: la migración de datos, la gestión de la infraestructura que tengan confiabilidad para el sistema, destinar recursos económicos, además la interfaz debe ser amigable con el desarrollador, y las actualizaciones de los profesionales con el sistema; el núcleo de

la investigación desarrolla los principios de la infraestructura del repositorio de código fuente, desarrollo evolutivo, continuidad de servicio, sistemas auto probados sistemas auto documentados, cambios pequeños de versionamiento. También servirá de fuente de información y consulta para todas las personas relacionadas o interesadas en el tema.

CAPÍTULO II

2. Marco teórico

2.1 ¿Qué es la Ingeniería de Software?

Desde el período de los sesenta empieza ha existir un movimiento con tendencia a cambiar sobre todo en la programación de computadoras de un estado artesanal al campo de la ingeniería; donde se puede evidenciar que la aparición del software en la ingeniería integra métodos, herramientas y procedimientos con el único objetivo el de introducir la técnica disciplinada en el desarrollo del software.

Según (Roa, 2017) indica que una de las primeras definiciones de la ingeniería de software aparece en el año de 1969 manifestando que es el establecimiento esencial en el desarrollo de la tecnología y que a través del uso de los mismos estarán orientados a obtener que el funcionamiento sea fiable sobre las máquinas reales". Por otro lado, la (UNIR, 2021) manifiesta que la ingeniería del software es una disciplina que aborda todas las fases del ciclo de vida de cualquier tipo de sistema de información y que la misma permite el control absoluto de un sistema operativo o un sistema de gestión de redes. Para los referidos autores la ingeniería de software es aplicable a la amplia gama de la ciencia de los ordenadores y el desarrollo de programas informáticos que son fundamentales y forma parte en cualquier operación humana en la actualidad.

2.2 ¿Qué es Software?

Según la Real Academia Española, 2016 (RAE) enfatiza que la palabra software es un vocablo inglés que hace referencia a un conjunto de programas que son aplicativos dentro de la informática, es decir permiten ejecutar distintas tareas en la parte lógica de la computadora. A lo que el Instituto de Ingenieros Eléctricos y Electrónicos, 2014 (IEEE) destaca que software es la suma de todos los programas

de ordenadores, reglas y procedimientos asociados a los datos que pertenecen a un sistema de computo; y el producto de software que es un producto elaborado directamente para el usuario.

En esencia el software es mucho más que un código que está dentro de un computador; es una aplicación inteligente de principios aprobados, técnicas y herramientas que son indispensables para las necesidades de los usuarios que con el tiempo han permitido la facilidad de resolver problemas en la administración, calidad, productividad fomentando de esta manera mejorar la competitividad mundial en adquirir información con mayor eficacia.

2.3 Importancia del Software

El software es un ingrediente indispensable para el funcionamiento eficaz del computador; en sí un computador solo es un conglomerado de componentes electrónicos, es así como el software le da vida al computador permitiendo aprovechar todos sus componentes y de esta manera funcione de forma ordenada; por lo que a la hora de crear una aplicación en el software este permitirá al ser humano realizar la gestión en un plazo determinado y buscando la aceptabilidad en el mundo tecnológico. Por ello se ha citado a varios autores que exponen lo siguiente:

(Maida, EG, Pacienzia, J, 2015) destaca que la importancia del software permiten una comunicación entre el ususaio y la máquina donde se ve evidenciado que en la actualidad podemos ejecutar tareas con más rapidez que hace decádas atrás nos hubiera llevado años de trabajo es así; como dan paso a una revolución mundial en la sociedad moderna.(p.14). A lo que responde (Hernández, 2008) “El software es indispensable; y en la tecnología ha avanzado de una manera impresionante y significativa”; ya que es la clave principal para el éxito de negocios y empresas, a través de la construcción de sistemas complejos, inteligencia artificial,

automatización de maquinarias entre otras facilitando de esta manera el trabajo de los seres humanos y en posibles años querer reemplazar al ser humano por una máquina como es un robot.

2.4 Clasificación del Software

Es importante analizar algunos tipos de software, este análisis permitirá entender el desarrollo tradicional del software llegando a la actualidad con rapidez y dinamismo que hacen que nuestro entorno se rodee de más competitividad al aumentar la productividad y satisfacer las necesidades del ser humano. En este contexto (Roger S. Pressman, 2010) clasifica al software de la siguiente manera:

2.4.1 Software de Sistemas

Se refiere al conjunto de programas escritos y que son necesarios para dar servicios a otros programas e interrelacionarse con los diferentes sistemas operativos, permitiendo de esta manera una interacción y un total control sobre el aplicativo hardware de la computadora y que de esta manera se puede realizar un uso intensivo por parte de usuarios múltiples, operación concurrente que requiere la secuenciación de recursos compartidos y administración de un proceso sofisticado, la cual permitirá una estructura más compleja de datos e interfaces externas y múltiples.

Entre los más conocidos del Software de Sistemas contamos con los siguientes operativos como son: Windows, Mac OS y Linux, que se complementa con herramientas de diagnóstico, controladores de dispositivos, entorno de escritorio, BIOS, servidores de información y herramientas de Corrección y Optimización, cargadores de programas, entre otras.

2.4.2 Software de Aplicación

Es aquel que nos ayuda a realizar una tarea determinada donde se ubican

programas que son funcionales a las necesidades de los individuos, pero especialmente a las empresas, en sí el uso de estas aplicaciones procesan operaciones por medio de datos comerciales o técnicos.

Además de las aplicaciones convencionales de procesamiento de datos, el software de aplicación se usa para controlar funciones de negocios en tiempo real (por ejemplo, procesamiento de transacciones en punto de venta, control de procesos de manufactura en tiempo real).

2.4.3 Software de Programación

Es el conjunto de herramientas que permite al desarrollador elaborar sistemas informáticos, que dan origen a los programas que utilizamos hoy en día es así como el programador va ubicando instrucciones en la computadora que dará la facilidad al usuario para complementar la aplicación. Entre ellos tenemos los:

- Editores de texto
- Compiladores
- Intérpretes
- Enlazadores
- Depuradores

2.5 Características del Software

Es importante analizar algunas características que tiene el Software donde nos indican que es un conjunto de programas asociadas directamente a un sistema informático incluyendo un sistema operativo, el Software es realizado directamente por los programadores a través de una escritura de códigos fuente, el cual puede ser traducido por medio de un compilador para que de esta manera la computadora

pueda pueda ejecutar con toda normalidad y sin tantas falencias.

En este sentido (Cavsi, 2016) menciona tres características principales del Software.

2.5.1 Características Operativas del Software

Son factores de funcionalidad, es como se presenta el software, es la parte interior del mismo. Incluye aspectos como:

- **Corrección:** El software que estamos haciendo debe satisfacer todas las especificaciones establecidas por el comprador.
- **Usabilidad:** Debe ser sencillo de aprender.
- **Integridad:** Un software de calidad no debe tener efectos secundarios.
- **Fiabilidad:** El producto del Software de calidad no debe tener ningún defecto. No solo esto, no debe fallar mientras se vaya ejecutando.
- **Eficiencia:** El software debe hacer un uso eficaz del espacio de almacenamiento y el comando ejecutar según los requisitos de tiempo deseado.
- **Seguridad:** Se deben tomar medidas apropiadas para mantener los datos a salvo de las amenazas externas.

2.5.2 De Transmisión de Software

- **Interoperabilidad**: Es la capacidad para el intercambio de información con otras aplicaciones.
- **Reutilización:** Es poder utilizar el código de software con algunas modificaciones para diferentes propósitos.
- **Portabilidad:** Capacidad para llevar a cabo las mismas funciones en todos

los entornos y plataformas.

2.5.3 De Revisión de Software

Son los factores de ingeniería, la calidad interior del software como la eficiencia, la documentación y la estructura. Incluye aspectos como:

- **Capacidad de mantenimiento:** El mantenimiento del software debe ser fácil para cualquier tipo de usuario.
- **Flexibilidad:** Los cambios del software deben ser fáciles de hacer.
- **Extensibilidad:** Debe ser fácil de aumentar nuevas funciones.
- **Escalabilidad:** Debe ser muy fácil de actualizar para más trabajo.
- **Modularidad:** Debe estar compuesto por unidades y módulos independientes entre sí.

2.6 Ciclo de la vida del Software

El ciclo de la vida del Software permite que los errores se detecten lo antes posible, y por lo tanto los desarrolladores se enfocan en mantener la calidad y años duraderos y precios módicos para la salida de venta al mercado, es indispensable que el programador garantice una verificación completa ya que estos programas deben ser eficientes, fiables y sobre todo seguros para el beneficio del cliente.

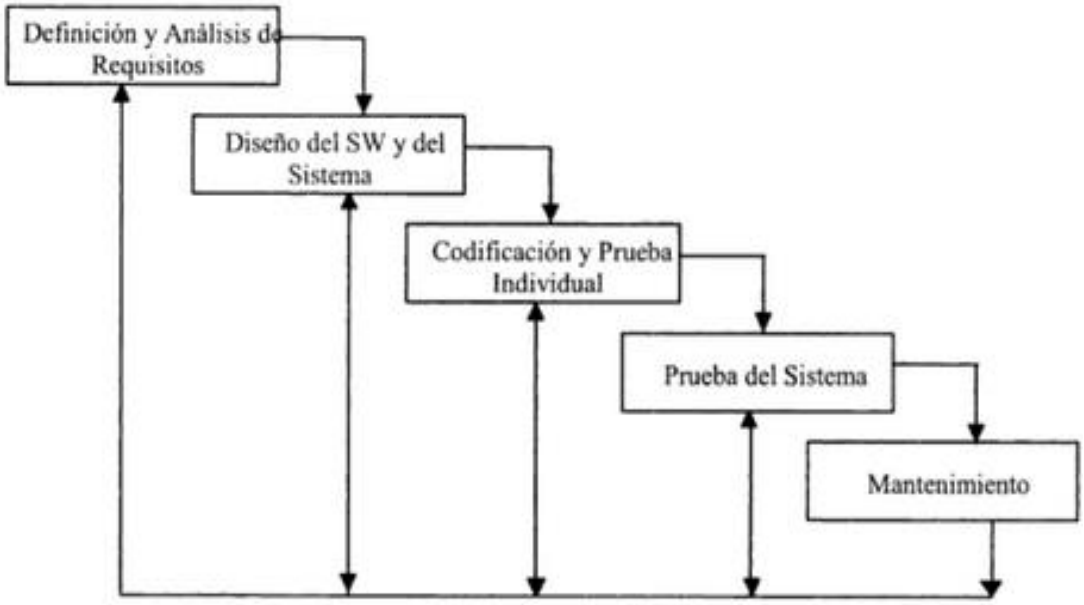
Es así como en la revista conocida como (Intelequelía, 2020) menciona que la vida de los programas informáticos debe tener una terminología bien definida, a la que pueda remitirse la industria del Software que surgió por necesidad del producto hasta que se cumplió el objetivo por el cual fue creado, y por el cual es importante para el desarrollo de la sociedad en la actualidad tomando en cuenta el surgimiento de la tecnología a nivel mundial.

De esta manera el ciclo de vida básico de un software consta de los

siguientes procedimientos.

Figura 1

“Ciclo de la vida del Software”



Nota: Representa cada proceso que el software debe pasar hasta llegar al producto terminado.

Para el Dr. (García, 2018) indica que el ciclo de vida del Software representa la evolución de un sistema construida por los seres humanos desde su concepción hasta la actualidad, es por ellos que debemos tener bien identificado por las distintas fases por las que pasa el Software que empieza como una necesidad de mecanizar un proceso hasta dar un orden lógico, es decir que vamos a encontrar de una manera comprensible que los seres humanos podamos entender y ejecutar.

Es así como citamos a (Sanchez, 2013) en donde el indica el proceso del ciclo de vida del Softaware que son los siguientes:

Fase de Requisitos del Sistema: consiste en establecer e identificar lo que los usuarios necesitan de este sistema y si existe falencias en los programas poder mejorarlos y así poder cumplir con las necesidades del usuario teniendo en cuenta posibles restricciones. Para el desarrollo del Software el analista debe comprender el ámbito, el entorno y el alcance del sistema además de la información obtenida y de

las funciones e interfaces que se requieran, en esta etapa es fundamental utilizar herramientas técnicas adecuadas que permitan estudiar el problema en detalle y dar una visión de la solución mediante modelos que tengan en cuenta.

Fase de Diseño del Sistema: Se enfoca al desarrollo de la estructura física, es decir a los datos que el sistema va a manejar y almacenar, en el Software, y de esta manera identificar como van a estar conectados los diferentes programas, módulos y procedimientos con las interfaces del sistema con los usuarios o con otros sistemas.

Fase de Codificación y Prueba Individual: Es la traducción del diseño del lenguaje que el Hardware comprenda, cuando el diseño se especifica de forma detallada, esta tarea suele ser casi mecánica.

Fase de Prueba de Sistemas: Consiste en probar todos los módulos de forma conjunta, su interconexión y la completitud del sistema.

Fase de Mantenimiento: Incluye todas las tareas que se llevan a cabo para incorporar posibles cambios una vez que el Software se pone en funcionamiento, se puede evidenciar errores y el mismo está obligado a adaptarse a cambios externos sin que exista una afectación al momento del uso; es así como se evidencia que los procesos muchas de las veces no siguen un orden secuencial se realiza a través de interacciones que surgen en la aplicación de un paradigma.

En esencia es difícil para el futuro usuario establecer desde el principio todos los requisitos, lo cual provoca dificultades al intentar incluir nuevas funcionalidades y proporcionar ciertos puntos de incertidumbre, es evidente que los resultados no son visibles hasta las últimas etapas del desarrollo. El futuro usuario en muchas ocasiones se impacienta ya que detectan errores en el funcionamiento del sistema es por ello que deben tener calma y esperar dar solución a este programa que son de vital importancia para el ser humano sin olvidar que todo es un proceso hasta que

se complemente su funcionamiento. (p.55.58).

2.7 Metodología de Software

(Maida G, 2015) informa que la metodología para el desarrollo del Software comprende un conjunto integrado de técnicas y métodos que permiten abordar de forma homogénea y abierta con la combinación de los modelos y procesos genéricos; además es importante tener conocimiento de estás metodologías por la demanda existente hoy en día por parte de muchas empresas , es por ello que se puede identificar que una de las etapas más importantes es poder distinguir que metodologías se asemeja al programador, para así lograr un mejor resultado en tiempo y forma. (p.12-13)

Es así como se puede identificar que la realización de la metodología del Software no es una tarea fácil; es decir que cada una de estas propuestas han llegado a través de un proceso empezando de una manera tradicional donde se centraron especialmente en el control del proceso sistemáticos, hoy en día podemos observar que con la ayuda idónea de la tecnología estos procesos nos han resultado más fáciles y con la apertura de que sí; existe un error inmediatamente se puede modificar sin que en ella existan afectaciones al usuario o al cliente, en este sentido podemos destacar algunas de las aplicaciones que van enmarcadas con la metodología del Software la cual;

- Optimiza el proceso y el producto del mismo.
- Guía los métodos de planificación y desarrollo del software.
- Define que hacer, cómo y cuándo y cuanto demora el desarrollo y mantenimiento de un proyecto.

Como una estrategia a utilizar en la metodología del Software contempla los siguientes elementos que forman parte como son:

- Fase: que es una tarea a realizar en cada etapa.

- Productos: es la fase de cada documento.
- Procedimientos y herramientas: es el apoyo a la realización de cada tarea.
- Criterios de evaluación: es el proceso y el producto, aquí podemos evidenciar el saber si se han logrado los objetivos.

Se concluye que la metodología del desarrollo del Software se enfoca en un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de los sistemas de información, durante los últimos años hemos ido observando la evolución en el marco de la tecnología; es así como se ha convertido en una industria con crecimiento vertical a nivel de todo el mundo y esto nos da la facilidad de crecimiento a nivel de conocimiento que es aquella que no debe quedarse estancada ya que todos los días es un nuevo aprendizaje, en este sentido el marco de trabajo de esta metodología consiste en:

- Una filosofía de desarrollo de software, con el enfoque en el proceso de desarrollo de software.
- Múltiples herramientas, modelos y métodos para ayudar en el proceso de desarrollo de software.

2.8 ¿Qué es una Metodología y Para que se Utiliza?

Una metodología que abarca el desarrollo del Software consiste en integrar a técnicas y métodos a través de una combinación de los modelos como un proceso genérico, adicional se pudo identificar que está vinculado con las actividades que involucran la creación, fabricación, actualización o modificación del mismo.

Es así que (Maida, 2015) en su investigación menciona que la metodología hace referencia al conjunto de procedimientos racionales utilizados para alcanzar un objetivo que requiera habilidades y conocimientos específicos para luego determinar cuál es el más adecuado y el más útil en el mercado que permitan a las empresas resaltar la calidad en la creación de estándares y modelos a utilizar. (p.18).

Finalmente, el uso de las metodologías en el Software es una de las etapas más específicas y relevantes de un proyecto que, si bien es cierto parte de la teoría y con ello conlleva a la búsqueda de una selección de técnicas y métodos para llegar a un procedimiento en el cumplimiento de los objetivos.

2.9 Metodología Tradicional

Las metodologías tradicionales no son más que la imposición de una disciplina de trabajo sobre el proceso del desarrollo del Software, para ello; es indispensable una planificación acorde al trabajo a realizar; una vez que todo este listo y detallado comienza una nueva etapa, donde se evidencia el control minucioso, es así como esta metodología no se adapta a todos los programas es difícil aceptar cambios por lo que no son unos métodos adecuados y mucho menos dan credibilidad a su uso, como el mundo van cambiando es imprescindible incluir nuevos conocimientos que van incluidos en el Software.

Es importante citar a (Maida, 2015) quién en su investigación destaca que para la existencia de desarrollar un buen Software depende de un gran número de actividades y etapas, donde el impacto de elegir la metodología para un equipo en un determinado proyecto es trascendental para el éxito del producto, es así como pone en énfasis que las metodologías tradicionales son denominadas, a veces, de forma despectiva, como metodologías pesadas. (p.18).

En esencia las metodologías tradicionales tratan de imponer una disciplina rigurosa de trabajo sobre el proceso del desarrollo del Software dejando a un lado el mejoramiento en el conocimiento para que este trabajo sea más ágil y eficiente, es necesario que cada desarrollador este adaptado a cambios que pueden ser importantes para agilizar el trabajo del usuario es así como resaltan dos de las características más importantes que tiene está metodología que son las siguientes:

2.10 Modelo en Cascada

Para (Sanz, 2015) este modelo define una metodología de construcción en el Software muy básica, que no permite introducir mejoras ni cambios durante el ciclo de vida del programa, lo que conlleva a desarrollar un modelo defectuoso en muchas ocasiones, debido a malas especificaciones o diseño de la misma es así; como la metodología tradicional en el Software descompone el desarrollo global de un proyecto en cinco fases únicas las cuales van en forma secuencial a través de las cuales se diseña o se construye un proyecto de principio a fin, y pone en énfasis a las siguientes fases que son indispensables para este modelo:

- **Análisis:** se contempla las necesidades del cliente para ejecutar un buen desarrollador del Software.
- **Diseño:** los desarrolladores de Software, a través de las necesidades del cliente diseñan y construyen sobre el papel una solución a medida de forma de diagramas UML organigramas y/o Pseudocódigo.
- **Implementación:** a partir del diseño, se crea una implementación del código del Software.
- **Verificación:** es la fase de pruebas en las que se comprueba que el código implementado que funcione de acuerdo con la especificación realizada.
- **Mantenimiento:** es la fase de corrección de errores.

2.10.1 Modelo Iterativo-incremental

La metodología en este modelo avanza con una evolución clara respecto al modelo en cascada, dado que en esta subdivide la construcción de un proyecto en pequeños bloques o componentes de funcionalidad, a los cuales se les aplica la metodología de guerra romana del “divide vencerás”, desarrollándose, en primer lugar, los módulos o bloques software de mayor importancia o prioridad para el

cliente, y descomponiendo los problemas más complejos en más pequeños y fáciles de desarrollar, lo que simplifica el desarrollo general de un proyecto, así como favorece su mantenibilidad y verificación.

Este incremento desarrolla una mejor funcionalidad para el cliente, permitiendo incluir pequeños cambios o modificaciones durante cada una de las iteraciones que se llevan a cabo sobre cada módulo del proyecto. La priorización de módulos del proyecto permite desarrollar de forma temprana prototipos que pueden ser probados por el propio cliente, lo que posibilita obtener una retroalimentación o feedback a tiempo del mismo, para poder introducir mejoras o cambio en el proyecto con el fin de adaptarlo a las necesidades particulares del cliente.

2.11 Metodología Ágiles

Para (Maida, 2015) indica que las metodologías tradicionales retrasan las decisiones y que se limitan en una planificación adaptiva es por ello que nace un enfoque como un requisito de respuesta a los problemas dando un status de solución fundamentado en la adaptabilidad del proceso del desarrollo del Software. (p19). Mientras que para (Jiménez, 2015) destaca que esta metodoloía es más ágil por que esta regida por doce principios que ayudan a que el proceso de desarrollo se vuelva menos complejo y responda de manera oportuna a los cambios que surgen a lo largo del mismo, siempre contando con el punto de vista del cliente. (p.7-8).

Finalmente, como conclusión podemos indicar que tanto los clientes como los desarrolladores de estos programas deben trabajar en una comunicación constante para agilizar el trabajo y que este proceso sea de manera oportuna y sencilla y sobre todo que sea adaptivo capaz de permitir cambios en el último momento sin que exista una afectación en el computador.

2.11.1 Scrum

Es una metodología de desarrollo ágil que tiene como base la idea de creación de ciclos breves para el desarrollo, que comúnmente se llama iteraciones y que en Scrum se llamara "**Sprint**". Para entender el ciclo de desarrollo de Scrum es necesario conocer las 5 fases que definen el ciclo de desarrollo ágil que son:

1 **Concepto. -** se define de forma general las características del producto y se asigna el equipo que se encargara de su desarrollo.

2 **Especulación. -** en esta fase se hacen disposiciones con la información obtenida y se establece límites que marcaran el desarrollo del producto, tales como costes y agendas son así como se construirá el producto a partir de las ideas principales y de la comprobación de las partes realizadas y su impacto en el entorno. Está fase se repite en cada iteración y consiste, en rasgos generales especialmente en:

   - Desarrollar y revisarlos los requisitos generales.
   - Mantener las listas de las funcionalidades que se esperan.
   - Plan de entrega donde se establece las fechas de las versiones, hotos e iteraciones la cual medirá el esfuerzo realizado en el proyecto.

3 **Exploración. -** en este aspecto se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación.

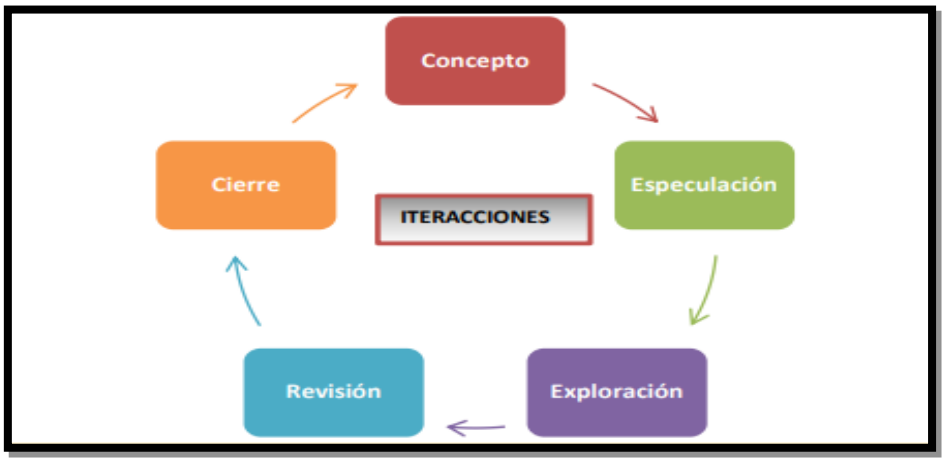
4 **Revisión. -** el equipo revisa todo lo que se ha construido y se contrasta con el objetivo deseado.

5 **Cierre. -** se entregará en las fechas acordadas una versión del producto deseado. Al tratarse de una versión, el cierre no indica que se ha finalizado el proyecto, sino que seguirá habiendo cambios, denominados "mantenimiento",

que hará que el producto final se acerque al producto final deseado.

Figura 2

"Metodología Scrum"

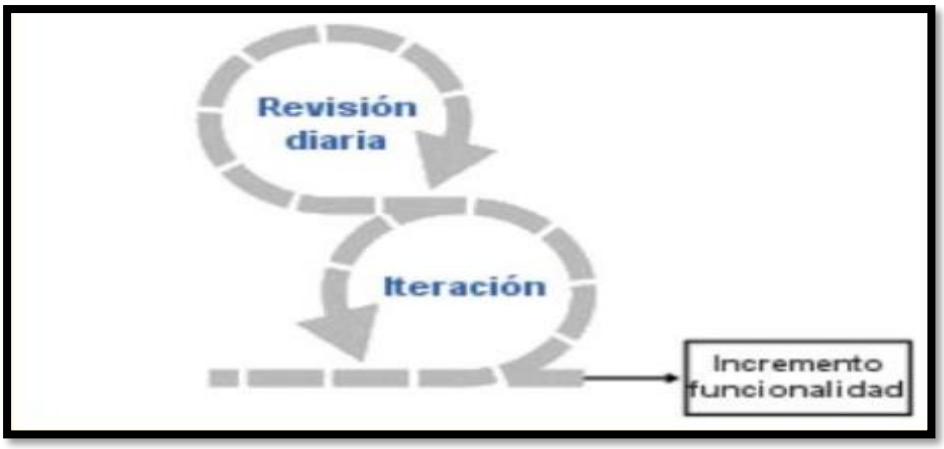


Nota: Representa la metodología ágil que se utilizó para realizar este desarrollo.

El equipo de desarrollo de Scrum consiste en la formación de profesionales en el área los cuales desempeñan su trabajo con el fin de proporcionar un producto terminado (Sprint), este equipo es formado de manera integral, contando con diversidad de competencias y estás cumplen con característica de ser auto dirigido, sin reconocimiento de títulos con libertad de decisión sobre las entregas, es así como Scrum gestiona estas iteraciones a través de reuniones diarias, uno de los elementos fundamentales de esta metodología es.

Figura 3

“Desarrollo de scrum”



Nota: Representa las reuniones periódicas que se realiza, en cada fase del software.

2.12 ¿Qué es un Repositorio?

Los repositorios en sí constituyen un medio de información que tiene por finalidad, organizar, preservar y difundir de una manera fácil controlando y estandarizando los recursos informáticos donde se puede lograr un mayor conocimiento, porque además de almacenar documentos y datos en el tiempo estos garantizarán un acceso adecuado a futuras generaciones.

Es así como (EcuRed, s.f.) indica que los repositorios son sistemas de información que preservan y organizan materiales científicos y académicos como apoyo a la investigación y el aprendizaje, donde se garantizan el acceso abierto a sus documentos y datos a través de internet, así como el respeto a los derechos de la institución o del autor sobre su producción científico tecnológico; mientras que para (Melero R, 2005) en su investigación destaca que los repositorios son archivos donde se almacenan recursos digitales que contienen información cientfífica que son importantes para el ser humano y su conocimento en la investigación estos documentos serán subidos de manera gratuita y libre a través de la Web.

Es así como estos dos autores tienen una relación en indicar que este

servicio es útil dentro de la investigación dando enfásis a la difusión de los contenidos en función al área del conocimiento obviamnete cada uno manteniedo sus características que contribuyen a los estándares de almacenamiento y preservación.

2.12.1 Tipos de Repositorios

Un repositorio de Software es aquella que dispone de una variedad del servicio que ofrecen dependiendo del tipo de licencia que sea usada por ello se identifica a dos tipos que son:

Licencia privada. - el administrador limita o restringe las propiedades del software. Ejemplo: Windows Update.

Licencia de Uso Libre. - es aquella que ofrece una plataforma de trabajo colaborativo y compartida de conocimientos libre sobre cualquier temática, sin ningún tipo de restricciones. Ejemplo: repositorio de Software libre, paquetes para el sistema operativo GNU/Linux, desde plataformas como SourceForge o Forja de Guadalinex.

2.12.2 Repositorios Institucionales

Son organismos políticos, sociales y educativos desarrollados para las universidades e institutos o asociaciones, para depositar, usar y preservar la producción científica y académica que generan en formato digital y haciéndola accesible al público. De esta manera la institución ofrece un servicio acorde al movimiento de acceso abierto.

2.12.3 Repositorio Temáticos

Creados por un grupo de investigadores, una institución la cual reúne documentos relacionados con una área o temática en particular, suele ser social de

educación ciudadana o académica.

2.12.4 Repositorio de Datos

Estos sirven para almacenar, conservar y comparten de manera eficaz todos los datos de las investigaciones.

2.13 Acerca del Control de Versiones

Los sistemas de control de versiones son herramientas del Software que ayudan a los equipos a gestionar los cambios de código en la fuente a lo largo del tiempo, también conocido como “control de código de fuente” que ayudan al Software a trabajar de una manera más rápida e inteligente, la importancia de esta versión es muy significativa dentro del programador por que facilita que si existe un error inmediatamente puede retroceder y hacer cambios sin que exista interrupciones en el equipo.

Desde la lógica citada de (Straub, 2014) destaca que un control de versiones es un sistema que registra cambios en un archivo a lo largo del tiempo, y para que exista una recuperación de estas versiones se debe ayudar con el código de fuente que sirve como protección, esta aplicación da apertura a la organización de carpetas en un árbol de archivos permitiendo cambios sin la existencia de ningún tipo de afectación al computador.

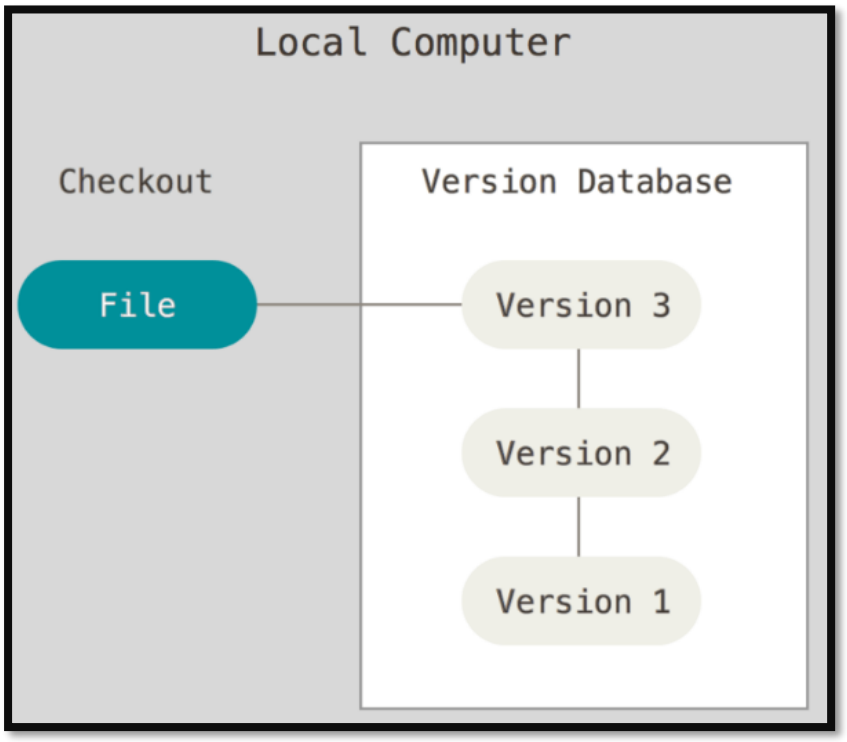
Visto de esta forma y si usted es un diseñador gráfico en la aplicación Web y desea conservar todas las versiones de una imagen o diseño debe aplicar el control de versiones (VCS) la cual es una opción muy inteligente que le permitirá revertir los archivos seleccionados a un estado anterior, es aquel que también puede hacer comparaciones a través de cambios a lo largo del tiempo, y todo esto lo obtiene con muy pocos gastos generales sino más bien la intelectualidad en el proceso del desarrollo del Software.

2.13.1 Sistemas de Control de Versiones Locales

Este método de control de versiones es muy utilizado por muchas personas ya que da la facilidad de copiar los archivos a un directorio con marca de tiempo, adicional que este control es muy simple y muy propenso a que tenga errores, pues es fácil de olvidarse en que directorio se encuentra y al momento de buscar y guardar los archivos por accidente podemos ubicar en carpetas no acordes al tema asignado, para hacerle frente a este problema, los programadores desarrollaron hace mucho tiempo VCS locales que tenían una base de datos simples que mantenía todos los cambios en los archivos bajo control de revisión.

Figura 4

"Sistemas de Control de Versiones Locales"



Nota: El gráfico representa el número de versiones actualizadas en el software.

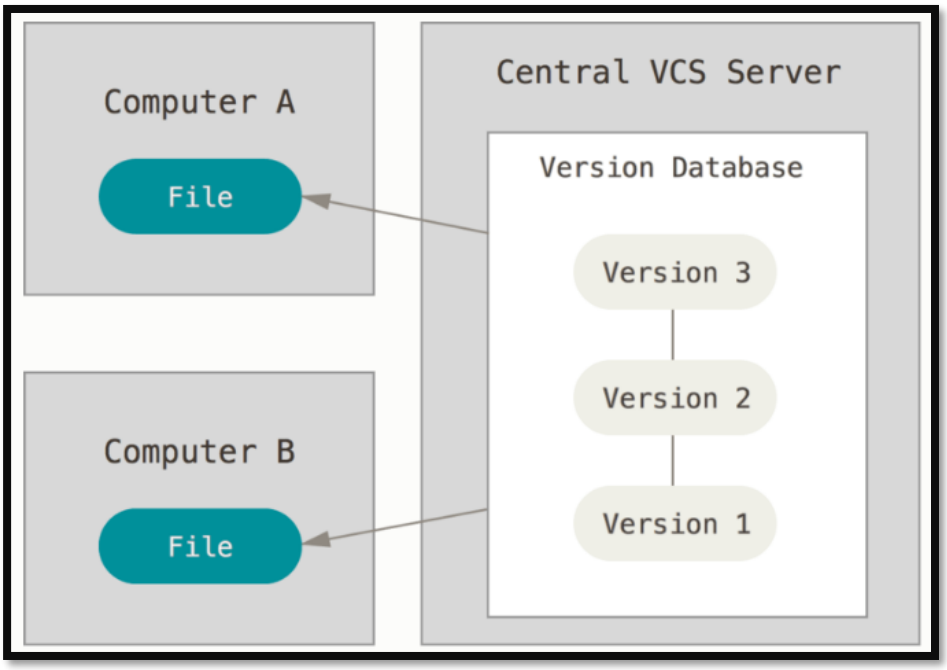
Unas de las herramientas de VCS más populares fue el sistema llamado RCS, que hoy en la actualidad todavía se distribuye en muchas computadoras que dan la función de mantenimiento en un formato especial en el disco, que luego se

podrá volver a crear el archivo en cualquier momento sumando todos los parches de un sistema de Control de Versiones Centralizados

Los problemas que enfrentan las personas que necesitan colaborar con los desarrolladores en otros sistemas es que se encontrará con un solo servidor que contiene todos los archivos versionados y una cantidad de clientes que extraen archivos desde ese lugar central. Durante muchos años, este ha sido el estándar para el control de versiones que ha imposibilitado trabajar de manera eficaz el cual pueda calcular el trabajo en el futuro con mucha precisión.

Figura 5

“Sistemas de Control de Versiones Centralizados”



Nota: Este grafico representa la copia de un solo proyecto.

Esta configuración ofrece muchas ventajas, especialmente sobre los VCS locales, por ejemplo, todo el mundo sabe hasta cierto punto lo que están haciendo los demás en el proyecto, mientras que los administradores tienen un control detallado sobre quien puede hacer que, y es mucho más creíble administrar un CVCS desde este punto de vista donde se va a tratarlos datos en base a cada

cliente.

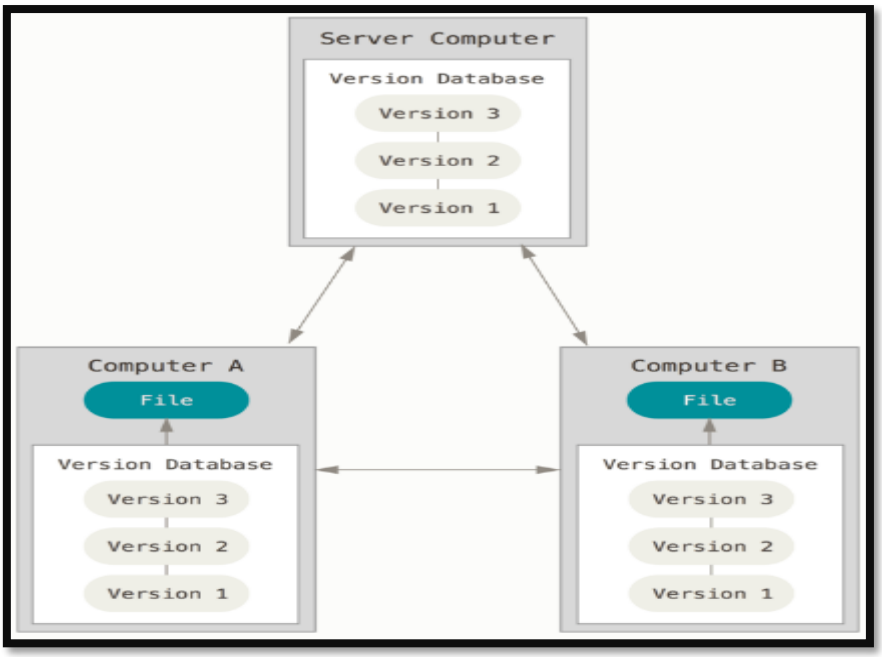
Sin embargo, esta configuración también tiene algunos inconvenientes graves, el más obvio es el punto de falla del servidor centralizado que deja de funcionar durante una hora, en esa hora nadie puede colaborar ni mucho menos guardar cambios versionados de lo que se estuvo trabajando, si el caso es que el disco duro se dañe durante la realización de la actividad y no existió una copia de seguridad adecuada, el usuario perderá absolutamente todo es así como los sistemas VCS locales sufren el mismo problema: siempre que tenga el historial completo del proyecto en un solo lugar, corre el riesgo de perderlo todo.

2.13.2 Sistemas de Control de Versiones Distribuidos

En este sistema intervienen versiones como Git, Mercurial, Bazzar o Darcs, donde los usuarios no solo revisan la última aplicación instantánea de los archivos; más bien, en este versionamiento se reflejan todos los archivos en el repositorio, incluido su historial completo. Por lo tanto, si algún servidor muere y estos sistemas colaboran a través de ese servidor, cualquiera de los repositorios de los clientes se puede copiar en el servidor para restaurarlo, este clon es realmente una copia de seguridad completa de todos los datos.

Figura 6

“Sistemas de Control de Versiones Distribuidos”



Nota: El grafico representa un clon el repositorio del proyecto al disco duro del dispositivo.

Este sistema permite configurar varios tipos de trabajo que muchas de las veces no son posibles en los sistemas centralizados como modelos jerárquicos, muchos de estos son bastantes adaptables a varios repositorios, por lo que pueden colaborar y trabajar con diferentes grupos de personas de diferentes maneras en este caso de manera simultánea dentro del mismo proyecto facilitando el uso adecuado del programa.

2.14 Una Breve Historia de Git

Empezamos indicando que un Git es una herramienta muy útil para el desarrollador dentro de un sistema de control, el mismo es utilizado para guardar diferentes versiones en un archivo y si este desea recuperarlo lo haga sin ninguna novedad; las cosas maravillosa de la vida van suscitando con el pasar del tiempo y con ella viene la adquisición de la tecnología que hoy en día está inmersa en nuestro

diario vivir es por ello que analizaremos un poco de la historia del surgimiento del Git como una manera creativa y al mismo tiempo la existencia de controversias.

En los años de 1991 se empieza a evidenciar los cambios en el Software a través de Kernel de Linux que facilita un código abierto de alcance bastante amplio y tratando de que exista una vida útil en el mantenimiento, para el 2002 el proyecto de kernel de Linux comienza a darle vida a un programador que es el DVCS donde su propietario lleva el nombre de BitKeeper.

Para el año 2005 las empresas de Kernel de Linux y BitKeeper se vino de baja y de esta manera la herramienta deja de ser gratuita, impulsando en este caso al desarrollador de Linux crear su propia herramienta basada en algunas lecciones que fue adquiriendo mientras usaba BitKeeper es así como identificamos algunos objetivos que fueron los siguientes:

- Velocidad
- Diseño simple
- Fuente apoyo para el desarrollo no lineal (miles de ramas paralelas)
- Totalmente destruido
- Capaz de manejar proyectos grandes como el kernel de Linux de manera eficiente (velocidad y tamaño de datos)

Es impresionante como dejamos ir oportunidades y estas oportunidades Linux las aprovecho de la mejor manera desde su nacimiento como propietario en el año 2005, Git ha evolucionado notablemente para la accesibilidad del ser humano de una manera sorprendente y eficiente con proyectos grandes fácil de usar y con un increíble sistema de ramificación para el desarrollo no lineal.

2.15 ¿Qué es Git?

Para (Castellanos, 2021) en su análisis sobre el Git menciona que es un

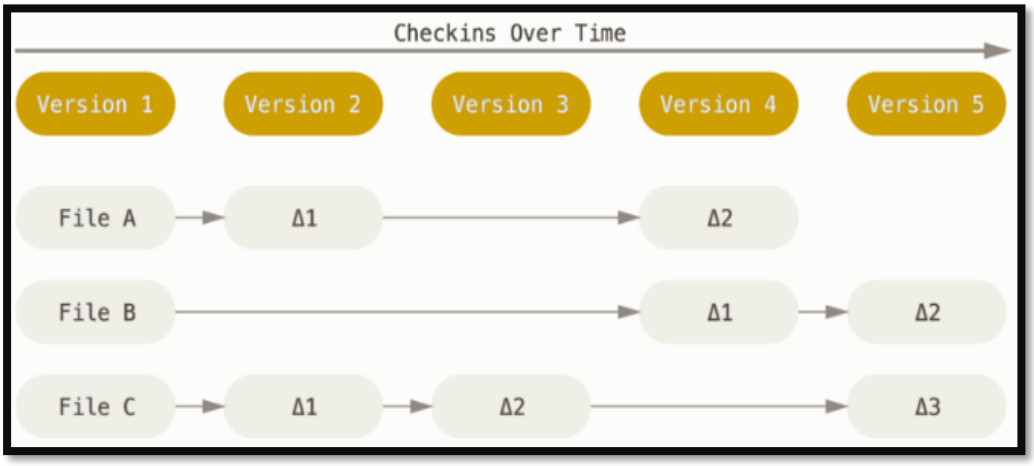
Sistema de Control Distribuido (DVSC) y que este es utilizado para archivar diferentes carpetas de una manera efectiva esto posibilita la revisión de estos documentos en cualquier momento sin la preocupación que se hayan perdido durante la realización de la misma. A medida que se aprende a manejar el Git, puede saber sobre otros VCS, como CVS, Subversion o Perforce; esto le ayudará a evitar confusiones sutiles al usar la herramienta; aunque la interfaz de usuario de Git es bastante similar a estos otros VCS, Git almacena y piensa en la información de una manera muy diferente, esto le ayudará a no confundirse mientras la usa.

2.15.1 Copias Instantáneas, no Diferencias

Las principales diferencias entre Git y cualquier otro VCS (Incluidos Subversión y amigos) es la forma como Git piensa en sus datos, conceptualmente, la mayoría de los otros sistemas almacenan información como una lista de cambios basados en archivos. Estos otros sistemas (CVS, Subversión, Perforce, Bazaar, etc.) manejan la información que almacenan como un conjunto de archivos y modificaciones hechas a cada uno de ellas a través del tiempo.

Figura 7

“Copias Instantáneas, no Diferencias”

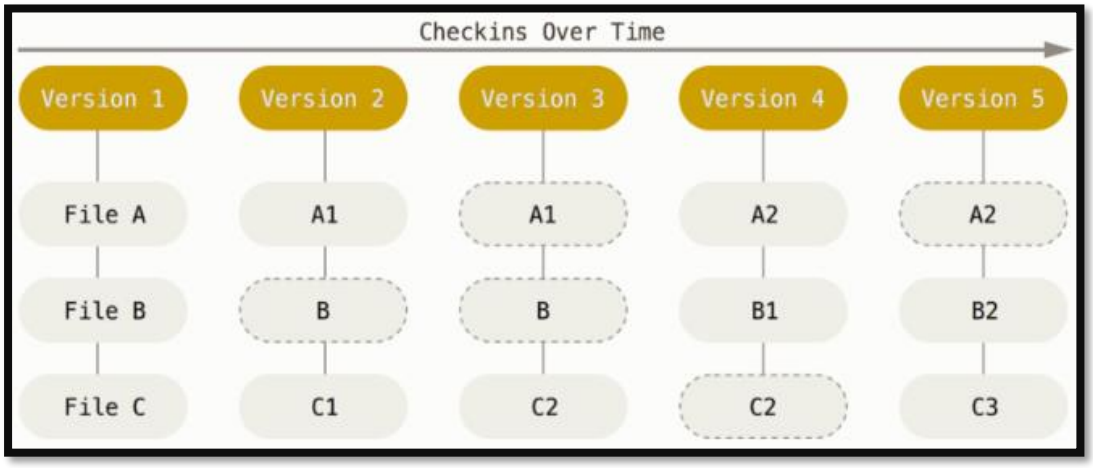


Nota: El grafico representa un conjunto de archivos y modificaciones hechas a cada una de ellas a través del tiempo.

Git no maneja ni almacena sus datos de esta forma Git maneja sus datos como un conjunto de copias instantáneas de un sistema de archivos en miniaturas. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente toma una foto del aspecto de todos tus archivos en ese instante, busca ser eficiente, si los archivos no se han modificado Git no almacena el archivo de nuevo, sino un enlace al archivo anterior idéntico ya tiene almacenado maneja sus datos como una secuencia de copias instantáneas.

Figura 8

“Copias Instantáneas”



Nota: Representa una foto del aspecto de todos tus archivos en el instante.

Esta es una diferencia importante entre Git y prácticamente todos los demás VCS. Hace que Git reconsidere casi todos los aspectos del control de versiones que muchos de los demás sistemas copiaron de la generación anterior. Esto hace que Git se parezca más a un sistema de archivos miniaturas con algunas herramientas extremadamente poderosas desarrolladas sobre él, que a un VCS de esta manera exploraremos algunos de los beneficios que obtiene al modular tus datos cuando veamos ramificación (branching) en Git.

2.15.2 Casi Todas las Operaciones son Locales

Para (Straub, 2014) en su investigación realizada indica que la mayoría de las operaciones en Git sólo necesitan archivos y recursos locales para funcionar; por lo general no se necesita información de ningún otro ordenador de su red. Sin embargo, la mayoría de las operaciones tienen un costo adicional y en este aspecto Git trae una velocidad impresionante con poderes sobrenaturales, en la cual se puede identificar toda una historia guardada en el disco local y la mayoría de las operaciones aparecen inmediatamente sin tanto esfuerzo en el programador como en el usuario.

Es así que podemos dar un ejemplo muy evidente dentro de la navegación donde Git no necesita conectarse al servidor para obtener la historia y mostrarla simplemente nos vamos directamente a la base de datos local., esto significa que este programa da a conocer la información inmediatamente, y si es necesario realizar cambios que se guardaron hace un mes Git con su capacidad puede realizar un cálculo entre la versión antigua con la actual que se manejaran desde la red y realizarla de una manera local sin que exista afectaciones.

2.15.3 Git tiene Integridad

En la aplicación Git todo es verificado mediante una suma de comprobación (checksum en inglés) esto significa que es imposible cambiar los contenidos de cualquier archivo o directorio sin que Git lo sepa; esta funcionalidad está integrada en Git al más bajo nivel y es parte integral de su filosofía, tampoco puedes perder información durante su transmisión sin que Git sea capaz de detectarlo para su aplicación se contempla de una cadena de 40 caracteres hexadecimales (0-9 y a-f), y se calcula con base en los contenidos del archivo o estructura del directorio en Git.

2.15.4 Git Generalmente solo Añade Información

Utilizar la aplicación Git debe ser placentero, porque de esta manera se va a experimentar el almacenamiento de información sin que este se estropee más bien teniendo el privilegio que se puede recuperar datos que aparentemente se encuentran perdidos, esto quiere decir que esta aplicación es muy importante dentro de la tecnología y la actualidad para los seres humanos.

Es así como nos enfocamos que Git añade información siempre y cuando este sea guardado correctamente, es por ello que Git viene de una manera avanzada a comparación de las otras aplicaciones que antes de culminar con sus trabajos existe un error y no se puede conseguir la confiabilidad en el programador para lo cual Git da la apertura y confianza en su programa al enviar los datos a otro repositorio con regularidad.

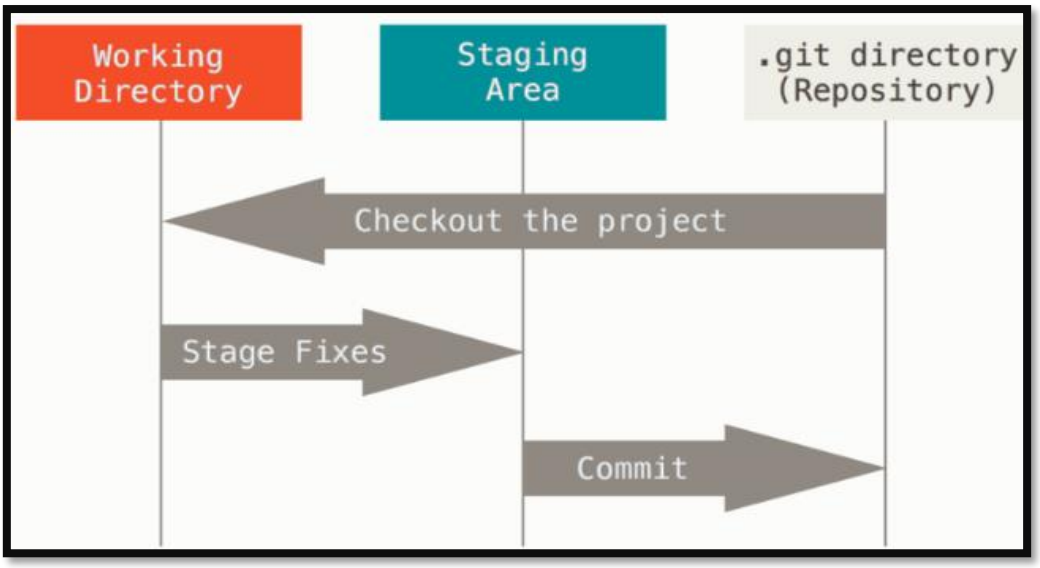
2.15.5 Los Tres Estados

Lo más importante de Git que debemos tener en cuenta para que siga el proceso de aprendizaje sin que exista problemas es la aplicación de 3 estados principales en los que se puede encontrar todos tus archivos almacenados como son: (committed), Modificado (modified) y Preparado (staged). Es así como el estado de confirmado: significa que los datos están almacenados de manera segura en tu disco local, en cambio en el estado modificado: significa que hecho cambios en el archivo, pero todavía no lo has confirmado a tu base de datos y finalmente en los datos de preparado: significa que ha marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Esto nos lleva a las tres secciones principales de un proyecto de Git: el directorio de Git (Git directory), el directorio de trabajo (working directory) y área de preparación (starging area).

Figura 9

“Los Tres Estados”



Nota: Representa el estado del software en el proceso de actualización.

1. **El directorio de Git** es la parte más esencial del Git, donde se almacenan los metadatos y la base de datos de objetos para tu proyecto y al momento de realizar una copia esta va como un clon al repositorio desde otra computadora.
2. **El directorio de trabajo** es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que los puedas usar o modificar.
3. **El área de preparación es un archivo**, generalmente contenido en tu directorio de Git, que almacena información acerca de lo que va a ir en tu próxima confirmación. A veces se le denomina índice (“índex”), pero se está convirtiendo en estándar al referirse a ella como el área de preparación.

   El flujo de trabajo básico en Git es algo así:

   1 Modificas una serie de archivos en tu directorio de trabajo.

   2 Preparar los archivos, añadiendo a tu área de preparación.

   3 Confirmar los cambios, lo que toma los archivos tal como están en el área

de preparación y almacena esa copia instantánea de manera permanente en tu directorio de Git.

Si una versión concreta de un archivo está en el directorio de Git, se considera confirmada (committed). Si ha sufrido cambios desde que se obtuvo del repositorio, pero ha sido añadida al área de preparación, está preparada (staged). Y si ha sufrido cambios desde que se obtuvo el repositorio, pero no se ha preparado, está modificada (modificad).

2.16 Ubuntu

Es una distribución de Linux basada en GNU la cual suministrará un sistema operativo actualizado y estable buscando la facilidad al usuario de la correcta instalación al sistema de manera correcta y rápida, este proyecto de Ubuntu establece que las personas accedan a un código teniendo ellos la potestad de poder descargarlo, modificarlo y usarlo de la manera que más le convenga, finalmente podemos indicar que Ubuntu ha reducido un sin número de paquetes quedando solo aplicaciones importantes y de alta calidad proporcionando al usuario calidad en la tecnología.

Es así como (Ubuntu, 2020) menciona que es una bifurcación del código base del proyecto, el objetivo de esta aplicación es de facilitar el uso y el correcto entendimiento habiendo más sencillas las tareas que corresponde a la gestión de proyectos para todos los usuarios, de esta manera la misma aplicación ayuda a que los desarrolladores puedan dar credibilidad al programa y este sea estable al momento de usarlo.

Desde su primer lanzamiento que fue el 20 de octubre del 2004 da inicio al proyecto su creador Shuttleworth proporcionando un soporte económico gracias a los beneficios obtenidos después de vender su empresa por la cantidad de 575 millones de dólares estadounidenses.

Es así como el 8 de julio del 2005, Shuttleworth anuncia la creación de la fundación Ubuntu donde aporto con 10 millones de dólares como un presupuesto inicial dando así el enfoque necesario para la implementación en la tecnología y el desarrollo para las futuras versiones, para ello en marzo del 2009 anuncia un nuevo soporte para las plataformas externas dentro de la administración de la computación en la nube como es Amazon EC2

Como un ejemplo evidente de superación y de ayuda idónea tanto a los usuarios como a la tecnología es la compañía británica propiedad del empresario sudafricano Mark Shuttleworth quien es el patrocinador directo de la creación de un sistema gratuito como es el Ubuntu , para que su empresa pueda funcionar de un amanera correcta este se financia por medio se servicios vinculados al sistema operativo y vendiendo soporte técnico a todos los usuarios que los requieran; es así como la comunidad de desarrolladores proporcionan soporte para otras derivaciones de Ubuntu, con otros entornos gráficos, como Kubuntu, Xubuntu, Ubuntu MATE, Edubuntu, Ubuntu studio, Mythbuntu, Ubuntu GNOME y Lubuntu.

2.16.1 Interfaz de Usuario

El interfaz del usuario da su credibilidad con su primer lanzamiento utilizando en el escritorio GNOME, la cual divide 2 paneles que son el inferior y el superior; el inferior sirve para listar ventanas y el superior para menús e indicadores del sistema desde la versión 11.04 el equipo de Canonical decidió lanzar su propia interfaz de usuario, de esa manera Unity fue diseñado para optimizar el espacio e interacción de la interfaz de Ubuntu. Aunque en la versión 18.04 regreso a GNOME

2.16.2 Diseño

La actual interfaz de usuario de Ubuntu está compuesta por tres elementos: la barra superior para indicadores de sistemas y menús donde aparecen iconos

superiores (Archivo, Editar) de cada aplicación, si es así se configura el lanzador de aplicaciones al costado izquierdo, y el tablero o Dash, ubicado en la parte superior del lanzador de aplicaciones, que despliega accesos a aplicaciones y medios.

Además de la interfaz Unity, Canonical ha diseñado varios elementos para ésta: set de iconos Ubuntu Mono y Humanity, temas visuales ligeros (Ambiance y Radiance), tipografía Ubuntu con varios estilos de escritura, barras de desplazamiento superpuestas, notificaciones OSD, pantallas de inicio de sesión Unity Greeter, gestos multitáctil u Touch, temas de sonido de inicio de sesión, y los menús globales de aplicación.

2.16.3 Características

En su última versión, Ubuntu soporta oficialmente dos arquitecturas de hardware en computadoras personales y servidores: 32-bit (x86) y 64-bit (x86_64) sin embargo, extraoficialmente, Ubuntu ha sido portado a más arquitecturas: ARM, PowerPC, SPARC e IA-64.

A partir de la versión 9.04, se empieza a ofrecer soporte extraoficial para procesadores ARM, comúnmente usados en dispositivos móviles. Al igual que la mayoría de los sistemas de escritorio basados en Linux, Ubuntu es capaz de actualizar todas las aplicaciones instaladas en la máquina a través de repositorios y también tiene la capacidad de traducir a más de 130 idiomas, y cada usuario es capaz de colaborar voluntariamente a esta causa, a través del uso del Internet.

2.16.4 Seguridad y Accesibilidad

El sistema incluye funciones avanzadas de seguridad y entre sus políticas se encuentra el no activa, que es una forma predeterminada de los procesos latentes al momento de instalarse. Por eso mismo, no hay unos cortafuegos predeterminados, ya que supuestamente no existen servicios que pueden atentar a la seguridad del

sistema. Para labores o tareas administrativas en la línea de comandos incluye una herramienta llamada sudo (de las siglas en Ingles de SwitchUser do), con la que se evita el uso del usuario administrador la misma que posee accesibilidad e internacionalización, de modo que el sistema esté disponible para tanta gente como sea posible. Desde la versión 5.04, se utiliza UTF-8 como codificación de caracteres predeterminados.

No solo se relacionan con Debian por el uso del mismo formato de paquetes también tiene uniones con esa comunidad, que, aunque raramente contribuyen con cualquier cambio directo o inmediatamente, o solo anunciándolos. Esto sucede en los tiempos de lanzamientos. La mayoría de los empaquetadores de Debian son los que realizan también la mayoría de los paquetes importantes de Ubuntu.

2.16.5 Organización del Software

Ubuntu internamente divide todo el software en cuatro secciones, llamada “componentes”, para mostrar diferencias en licencias y la prioridad en la que se atienden los problemas que informen los usuarios. Estos componentes son: **main, resticted, universe y multiverse.**

De forma predeterminada se instala paquetes de los componentes **main y restricted.** En cambio, **universe** de Ubuntu generalmente se basa en los paquetes de la rama inestable (sid) y en donde el repositorio experimental de Debian

- **Main** contiene solamente los paquetes que cumplen los requisitos de la licencia de Ubuntu y para los que hay soporte disponible por parte de su equipo. Está pensando para que incluya todo lo necesario para la mayoría de los sistemas de Linux de uso general. Los paquetes de este componente poseen ayuda técnica garantizada y mejoras de seguridad oportunas.
- **Restricted** contiene paquete soportados por los desarrolladores de Ubuntu debido a su importancia, pero que no está disponible bajo ningún tipo de

licencia libre para incluir en main. En este lugar se incluyen los paquetes tales como controladores propietarios de alguna tarjeta gráficas, como los de ATI y NVIDIA. El nivel de la ayuda es más limitado que para main, puesto que los desarrolladores pueden no tener acceso al código fuente.

- **Universe** contiene una amplia gama de prórrogas, que pueden o no tener una licencia restringida, pero que no recibe apoyo por parte del equipo de Ubuntu sino por parte de la comunidad. Esto permite que los usuarios instalen toda clase de programas en el sistema guardándolos en un lugar aparte de los paquetes soportados. Main y restricted.
- **Multiverse** contiene los paquetes sin soporte debido a que no cumple los requisitos de software libre.

2.17 ¿Qué es GITLAB?

(Martinez, 2018) indica que Gitlab es una herramienta de servicio web que controla las versiones y el desarrollo de Software colaboritavo basado en Git , una de las opciones más confiables y con menos costo es esta aplicación que la usan en las empresas ya que es necesario que el servidor sea configurado, actualizado y monitoreado todo el tiempo, es una ventaja muy grande ya que posibilita el acceso al repositorio con opciones a traer cambios sin que exista afectaciones en el computador.

Es así como se da a conocer que está aplicación de GitLab nace en el año 2011 como un sistema de alojamiento de repositorio Git, es decir, un hosting para proyectos gestionados por el sistema de versiones; sin embargo, alrededor de esta herramienta han surgido muchas otras más muy interesantes para programadores y equipos de desarrollo, que envuelve todo el flujo del desarrollo y el despliegue de aplicaciones y test.

Sin duda, para dar una idea de lo que es GitLab, lo más rápido seria

compartirlo con uno de sus competidores, que en este caso es GitHub pues este último es especialmente conocido en el mundo del desarrollo de Software como una potencia en la tecnología. GitLab sería algo muy similar a lo que encontramos en GitHub, aunque a veces con otros nombres, otras alternativas de programas o servicios para Git como Bitbucket están muy por detrás en posibilidades.

Aunque GitHub es un monstro, en cuanto a números de repositorios y funcionalidad, GitLab ha conseguido llegar aún más lejos, ofreciendo un conjunto más amplio de servicios que harán las delicias no solo de desarrolladores, sino también de Devops.

2.17.1 Instalar en tu Servidor o usarlo como Servicio Web

La principal diferencia entre GitLab y sus competidores es que GitLab se ofrece como un Software libre que puedes descargar e instalar en cualquier servidor esta posibilidad permite que las empresas los profesionales u organizaciones adquieran sin ningún coste adicional.

La otra alternativa es usar GitLab directamente de GitLab.com, pagando por un servicio esto permite disponer de todo el poder de GitLab y sus herramientas colindantes, sin invertir tiempo en configuración, aprovechando sus ventajas desde el primer minuto. Además, las versiones del servicio “en la nube” tienen muchas herramientas adicionales, funcionalidades que superan con diferencia a la versión que se ofrece para instalar como software libre.

Por último, GitLab también se ofrece sin coste para publicar repositorios de software libre, igual que su competidor GitHub. En este caso, aunque GitLab pueda disponer de algunos servicios extras que justifiquen trabajar con la herramienta, lo cierto es que GitHub sigue siendo el sitio preferido donde ubicar un proyecto, dado que diversos sistemas de dependencia, como NPM, Composer, etc., trabajan directamente contra ellos.

En resumen, si queremos usar GitLab gratis, para alojar en remoto un repositorio Git en general, la mejor alternativa es que instalemos gratuitamente en una de nuestras máquinas. Sin embargo, si queremos ahorrarnos tiempo y no nos importa pagar un poco, la versión en la nube está mucho más completa y nos permite olvidarnos del servicio y concentrarnos en el desarrollo de nuestros programas.

2.17.2 ¿Cómo Usar GitLab?

GitLab es una herramienta basada en Git, que usas de la misma manera que cualquier otra herramienta similar. Generalmente usas Git a través de la línea de comandos, o a través de programas de interfaz gráfica, o del propio editor de código. Toda esa operativa que ya conoces y que hemos explicado en el manual de Git no cambia.

Además del hosting remoto para repositorios GitLab ofrece una interfaz web para controlar el repositorio y muchas otras herramientas. Ofrece la posibilidad de examinar el código en cualquier de sus versiones, realizar acciones relacionadas con el sistema de repositorios como mergear el código de versiones de proyecto o gestionar las “pull request” (que en Gitlab se llaman “merge request”), gestionar problemática de tu software diversa, automatizar procesos como el despliegue o la ejecución de pruebas del software, etc. Toda esta operativa la realiza, o configuras, en GitLab por medio de una web.

Por lo tanto, para usar GitLab simplemente necesitas las mismas herramientas que ya utilizas en tú día a día, el terminal o un programa de interfaz gráfica para gestionar tu repositorio, así como el navegador web para acceder a el ecosistema de la herramienta disponible en el sitio de GitLab. Por supuesto, todas estas herramientas las puedes usar desde cualquier ordenador conectado a internet, independientemente de su sistema operativo.

2.17.3 Funcionalidades de GitLab

En GitLab podemos gestionar principalmente proyectos, grupos y Sinppets. Los proyectos son los protagonistas del sistema, básicamente repositorios de software gestionados por GitLab y todo el ecosistema GitLab. Los snippets por su parte son como pedazos de código que puedes dejar para hacer cualquier cosa como decimos, dentro de los proyectos es donde se aglutinan la mayoría de las funcionalidades que vamos a resumir:

2.17.4 Overview

Es un listado de todo el proyecto, los archivos, los README.md es parecido a lo que vemos cuando accedemos a un proyecto con GitHub donde te dan el resumen del repositorio, archivo, commits luego tiene dos subsecciones: en Activity del proyecto te ofrece toda la actividad, de una manera estadística, en Cycle Analytics además te ofrece algo muy novedoso, no disponible en otras herramientas.

Básicamente informa el tiempo que se tarda en realizar una funcionalidad, desde que tienes la idea hasta que se incorpora al Software, de modo que cualquier persona, incluso sin conocimientos de programación, puede saber el tiempo que ocupo el hacer las tareas. Una información muy valiosa que puede ayudar a futuro a estimas mejor el tiempo de trabajo necesario para nuevas funcionalidades, obviamente, cuantas más issues tengas en el sistema, más datos tendrás para saber el tiempo que necesitas para las próximas tareas.

2.17.5 Repository

Dentro de la sección “Repository” tenemos varias opciones diversas que afectan al repositorio del proyecto como son:

“Files”, donde se puede navegar por los directorios y archivos, cuyo código podemos ver, e incluso editar los ficheros permite una excelente visualización por

ramas y dispone de utilidades diversas para poder hacer cosas relacionadas con el repositorio remoto, ahorrando la necesidad de lanzar comandos tiene un buscador de archivos muy potente.

“Commits” las ramas “Branches” sirven para ver los repositorios que tenemos.

“Tags”, es importante por el mecanismo disponible en Git para definir puntos del estado del código, correspondientes a cada reléase.

“Locked files”, disponible solo en GitLab como servicio, que es algo que no ofrece el propio sistema de control de versiones Git pero que han implementado dentro de GitLab, que permite bloquear un fichero para que solo ciertas personas lo puedan editar.

2.17.6 Issues

Este es otra de las grandes utilidades de GitLab, que permite definir cualquier problema que se detecta en el Software y darle seguimiento, seguro que las conocemos porque es una de z, el tiempo estimado el uso, la fecha límite, el peso de las tareas, etc.

En GitLab han publicado otra interesante innovación que es un tablero de ISSUES (Issue Boards), que permite visualizar las tareas, de una manera similar a los boards de Trello. Como gestores somos capaces de definir los tableros y las etiquetas, por medio de la gestión de las Issues, permite las actualizaciones el estado de las tareas a través de visualizar su evolución por medio de los tableros.

Otra cosa muy interesante es el “Service desk”, que te ofrece un email que lo puedes proporcionar al cliente. Sin que el cliente se registre en GitLab, ni tenga acceso al proyecto, puede enviar mensaje a ese email, adjuntando texto, imágenes y archivos. GitLab, al recibir el correo, da de alta automáticamente una issue con ese contenido.

2.17.7 Merge Request

Son como las Pull Request de GitHub en donde te permiten controlar todas las solicitudes de combinación o unión de código de distintas ramas o forks. Es muy importante que los merges se resuelvan mediante la interfaz gráfica, ya que nos ofrece muchas posibilidades interesantes, como automatización de tesis, la posibilidad de revisión de los cambios por parte de componentes del equipo, implementar diversas políticas de control sobre el código del proyecto, etc.

2.17.8 CI/CD

Es una de las maravillas que dispone GitLab, una herramienta sencilla y muy útil para los procesos de integración continúa y despliegue continúo es así como existen muchas herramientas que se puede integrar para automatizar los procesos y llegar a crea flujos de trabajo completamente automatizados. De modo que se lancen los test y si todo va bien se puedan realizar una serie de tareas definida, que puede llegar a producir el despliegue automático de las aplicaciones.

Solo disponer de esta sección es suficiente motivo para pasarse a GitLab. No llega la complejidad de herramientas específicas como Jenkins, pero resuelve de manera muy potente problemas similares.

2.18 Subversion

Según (Collins-Sussman, 2018) es un sistema de control de versiones libre y de código abierto es decir está aplicación maneja ficheros y directorios a través del tiempo donde existe un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. En este aspecto,

mucha gente piensa en los sistemas de versiones como en una especie de “máquina del tiempo”.

Subversión puede acceder al repositorio a través de redes, lo que permite ser usado por personas que se encuentran en distintos ordenadores a cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada por la pérdida de ese conducto único, si se ha hecho un cambio incorrecto los datos, simplemente deshaga ese cambio.

Algunos sistemas de control de versiones son también sistemas de administración de configuración de Software estos sistemas son diseñados específicamente para la administración de árboles de código fuente, y tiene muchas características que son específicas de desarrollo de Software, tales como el entendimiento nativo de lenguajes de programación, o el suministro de herramientas para la construcción del Software. Sin embargo, Subversión no es uno de estos sistemas. Subversión es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros.

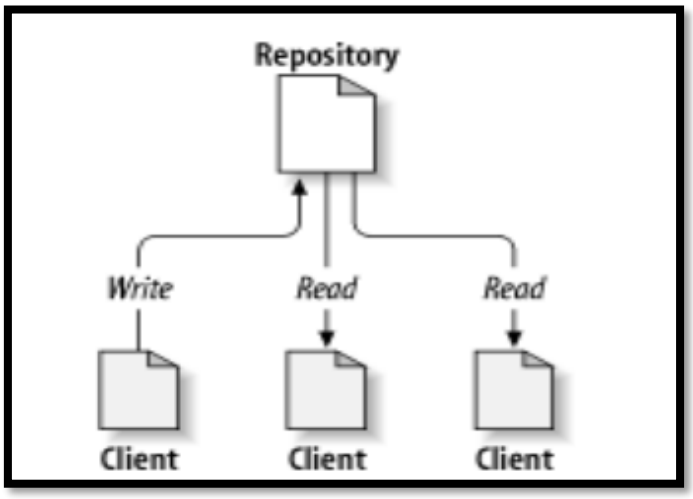
Repositorio Subversión

Según (Collins-Sussman, 2018) Subversión es un sistema centralizado para compartir información donde se pone en énfasis la parte principal que llega a ser el repositorio el cual es un almacén central de datos el mismo guarda información en forma de árbol de archivos donde se evidencia una típica jerarquía que ayuda a mantener un orden lógico en la colocación de tareas.

Cualquier número de clientes pueden conectarse al repositorio y luego leer o escribir en esos archivos al escribir datos, un cliente pone a disposición de otros la información; al leer datos, el cliente recibe información de otros.

Figura 10

“Un sistema cliente/servidor típico”



Nota: Representa un almacén central que guarda información de forma de un árbol de archivo.

Entonces, que tiene esto de interesante. Hasta ahora, suena como la definición del típico servidor de archivos y de hecho, el repositorio es una especie de servidor de archivos, pero no del tipo habitual. Lo que le hace especial a este repositorio de Subversión es que recuerda todos los cambios hechos sobre el: cada cambio a cada archivo, e inclusive cambios al propio árbol de directorios, tales como la adición, borrado y ubicación de archivos y directorios.

Cuando un cliente lee datos del repositorio, normalmente solo ve la última versión del árbol de archivos; sin embargo, el cliente también tiene la posibilidad de ver estados previos del sistema de archivos. Por ejemplo, un cliente puede hacer consultas históricas como, “Que contenía ese directorio el martes pasado” esta es la clase de preguntas que resulta esencial en cualquier sistema de control de

versiones: sistema que están diseñados para registrar y seguir los cambios en los datos a través del tiempo.

Según (Master, 2019) El número de desarrolladores creció debido a que el proyecto atrajo a mucha gente que compartía los mismos deseos como es: mejorar los CVS, después de poco más de un año de programación, los desarrolladores de Subversión empezaron a usar su propio sistema en lugar de CVS, que era lo que usaban hasta ese momento algunas de las características que hacen de Subversión un digno sucesor de CVS son:

- Histórico de Directorios: a diferencia de CVS, la nueva implementación permite mantener un histórico de directorios creados, renombrados y/o borrados.
- Histórico Real de Archivos: CVS guardaba todos los cambios realizados sobre un archivo en un fichero de texto. Esto representaba un serio problema porque a la hora de, por ejemplo, crea un fichero con el nombre de otro que ya existía, se heredaba el histórico del fichero viejo, aunque ya no tuvieran nada en común. Eso ha dejado de ocurrir ahora cualquier operación que incluya copiado, renombrado, borrado o añadido tendrá su propio histórico.
- Consistencia de Datos Almacenados: Subversión maneja de igual manera ficheros de texto y binarios: hace uso de un algoritmo de “diferenciación” que permite comprensión de los daos dentro del repositorio y, además, se apoya en una base de datos.
- Transacciones: igual que en el banco cuando efectuamos una operación, en Subversión las modificaciones del repositorio se realizan completas o no se realizan. O se realizan todos los pasos o no se realiza ninguno. Nunca se quedará un commit a mitad.

- Modificable: Subversión ha sido diseñado de forma que cualquier modificación no conlleve demasiado esfuerzo: consiste básicamente en una serie de APIS bien definidas programadas en varias librerías compartidas.

CAPÍTULO III

3. Desarrollo

3.1 ¿Qué es Putty?

(Perez, 2019) Putty es un cliente SSH que nos permite administrar de forma remota un servidor. Seguro que aquellos que hayan necesitado conectarse por SSH a un sistema Linux, ya saben de lo que hablo.

Algunos prefieren usar directamente SSH desde la terminal, pero la verdad es que Putty es un frontend para SSH que nos aporta bastante más características que el propio SSH. Por ello, en Ubunlog queremos explicar cómo lo podemos instalar y usarlo para podernos conectar a otro sistema remoto y desde Ubuntu.

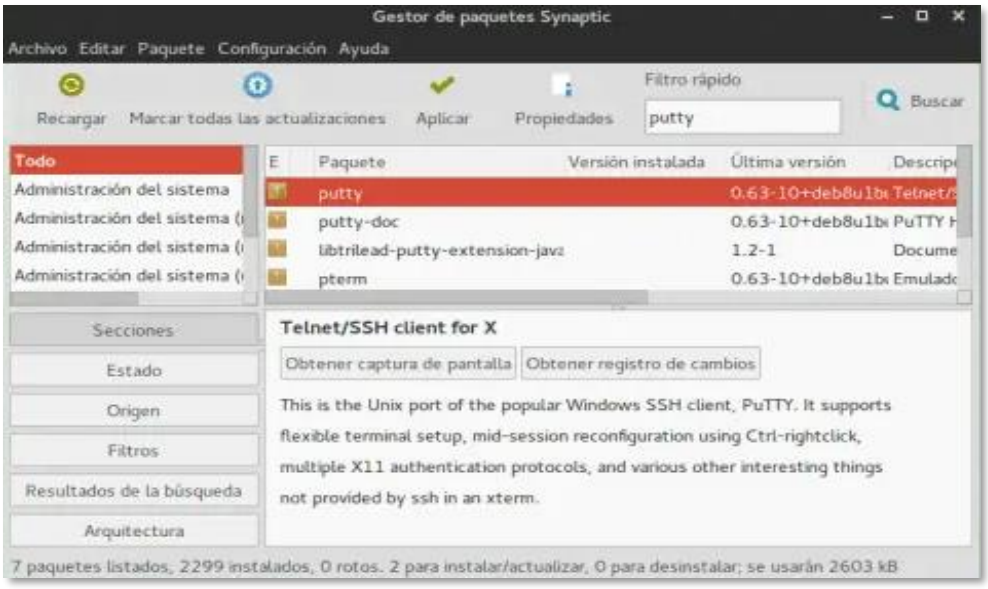
En realidad, Putty es el cliente SSH más popular de Windows, pero también cuenta con una versión para Linux. Putty nos permite configurar la terminal de forma flexible, cuenta con múltiples protocolos de autenticación de X11 y más características no soportadas por SSH.

3.2 Instalando Putty

(Perez, 2019) Para instalarlo podemos hacerlo a través del Gestor de paquetes Synaptic, simplemente buscando el paquete “putty”, marcándolo para instalar y procediendo con la descarga, como vemos en a la siguiente imagen.

Figura 11

“Instalando Putty”



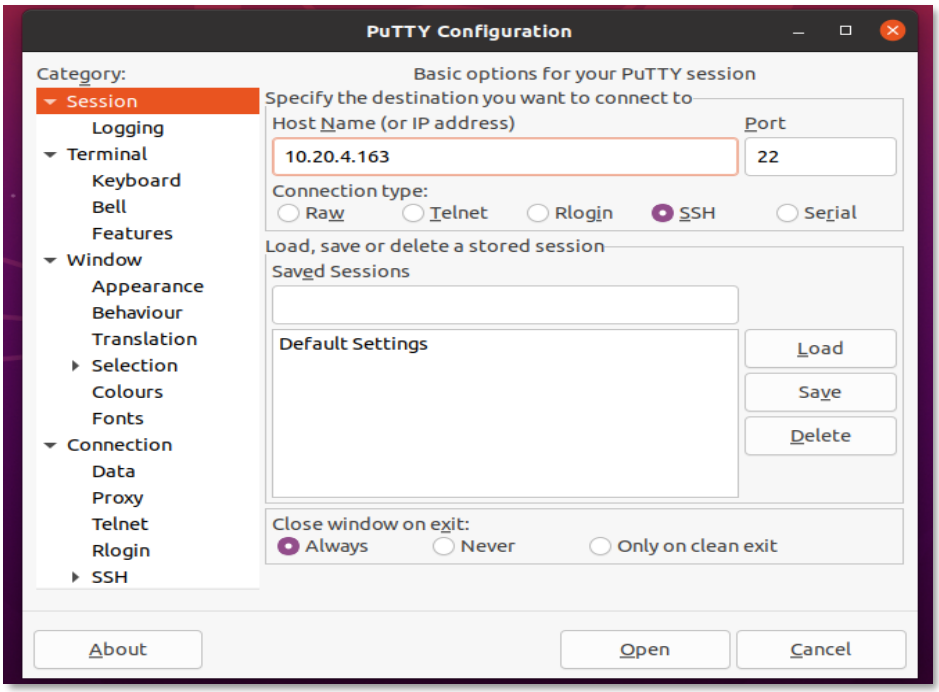
Nota: Representa la instalación través del Gestor de paquetes Synaptic.

3.2.1 Como Usar Putty

(Perez, 2019) Una vez hemos instalado Putty, usarlo es bastante sencillo. Simplemente tenemos que buscar la aplicación Putty y ejecutarla. Para iniciar una sesión SSH, simplemente tenemos que introducir el nombre del host o la IP donde nos queremos conectar de forma remota, y seccionar SSH como tipo de conexión, tal como vemos en la siguiente imagen

Figura 12

“Como Usar Putty”



Nota: Representa para iniciar una sesión hay que introducir el nombre del host o la IP donde nos queremos conectar de forma remota.

Cuando hagamos clic en aceptar, se nos pedirá usuario y contraseña, y ya podrás iniciar tu sesión remota al servidor Linux. Exactamente igual que si tuvieras un monitor y un teclado conectados al servidor y estuvieras gestionarlo a través de ellos.

Además, como vemos en la imagen anterior, como ya decíamos, Putty no solo les sirve para sesiones SSH, sino que nos proporciona un abanico e configuraciones muy amplios. Por ejemplo, en la pestaña Terminal podemos configurar la terminal que nos saldrá al iniciar la sesión SSH, o también podemos configurar el modo en que queremos que Putty nos codifique el texto en la opción Translation de la pestaña de Windows.

Esperemos que a partir Putty te ayude y te simplifique un poco más el trabajo a la hora de conectar de forma remota a un servidor con Linux.

3.3 ¿Qué es Postfix?

(Reyes, 2006) Postfix es un programa al que podemos llamar daemon (demonio) que está en escucha en nuestra maquina en un determinado puerto y que cumple determinadas funciones respondiendo a ciertos datos de entrada que recibe.

Postfix es un servidor de correo, un daemon, que gestiona la entrada y la salida de correo de internet a la intranet o de la intranet a internet o sin salir de la propia intranet. Postfix fue diseñado por Wistse Venema como alternativa a sendmaail. Postfix como rige el estándar del protocolo smtp (Simple Mail Transfer Protocolo o Protocolo Simple de transferencia de correo electrónico).

3.3.1 Razones para Utilizar Postfix

(Reyes, 2006) Las razones para usar Postfix fueron básicamente su sencillez, potencia y versatilidad a respuesta a todas las interrogantes, porque es tan potente como sendmail, fácil de configurar y además, hasta es entretenido.

- Diseño modular (no es un único programa monolítico).
- La seguridad ha sido un condicionante desde el comienzo de su diseño.
- Lo mismo cabe decir del rendimiento (seguramente Sendmail no se diseñó pensando que algún día habría sitios necesitaran procesar cientos de miles o millones de mensajes al día).
- Soporte para las tecnologías más usadas hoy en día. (MySQL), autentificación mediante SASL, entre otras.
- Estricto cumplimiento de los estándares de correo-e.
- Factibilidad de configuraciones.
- Abundante documentación y de calidad.
- Fácil integración con antivirus.

- Uso sencillo de listas negras.
- Tiene múltiples formas de obtener información de lo que está pasando para resolver problemas o simplemente, para aprender.
- Se puede lanzar varias instancias de Postfix en la misa maquina con distintas configuraciones, usando cada una distintas direcciones IP, distintos puertos, etc.
- Filtrado de cabeceras y cuerpos de mensajes por expresiones regulares.
- Utilidades para varias cosas, como gestionar las colas de mensajes.

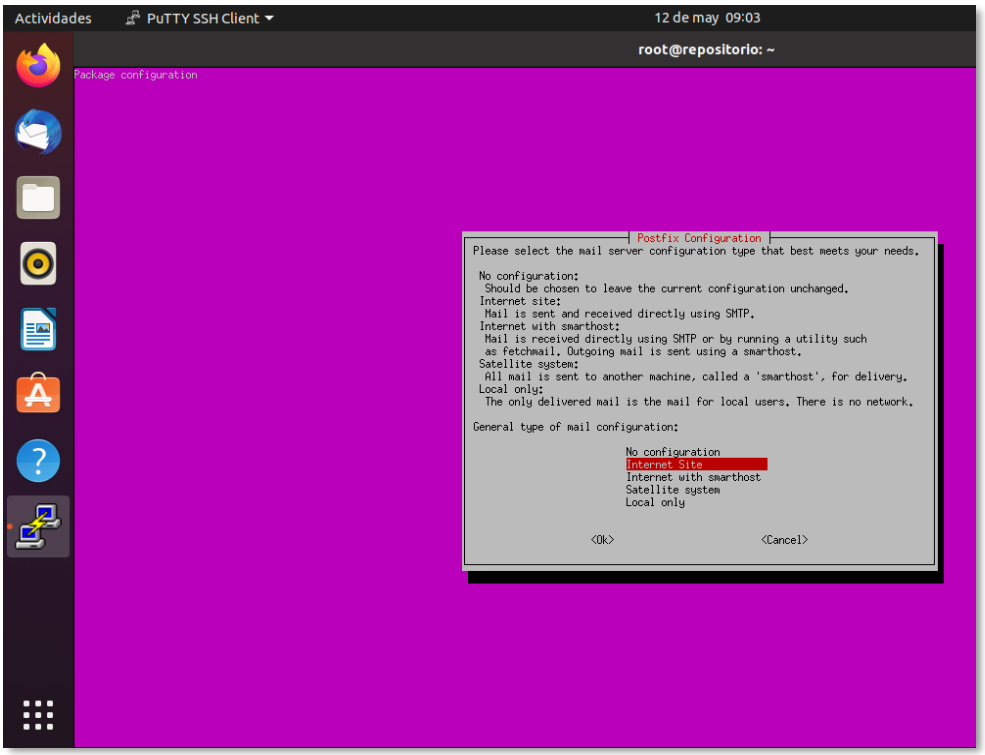
Por último, pero no menos importante, hay que decir que el código fuente de Postfix (por su puesto de dominio público) es un ejemplo de diseño, claridad y documentación, lo cual facilita su mantenimiento, así como la incorporación de nuevas capacidades, correcciones de errores, etc.

3.3.2 Características de Seguridad de Postfix

- Arquitectura modular: cada proceso se ejecutaron privilegios mínimos para su tarea.
- Proceso que no se necesita se deshabilita: No se puede explotar.
- Los procesos se aíslan unos de otros. Muy poca comunicación entre procesos.
- Evita utilizar buffers de tamaño fijo, evitando que tengas éxito ataques buffers overflow.
- Puede ejecutarse en modo chroot.
- Preparado para ataques DoS (Deny of Service, Denegación de Servicio). Cantidad de memoria controlada

Figura 13

“Instalación del Postfix”



Nota: Representa la instalación con el commando “Sudo apt-get install -y postfix”

3.4 Desarrollo de GitLab

Según (Straub, 2014) GitWeb es muy simple. Si buscas un servidor Git más modernos, con todas las funciones, tiene algunas soluciones de código abierto que puedes utilizar en su lugar. Puesto que GitLab es una de las más populares, vamos a ver como se instala y se usa, a modo de ejemplos. Es algo más complejo que GitWeb y requiere algo más de mantenimiento, pero es una opción con muchas más funciones

No se especifica ningún origen. GitLab es una solución web de software libre de forja, control de versiones e integración continua que ayuda al desarrollo de software colaborativo basado en Git. Se trata de una solución con el modelo de negocio “open core” teniendo una versión libre y gratuita con la funcionalidad

principal y otra versión de pago con soporte y algunas funcionalidades más avanzadas.

(IONOS, 2020) GitLab es un popular sistema de control e versiones (CVS) que se usan, sobre todo, en el ámbito del desarrollo de software. En el año 2011, Dimitri Saparoschez escribió y publico este Software, basado en la web, en el lenguaje Ruby on Rails. Hoy en día, se ha convertido en imprescindible para los desarrolladores.

La ventaja principal de GitLab es que facilita noblemente el desarrollo de software fácil entre varios equipos. De esta manera, varios desarrolladores pueden trabajar simultáneamente en un proyecto y edita, por ejemplo, diferentes funciones de forma paralela. La protocolizan continua de todos los procesos garantiza que no se pierda ninguna modificación del código ni que se sobrescriba de forma no intencionada. También es posible deshacer rápidamente los cambios ya aplicados.

GitLab se basa en el Software de gestión de versiones de uso común Git. Git está disponible de forma gratuita como Software de código abierto y es considerado uno de los sistemas de control de versiones más usados en general. GitLab es una de las alternativas a GitHUb más populares cuando Microsoft se hizo con GitHUb en 2018, muchos usuarios se cambiaron a GitLab.

3.5 ¿Cómo Funciona GitLab?

(IONOS, 2020) GitLab es una aplicación basada en la web con una interfaz gráfica de usuario que también puede instalarse en un servidor propio. El núcleo de GitLab lo forman los proyectos en los que se guarda el código que va a editarse en archivos digitales, los denominados repositorios. En estos directorios de proyecto se encuentran todos los contenidos y archivos de un proyecto de software, es decir, archivos JavaScript, HTML, CSS o PHP, entre otros.

En este tutorial de GitLab te explicamos cómo funciona. Para comenzar, todos los implicados en el proyecto se descargan una copia del repositorio central en su ordenador. A partir de ahí, los cambios del código se realizan mediante dominados commit. Una vez realizada la edición, los cambios se integran en el repositorio principal.

Otra función importante es la ramificación. Esta permite a los usuarios crear una “rama” que se bifurca de la parte principal del código y que se puede editar de forma independiente a este. Esta función es especialmente útil a la hora de introducir y probar nuevas funciones sin que esto afecte al desarrollo de la línea principal.

Gracias a la entrega e integración continúas integradas, GitLab es perfecto para trabajar por ramificaciones y ofrece funciones muy útiles como solicitudes de combinación y la creación de bifurcaciones. Por ello, el software es una de las herramientas de integración continua más populares.

Entre las funciones más importantes de GitLab se encuentran las siguientes:

- Interfaz fácil de usar
- Las ramificaciones pueden permanecer privadas o publicarse
- Posibilidad de gestionar varios repositorios
- Revisión de códigos
- Localización integrada de errores y problemas
- Integración continua y entrega continua (CI/CD, por sus siglas en inglés) integradas de forma gratuita.
- Wikis de proyectos.
- Creación sencilla de fragmentos de código para dividir partes del código

3.5.1 Modelos de Licencia y uso de GitLab

(IONOS, 2020) GitLab está basada en un código fuente abierto y de acceso libre. En el año 2013 se creó una versión Enterprise propia para empresas, por lo que ahora hay dos modelos de uso:

- GitLab CE: Community-Edition (gratuita)
- GitLab EE: Enterprise-Edition (de pago)

Ambas versiones están basadas en la licencia de código abierto MIT. La Enterprise-Edition cuenta con algunas funciones adicionales en comparación con la Community. En este contexto, GitLab ofrece tres modelos de suscripción en función del alcance en las funciones adicionales deseadas.

La Enterprise-Edition también se puede usar de manera gratuita, pero solo con las funcione básicas de la Community-Edition. Esta acción está pensada para casos en los que más adelante convengan pasarse a la variante Enterprise, ya que basta un clic para realizar el cambio. El paso desde la versión Community es bastante complicado y quiere más tiempo.

3.5.2 GitLab en servidor propio o en la Nube

(IONOS, 2020) La instalación de GitLab en un servidor propio no supone ninguna complicación para cualquier usuario que conozca Linux, pero lleva bastante tiempo. Al margen de la instalación en sí, hay que dedicar algo de tiempo a la configuración y al trabajo de mantenimiento regular.

Si quieres evitar todo esto, también puedes usar GitLab como Software as a service (SaaS) e instalarlo y ejecutarlo en un servidor en la nube (hay varios proveedores). De esta manera podrás usar el software rápidamente y ya configurado, sin necesidad de instalarlo por norma general, también viene incluida la instalación del GitLab Runner, así que podrás comenzar inmediatamente.

La ventaja de la instalación manual en un entorno de servidor propio es que eso aporta una mayor flexibilidad. Los usuarios gozan de total libertad en la instalación: pueden decidir libremente acerca de copias de seguridad, actualizaciones o recursos adicionales e instalar solo lo que necesiten en su caso concreto. Aun así, la solución en la nube es muy interesante, sobre todo, si el administrador de sistema ya que tiene mucho trabajo.

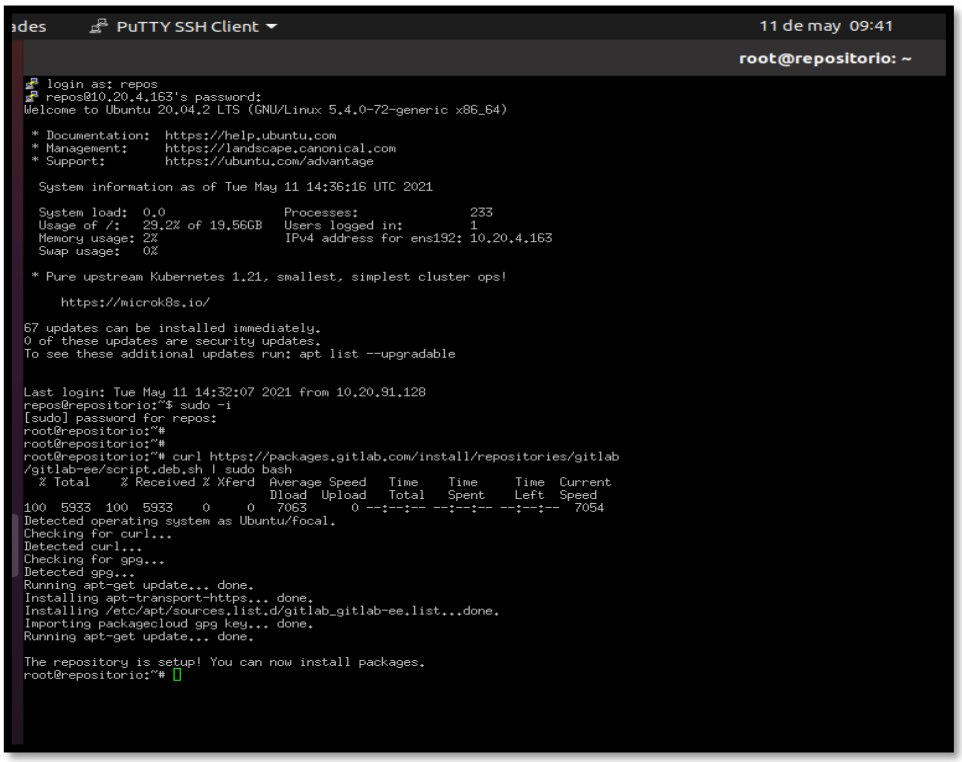
3.6 Instalación de GitLab

(IONOS, 2020) Por norma general, recomendamos un entorno Linux para usar GitLab, ya que al igual que Git, GitLab está concebido de fábrica para Linux. El uso en Windows implica una serie de restricciones, en estos casos, puede usarse una máquina virtual que simula un entorno Linux en un ordenador con Windows. Una opción menos complicada es la instalación del denominado GitLab Runner que también se requiere para usar la integración continua de GitLab.

Se instala el paquete Omnibus de GitLab. Para ello, tiene que añadir primero el “repositorio GitLab Package” con el siguiente comando.

Figura 14

"Instalación de GitLab"



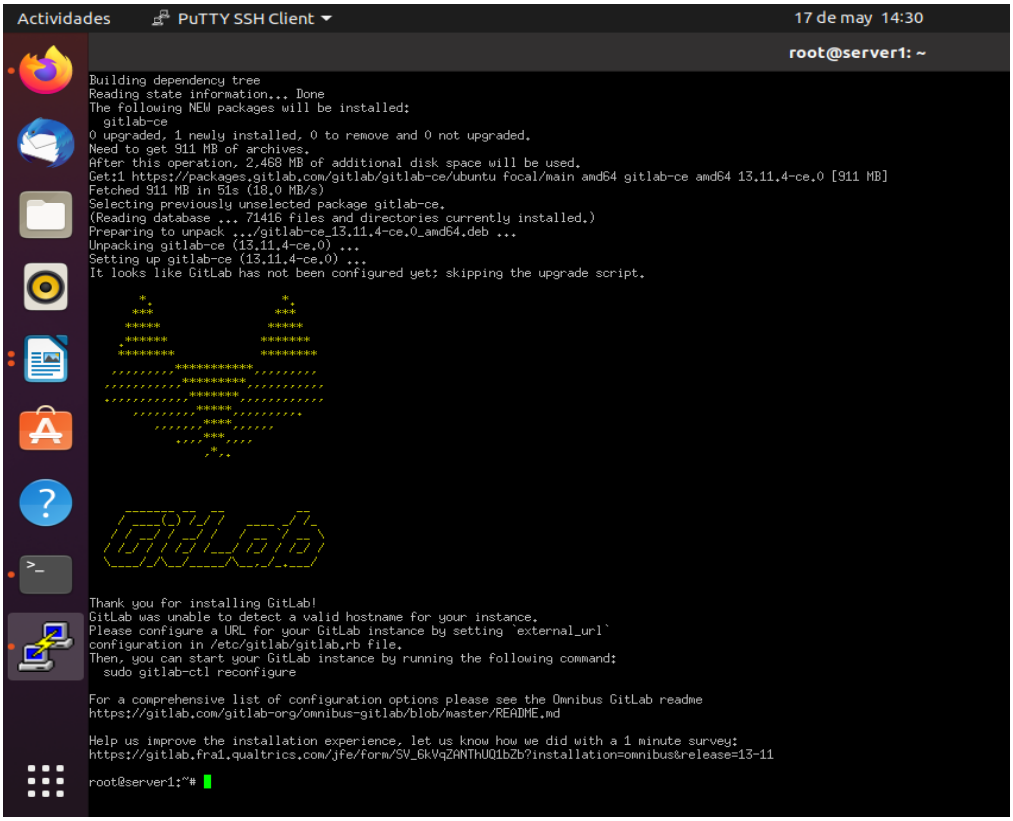
Nota: La imagen representa GitLab en servidor propio o en la Nube curl

- A continuación, se instala el paquete GitLab; para ello, debe asegurarse de haber configurado correctamente su DNS y cambie https://gitlab.example.com que es la URL en la que desea acceder a su instancia de GitLab. La instalación se configurará e iniciara GitLab automáticamente es esa URL.

- Para http://url, GitLab solicitara automáticamente un certificado con Let's Encrypt, que requiere acceso http entrante y un nombre de host valido. También puede usar su propio certificado o simplemente usar https:

Figura 15

“Instalación del paquete de GitLab”



Nota: La imagen representa sudo EXTERNAL_URL=”http://10.20.4.163:8090 “apt-get install gitlab-ee.

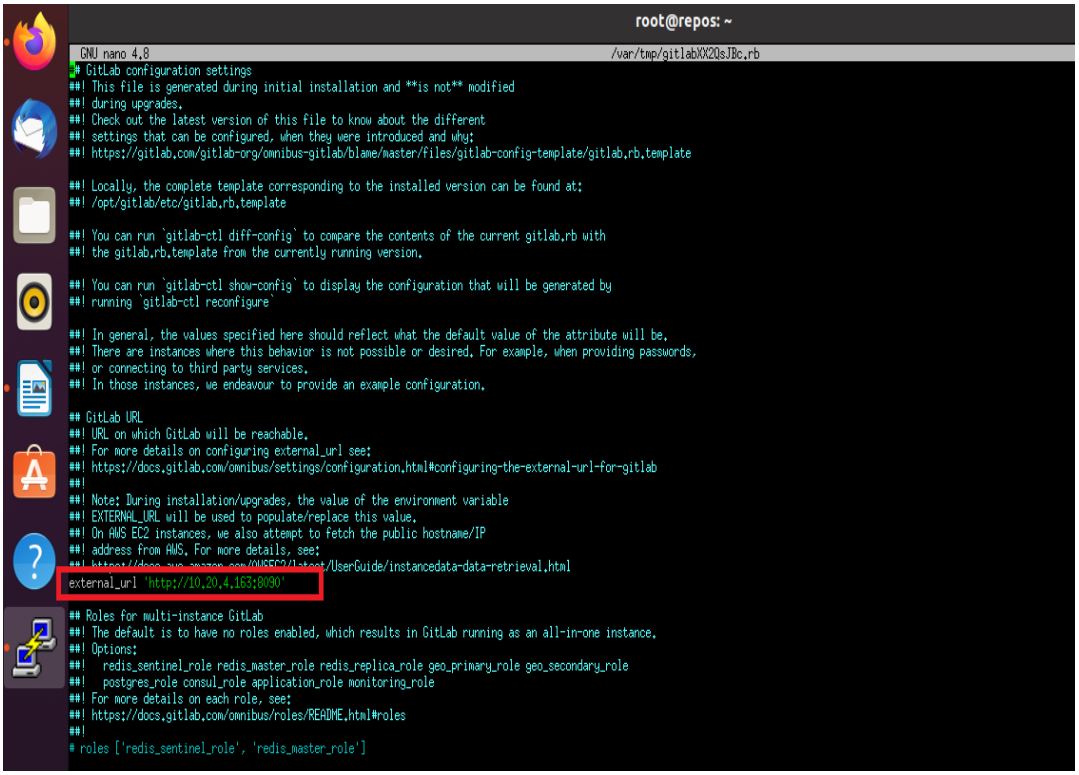
El usuario “root” que es el administrador del sistema se debe de configurar por primera vez el password, para esto se necesita acceder mediante un navegador a la IP y puerto configurado, en este caso “https:// 10.20.4.163:8090”. Las credenciales que se debe utilizar por primera vez en nuestro caso son:

Ahora es el turno de cambiar el puerto por defecto para el acceso via web. El puerto por defecto es el puerto 80, pero queremos cambiar al puerto 8090.

Ingresamos a la terminal SSH donde está el servidor GitLab e ingresamos el siguiente comando

Figura 16

“Sudo -e /etc/gitlab/Gitlab.rb”



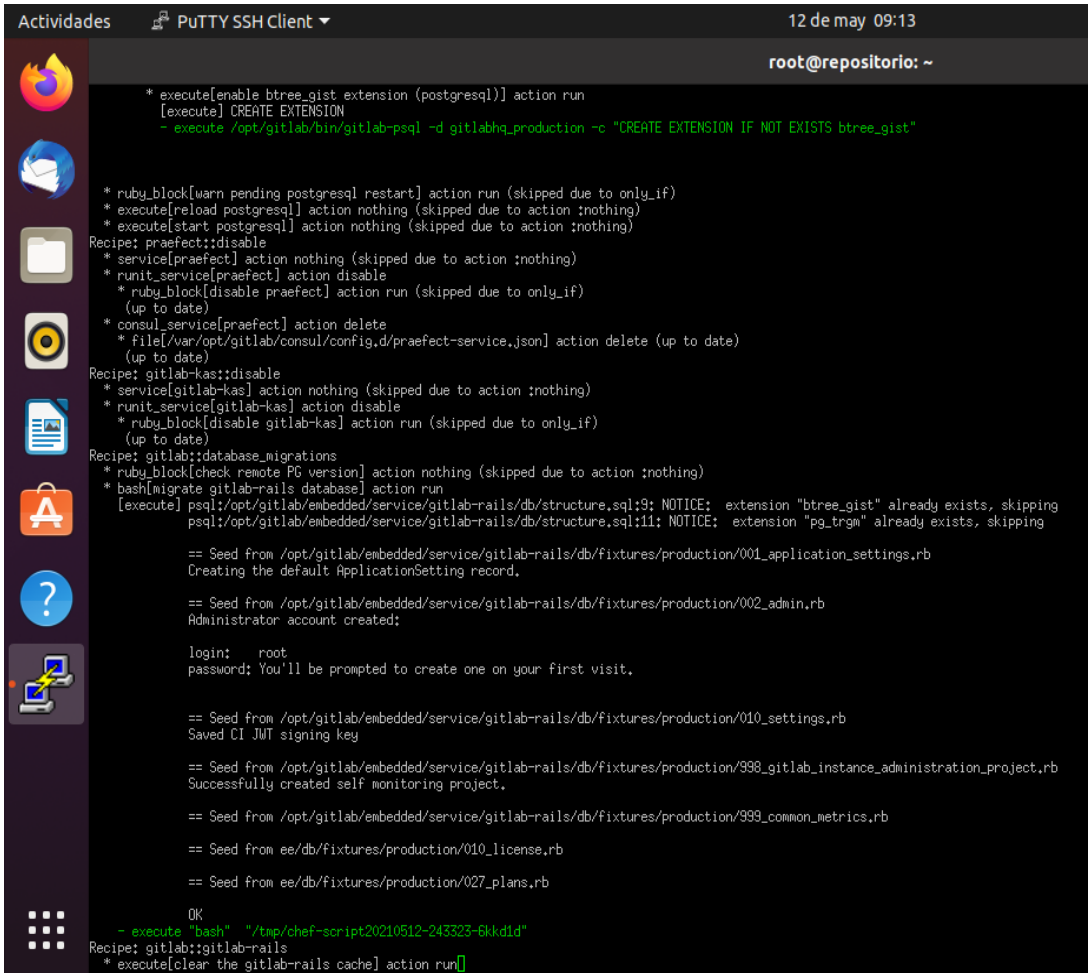
Nota: La imagen representa la Ip https:// 10.20.4.163:8090 como administrador.

Agregamos el puerto que necesitamos configurar en este caso es el puerto “8090”

external_url ‘http://gitlab.example.com:8090’ como indica en la imagen anterior, por último, guardamos y volvemos a ingresar al terminal el siguiente comando para poder acceder a la web.

Figura 17

“Sudo gitlab-ctl Reconfigure”



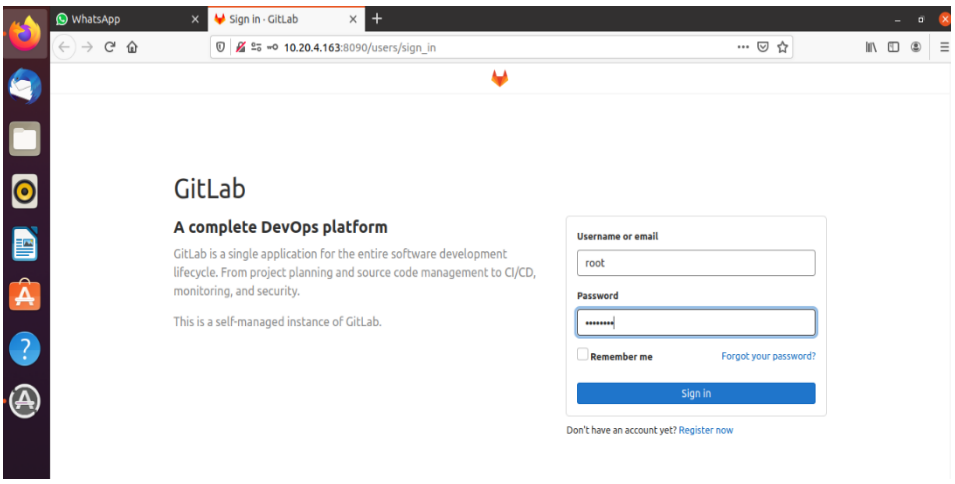
Nota: La imagen representa el acceso a la web de GitLab.

3.7 Manual del Administración del Repositorio GItLab.

Designar una persona responsable del sistema de repositorio el administrador, persona indicada de crear los grupos de repositorios con los permisos de acceso necesarios al código fuente del software de los sistemas que esté a cargo en los desarrolladores. La finalidad del administrador es que lleve documentado de todo el personal desarrollador tiene acceso al código fuente como la última versión del software puesto en producción.

Figura 18

“Manual del Administración del Repositorio GItLab”



Nota: La imagen representa el login como administrador de GitLab.

Llegamos a este punto donde el administrador debe registrarse para empezar como administrador y poder crearlos grupos de trabajo en el proyecto que pudieran estar desarrollado en conjunto o individualmente y puedan registrar versión actual en esta herramienta de GitLab

3.7.1 Manual del Usuario

Antes de empezar a utilizar el repositorio de GitLab debemos de haber instalado todos los programas dentro del sistema operativo

- Git - versiones actuales
- Visual Studio Code – versiones actuales

3.7.2 Uso de GitLab

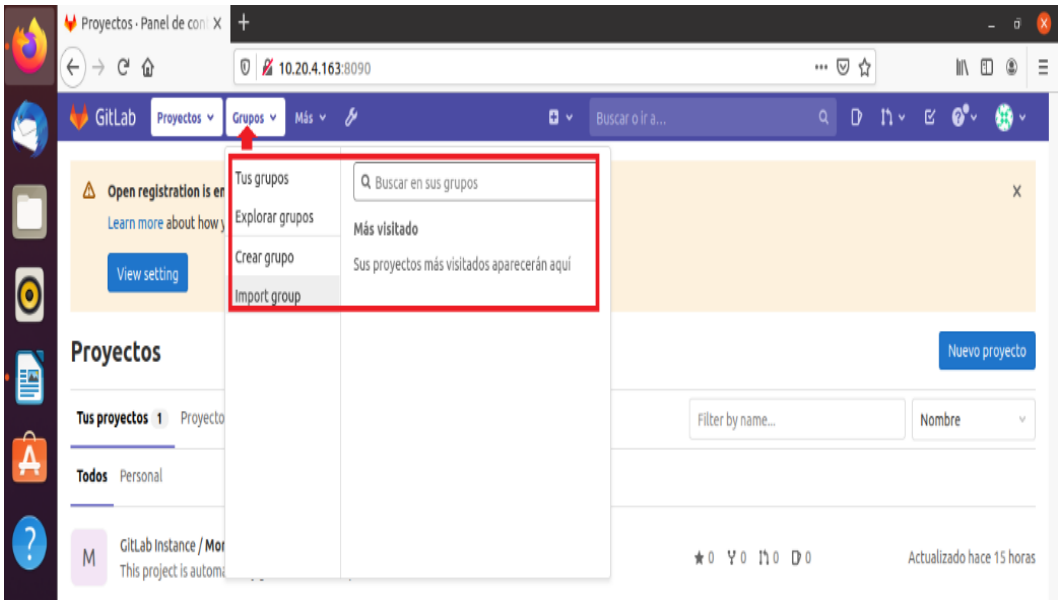
El uso de GitLab es como cualquier otra herramienta de repositorio en Git, para se necesita utilizar un cliente Git para poder hacer un commit y push a la rama dentro del servidor.

Lo primero que vamos hacer es crear grupos de repositorios. Estos grupos

pueden tener permisos y roles. Para crear un grupo nos dirigimos a la opción “Groups→Your Groups”.

Figura 19

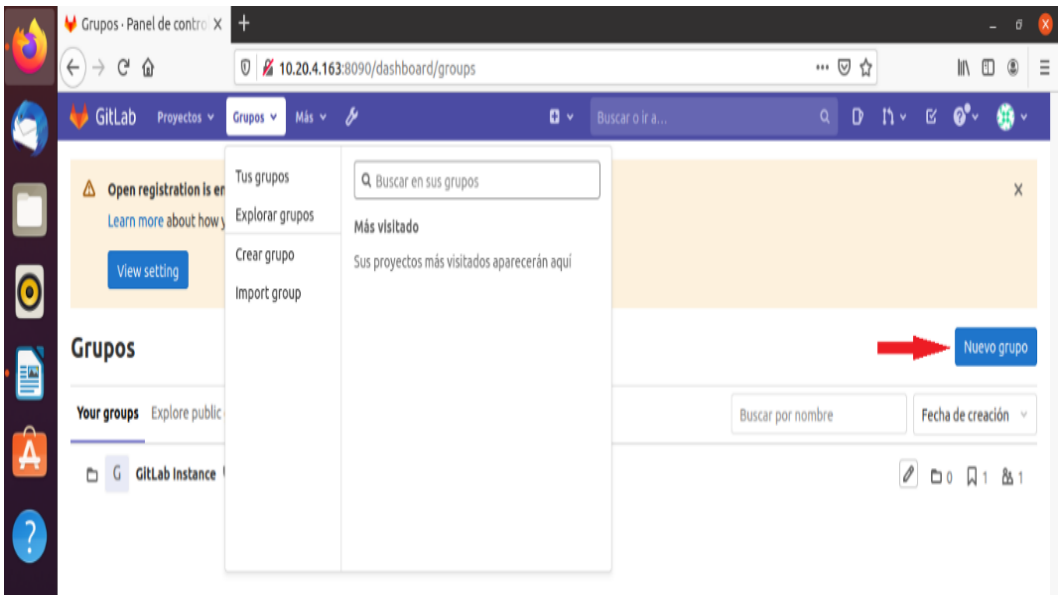
“Uso de GitLab”



Nota: La imagen representa la creación de grupos de trabajo.

Figura 20

“Luego damos click en “Nuevo Grupo”

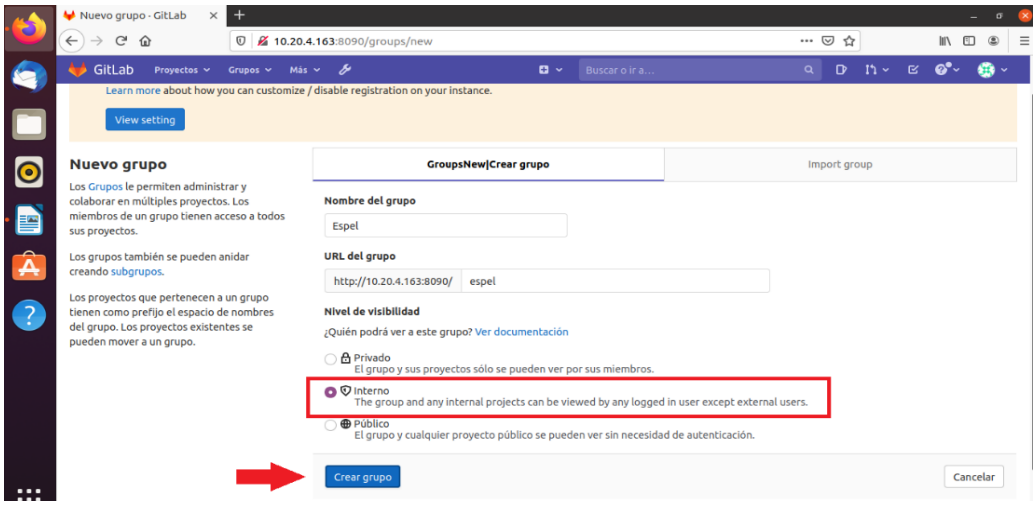


Nota: La imagen representa el nuevo grupo que se va a crear.

Le ponemos el nombre y luego seleccionamos la visibilidad del proyecto "**Interno**" y damos clic en el botón "**Crear Grupo".**

Figura 21

"Crear Grupo".

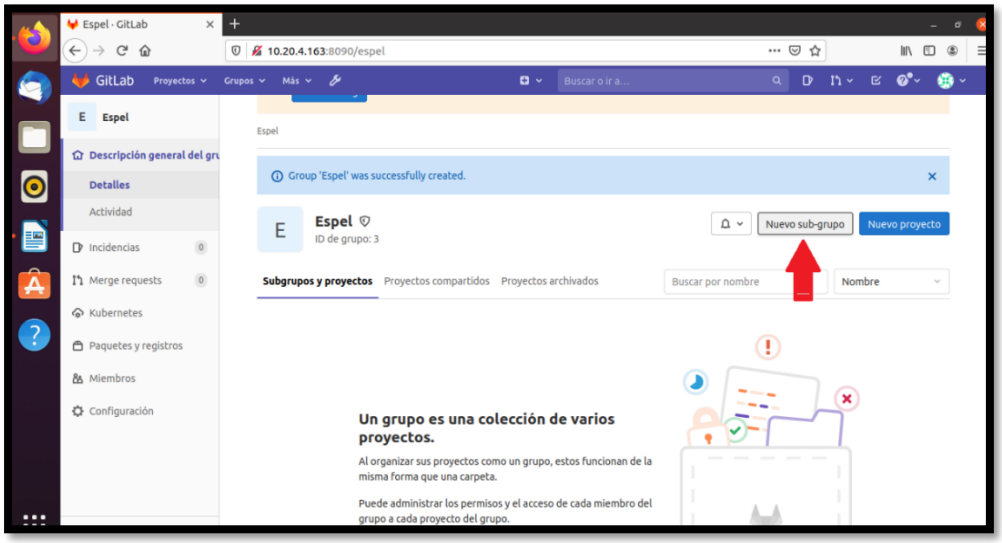


Nota: La imagen represente las configuraciones previas a la creación del grupo.

Dentro de los grupos podemos tener otros grupos en la opción "**Nuevo Subgrupo**" y seguimos los mismos pasos de anterior pasamos.

Figura 22

"Nuevo Subgrupo"

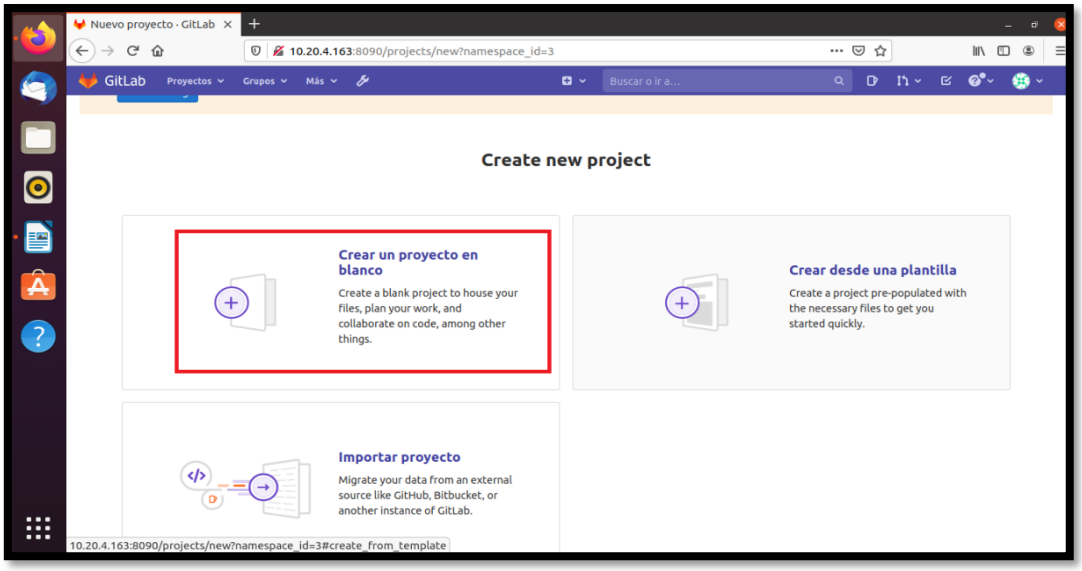


Nota: La imagen representa un subgrupo dentro de un grupo.

Seleccionamos el tipo de proyecto en este caso un proyecto en blanco "**Crear un Proyecto en Blanco**"

Figura 22

"Crear un Proyecto en Blanco"

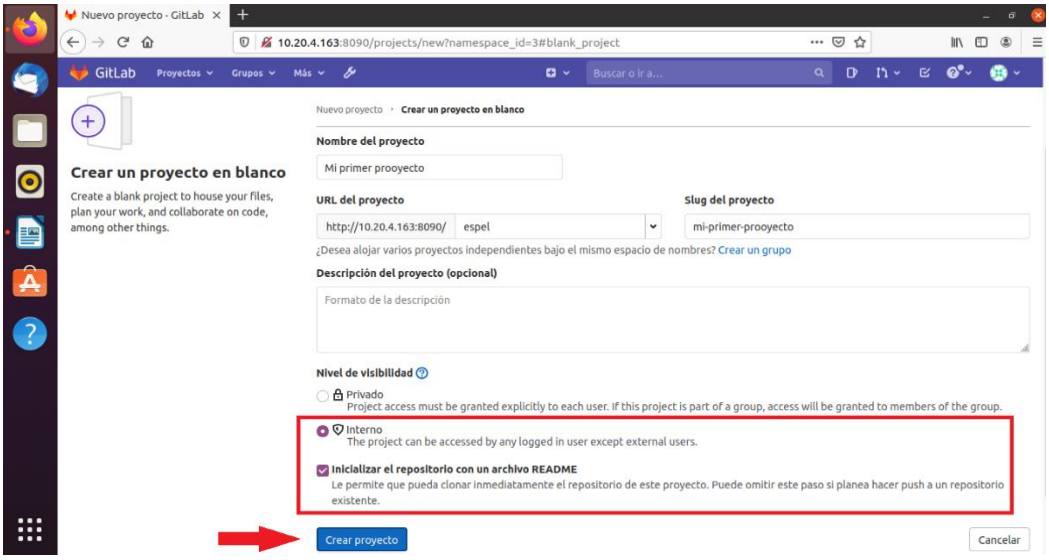


Nota: La imagen representa un proyecto en blanco.

Dentro del proyecto ingresamos el nombre del proyecto, y seleccionamos la visibilidad e inicializamos con un README. Finalmente, clic en "**Crear Proyecto**"

Figura 23

"Crear Proyecto"

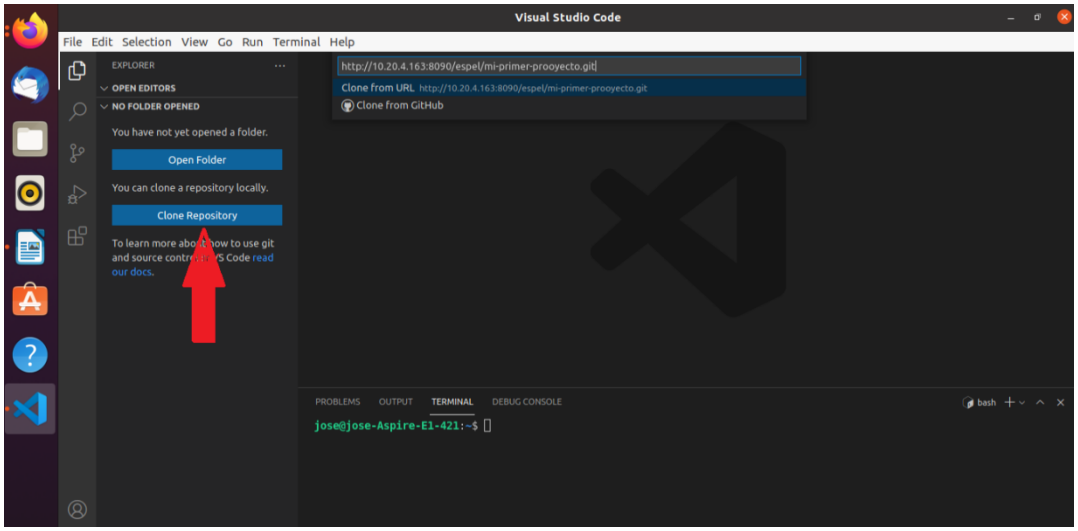


Nota: La imagen represente el nombre del proyecto e inicializamos con un README.

Para clonar el repositorio se necesita tener instalado Git y cualquier cliente de Git. En nuestro caso vamos a usar Visual Studio Code. Nos vamos a dirigir al tercer menú de control de versiones y vamos a dar clic "**Clonar Repositorio**"

Figura 24

"Visual Studio Code"



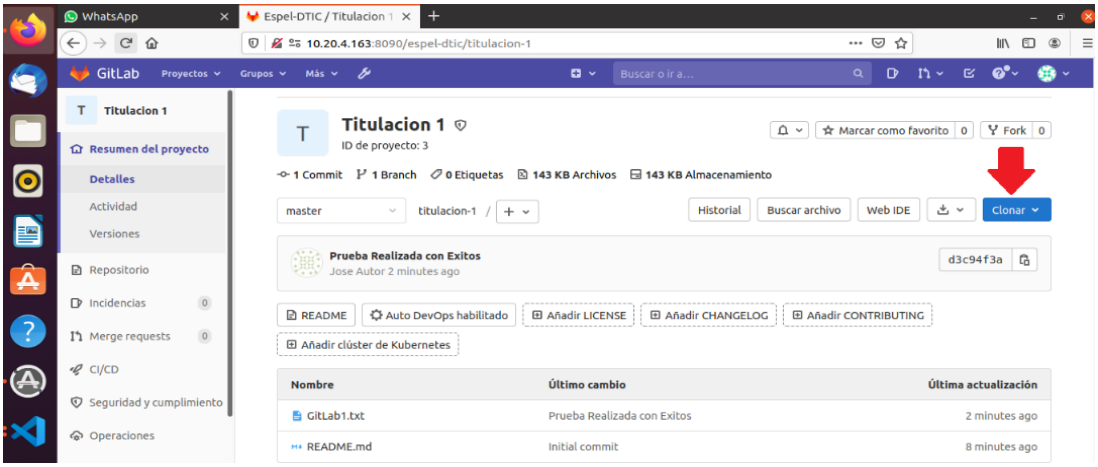
Nota: La imagen representa clonar el repositorio en Visual Studio Code

Una vez realizado este paso debemos regresar a nuestro repositorio de

GitLab y nos ubicamos en la pestaña que dice “**CLONAR**” nos desplazara la URL de nuestro repositorio, copiamos.

Figura 25

“Clonar Repositorio”

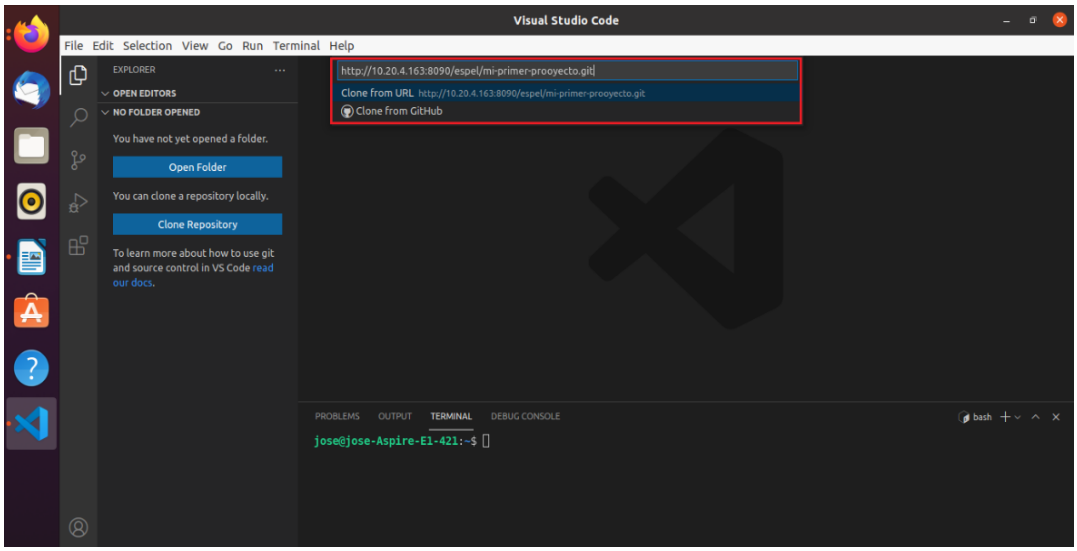


Nota: La imagen representa la copia del URL del repositorio que vamos a clonar.

Luego se debe pegar el url copiado dentro de la barra superior de comandos de Visual Studio Code. Seleccionamos la opción que dice “**Clone From** http://10.20.4.163:8090/espel/mi-primer-proyecto.git”.

Figura 26

“Clone From http://10.20.4.163:8090/espel/mi-primer-proyecto.git”

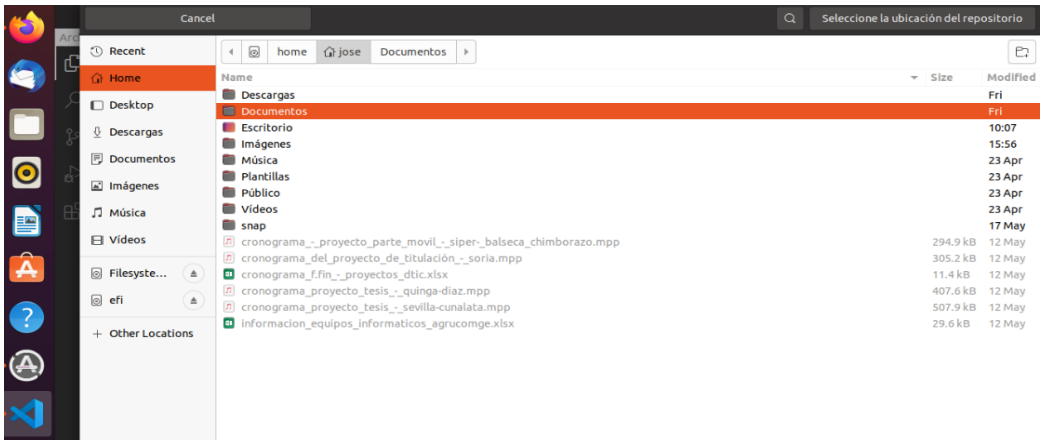


Nota: La imagen representa pegar el URL copiada en Visual Studio Code.

Seleccionamos la carpeta donde se va a clonar los archivos para posteriormente poder identificarlos con facilidad y en el caso de ser necesario revertir cualquier control de versiones con la debida autorización para dicha reversión.

Figura 27

Selección de carpeta donde se va a clonar

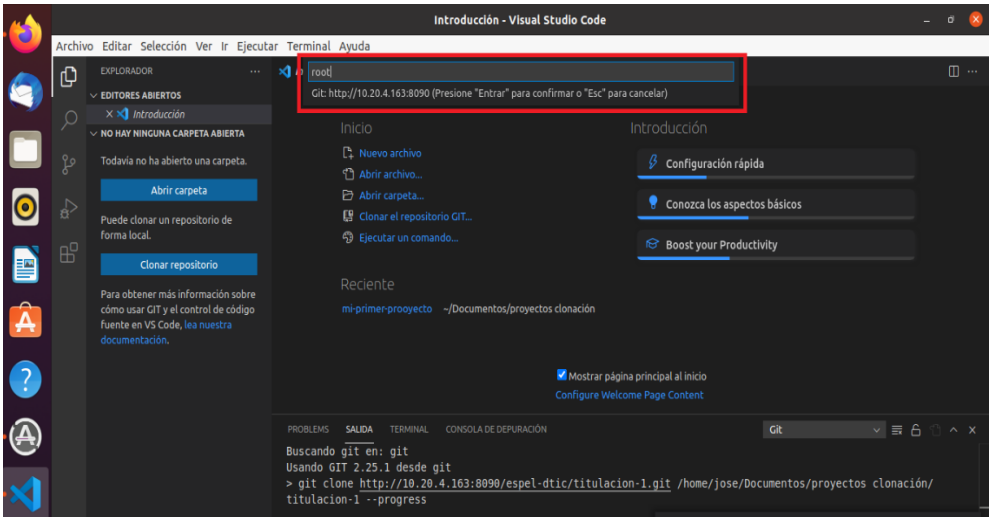


Nota: La imagen representa el lugar donde se va a guardar el clon del repositorio.

Después te pedirá las credenciales de GitLab para poderlo clonar. Cada usuario debe tener su propia clave y usuario en GitLab.

Figura 28

"Validación usuario en GitLab"

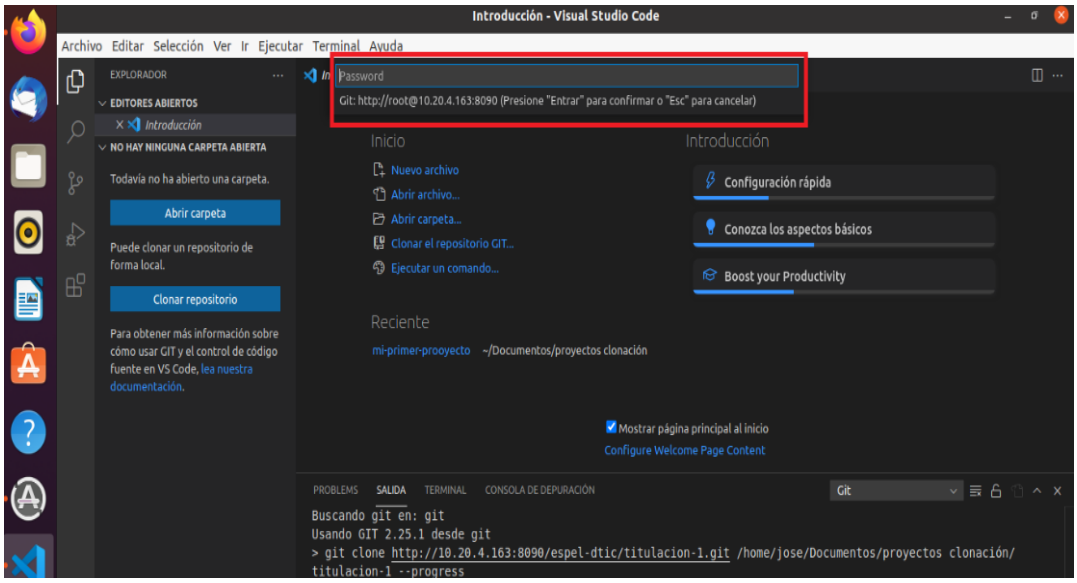


Nota: La imagen nos indica la validación del usuario en el repositorio.

Seguido debemos ingresar la contraseña para que nos valide las credenciales de Git

Figura 29

“Validar Contraseña”

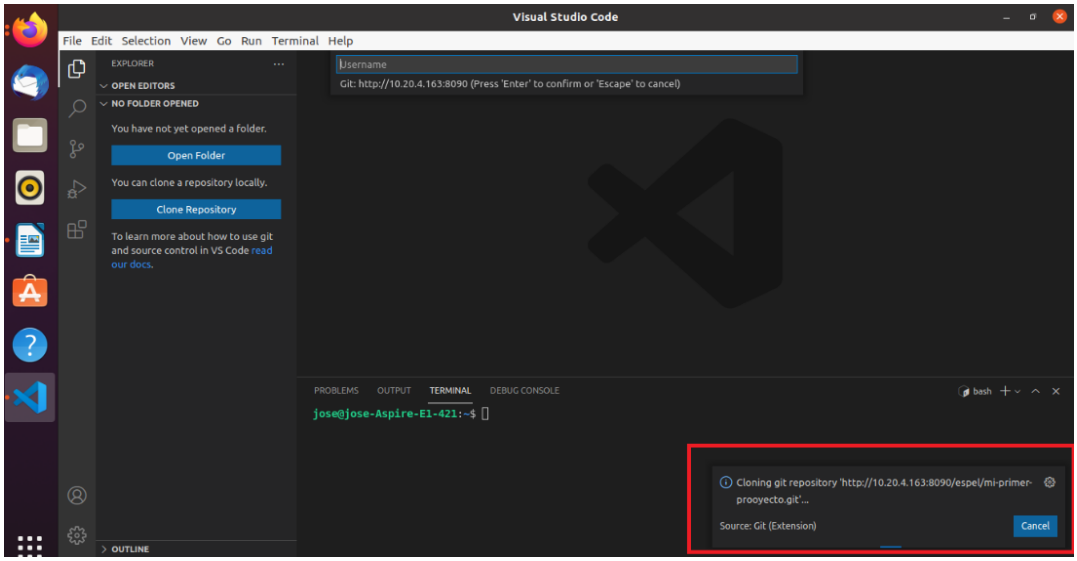


Nota: La imagen represente la validación de la contraseña de GitLab y sus credenciales.

Abrimos el proyecto y nos reflejara un mensaje de confirmación en la parte inferior derecha de la pantalla de Visual Studio Code.

Figura 30

“Mensaje de Confirmación”

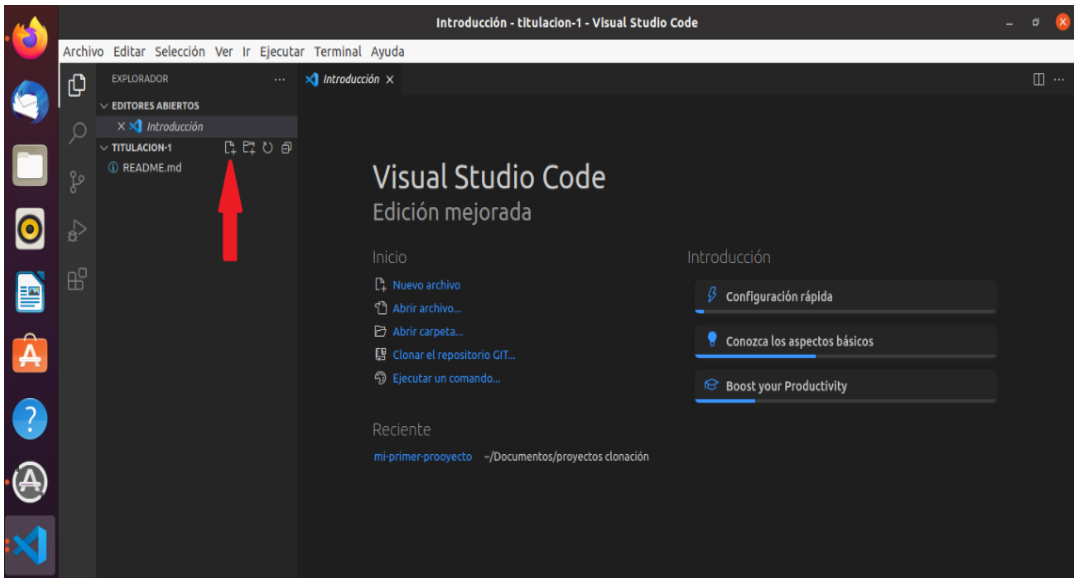


Nota: La imagen representa el ingreso al repositorio de GitLab.

Ahora nos dirigimos al repositorio de GitLab. Vamos a realizar una prueba creando un archivo con por ejemplo.txt de la siguiente manera.

Figura 31

“Realizar una Prueba”



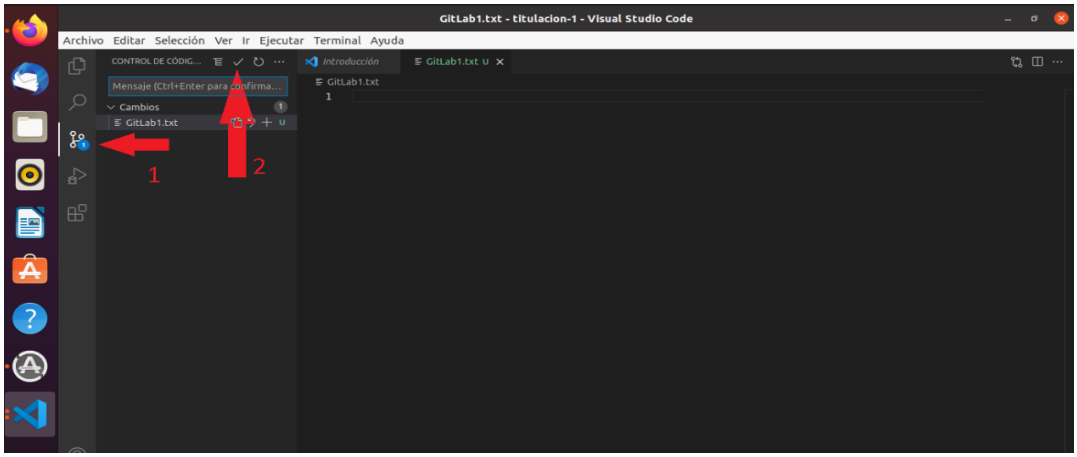
Nota: La imagen representa las pruebas previas del funcionamiento del repositorio.

Nos dirigimos a la tercera pestaña de control de versiones en Visual Studio

Code y añadimos los archivos para hacer un commit LOCAL. Escribimos un mensaje de acuerdo al repositorio que clonamos y le damos clic en el icono de visto "**Commit**" en la parte superior como lo detalla en la imagen.

Figura 32

"Icono de visto Commit"

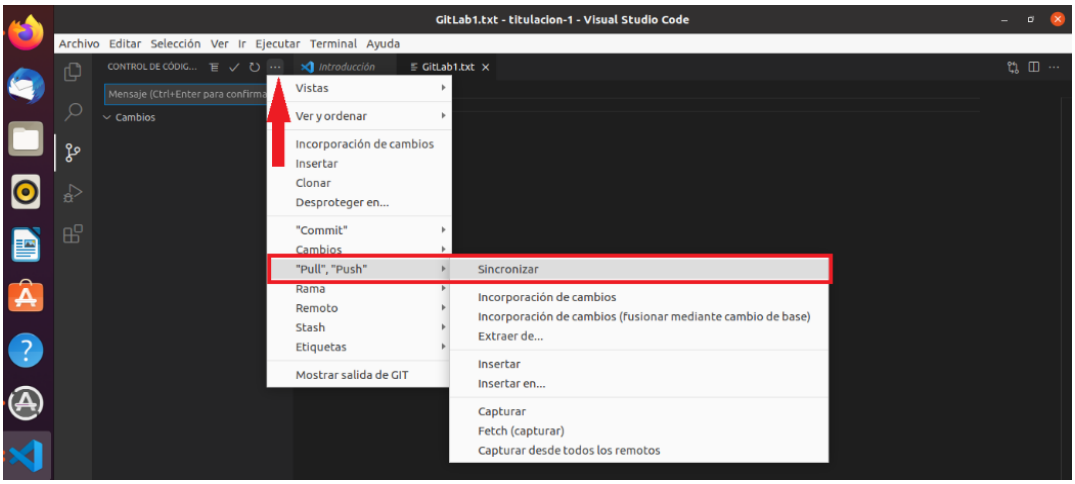


Nota: La imagen representa que esta listo para ser utilizado como repositorio.

Ahora debemos enviar los cambios al servidor. Para esto abrimos el menú, esto conseguimos dando clic en los tres puntos en la parte superior y seleccionamos la opción de "push……" como indica en la imagen.

Figura 33

"Seleccionamos la opción de "push……"

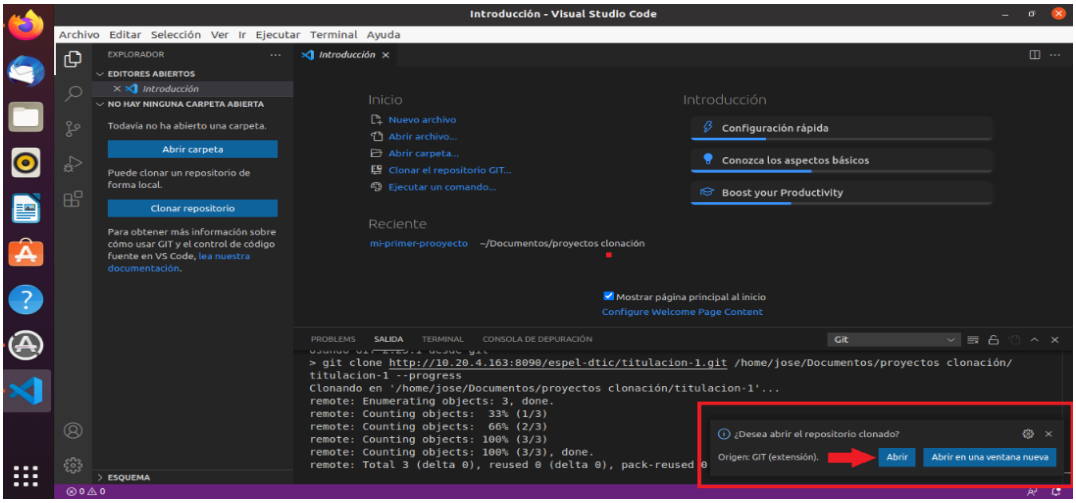


Nota: La imagen representa el envío de los cambios al servidor.

La primera vez que clonemos nos saldrá un mensaje de confirmación en la parte inferior derecha la misma que debemos aceptar.

Figura 34

"Mensaje de confirmación"

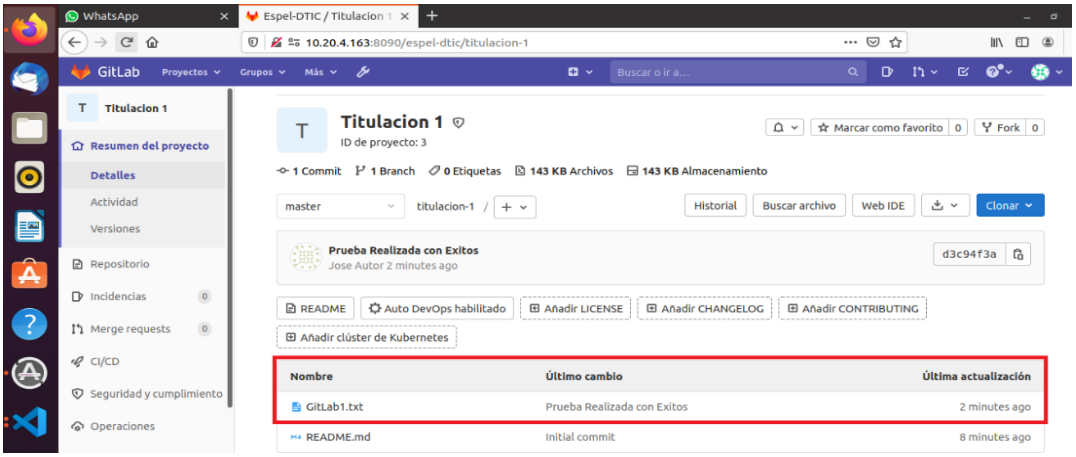


Nota: La imagen representa la confirmación de los cambios en el servidor.

Por último, constatamos que los cambios efectuados se hayan subido al servidor, en nuestro caso debe estar GitLab.txt reflejado en nuestro repositorio.

Figura 35

"GitLab.txt reflejado en nuestro repositorio"



Nota: De esta manera hemos creado satisfactoriamente nuestro repositorio y también logramos establecer la versión de un archivo editado en txt.

CAPÍTULO IV

4. Conclusiones y recomendaciones

4.1 Conclusiones

- La Dirección de Tecnologías de la Información y Comunicaciones del Ejército Ecuatoriano es el departamento encargado de desarrollar los proyectos de sistemas de Software y sus aplicativos, los mismos que utilizan herramientas descontinuadas para el registro de control de versiones.

- Con la instalación del repositorio GitLab ayudará de manera significativa a la administración ordenada del control de versiones de todos los sistemas que se produzcan en este departamento dedicado al desarrollo del Software.

- Existen diferentes ventajas por la instalación del repositorio GitLab, la misma que cuenta con las herramientas del trabajo en equipo, esto facilitará mucho a los desarrolladores en un proyecto extenso, que se necesita la colaboración de más personas para lograr un trabajo eficiente.

4.2 Recomendaciones

- La Dirección de Tecnologías de la Información y Comunicaciones del Ejército Ecuatoriano está en la obligación de actualizar las herramientas que utilizan los desarrolladores de Software para mejorar la producción y la calidad de los sistemas.

- Es necesario capacitar a todo el personal que se dedica al desarrollo del Software sobre el funcionamiento de este repositorio con la herramienta GitLab para que se familiaricen y sea de fácil manejo.

- Aprovechar al máximo todas las bondades que nos ofrece la tecnología especialmente la herramienta GitLab que además de ser gratuita nos facilita el trabajo a todos los desarrolladores de Software.

- Finalmente se debe delegar a una persona que éste encargada directamente como administrador del repositorio de GitLab para llevar los controles de versión de todos los sistemas que se desarrollan en la Dirección de Tecnologías de la Información y Comunicaciones del Ejército Ecuatoriano.

Bibliográficas

A. Qloudea. (04 de Septiembre de 2014). *Qué es un disco duro*. Recuperado el 20 de Marzo de 2021, de Qloudea: https://qloudea.com/blog/que-es-un-disco-duro/

Alejandro, M. T. (14 de Julio de 2020). Recuperado el 15 de Marzo de 2021, de http://repositorio.unesum.edu.ec/handle/53000/2255

Aljarafe, M. d. (03 de Novienbre de 2020). *INTPLUS*. Recuperado el 15 de Abril de 2021, de http://www.videovigilancia.com/respvideovigilancia.htm

Angel, M. M. (2018). *El uso y herramientas de Git y GitLab.* México, México.

Barcell Manuel, F. (2014). *Medios de transmisión*. Recuperado el 12 de Abril de 2021, de rodin uca: https://rodin.uca.es/xmlui/bitstream/handle/10498/16867/tema05_medios.pdf

Camaras ocultas. (27 de Febrero de 2019). *Elección sistema camaras adecuado*. Recuperado el 28 de Mayo de 2021, de Camaras ocultas: https://www.camarasocultas.cl/eleccion-sistema-camaras-de-seguridad-adecuado/

CARLOS NOVILLO, M. (2014). *Repositorio.ug.edu.ec*. Recuperado el 15 de Febrero de 2021, de UNIVERSIDAD DE GUAYAQUIL: http://repositorio.ug.edu.ec/bitstream/redug/6529/1/TesisCompleta-523.pdf

Cavsi. (2016). *Computer Audio Video System Integrator*. Recuperado el 03 de mayo de 2021, de https://www.cavsi.com/preguntasrespuestas/caracteristicas-del-software/

Charlene. (24 de 09 de 2017). *Cómo usar el switch PoE para cámaras de vigilancia IP*. Recuperado el 17 de Marzo de 2021, de fs.community: https://community.fs.com/es/blog/using-8-port-poe-switch-for-ip-surveillance.html

Chavarría Neira, B. (2017). *Repositorio Institucional*. Recuperado el 04 de Mayo de 2021, de https://dspace.ups.edu.ec/handle/123456789/14162

CISCO. (Enero de 2021). *cisco.com*. Recuperado el 21 de Abril de 2021, de https://www.cisco.com/c/en/us/support/switches/catalyst-2960x-24ps-l-switch/model.html#~tab-specs

Coelma. (2019). *Tipos de cámaras de seguridad.* Recuperado el 20 de Abril de 2021, de todo electronica: https://www.todoelectronica.com/manuales/tipos_camars_seguridad.pdf

Collins-Sussman, B. (2018). *Control de versiones con Subversion.* Recuperado el 05 de abril de 2021

Ctronics. (2020). *camara ip wifi*. Recuperado el 20 de Abril de 2021, de https://camaraipwifi.com/tipo-bullet-bala/

Diaz, G. (2010). Redes de Computadoras. https://cmapspublic2.ihmc.us/rid=1H8K8MN9X-1SXK3SB-K1K/conceptos-basicos.pdf.

EcuRed. (s.f.). *EcuRed*. Recuperado el 10 de mayo de 2021, de https://www.ecured.cu/Repositorio

Española, R. A. (15 de 03 de 2016). *https://definicion.de/software/*. Recuperado el 14 de junio de 2021, de https://definicion.de/software/: https://definicion.de/software/

ESPE. (2020). *Universidad de las Fuerzas Armadas ESPE*. Recuperado el 21 de Abril de 2021, de https://espe-el.espe.edu.ec/filosofia/

Ezequiel, C. (2021). Git vs GitHub. *freeCodeCamp.org*, 5.

Figueiras Vidal, A. R. (2002). *Una panorámica de las telecomunicaciones. .* Madrid: PEARSON EDUCACIÓN S.A.

FOSCAM. (21 de Enero de 2021). *Foscam.es*. Recuperado el 10 de Mayo de 2021, de Descarga de Programas y Manuales para cámaras IP: https://www.foscam.es/descarga/

Frezzo & DiCerbo. (2009). *Psychometric and Evidentiary Approaches to Simulation Assessment in Packet Tracer Software.* Valencia España: IEEE.

Gabriel, M. (Diciembre de 2015). *Repositorio.* Recuperado el 08 de Junio de 2021, de https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

García, F. J. (2018). *Ingeniería del Software.* España.

GARCIA, M. (2019). INTRODUCCIÓN A LA COMPUTACIÓN, REDES DE DATOS E. http://148.215.1.182/bitstream/handle/20.500.11799/108235/secme-16870_1.pdf?sequence=1.

Gebhart, A. (27 de Marzo de 2019). *CNET*. Recuperado el 16 de Mayo de 2021, de https://www.cnet.com/es/noticias/reconocimiento-facial-apple-amazon-google-ai/#:~:text=Todos%20los%20sistemas%20de%20reconocimiento,de%20datos%20de%20im%C3%A1genes%20conocidas.

González, Yolanda. (27 de Julio de 2020). *Cámaras de reconocimiento facial*. Recuperado el 17 de Mayo de 2021, de Grupo Atico34: https://protecciondatos-lopd.com/empresas/camaras-reconocimiento-facial/

HIKVISION. (Enero de 2021). *hikvision.com*. Recuperado el 02 de Junio de 2021, de https://www.hikvision.com/es-la/products/ITS-Products/

Huidobro Moya, J. M. (2006). *Redes y servicios de telecomunicaciones.* Madrid: Paraninfo S.A.

Intelequelía. (2020). Ciclo de vida del software. *Intelequelía*, 2.

IONOS. (21 de 10 de 2020). *Digital Guide*. Recuperado el 19 de Abril de 2021, de Digital Guide: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/tutorial-de-gitlab/#:~:text=GitLab%20es%20un%20popular%20sistema,en%20imprescindible%20para%20los%20desarrolladores.

ISO/IEC 9126. (2015). *Funcionalidad*. Recuperado el 19 de 06 de 2020, de Funcionalidad: https://diplomadogestioncalidadsoftware2015.wordpress.com/norma-iso-9126/calidad-interna-y-externa/funcionalidad/#:~:text=to%20primary%20content-,Funcionalidad,ser%20utilizado%20bajo%20condiciones%20específicas.&text=Esto%20depende%2C%20en%20gran%20parte,

Jiménez, R. E. (2015). Metodologías Ágiles de Desarrollo de Software Aplicadas. *Revista Tecnologica*, 7-8-9.

JOSÉ L., M. L., & CARLOS E., P. C. (2014). *Repositorio upao.* Recuperado el 29 de Octubre de 2019, de http://repositorio.upao.edu.pe/bitstream/upaorep/653/1/MANTILLA%20_JOSE_SISTEMA_VIDEOVIGILANCIA_UNILAP.pdf

Juan Worton. (11 de Enero de 2019). *Fs Comunidad*. Recuperado el 22 de Mayo de 2021, de https://community.fs.com/es/blog/fiber-media-converter-what-is-it-and-how-it-works.html

JULIAN RODRÍGUEZ, F. (2018). *Circuito cerrado de televisión y seguridad electrónica 2.ª edición.* Madrid: Ediciones Paraninfo, S.A.

Lacoma, Tyler. (15 de Febrero de 2021). *camaras PTZ*. Recuperado el 20 de Mayo de 2021, de techlandia: https://techlandia.com/camara-ptz-sobre_76400/

Lardear, J. (2006). *NFPA*. Recuperado el 23 de Mayo de 2021, de https://www.nfpajla.org/archivos/exclusivos-online/otros/937-normas-nfpa-730-y-nfpa-731

Lazo Cajilima, J. S. (2015). *Desarrollo de un Prototipo web para la inscripción de nuevos alumnos empleando la tegnología Java Server Faces con componentesPrimeFaces.* Universidad del Azuay, Cuenca, Azuay, Ecuador. Recuperado el 26 de mayo de 2021, de http://201.159.222.99/bitstream/datos/5054/1/11493.pdf

Loor Rodríguez, J. G., & Ortiz Rodríguez, N. A. (2015). Tesis de Pregrado. *Sistema Web de Gestion Administrativa en la Operadora Turistica Ecuador Fourexperiences S.A. de la Ciudad de Chone Provincia de Manabí.* Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, Calceta.

Lozada, D. F. (Enero de 2020). *TECNOSeguro*. Recuperado el 22 de Mayo de 2021, de https://www.tecnoseguro.com/faqs/electronica/que-es-poe

Lucas Vega, K. B. (2017). Desarrollo e Implementacion de Aplicacion web Para el Control de Inventario del Local Comercial Máquinas Hidalgo. *Proyecto Tecnico de Ingenieria.* Universidad Politacnica Salesiana Sede Guayaquil, Guayaquil. Recuperado el 19 de mayo de 2021, de http://dspace.ups.edu.ec/handle/123456789/15097

LUIS, C. P. (Junio de 2018). *GOOGLE ACADEMICO*. Recuperado el 24 de Mayo de 2021, de https://bibdigital.epn.edu.ec/bitstream/15000/19845/1/CD-9251.pdf

Maida, E. G. (2015). *Metodologia de Desarrollo del Software.* Recuperado el 07 de Junio de 2021, de https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Master, B. (2019). *Subversion.*

Menendez, R. (04 de 05 de 2014). *Informática Aplicada a la Gestión Pública. Facultad Derecho. UMU*. Recuperado el 11 de Junio de 2021, de Informática Aplicada a la Gestión Pública. Facultad Derecho. UMU: https://www.um.es/docencia/barzana/IAGP/IAGP2-Ingenieria-software-introduccion.html#BM2

Mónica Cánovas, M. (s.f.). Empresa de Seguridad Mallorca y Camaras de Seguridad. *camaras analogicas.* TTCS, S.L.Camí Can Frontera, 26 B, Mallorca.

Perez, M. (14 de 08 de 2019). *Ubunlog*. Recuperado el 18 de Mayo de 2021, de Ubunlog: https://ubunlog.com/instala-putty-en-tu-ubuntu/#comments

PREVENT. (2019). *Partes de una cámara de video vigilancia*. Recuperado el 27 de Octubre de 2019, de prevent: https://www.prevent.es/servicios-de-seguridad/camaras-de-seguridad/comunidades-de-vecinos/partes-de-una-camara-de-videovigilancia-antivandalica-exterior

Quinde Pomagualli, W. F. (30 de Octubre de 2020). Recuperado el 10 de Junio de 2021, de https://bibdigital.epn.edu.ec/bitstream/15000/20280/1/CD%209745.pdf

R., E. T. (2012). *Revisión de los sistemas de control de versiones utilizados en el desarrollo de software.* https://doi.org/10.21500/20275846.267. Recuperado el 20 de Mayo de 2021

R., M. (2005). El profesional de la información. *glosaiumBITRI*, 5.

REDATEL. (22 de Marzo de 2016). *REDATEL*. Recuperado el 11 de Junio de 2021, de https://www.redatel.net/html/cable-coaxial.html

Reyes, M. M. (2006). *Configuracion e instalacion de un completo servidor de correo con Postfix y Cyrus.*

Rio, E. d. (08 de Febrero de 2014). *Tartanga*. Recuperado el 10 de Abril de 2021, de http://fibraoptica.blog.tartanga.eus/2014/02/08/la-importancia-de-un-etiquetado-correcto-en-las-instalaciones-de-cableado-estructurado/

Roa, O. (2017). *Ingeniería de Software.* Recuperado el 27 de Mayo de 2021, de http://cic.javerianacali.edu.co/wiki/lib/exe/fetch.php?media=materias:introingsistemas:ingenieria_de_software-i.pdf

RODRÍGUEZ FERNÁNDEZ, J. (2013). *Circuito cerrado de televisión y seguridad.* Madrid: Paraninfo, S.A.

Roger S. Pressman, P. (2010). *Ingeniería del software.* Mexico: McGRAW-HILL.

S.Información.DROSE. (s.f.). *SISTEMAS DE INFORMACIÓN*. Recuperado el 15 de Junio de 2021, de https://sites.google.com/site/sinformaciondrose/ciclo-de-vida-del-software

Salazar, B. (2020). *Ingeniería Industrial*. Recuperado el 28 de Julio de 2020, de Ingeniería Industrial: https://www.ingenieriaindustrialonline.com/gestion-de-almacenes/que-es-la-gestion-de-almacenes/

Sanchez, D. (17 de Mayo de 2020). *cámara Fisheye / Ojo de Pez*. Recuperado el 19 de Mayo de 2021, de ITA.TECH: https://info.ita.tech/blog/que-son-las-camaras-fisheye-ojo-de-pez

Sanchez, J. S. (2013). *Ingenieria de Proyectos Informaticos.* Graphic Group S.A.

Sanz, M. L. (2015). *Programacion Web en el Entorno Cliente.* Madrid: RA_MA S.A.

Saulo, H. (2008). *LA IMPORTANCIA DEL DESARROLLO DEL SOFTWARE.* México.

SEGURIDAD, C. (27 de Octubre de 2017). *A. COFERSA SEGURIDAD*. Recuperado el 20 de Abril de 2021, de http://cofersaseguridad.com/que-es-un-sistema-de-videovigilancia/

Silva Perez & José Olger. (2016). *Introducción al AutoCAD en tres dimensiones.* Cuenca-Ecuador: Universitaria Abya-Yala.

Silvia Martí, M. (2013). *riunet.upv.es*. Recuperado el 09 de Junio de 2021, de UNIVERSIDAD POLITECNICA DE VALENCIA: https://riunet.upv.es/bitstream/handle/10251/34082/memoria.pdf

Sosio, N. (2013). *dvr*. Recuperado el 11 de Junio de 2021, de S.O.S seguridad: http://www.seguridadsos.com.ar/dvr/

Straub, S. C. (2014). *Pro Git.* Apress.

Tanenbaum, A. S. (2003). *Redes de Computadoras.* México: Pearson Educación.

tecnitran. (2019). *Tecnitrán Telecomunicaciones*. Recuperado el 13 de Mayo de 2021, de https://www.tecnitran.es/videovigilancia/camaras-de-videoviglancia-ip/camaras-ip-domos-fijas/

TODOELECTRONICA. (2016). *INTRODUCCIÓN A LA VIDEOVIGILANCIA*. Recuperado el 03 de Junio de 2021, de https://www.todoelectronica.com/manuales/tipos_camars_seguridad.pdf

Toro, D. L. (28 de Mayo de 2015). (E. aPtito, Editor) Recuperado el 10 de Febrero de 2021, de https://bibdigital.epn.edu.ec/bitstream/15000/10770/1/CD-6313.pdf

Tp link. (Enero de 2021). *tp-link.com*. Recuperado el 20 de Marzo de 2021, de https://www.tp-link.com/ec/business-networking/accessory/mc112cs/

Ubuntu. (21 de diciembre de 2020). *Lunux en Español*. Recuperado el 02 de Junio de 2021, de https://www.xn--linuxenespaol-skb.com/distribuciones/ubuntu/

UNAM. (10 de junio de 2004). Estándar. *Revista unam*, http://www.revista.unam.mx/vol.5/num5/art28/art28-1c.htm. Recuperado el 10 de mayo de 2021

UNIR. (06 de 04 de 2021). *https://mexico.unir.net/vive-unir/ingenieria-de-software-que-es-objetivos/*. Recuperado el 15 de Junio de 2021, de https://mexico.unir.net/vive-unir/ingenieria-de-software-que-es-objetivos/: https://mexico.unir.net/vive-unir/ingenieria-de-software-que-es-objetivos/

vivotek. (2021). *Herramienta de diseño de sistemas de video IP*. Recuperado el 02 de Junio de 2021, de https://www.vivotek.com/es/ip_video_system_design_tool

Wikimedia, F. (2021). *Wikipedia*. Recuperado el 21 de Junio de 2021, de https://es.wikipedia.org/wiki/Software#:~:text=Software%20de%20programación%3A%20Es%20el,Compiladores

Zambrano Loor, J. M., & Hecheverria Hidrovo, J. E. (2014). Aplicaión web para la administracion de los materiales almacenados en las bodegas de la empresa constructora coinfra s.a. *Tesis de Ingenieria.* Escuela Superior Politecnica Agropecuaria de Manabí Manuel Félix López, Calceta. Recuperado el mayo de 31 de 2021, de http://repositorio.espam.edu.ec/handle/42000/74