**Universidad Tecnológica del Centro de Veracruz**

**Carrera**

Ingeniería Desarrollo y Gestión de Software Multiplataformas

**Cuatrimestre**

9°A

**Datos de Alumno**

Marin Salazar Ribaldo de Jesús

**Datos del docente**

ITI.Francisco Trujillo Romero

**Fecha**

04-junio-2022

# **Introducción**

La importancia de una implementación de un sistema de versiones para una administración de un proyecto es vital ayuda a los desarrolladores a moverse más rápido y permite que los equipos de software mantengan la eficiencia y la agilidad a medida que el equipo se escala para incluir más desarrolladores.

Estos proyectos pueden verse afectados por movimientos de los demás desarrolladores ya que si mueven ciertas áreas que no les corresponde pueden afectar el proyecto y retrasar una entrega quedando mal para las empresas.

El control de versiones puede dividirse en ramos evitando el colapso del mismo ya que si encuentran fallos puedes regresar una versión estable del mismo optimizando buscar el control de daños.

Ahora preguntaran ¿Cuál es el mas recomendado para mi proyecto? Para esa respuesta no hay un control de versiones exacto todo depende del proyecto para seleccionar un control de versiones adecuados.

# **Capítulo I**

Las plataformas mas usadas y recomendadas para el versiona miento son las siguientes:

1. GitHub. - Git es uno de estos sistemas de control, que permite comparar el código de un archivo para ver las diferencias entre las versiones, restaurar versiones antiguas si algo sale mal, y fusionar los cambios de distintas versiones. También permite trabajar con distintas ramas de un proyecto, como la de desarrollo para meter nuevas funciones al programa o la de producción para depurar los bugs.
2. GitLab. - GitLab ha dado un gran salto desde solo la administración de código hasta el monitoreo y la implementación, llamándose a sí mismo "DevOps completo". De hecho, ha aumentado la comodidad y ofrece muchos beneficios para los desarrolladores, sobre todo para los que trabajan en equipos más grandes.
3. Bitbucket. - Es una herramienta de alojamiento de código y colaboración basada en Git diseñada para equipos. Las inmejorables integraciones de Bitbucket con Jira y Trello están concebidas para unir a todo el equipo de software con el fin de poner en práctica un proyecto.

# **Capítulo II**

Los flujos de trabajo para los controladores de versiones mas utilizados y recomendados son los siguientes:

1. GitFlow. - GitFlow puede utilizarse en proyectos que tienen un ciclo de publicación programado, así como para la práctica recomendada de DevOps de entrega continua. Este flujo de trabajo no añade ningún concepto o comando nuevo, aparte de los que se necesitan para el flujo de trabajo de ramas de función.
2. GitHub Flow. - GitHub Flow, fue creado por GitHub y es conocido en la comunidad de desarrolladores como una alternativa simple y ligera a GitFlow. GitHub Flow se basa en un flujo de trabajo basado en branches que permite a equipos de desarrollo enfocarse principalmente en la entrega continua.
3. GitLab Flow. - GitLab Flow se caracteriza por un flujo de trabajo bastante similar a GitHub Flow. La principal diferencia es que no solo permite el uso de release branches, sino que adiciona el uso de environment branches, por ejemplo, QA, Pre-Producción y Producción. Esto debido a que considera los casos donde una nueva funcionalidad no siempre puede implementarse en entornos de producción.
4. One Flow. - OneFlow es necesario tener un master branch que tendrá una vida infinita en el repositorio de código, además, cada nueva versión de producción debe basarse en la versión anterior, una condición que la mayoría de proyectos de software cumple. La mayor diferencia entre One Flow y Git Flow es que en OneFlow no existe develop.

# **Conclusiones**

Para descartar que utilizar un software de control de versiones es lo mas recomendado y eficiente para los equipos de desarrollo de alto rendimiento ya que su uso practico de control de ramificaciones y evitar dañar en proyecto de manera directa es una manera muy eficiente para agilizar la velocidad de desarrollo.

# **Bibliografías**

<https://openwebinars.net/blog/estrategias-de-branching-gitflow-gitlab-flow-oneflow-github-flow/>

<https://openwebinars.net/blog/estrategias-de-branching-gitflow-gitlab-flow-oneflow-github-flow/#:~:text=GitLab%20Flow%20se%20caracteriza%20por,%2C%20Pre%2DProducci%C3%B3n%20y%20Producci%C3%B3n>.

<https://openwebinars.net/blog/estrategias-de-branching-gitflow-gitlab-flow-oneflow-github-flow/#:~:text=GitHub%20Flow%2C%20fue%20creado%20por,principalmente%20en%20la%20entrega%20continua>.

<https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=Gitflow%20es%20un%20modelo%20alternativo,vez%20y%20quien%20lo%20populariz%C3%B3>.

<https://bitbucket.org/product/es/guides/getting-started/overview>

<https://geekflare.com/es/gitlab-hosting/>

<https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores>