**HERRAMIENTAS DEL DESARROLLO**

**Formato de Proyecto de Entrega de Trabajo: Soluciones Complementarias para el Desarrollo de Software en Equipo para Control de Versiones**

**Objetivo del Proyecto:**

El objetivo de este proyecto es desarrollar una solución integral para el manejo de control de versiones en un equipo de desarrollo de software. Se debe implementar y documentar el uso de herramientas y flujos de trabajo relacionados con el control de versiones y la integración de equipos, con el fin de optimizar la colaboración, reducir conflictos y mejorar la calidad del código.

**Requerimientos del Proyecto:**

1. **Uso de herramientas de control de versiones:**
   * Se debe utilizar un sistema de control de versiones como **Git**, **SVN** o **Mercurial**.
   * Configuración de un repositorio remoto en plataformas como **GitHub**, **GitLab** o **Bitbucket**.
2. **Estrategias de Ramificación (Branching):**
   * Implementación de estrategias de ramificación adecuadas para equipos de desarrollo, como **GitFlow**, **Feature Branching** o **Trunk-Based Development**.
3. **Flujo de Trabajo de Desarrollo:**
   * Implementación de un flujo de trabajo eficiente de integración continua (CI) y entrega continua (CD), utilizando herramientas como **Jenkins**, **GitHub Actions**, **GitLab CI/CD** o **Travis CI**.
4. **Resolución de Conflictos:**
   * Estrategias para la resolución de conflictos en el código cuando dos desarrolladores trabajen en el mismo archivo o en áreas relacionadas del proyecto.
5. **Trabajo Colaborativo:**
   * Implementación de buenas prácticas de colaboración en equipo, como **Pull Requests (PR)**, **Code Reviews**, y comunicación eficiente mediante herramientas como **Slack** o **Trello**.
6. **Automatización de Tareas:**
   * Configuración de scripts automatizados para pruebas, despliegues y verificaciones de calidad de código.

**Estructura del Proyecto:**

1. **Portada:**
   * Título del proyecto.
   * Nombre del equipo de desarrollo.
   * Nombre del profesor
   * Logo o imagen representativa (opcional).
2. **Introducción:**
   * Descripción del objetivo del proyecto.
   * Justificación del uso de herramientas y estrategias seleccionadas.
   * Breve descripción de la metodología de trabajo empleada (ágil, iterativa, etc.).
3. **Configuración del Repositorio:**
   * **Descripción del Repositorio:** Nombre, descripción y link del repositorio en la plataforma seleccionada.
   * **Estructura de Ramas:** Descripción de las ramas principales y sus roles (e.g., main, develop, feature/, hotfix/).
   * **Configuración de Accesos:** Detalles sobre la gestión de accesos y permisos para los miembros del equipo.
4. **Estrategia de Control de Versiones y Ramificación:**
   * **GitFlow:** Explicación de cómo se implementa la estrategia de GitFlow, con un diagrama de flujo de trabajo.
   * **Branching Model:** Descripción detallada de cómo y cuándo se crean ramas para características, correcciones de errores y producción.
   * **Mezcla de Ramas y Resolución de Conflictos:** Procedimiento para realizar merges y resolver conflictos cuando se presenten.
5. **Flujo de Trabajo de Desarrollo:**
   * **Integración Continua (CI) y Entrega Continua (CD):** Descripción de las herramientas utilizadas y cómo se configuran para automatizar la construcción, las pruebas y el despliegue del software.
   * **Pull Requests (PR) y Revisiones de Código:** Explicación de la política de PR en el equipo y los estándares para la revisión de código.
6. **Automatización y Herramientas Complementarias:**
   * **Scripts de Automatización:** Descripción de los scripts utilizados para pruebas automatizadas, compilación, y despliegue.
   * **Herramientas de Gestión de Proyectos:** Uso de herramientas como **Trello**, **Jira**, o **Asana** para la gestión de tareas y seguimiento del progreso del proyecto.
   * **Herramientas de Comunicación:** Uso de **Slack**, **Teams**, o **Zoom** para la comunicación del equipo.
7. **Pruebas y Resultados:**
   * **Pruebas Unitarias:** Descripción de las pruebas automatizadas configuradas en el pipeline de CI/CD.
   * **Pruebas de Integración:** Detalles sobre cómo se realizaron las pruebas de integración y su automatización.
   * **Despliegue a Producción:** Explicación del proceso de despliegue y los controles de calidad previos al despliegue.
8. **Conclusiones:**
   * Reflexión sobre el proceso de implementación del control de versiones en equipo.
   * Lecciones aprendidas durante el proyecto.
   * Recomendaciones para equipos de desarrollo futuros.

**Rúbrica para Evaluación (Nota Total: 20 puntos):**

| **Criterio** | **Puntos** |
| --- | --- |
| **Configuración del Repositorio** | 4 puntos |
| - Repositorio bien configurado (con enlace). - Uso correcto de ramas principales (main, develop) y ramas de características. |  |
| **Estrategia de Control de Versiones y Ramificación** | 4 puntos |
| - Implementación correcta de la estrategia de ramificación (GitFlow, Feature Branching). - Uso adecuado de ramas para tareas específicas. |  |
| **Flujo de Trabajo de Desarrollo** | 4 puntos |
| - Implementación de CI/CD con herramientas adecuadas (GitHub Actions, Jenkins, GitLab CI/CD). - Flujo de trabajo claro y eficiente (Pull Requests, merges). |  |
| **Automatización y Herramientas Complementarias** | 4 puntos |
| - Uso adecuado de herramientas complementarias como Jira, Trello, Slack para gestión y comunicación. - Scripts de automatización efectivos para pruebas y despliegue. |  |
| **Pruebas y Resultados** | 2 puntos |
| - Pruebas unitarias e integración configuradas correctamente. - Despliegue realizado sin errores y con controles de calidad. |  |
| **Conclusiones y Presentación** | 2 puntos |
| - Conclusiones claras y detalladas. - Informe bien estructurado y presentación del proyecto. |  |

**Total: 20 puntos**

**Caso Detallado (Ejemplo):**

**Proyecto: "Desarrollo de una Plataforma de Gestión de Tareas"**

**Configuración del Repositorio:**

* Repositorio creado en **GitHub** llamado **task-manager-app**.
* Ramas principales: main (producción) y develop (desarrollo).
* Ramas de características: feature/login, feature/task-crud, feature/user-profile.

**Estrategia de Ramificación (GitFlow):**

* **main:** Contiene la versión estable de la aplicación.
* **develop:** Desarrollo continuo; se fusionan las ramas de características aquí.
* **feature/xyz:** Cada nueva característica se desarrolla en su propia rama feature/nombre-característica.
* **hotfix/xyz:** Para arreglos urgentes en producción.

**Flujo de Trabajo:**

1. Los desarrolladores crean ramas de características (feature/login, feature/task-crud).
2. Desarrollan nuevas funcionalidades y abren **Pull Requests** hacia la rama develop cuando están listos.
3. Después de la revisión de código y la aprobación del PR, se fusiona la rama feature a develop.
4. Se realizan pruebas automáticas a través de **GitHub Actions** antes de cada despliegue.
5. Una vez la rama develop está lista, se fusiona a main para producción.

**Automatización y Herramientas Complementarias:**

* **GitHub Actions** para ejecutar pruebas unitarias automáticamente en cada push.
* **Trello** para gestionar las tareas del equipo: tarjetas de tareas creadas para cada funcionalidad.
* **Slack** para la comunicación constante del equipo.

**Pruebas y Resultados:**

* Se configuraron pruebas unitarias utilizando **Jest**.
* **Jenkins** automatiza el despliegue a un servidor de pruebas después de cada push a develop.

**Conclusión:** El uso de GitFlow, CI/CD, y herramientas de comunicación como Slack y Trello permitió una gestión eficiente del equipo, reducción de conflictos y despliegues rápidos y seguros.