Logotipo

Descripción generada automáticamente

**Escuela Politécnica Nacional**

**Facultad de Ingeniería de Sistemas**

**Construcción y Evolución de Software (ISWD622)**

**GR2SW**

**Grupo: EvoLogic**

**Proyecto [1B]**

**Documento de Flujo de Trabajo**

**Alumnos:**

**Molina Anael**

**Palma Stuart**

**Quillupangui Andrés**

**Sánchez Ariel**

**Profesora: Evelyn Mosquera**

**Fecha de entrega: --/--/2024**

**Objetivo:**

Definir el proceso y las reglas para la gestión de cambios en el código fuente utilizando las ramas Master, Develop y Bugfix. Este flujo asegura una colaboración eficiente, control de calidad y una integración organizada en el repositorio principal.

**Flujo de Trabajo**

1. **Estructura de Ramas**

* **Master:**
  + Contiene el código estable en producción.
  + Solo se actualiza con cambios completamente probados y aprobados.
* **Develop:**
  + Contiene el código en desarrollo.
  + Recibe nuevas funcionalidades y correcciones antes de ser probadas para producción.
* **Bugfix:**
  + Se usa para corregir errores detectados en el código ya implementado.
  + Las correcciones se integran primero en Develop y, en casos críticos, directamente en Master.
* **Documentos:**
  + Contiene la documentación del proyecto.

1. **Creación de una Nueva Rama**

* **Regla:**  
  Cada tarea se desarrolla en una rama independiente que se deriva de Develop (para nuevas funcionalidades) o de Master (para correcciones urgentes).
* **Tipos de ramas:**
  + **Feature branches (nueva funcionalidad):** feature/nombre-descriptivo
  + **Bugfix branches (corrección de errores):** bugfix/nombre-descriptivo
* **Formato de nombres:**  
  Usar nombres descriptivos que reflejen la tarea. Ejemplo: bugfix/corrige-error-login.

1. **Desarrollo en la Rama**

* **Regla:**  
  Los cambios deben realizarse exclusivamente en la rama correspondiente.
* **Proceso:**
  + Realizar *commits* pequeños y descriptivos.
  + Seguir las guías de estilo del proyecto.
* **Validación:**
  + Ejecutar pruebas unitarias y funcionales antes de realizar un commit.

1. **Sincronización con Ramas Base**

* **Regla:**  
  Las ramas deben mantenerse actualizadas con su rama base (Develop o Master).
* **Proceso:**
  + Hacer *pull* regularmente desde la rama base.
  + Resolver conflictos de manera local antes de avanzar al siguiente paso.

1. **Solicitud de Revisión de Código (Pull Request / Merge Request)**

* **Regla:**  
  Cada cambio debe ser revisado antes de integrarse en las ramas Develop o Master.
* **Proceso:**
  1. Crear una solicitud de PR hacia la rama base (Develop o Master).
  2. Incluir una descripción detallada de los cambios realizados.
* **Responsables:**
  1. Autor del PR.
  2. Revisores asignados (mínimo dos).

1. **Revisión de Código**

* **Regla:**  
  Todos los PR deben pasar por revisión antes de ser aprobados.
* **Criterios de revisión:**
  + Correcta implementación de la funcionalidad o solución al error.
  + Cumplimiento de las guías de estilo del proyecto.
  + Cobertura adecuada de pruebas automatizadas.
* **Decisiones posibles:**
  + **Aprobado:** El PR cumple con los requisitos.
  + **Solicitar cambios:** Se deben realizar ajustes antes de proceder.

1. **Integración de Cambios**

* **Regla:**  
  Solo los PR aprobados pueden integrarse en la rama base correspondiente.
* **Proceso:**
  + Funcionalidades nuevas: Se integran en Develop.
  + Correcciones urgentes: Se integran en Master y se sincronizan con Develop.
  + Eliminar la rama temporal después del merge.

1. **Pruebas Finales e Implementación en Producción**

* **Regla:**  
  Antes de desplegar a producción, los cambios en Master deben ser probados exhaustivamente en un entorno de preproducción.
* **Proceso:**
  + Ejecutar pruebas de integración, funcionales y de aceptación en Master.
  + Implementar en producción solo después de que todas las pruebas sean satisfactorias.