
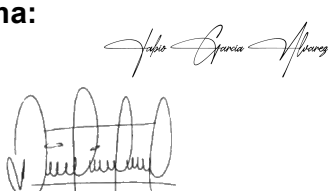
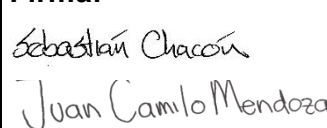


 <p>ALCALDÍA MAYOR DE BOGOTÁ D.C. AMBIENTE Instituto Distrital de Protección y Bienestar Animal</p>	PROCESO GESTIÓN TECNOLÓGICA		 <p>BOGOTÁ INSTITUTO DISTRITAL DE PROTECCIÓN Y BIENESTAR ANIMAL</p>
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

AUTORIZACIONES



ELABORÓ:	REVISÓ	APROBÓ
ÁREA TÉCNICA	OFICINA Y/O SUBDIRECCIÓN	LÍDER DEL PROCESO
Nombre: Fabio García Álvarez Diana Lorena Sánchez	Nombre: Sebastián Chacón Calvo Juan Camilo Mendoza	Nombre: Hugo Alberto Carillo Gómez
Firma: 	Firma: 	Firma:
Cargo: Contratistas Subdirección de Gestión Corporativa	Cargo: Contratistas Oficina Asesora de Planeación	Cargo: Subdirector de Gestión Corporativa

	PROCESO GESTIÓN TECNOLÓGICA		
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

GUÍA METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE

Tabla de contenido

1.	OBJETIVO	3
2.	ALCANCE	3
3.	TÉRMINOS Y DEFINICIONES	3
4.	DESARROLLO	4
4.1.	RESPONSABILIDADES GENERALES	4
4.2.	LINEAMIENTOS Y/O POLÍTICAS DE OPERACIÓN	5
4.2.1.	Diseño	6
4.2.2.	Gestión del Trabajo con Azure Boards	6
4.2.3.	Uso de Repositorio de Código Fuente con Azure Repos	7
4.2.4.	Integración Continua con Azure Pipelines	12
4.2.5.	Pruebas Unitarias	12
4.2.6.	Lineamientos para Desarrollo de Código Seguro	12
4.2.7.	Eventos de SCRUM en el IDPYBA	13
4.2.8.	Referencias	14
5.	FORMATOS Y DOCUMENTOS ANEXOS	14
6.	NORMATIVIDAD ASOCIADA.....	14

	PROCESO GESTIÓN TECNOLÓGICA		
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

1. OBJETIVO



Establecer los lineamientos técnicos y las mejores prácticas que se deben seguir durante el ciclo de vida de desarrollo de software en el Instituto Distrital de Protección y Bienestar Animal - IDPYBA, utilizando el ecosistema de Azure DevOps como plataforma centralizada para la gestión de repositorios, el control de trabajo y la integración continua.

2. ALCANCE

Este documento abarca el flujo de trabajo completo, desde la planificación de una historia de usuario en Azure Boards hasta su despliegue, pasando por el diseño de la solución, el control de versiones del código fuente con Azure Repos y la automatización de la compilación y pruebas con Azure Pipelines. Estos lineamientos buscan estandarizar y mejorar la calidad tanto del proceso como de los productos de software desarrollados en el IDPYBA.

3. TÉRMINOS Y DEFINICIONES

- **Azure DevOps:** Una plataforma de Microsoft que proporciona un conjunto de servicios para desarrolladores para soportar la planificación, colaboración, compilación y despliegue de aplicaciones.
- **Azure Boards:** Un servicio de Azure DevOps para la gestión del trabajo. Permite el seguimiento de épicas, características, historias de usuario, tareas y bugs mediante tableros Kanban y Scrum.
- **Azure Repos:** Un servicio de Azure DevOps que proporciona repositorios de código Git privados y escalables.
- **Azure Pipelines:** Un servicio de Azure DevOps que permite la creación, prueba y despliegue continuo (CI/CD) de cualquier aplicación en cualquier plataforma o nube.
- **CI/CD:** Integración Continua y Despliegue Continuo (o Entrega Continua). Prácticas que automatizan las fases de construcción, prueba y despliegue del software.
- **Develop:** Una rama de integración donde se fusionan todas las características completadas antes de ser incorporadas a la rama principal.
- **Diagrama de secuencia:** Es un tipo de diagrama de interacción UML que representa el comportamiento dinámico de un sistema. Ilustra cómo los objetos interactúan entre sí en un orden secuencial cronológico para realizar una tarea específica, utilizando líneas de vida verticales para el tiempo y flechas horizontales para los mensajes.
- **GitFlow:** Un modelo de ramificación de Git que utiliza ramas específicas para características, lanzamientos y correcciones urgentes para organizar el desarrollo.
- **Historia de usuario** Descripción breve, informal y centrada en el valor funcional de un producto, escrita desde la perspectiva del usuario final. Utilizada en metodologías ágiles (Scrum,

 ALCALDÍA MAYOR DE BOGOTÁ D.C. AMBIENTE <small>Instituto Distrital de Protección y Bienestar Animal</small>	PROCESO GESTIÓN TECNOLÓGICA		 INSTITUTO DISTRITAL DE PROTECCIÓN Y BIENESTAR ANIMAL
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

Kanban), define **quién** necesita qué, **qué** quiere, y **por qué**, con el formato: "Como [rol], quiero [funcionalidad] para [beneficio]".

- **Main / master:** La rama principal del repositorio que siempre debe contener código estable y listo para producción. En Azure DevOps, el nombre por defecto es main.
- **Pull Request (PR):** Mecanismo para que los desarrolladores notifiquen a los miembros del equipo que han completado una característica y soliciten la revisión y fusión de su código en otra rama.
- **Repositorio:** Espacio de almacenamiento digital centralizado y seguro, diseñado para guardar, gestionar y organizar el código fuente, la documentación y el historial de revisiones de un proyecto de software.
- **Work Item:** Un registro en Azure Boards que representa una unidad de trabajo, como una historia de usuario, una tarea o un bug.

4. DESARROLLO

4.1. RESPONSABILIDADES GENERALES

Rol	Descripción	Responsabilidades clave
Product Owner del Producto	Representante de los stakeholders y la voz del cliente. En el IDPYBA, este rol podría ser asumido por un líder de proyecto, un director de área o un subdirector técnico.	<ul style="list-style-type: none"> • Definir y priorizar el Product Backlog. • Asegurar que el equipo de desarrollo entienda los requerimientos. • Maximizar el valor del producto resultante del trabajo del equipo de desarrollo.
Scrum Master	Facilitador del proceso SCRUM. En el IDPYBA, este rol podría ser asumido por un gerente de proyectos o un líder de equipo con experiencia en metodologías ágiles.	<ul style="list-style-type: none"> • Asegurar que el equipo SCRUM siga las prácticas y valores de la metodología. • Eliminar impedimentos que puedan afectar al equipo. • Facilitar los eventos de SCRUM.
Development Team (Equipo de Desarrollo)	Equipo multifuncional encargado de crear el incremento del producto. En el IDPYBA, este equipo podría estar compuesto por desarrolladores de software, analistas de datos, diseñadores UX/UI, y expertos en bienestar animal (veterinarios, biólogos, etc.) según el proyecto.	<ul style="list-style-type: none"> • Auto-organizarse para completar el trabajo del Sprint Backlog. • Entregar un incremento de producto "Terminado" al final de cada Sprint. • Asegurar la calidad del incremento.

4.2. LINEAMIENTOS Y/O POLÍTICAS DE OPERACIÓN

Cuando una historia de usuario es asignada a un desarrollador, se deben llevar a cabo una serie de acciones con el objetivo de garantizar el cumplimiento y la calidad de los productos entregados.

Cada historia de usuario que se construya debe tener un diseño de software, su implementación debe permanecer en un repositorio de código fuente en **Azure Repos**, se deben realizar las pruebas unitarias correspondientes y se deben cumplir con la definición de HECHO (Definition of Done). A continuación, se presenta el detalle de lo que se espera con cada una de las historias desarrolladas.

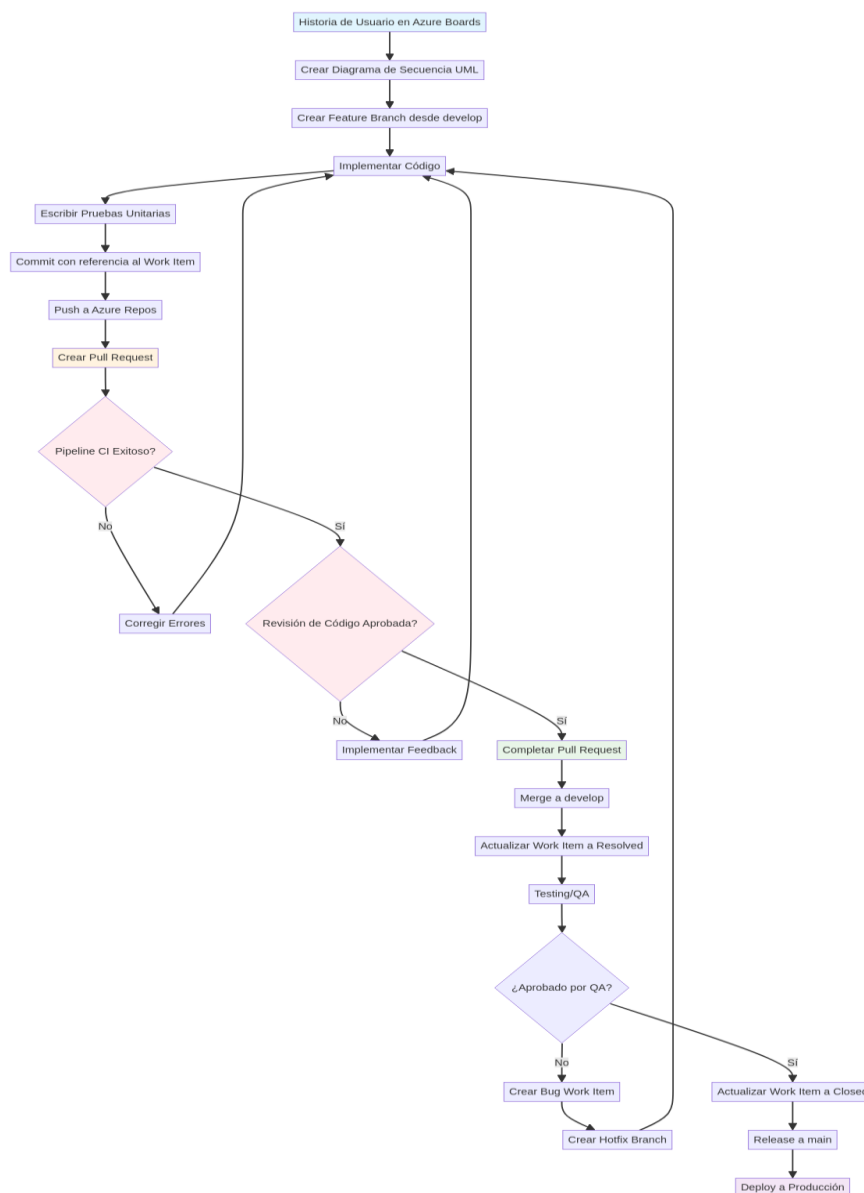




Ilustración 1 - Estructura Historias de Usuario

	PROCESO GESTIÓN TECNOLÓGICA		
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

4.2.1. Diseño

El objetivo de la fase de diseño es construir uno o varios artefactos que permitan garantizar:

- La verificación del adecuado entendimiento del requerimiento.
- La definición de un camino claro para implementar la solución.
- El dimensionamiento del impacto en los diferentes módulos del sistema de información.
- Contar con una herramienta para la validación de la propuesta de solución antes de iniciar actividades de codificación.
- La transferencia del conocimiento al equipo de trabajo.

Como requisito obligatorio se define la construcción del **Diagrama de Secuencia (UML)** del flujo principal, también conocido como “camino feliz” de la historia de usuario. Queda a discreción del desarrollador y el líder técnico el complementar la documentación con los diagramas de secuencia de los flujos alternos más importantes. Los artefactos complementarios podrían ser vistas estáticas (diagramas de componentes, paquetes, clases) o vistas dinámicas (diagramas de actividades, estados).

IMPORTANTE: Los diagramas de secuencias son artefactos que pretenden garantizar una representación muy cercana al código fuente. Por tanto, los objetos corresponden a clases reales y los mensajes a llamados a métodos existentes en el objeto receptor.

HERRAMIENTA RECOMENDADA: Se recomienda el uso de **draw.io** (ahora diagrams.net) para la creación de los diagramas UML, vinculada a la cuenta de correo institucional para facilitar la colaboración y el almacenamiento en la nube.



4.2.2. Gestión del Trabajo con Azure Boards

Toda historia de usuario debe ser gestionada como un **Work Item** dentro de **Azure Boards**. Esta herramienta es fundamental para planificar, seguir y discutir el trabajo a lo largo del ciclo de desarrollo, proporcionando una trazabilidad completa desde el requisito hasta el despliegue.

4.2.2.1. Jerarquía de Work Items

Para organizar el trabajo de manera efectiva, se utilizará la siguiente jerarquía, basada en el proceso **Agile** de Azure DevOps:

- **Épicas (Epics):** Representan grandes iniciativas de negocio. Se descomponen en Features.
- **Características (Features):** Agrupan un conjunto de Historias de Usuario que, en conjunto, entregan un valor significativo. Se descomponen en User Stories.
- **Historias de Usuario (User Stories):** Describen una funcionalidad desde la perspectiva del usuario final. Son la unidad principal de trabajo para el equipo de desarrollo y deben poder completarse dentro de un sprint. Se descomponen en Tareas.

 ALCALDÍA MAYOR DE BOGOTÁ D.C. AMBIENTE Instituto Distrital de Protección y Bienestar Animal	PROCESO GESTIÓN TECNOLÓGICA		 BOGOTITA	INSTITUTO DISTRITAL DE PROTECCIÓN Y BIENESTAR ANIMAL
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE			
	Código: PA04-PR06-G01	Versión: 1.0		

- **Tareas (Tasks):** Representan el trabajo técnico específico necesario para completar una Historia de Usuario (ej: "Crear el diagrama de secuencia", "Implementar el endpoint de la API", "Crear pruebas unitarias").
- **Bugs:** Registran defectos o errores encontrados en el código. Los bugs pueden ser gestionados al mismo nivel que las Historias de Usuario o como trabajo subordinado a ellas, según la configuración del equipo.

4.2.2.2. Flujo de Trabajo (Workflow)

El estado de los Work Items (especialmente las Historias de Usuario) debe ser actualizado constantemente en el tablero para reflejar su progreso real. El flujo de trabajo estándar para una Historia de Usuario es:

- **New:** La historia ha sido creada y está en el backlog, pendiente de ser priorizada.
- **Active:** El equipo ha comenzado a trabajar en la historia.
- **Resolved:** El desarrollo y las pruebas unitarias han sido completados. El código está listo para ser fusionado a la rama develop tras la aprobación del Pull Request.
- **Closed:** La historia ha sido verificada por el equipo de calidad o el Product Owner y se considera completada.

4.2.3. Uso de Repositorio de Código Fuente con Azure Repos

El control de versiones es una práctica esencial para gestionar los cambios en el código fuente a lo largo del tiempo. En nuestra organización, **Azure Repos** será la herramienta centralizada para alojar los repositorios Git, permitiendo la colaboración, el seguimiento de cambios y la integración con el resto de los servicios de Azure DevOps.

4.2.3.1. Estructura de Proyectos y Repositorios

La estructura organizativa en Azure DevOps sigue una jerarquía que facilita la gestión de múltiples aplicaciones y equipos:

- **Organización:** Es el nivel más alto y generalmente corresponde al nombre de la entidad o empresa.
- **Proyecto:** Un proyecto en Azure DevOps es un contenedor para un producto de software. Por ejemplo, para un sistema de información misional como "SIPYBA", se crearía un único proyecto llamado "SIPYBA".
- **Repositorio:** Dentro de un proyecto, se pueden crear múltiples repositorios Git. La práctica recomendada es tener un repositorio por cada aplicación o microservicio que compone el sistema. Por ejemplo, dentro del proyecto "SIPYBA", podrían existir repositorios como "codigo-inscripcion", "gestion-pagos", etc.

Esta estructura permite a los equipos centrarse en un objetivo, producto o proceso, organizando los repositorios de manera lógica dentro de un proyecto contenedor.

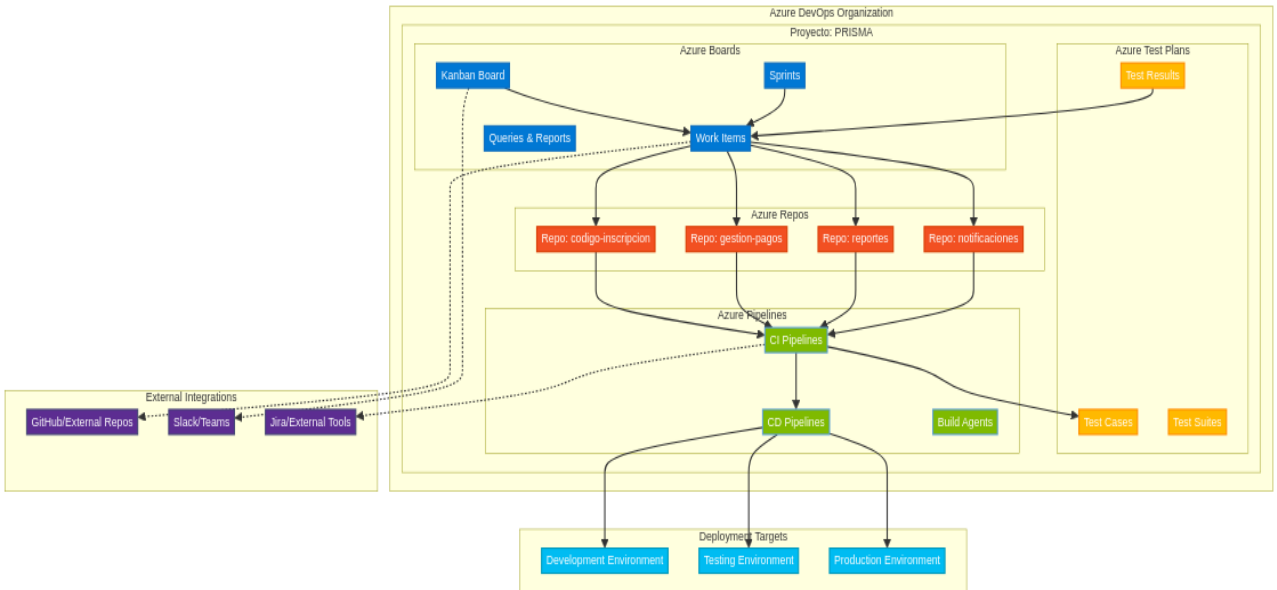


Ilustración 2 - Estructura organizativa en Azure DevOps

4.2.3.2. Modelo de Ramificación: GitFlow

Para asegurar un proceso de desarrollo ordenado y predecible, se adoptará el modelo de ramificación **GitFlow**. Este modelo define un conjunto de ramas con roles específicos y un flujo de trabajo claro para la integración de código.

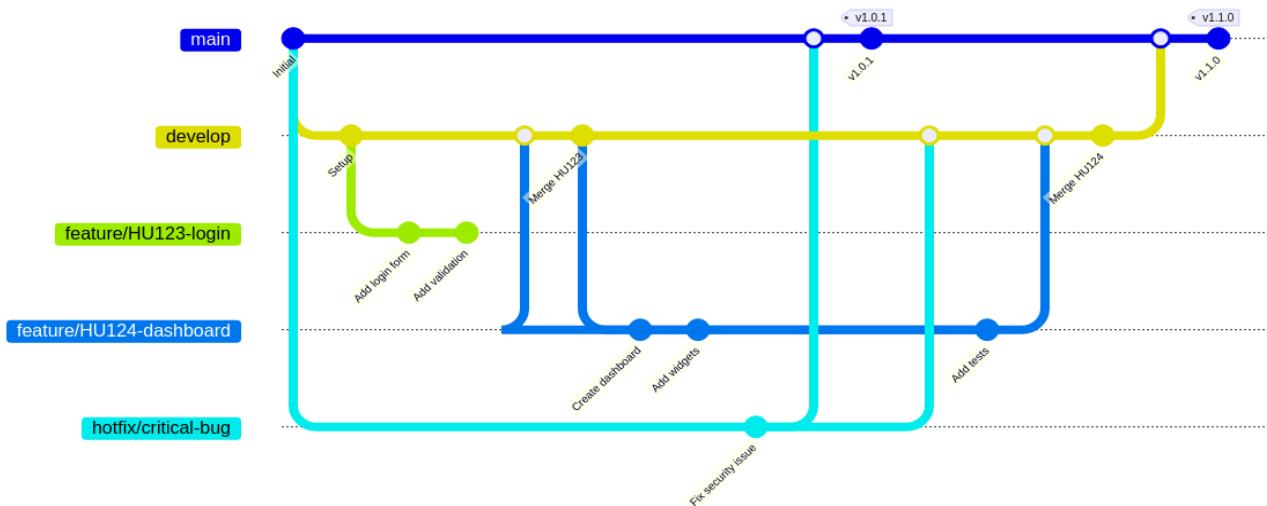




Ilustración 3 - Modelo de ramificación GitFlow

Ramas Principales

Se definen dos ramas principales que tienen un ciclo de vida infinito:

 ALCALDÍA MAYOR DE BOGOTÁ D.C. AMBIENTE <small>Instituto Distrital de Protección y Bienestar Animal</small>	PROCESO GESTIÓN TECNOLÓGICA		
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

- **main:** Esta es la rama principal que refleja el estado actual del código en **producción**. El código en main debe ser siempre estable y desplegable. Todo commit en main debe ser una nueva versión de producción y debe estar etiquetado (taggeado) con un número de versión (ej: v1.0.1).
- **develop:** Esta es la rama de **integración** para el desarrollo activo. Contiene los últimos cambios de desarrollo entregados y consolidados para el próximo lanzamiento. Cuando el código en develop alcanza un punto estable y está listo para ser lanzado, se fusiona en main.

Ramas de Soporte

Son ramas auxiliares con un ciclo de vida limitado, creadas para tareas específicas y que eventualmente son eliminadas.

- **Ramas de Característica (Feature Branches):**

- **Propósito:** Para desarrollar nuevas características o historias de usuario.
- **Origen:** Se crean a partir de la rama develop.
- **Integración:** Se fusionan de nuevo en develop a través de un **Pull Request**.
- **Nomenclatura:** Deben seguir la convención feature/HUXXX-descripcion-corta o users/nombre.apellido/HUXXX para identificar claramente el trabajo y el responsable.

- **Ramas de Corrección Urgente (Hotfix Branches):**

- **Propósito:** Para corregir errores críticos detectados en producción.
- **Origen:** Se crean a partir de la rama main.
- **Integración:** Una vez finalizada la corrección, la rama de hotfix debe ser fusionada tanto en main (para desplegar la corrección a producción) como en develop (para asegurar que la corrección esté en el próximo lanzamiento).
- **Nomenclatura:** Deben seguir la convención hotfix/TICKET-XXX-descripcion-corta.

4.2.3.3. Ciclo de Desarrollo y Pull Requests



El flujo de trabajo diario para un desarrollador se centra en la creación y fusión de ramas de características. Este proceso garantiza que la rama develop se mantenga limpia y que todo el código nuevo sea revisado antes de su integración.

El ciclo es el siguiente:

1. **Sincronizar el Repositorio Local:** Antes de iniciar cualquier trabajo, asegúrate de tener la última versión de la rama develop del repositorio remoto.

git checkout develop

git pull origin develop

 ALCALDÍA MAYOR DE BOGOTÁ D.C. AMBIENTE Instituto Distrital de Protección y Bienestar Animal	PROCESO GESTIÓN TECNOLÓGICA		 BOGOTITA	INSTITUTO DISTRITAL DE PROTECCIÓN Y BIENESTAR ANIMAL
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE			
	Código: PA04-PR06-G01	Versión: 1.0		

2. **Crear una Rama de Característica:** Crea una nueva rama a partir de develop para tu historia de usuario, siguiendo la convención de nomenclatura establecida.

```
git checkout -b feature/HU456-login-con-ss0
```

3. **Vincular el Trabajo a Azure Boards:** Es **mandatorio** que cada rama y sus commits estén vinculados al Work Item correspondiente en Azure Boards. Esto se logra incluyendo `#{ID_DEL_WORK_ITEM}` en el mensaje del commit.

```
git commit -m "Implementa la lógica de autenticación SSO #{ID_DEL_WORK_ITEM}"
```

4. **Desarrollar y Hacer Commits:** Realiza los cambios necesarios en el código, creando commits pequeños y atómicos que representen unidades lógicas de trabajo.

5. **Publicar la Rama:** Una vez que el desarrollo esté listo para ser revisado, publica tu rama en el repositorio remoto de Azure Repos.

```
git push -u origin feature/HU456-login-con-ss0
```

6. **Crear un Pull Request (PR):** En la interfaz web de Azure DevOps, crea un nuevo Pull Request para fusionar tu rama de característica en develop. El PR es una solicitud formal para revisar e integrar tu código.

- **Título y Descripción:** El título del PR debe ser claro y conciso. La descripción debe explicar qué cambios se hicieron, por qué y cómo pueden ser probados. Debe hacer referencia al Work Item que resuelve.
- **Revisores (Reviewers):** Asigna al menos a un miembro del equipo (preferiblemente el líder técnico o un par con experiencia) para que revise el código.
- **Work Items Vinculados:** Asegúrate de que el PR esté vinculado al Work Item correcto de Azure Boards. Esto automatizará la trazabilidad.

7. **Revisión de Código y Políticas de Rama:** El revisor examinará el código en busca de errores, inconsistencias y oportunidades de mejora. Además, Azure DevOps ejecutará automáticamente las **Políticas de Rama** configuradas, que pueden incluir:

- **Compilación Exitosa (CI Build):** El código debe compilar sin errores.
- **Pruebas Unitarias Aprobadas:** Todas las pruebas unitarias deben pasar.
- **Sin Conflictos de Fusión:** La rama no debe tener conflictos con develop.
- **Número Mínimo de Aprobadores:** El PR debe ser aprobado por el número requerido de revisores.

8. **Completar el Pull Request:** Una vez que el PR es aprobado y todas las políticas se cumplen, el creador del PR puede completarlo. Al completar, se recomienda seleccionar la opción **"Squash commit"** para combinar todos los commits de la rama en uno solo en develop, manteniendo el historial de la rama principal limpio y legible. La rama de característica debe ser eliminada automáticamente después de la fusión.

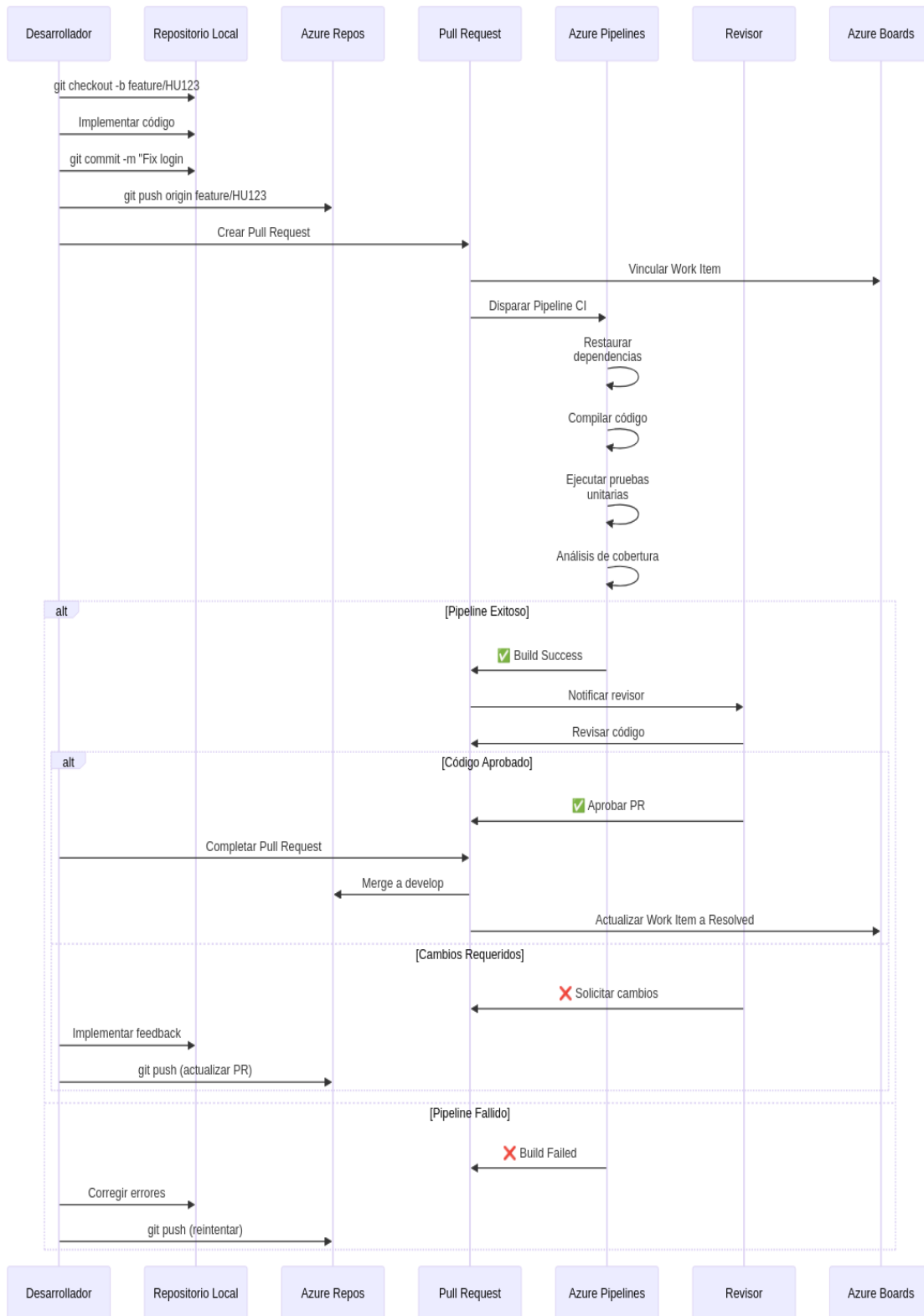




Ilustración 4 - Ciclo de Desarrollo y Pull Requests

	PROCESO GESTIÓN TECNOLÓGICA		
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

4.2.4. Integración Continua con Azure Pipelines

La **Integración Continua (CI)** es una práctica de desarrollo que consiste en automatizar la compilación y prueba del código cada vez que un miembro del equipo confirma cambios en el control de versiones. **Azure Pipelines** es el servicio que nos permite implementar CI/CD de manera robusta y escalable.

4.2.4.1. Pipeline de Integración Continua

Se debe configurar un pipeline de CI para cada repositorio. Este pipeline se definirá en un archivo `azure-pipelines.yml` en la raíz del repositorio, permitiendo que la definición del pipeline sea versionada junto con el código.

El pipeline de CI se activará automáticamente cada vez que se cree o actualice un **Pull Request** dirigido a la rama `develop` o `main`. Sus responsabilidades principales son:

1. **Restaurar Dependencias:** Descargar todas las librerías y paquetes necesarios para el proyecto.
2. **Compilar el Código:** Verificar que el código fuente se compila correctamente sin errores.
3. **Ejecutar Pruebas Unitarias:** Correr la suite completa de pruebas unitarias para asegurar que los nuevos cambios no han introducido regresiones.
4. **Publicar Resultados:** Publicar los resultados de las pruebas y la cobertura de código para que puedan ser revisados directamente en la vista del Pull Request.

Un pipeline de CI exitoso es un requisito indispensable para poder completar un Pull Request, asegurando así la integridad y calidad de las ramas principales.



4.2.5. Pruebas Unitarias

Con el fin de verificar que el desarrollo realizado para una historia de usuario cumple con el objetivo y la calidad esperados es mandatorio realizar pruebas unitarias. Una prueba unitaria permite probar partes del código de forma aislada y rápida, garantizando que las funcionalidades implementadas tengan bajo acoplamiento, alta cohesión y sean fácilmente mantenibles, escalables y reutilizables.

La implementación de pruebas unitarias es una responsabilidad del desarrollador y deben ser entregadas como parte del mismo Pull Request que contiene el código de la funcionalidad. Como se mencionó en la sección de **Azure Pipelines**, estas pruebas se ejecutarán automáticamente como parte del proceso de integración continua, y un fallo en cualquiera de ellas impedirá la fusión del código a la rama `develop`.

4.2.6. Lineamientos para Desarrollo de Código Seguro

La seguridad es una parte integral del desarrollo de software. Todos los desarrollos deben adherirse a las mejores prácticas de codificación segura para minimizar las vulnerabilidades. Se deben tener como base estándares del mercado como el **OWASP Application Security Verification Standard (ASVS)**.



 ALCALDÍA MAYOR DE BOGOTÁ D.C. AMBIENTE <small>Instituto Distrital de Protección y Bienestar Animal</small>	PROCESO GESTIÓN TECNOLÓGICA		 INSTITUTO DISTRITAL DE PROTECCIÓN Y BIENESTAR ANIMAL
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

Los siguientes lineamientos son de obligatorio cumplimiento:

- **Validación de Entradas:** Nunca confíes en los datos provenientes del cliente. Valida, sanea y escapa toda entrada del usuario para prevenir ataques de inyección (SQL, XSS, etc.).
- **Nombres Descriptivos:** Emplea nombres descriptivos en la declaración de variables, métodos y clases. El código debe ser lo más auto-explicativo posible.
- **Evitar Variables Globales:** El uso de variables globales debe ser minimizado, ya que pueden ser modificadas de forma inesperada, introduciendo errores difíciles de rastrear y potenciales brechas de seguridad.
- **Inicializar Siempre las Variables:** Asegúrate de que todas las variables tengan un valor inicial definido para evitar comportamientos indefinidos.
- **Principio de Menor Privilegio:** El código debe ejecutarse con los mínimos privilegios necesarios para realizar su función.
- **Gestión de Secretos:** Nunca almacenes secretos (contraseñas, claves de API, strings de conexión) directamente en el código fuente. Utiliza servicios como **Azure Key Vault** para gestionar y acceder a estos secretos de forma segura.
- **Logs de Auditoría:** La aplicación deberá generar y almacenar un log de auditoría sobre las tablas y transacciones críticas, que permita consultar como mínimo: ID de usuario, fecha, hora, dirección IP, tabla modificada y acción ejecutada (creación, modificación, borrado).

4.2.7. Eventos de SCRUM en el IDPYBA

Evento	Duración	Propósito
Sprint	2 a 3 semanas	Un período de tiempo fijo durante el cual se crea un incremento de producto "Terminado" y potencialmente entregable.
Sprint Planning (Planificación del Sprint)	4 a 6 horas (para un Sprint de 3 semanas)	Planificar el trabajo a realizar durante el Sprint. Se define el objetivo del Sprint y se seleccionan los elementos del Product Backlog que se incluirán en el Sprint Backlog.
Daily Scrum (Scrum Diario)	15 minutos	Sincronizar las actividades del equipo y planificar el trabajo para las próximas 24 horas. Se responden tres preguntas: ¿Qué hice ayer? ¿Qué haré hoy? ¿Qué impedimentos tengo?
Sprint Review (Revisión del Sprint)	2 a 3 horas (para un Sprint de 3 semanas)	Inspeccionar el incremento y adaptar el Product Backlog si es necesario. El equipo de desarrollo presenta el trabajo realizado a los stakeholders.
Sprint Retrospective (Retrospectiva del Sprint)	1.5 a 2 horas (para un Sprint de 3 semanas)	Inspeccionar el último Sprint en cuanto a personas, relaciones, procesos y herramientas, y crear un plan de mejoras para el próximo Sprint.

	PROCESO GESTIÓN TECNOLÓGICA		
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

4.2.8. Referencias



- [Microsoft Learn: Git workflow in Azure Repos](#)
- [Microsoft Learn: Adopt a Git branching strategy](#)
- [Microsoft Learn: About work items and work item types](#)
- [Microsoft Learn: CI/CD baseline architecture with Azure Pipelines](#)
- [A successful Git branching model by Vincent Driessen](#)

5. FORMATOS Y DOCUMENTOS ANEXOS

No.	Código	Nombre	Característica Físico/Digital
1.	PA04-PR06-G01	Metodología para el desarrollo de software	Digital
2.	PA04-PR04-F01	Solicitud de Sistemas de Información	Digital
3.	PA04-PR06-F06	Acta de Refinamiento y Aceptación	Digital
4.	PA04-PR06-F07	Certificación de Pruebas de Desarrollo	Digital
5.	PA04-PR06-F08	Especificación Historias de Usuario	Digital

6. NORMATIVIDAD ASOCIADA

TIPO DE NORMA	NÚMERO DE IDENTIFICACIÓN	TÍTULO DEL DOCUMENTO	CAPÍTULOS O ARTÍCULOS	FECHA EXPEDICIÓN (DD/MM/AAAA)
Ley	1581	Por la cual se dictan disposiciones generales para la protección de datos personales.	Todos	17/10/2012
Decreto	1078	Por medio del cual se expide el Decreto Único Reglamentario del Sector de Tecnologías de la Información y las Comunicaciones.	Título 9. Políticas y Lineamientos de Tecnologías de la Información	26/05/2015
Resolución	500	Por el cual se establecen los lineamientos y estándares para la estrategia de seguridad digital y se adopta el Modelo de Seguridad y Privacidad como habilitador de la Política de Gobierno Digital.	Todos	10/03/2021
Decreto	767	Por el cual se establecen los lineamientos generales de la Política de Gobierno Digital y se subroga el Capítulo 1 del Título 9 de la Parte 2 del Libro 2 del Decreto 1078 de 2015, Decreto Único Reglamentario del Sector de Tecnologías de la Información y las Comunicaciones	Todos	16/05/2022
Resolución	2277	Por la cual se actualiza el Anexo 1 de la Resolución 500 de 2021 y se derogan otras disposiciones relacionadas con la materia	Todos	03/06/2025

	PROCESO GESTIÓN TECNOLÓGICA		
	METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE		
	Código: PA04-PR06-G01	Versión: 1.0	

CONTROL DE CAMBIOS

NO. DE ACTA DE APROBACIÓN	FECHA	VERSIÓN	DESCRIPCIÓN
El número del acta de aprobación lo asigna el profesional de la OAP	/03/2026	1.0	Creación y adopción documento como guía donde se define la metodología de desarrollo de software que se aplicará en el IDPYBA.