MANUAL DE ESPECIFICACION DE REPOSITORIO GITHUB

ALEJANDRO HERNANDEZ

DANIEL VELOZA

FICHA 3147238

SENA



Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

SENA – SOFT 2025 – DESARROLLO LIBRE

21/10/2025

**INTRODUCCIÓN PRAGMA**

El presente documento define los lineamientos de trabajo colaborativo, control de versiones y manejo del repositorio GitHub del proyecto   
Su objetivo es estandarizar la forma en que los integrantes del equipo gestionan el código, documentan los avances y estructuran las ramas de desarrollo, garantizando trazabilidad, organización y coherencia durante todo el ciclo de desarrollo.

Se emplea GitHub como plataforma de control de versiones y el flujo de trabajo GitFlow, lo que permite trabajar de forma paralela y ordenada, evitando conflictos y manteniendo una integración continua del proyecto.

**OBJETIVOS DEL MANUAL**

Establecer las normas para el uso del repositorio.

* Definir la estructura de ramas según GitFlow.
* Estandarizar los mensajes de commit y nombres de ramas.
* Facilitar el trabajo colaborativo entre los integrantes.
* Asegurar la trazabilidad del código fuente y documentación técnica.

**FLUJO DE TRABAJO (GITFLOW)**

El proyecto se desarrollará bajo la metodología GitFlow, la cual divide el trabajo en ramas con propósitos específicos.

🔸 Ramas principales

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rama | Descripción | Uso |  |  |  |
| main | Contiene la versión estable del proyecto, lista para entrega o despliegue. | Solo recibe merges desde release o hotfix. |  |  |  |
| develop | Rama base del desarrollo. | A partir de ella se crean las ramas feature/. |  |  |  |

🔸 Ramas secundarias

|  |  |  |
| --- | --- | --- |
| Rama | Descripción | Uso |
| feature/ | Se crean desde develop para desarrollar funcionalidades nuevas. | Cada integrante trabaja en una rama propia. |
| release/ | Se crean desde develop para pruebas y ajustes previos a la entrega. | Al finalizar, se fusiona con main. |
| hotfix/ | Corrige errores críticos detectados en main. | Luego se sincroniza con develop. |

**CONVENCIONES PARA NOMBRES DE RAMAS**

Para mantener consistencia, todas las ramas deben nombrarse en minúsculas, separadas por guiones (-) y con prefijo según su tipo:

|  |  |  |
| --- | --- | --- |
| Tipo | Ejemplo | Descripción |
| feature/ | feature/compra-tiquete | Nueva funcionalidad. |
| feature/ | feature/crud-vuelos | CRUD o módulo del admin. |
| release/ | release/v1.0 | Versión de entrega estable. |
| hotfix/ | hotfix/corregir-pago | Corrección urgente en producción. |

Regla General: los nombres de ramas deben describir claramente su propósito.  
Ejemplo correcto: feature/validar-asientos  
Ejemplo incorrecto: branch1 o nueva.

**Estructura del repositorio**

senasoft2025-myticket/

│

── backend/

│ ── controllers/

│ ── models/

│ ── routes/

│ ── views/ (EJS)

│ ── config/

│ └── app.js

│

── frontend/

│ |── assets/

│ |── css/

│ |── js/

│ ── img/

│

── docs/

│ |── 1\_ANALISIS\_REQUERIMIENTOS.docx

│ |── 2\_DISENO\_SOLUCION.docx

│ |── 3\_DICCIONARIO\_DATOS.docx

│ |── 4\_STACK\_TECNOLOGICO.docx

│ ── diagramas/

│

── .gitignore

── README.md

|── package.json

📁 docs/ contiene toda la documentación técnica del proyecto.  
📁 backend/ aloja la lógica MVC (Node.js).  
📁 frontend/ guarda los recursos visuales estáticos.

**CONVENCIONES PARA MENSAJES DE COMMIT**

Los mensajes de commit deben ser breves, descriptivos y en tiempo presente.  
Formato recomendado:

[Tipo]: Descripción breve

Ejemplos:

* feat: agrega validación de fechas en reserva
* fix: corrige error en cálculo de valor total
* docs: añade diagrama UML al documento
* style: mejora diseño de formulario de pago

Tipos más comunes que se puedan presentar en el desarrollo de nuestro proyecto:

|  |  |
| --- | --- |
| **Tipo** | **Uso** |
| feat | Nueva funcionalidad |
| fix | Corrección de errores |
| refactor | Mejora del código sin cambiar funcionalidad |
| docs | Cambios en la documentación |
| style | Cambios de formato o diseño |
| test | Pruebas unitarias o funcionales |

**FLUJO DE TRABAJO COLABORATIVO**

Paso 1: Crear la rama de desarrollo

Cada integrante crea su rama a partir de develop.  
Ejemplo:

feature/compra-tiquete

feature/crud-vuelos

Paso 2: Realizar commits frecuentes

Se deben realizar commits por cada avance importante, con mensajes descriptivos.

Paso 3: Subir cambios al repositorio remoto

En GitHub Desktop → *Push origin* para mantener sincronizado el trabajo.

Paso 4: Crear un Pull Request (PR)

Cuando una funcionalidad esté terminada:

1. Hacer *Push origin*.
2. Crear un Pull Request desde feature/... hacia develop.
3. Otro integrante revisa y aprueba antes de hacer el merge.

Paso 5: Versiones de entrega

Antes de cada entrega:

* Crear rama release/vX.X desde develop.
* Probar todo el sistema.
* Si es estable, hacer merge hacia main.

**ROLES DEL EQUIPO EN EL REPOSITORIO**

|  |  |  |
| --- | --- | --- |
| Integrante | Rol | Responsabilidades |
| Daniel Veloza | Analista / Arquitecto | Estructura del repo, documentación técnica, diseño UML y control de versiones. |
| Alejandro Hernandez | Desarrollador Full Stack | Desarrollo de módulos backend/frontend y despliegue local. |

**BUENAS PRACTICAS**

* No subir archivos de dependencias (node\_modules/, .env, etc.).
* Incluir un .gitignore correctamente configurado.
* Actualizar siempre develop antes de crear nuevas ramas.
* No hacer commits directos a main.
* Nombrar commits y ramas en minúsculas, sin tildes ni espacios.
* Revisar y aprobar Pull Requests antes de fusionar.
* Sincronizar el repositorio local con frecuencia.
* Mantener actualizada la carpeta /docs con los últimos documentos del proyecto.

**CONCLUSIÓN**

Este manual establece las normas para el uso y administración del repositorio GitHub del proyecto My Ticket, garantizando un flujo de trabajo colaborativo, ordenado y eficiente.  
El cumplimiento de estas directrices permitirá mantener la integridad del código, la trazabilidad de los cambios y la correcta organización del proyecto durante la competencia SENASoft 2025.