Informe de Ambientes

Proyecto: Experiencias Significativas

ÍNDICE

1. Introducción

Este informe describe el manejo de ambientes implementado en el proyecto Experiencias Significativas. La separación por ambientes permite mantener un flujo organizado, garantizando que cada cambio pase por distintas etapas de validación antes de ser desplegado.

2. Objetivos del documento

- Describir los ambientes utilizados durante el desarrollo.
- Explicar el flujo de trabajo aplicado en GitHub.
- Documentar el proceso aplicado en frontend, backend y móvil.
- Presentar ejemplos reales con comandos y ramas creadas.
- Proveer una guía clara para futuros desarrolladores del proyecto.

1. **Estructura de Ambientes en GitHub**

3.1 ¿Qué es un Ambiente de control de versions?

Un ambiente es un espacio lógico que representa una etapa del software dentro del ciclo de vida del Desarrollo. Permite separar Código experimental de Código estable.

3.2 Ambiente utilizados en Experiencias Significativas

Ambiente | Propósito

dev | Desarrollo diario y nuevas funcionalidades.

qa | Validación funcional y pruebas internas.

staging | Simulación de producción antes del despliegue.

main | Versión final estable del Proyecto.

2. **Flujo de trabajo por ambientes**

4.1 Ambiente dev

Primer entorno de desarrollo donde se crean y prueban nuevas funcionalidades.

Flujo estándar:

- git pull origin dev
- git switch dev
- git switch -c HU-01-dev
- realizar Desarrollo
- crear MR de HU-01-dev → dev
- Nota: El desarrollo se cierra con la rama dev update.

4.2 Ambiente qa

Ambiente para pruebas de calidad, donde se valida que las funcionalidades que vienen de dev cumplen lo requerido.

Flujo estándar:

- git pull origin qa
- git switch qa
- git switch -c HU-01-qa
- realizar desarrollo QA
- crear MR de HU-01-qa → qa
- Nota: El desarrollo se cierra con la rama qa update.

4.3 Ambiente staging

Ambiente previo a producción. Aquí se valida la integración completa del sistema.

Flujo estándar:

- git pull origin staging
- git switch staging
- git switch -c HU-01-staging
- realizar desarrollo staging
- crear MR de HU-01-staging → staging
- Nota: El desarrollo se cierra con la rama staging update.

4.4 Ambiente main

Ambiente principal donde solo se almacenan releases aprobados. No se trabaja directamente sobre esta rama.

Flujo estándar:

- git pull origin main
- git switch main
- git switch -c release.1.1
  HU-01-release.1.1
  HU-02-release.1.1
- crear MR release.1.1 → main

5. Uso de ambientes por repositorio

- Frontend Web – Implementación completa y correcta.
- Backend – Implementación completa y correcta.
- Móvil – Implementación completa y correcta.

6. Problemas encontrados y soluciones implementadas

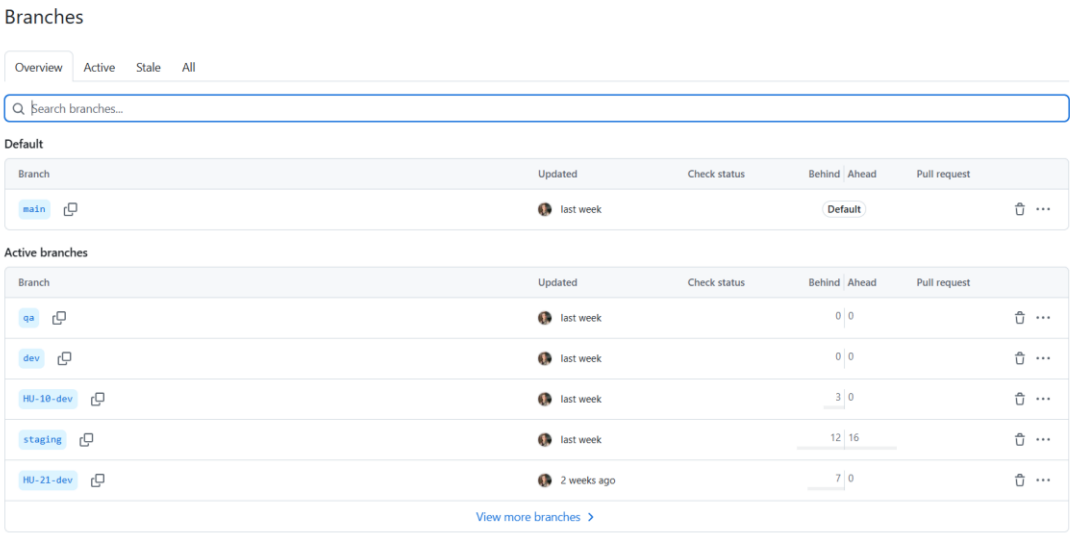
- Conflictos entre ramas → solución: orden en commits.
- Fallos de control de versiones en móvil → establecer estándares.
- Dependencias rotas en backend → unificación de versiones.
- Muchas ramas abiertas → políticas estrictas de cierre.

7. Ejemplos con capturas

7.1 Ambientes principales configurados en GitHub.

Aqui se muestran las cuatro ramas base del flujo de desarollo:

- **main**: versión estable y publicada (producción).
- **dev**: ambiente destinado al desarrollo activo de funcionalidades.
- **qa**: ambiente donde se realizan pruebas internas de validación.
- **staging**: ambiente de preproducción donde se simula el despliegue final.



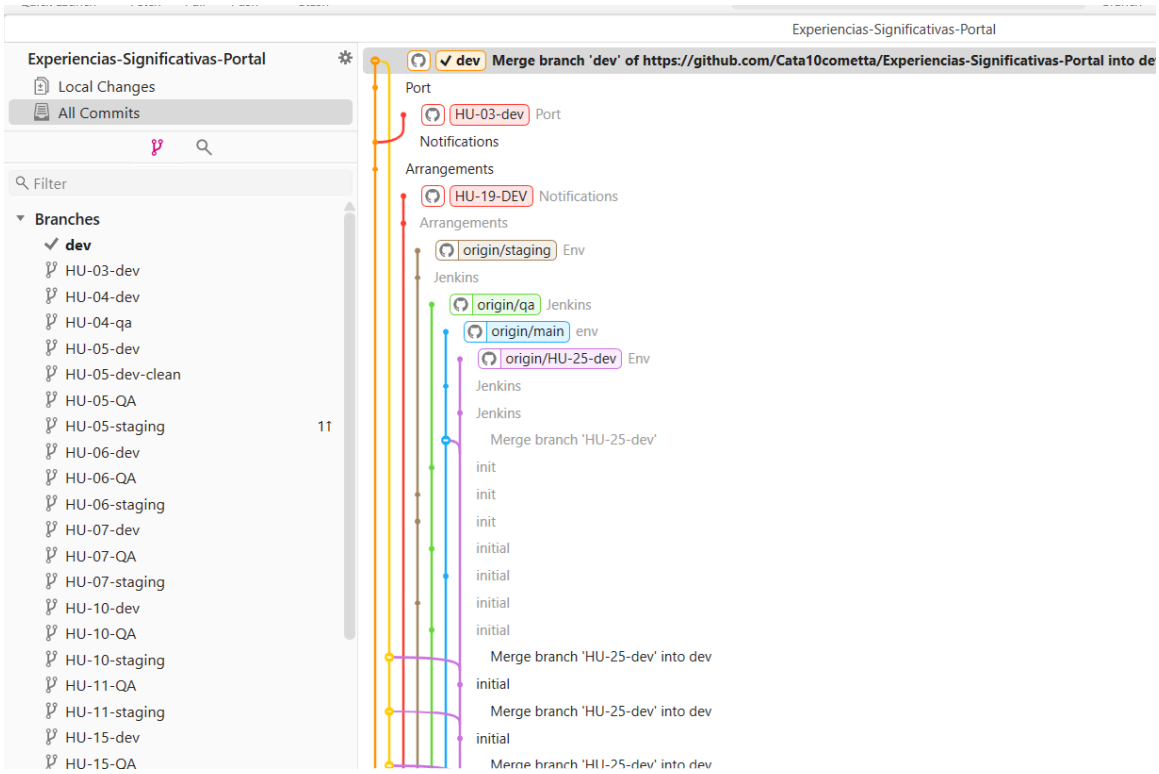
También se visualizan ramas individuales de Historia de Usuario, como HU-10-dev, lo que demuestra la aplicación correcta del flujo GitFlow adaptado al Proyecto.

7.2 Vista de ramas y flujo de commits del Proyecto

En esta imagen se observa la estructura completa de las ramas utilizadas en el proyecto *Experiencias Significativas*, incluyendo las ramas por Historia de Usuario en sus diferentes ambientes (*dev*, *qa*, *staging*).

Esta vista permite visualizar :

- Las ramas locales HU-XX-dev, HU-XX-qa y HU-XX-staging.
- La línea de tiempo de los commits y los merges hacia *dev*.
- El historial de cambios por ambiente.



Además, este es el espacio donde se realizan los ***cherry-picks*** para traer commits específicos desde una rama hacia otra.

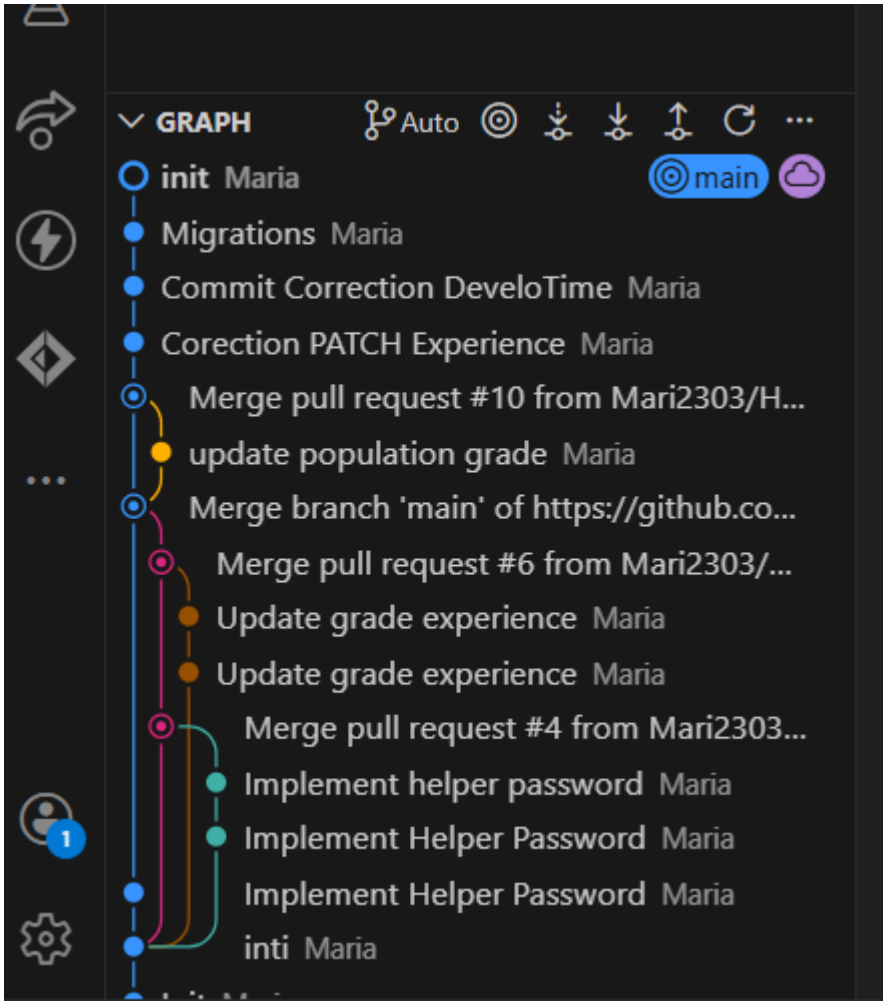
7.3 Visualización del gráfico de commits (GRAPH) que muestra el historial de ramas y movimientos del proyecto.

En esta imagen se observa la representación gráfica del flujo de trabajo Git utilizado en el proyecto *Experiencias Significativas*.

El GRAPH permite visualizar:

- La línea de tiempo de todos los commits realizados.

- La separación entre ramas como *dev*, *qa*, *staging* y *main*.
- Las ramas individuales creadas para cada Historia de Usuario (HU-XX-dev, HU-XX-qa, HU-XX-staging).
- Los merges entre ramas y el recorrido que siguen las funcionalidades.
- Los puntos específicos donde se realizan **cherry-picks**, permitiendo traer commits puntuales desde otras ramas sin necesidad de hacer un merge completo.



Esta vista es fundamental para entender cómo evoluciona el proyecto y cómo se integran los cambios entre los diferentes ambientes, asegurando trazabilidad y orden en el ciclo de desarrollo.

8. Conclusiones

- El manejo de ambientes permitió mantener un flujo organizado.
- El frontend fue el repositorio con implementación completa.
- Backend y móvil aplicaron el flujo parcialmente.
- El esquema garantiza calidad, orden y trazabilidad.