Flujo Colaborativo en GitHub

Área de Desarrollo Móvil Web y Escritorio
Centro De Electricidad Y Automatización Industrial, SENA
3067594: Análisis y Desarrollo de Software
Cómite Evaluador
5 de julio del 2026.

Cali, Valle del Cauca

Tabla de Contenido

1. Introducción General

El proyecto TAM se desarrolló de manera simultánea en tres frentes tecnológicos: una aplicación móvil, una aplicación web y una aplicación de escritorio, cada una gestionada por un equipo de trabajo independiente dentro de un mismo proceso formativo de Análisis y Desarrollo de Software (ADSO) del SENA. Debido a esta organización, cada área mantuvo su propio repositorio en GitHub y adoptó una dinámica particular de trabajo colaborativo para gestionar el código fuente.

El presente informe consolida la evidencia del flujo colaborativo aplicado en las tres áreas del proyecto (Móvil, Web y Escritorio), documentando el uso de Git y GitHub como sistema de control de versiones, la forma en que se registraron y organizaron los cambios mediante commits, y las prácticas de integración y coordinación de equipo empleadas en cada caso.

A diferencia de un informe individual por área, este documento permite comparar las distintas metodologías de trabajo adoptadas por cada equipo, identificar buenas prácticas y detectar oportunidades de mejora comunes a todo el proyecto.

2. Objetivo del Informe

Documentar y analizar el flujo de trabajo colaborativo desarrollado en GitHub por los equipos de Móvil, Web y Escritorio del proyecto TAM, evidenciando el uso del control de versiones, la trazabilidad de los cambios mediante el historial de commits y las dinámicas de integración y comunicación adoptadas por cada equipo durante el desarrollo del software.

3. Área Móvil — Repositorio “tam-app”

El equipo de Desarrollo Móvil gestionó su código fuente en el repositorio de GitHub “tam-app”, aplicando un modelo de ramas por funcionalidad (feature branches) integradas a la rama principal mediante Pull Requests. A continuación se documenta este flujo de trabajo con base en la evidencia disponible del repositorio.

3.1 GitHub como sistema de control de versiones durante el proyecto

Durante el desarrollo del proyecto móvil “tam-app”, GitHub se utilizó como el repositorio central donde se almacenó y versionó todo el código fuente de la aplicación. Esto se evidencia en la estructura de carpetas del proyecto visible en la página principal del repositorio, la cual incluye directorios propios de una aplicación móvil desarrollada en un framework tipo React Native/Expo, tales como app, assets, components, constants, context, services y types, además de un módulo independiente TAM_API destinado a la conexión con el backend.

Cada uno de estos directorios muestra la fecha y el mensaje del último commit que lo modificó, lo cual confirma que el control de versiones no se limitó a un respaldo estático del código, sino que acompañó de forma continua la evolución del proyecto: desde el commit inicial que estableció la estructura base (“Primer commit - estructura inicial TAM-Movil”, hace 4 meses) hasta los últimos cambios funcionales incorporados dos días antes de la captura.

El repositorio registra un total de 67 commits y 32 ramas, cifras que reflejan un proyecto con un ciclo de desarrollo sostenido en el tiempo y con múltiples líneas de trabajo paralelas, características propias de un flujo de control de versiones aplicado de manera constante y no ocasional.

3.2 ¿Qué es un commit y cuál fue su propósito en el proyecto?

Un commit puede entenderse como una "fotografía" del estado del código en un momento específico, acompañada de un mensaje que describe qué se modificó y por qué. En el proyecto "tam-app" los commits cumplieron el propósito de documentar el avance funcional de la aplicación de forma incremental, con mensajes descriptivos y precisos, por ejemplo:

- **"Primer commit - estructura inicial TAM-Movil":** Registra la creación de la estructura base del proyecto móvil.
- **"feat: conexión de login, registro y verificación con API backend":** Documenta la integración del módulo de autenticación con el backend.
- **"feat: integrar imágenes remotas, ProductImage y actualización del comparador":** Documenta la incorporación de imágenes remotas y ajustes al módulo comparador de productos.

3.3 Flujo de trabajo con Git aplicado en el proyecto

La estructura del repositorio, el historial de commits y el listado de ramas permiten reconstruir el flujo de trabajo de Git que se siguió durante el desarrollo:

```
git init / git clone
```

Utilizados para crear e inicializar el repositorio local del proyecto y para obtener copias locales completas del repositorio remoto, respectivamente.

```
git add / git commit
```

Utilizados para preparar los archivos modificados y confirmarlos de forma definitiva. Se identifican identificadores de commit como c0a4f01, 27a92a3, 03dea13 y 68ebfbd, cada uno con un mensaje explicativo.

```
git push / git pull
```

Utilizados para sincronizar los cambios entre el repositorio local y el repositorio remoto en GitHub, especialmente relevante dado que el proyecto maneja 32 ramas simultáneas.

```
git branch
```

Utilizado para crear y gestionar ramas independientes de desarrollo, organizadas por convención de nombres: fix/drawer-sesion, feature/comparacion, feature/carrito, feature/auth, feat/verificacion-con-nodejs, entre otras, hasta completar 32 ramas.

```
git merge
```

La fusión de ramas no se realizó de forma directa por línea de comandos, sino a través de Pull Requests en GitHub, como se evidencia en los commits de fusión "Merge pull request #32...", "#31..." y "#30...", todos integrados hacia la rama main.

3.4 Análisis de los commits visibles en el repositorio

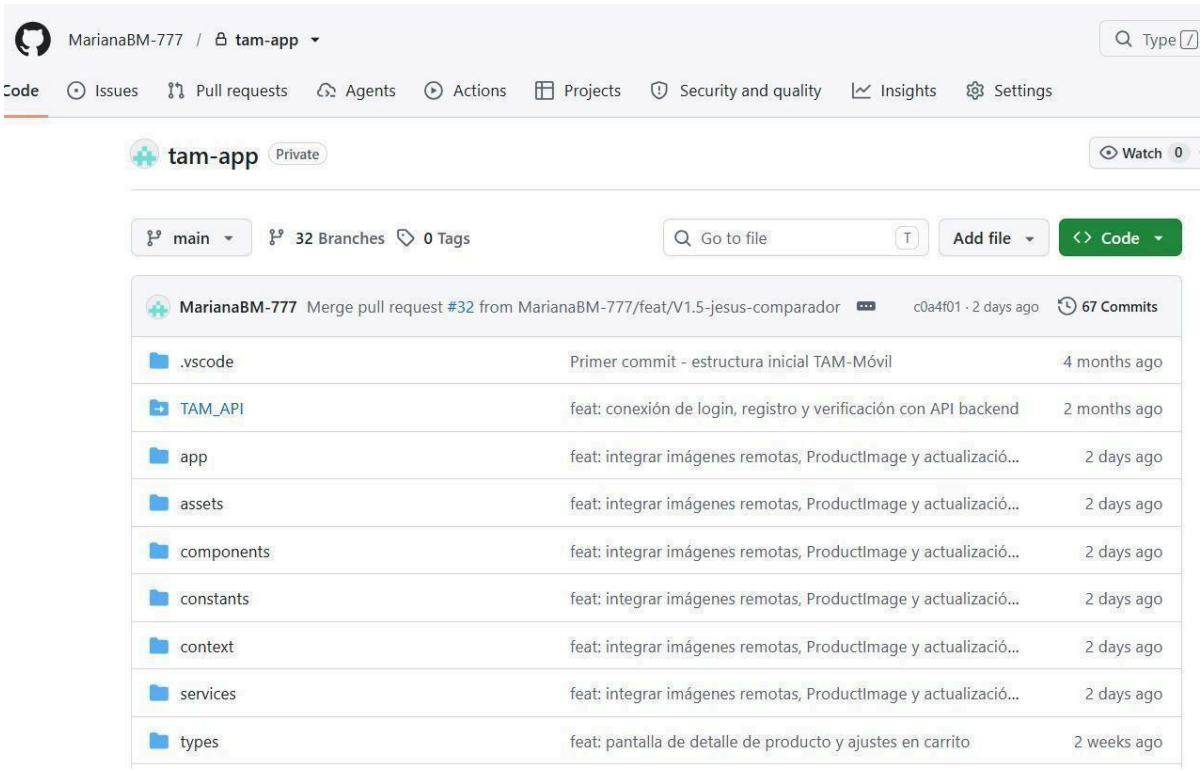
El historial de commits muestra una secuencia ordenada de trabajo en la que se alternan commits de desarrollo funcional con commits de fusión (merge), evidenciando el ciclo: rama de funcionalidad → commits de desarrollo → Pull Request → merge a main. Esto demuestra que el proyecto no se desarrolló directamente sobre la rama principal, sino que cada nueva funcionalidad fue aislada, desarrollada y validada antes de integrarse.

3.5 Pull Requests y evidencia de trabajo colaborativo

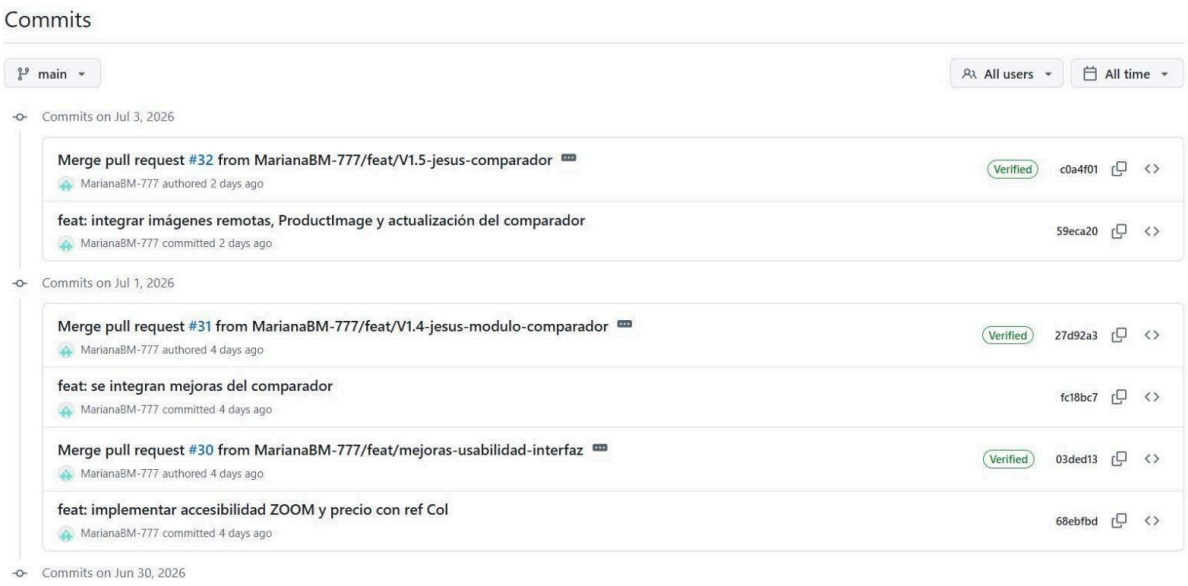
Se identificaron al menos tres Pull Requests fusionadas hacia la rama main: la #30 (mejoras de usabilidad e interfaz), la #31 (módulo comparador V1.4) y la #32 (comparador V1.5 e imágenes remotas). Esto evidencia una metodología de trabajo ordenada basada en ramas independientes y revisión previa a la integración, característica de un flujo de trabajo colaborativo tipo Feature Branch Workflow.

Cabe precisar que, en las capturas analizadas, todas las Pull Requests y commits visibles aparecen asociados al mismo usuario (MarianaBM-777), por lo que no es posible confirmar si el trabajo colaborativo involucró a otros integrantes del equipo o si se trató de una organización personal del flujo mediante ramas.

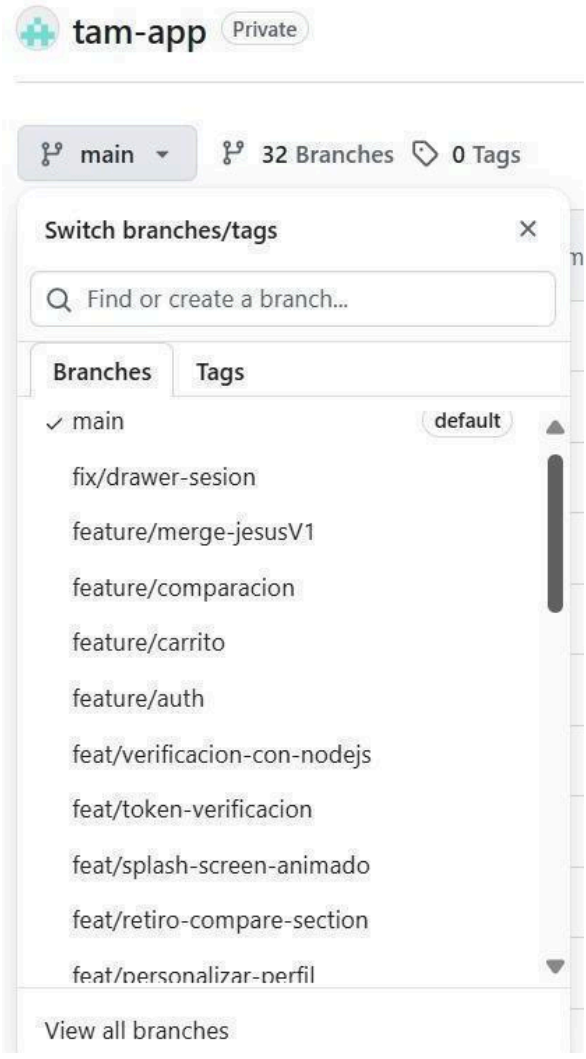
3.6 Evidencias fotográficas — Área Móvil



Área Móvil. Página principal del repositorio tam-app: estructura de carpetas, último commit (Merge PR #32) y 67 commits totales.



Área Móvil. Historial de commits: fusiones (merge) de las Pull Requests #30, #31 y #32 junto con los commits de desarrollo asociados.



Área Móvil. Listado de ramas del repositorio (32 ramas), organizadas por convención de nombres feature/, feat/ y fix/.

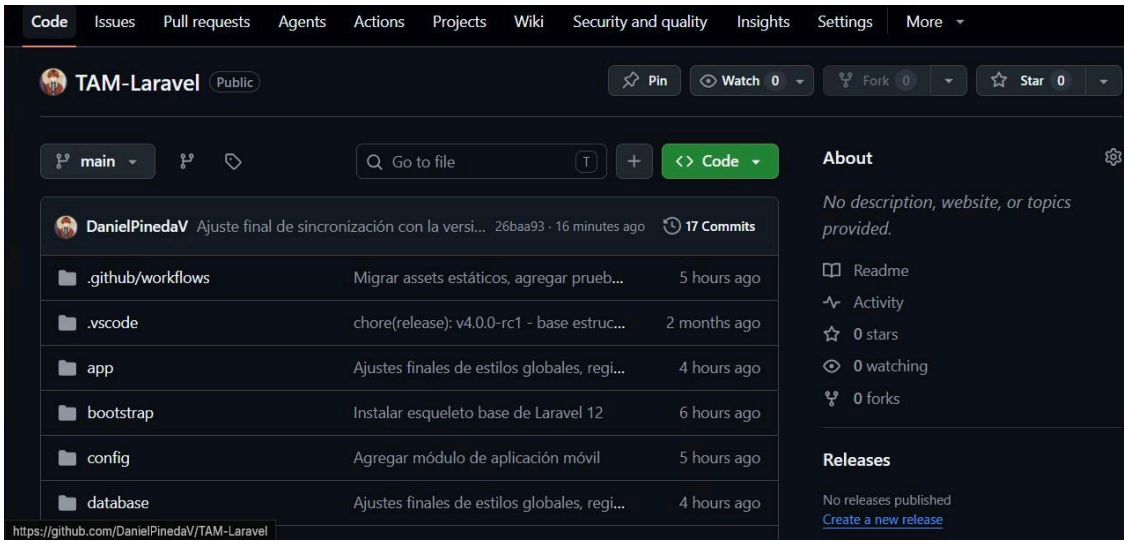
4. Área Web — Repositorio “TAM-Laravel”

El equipo de Desarrollo Web utilizó GitHub como plataforma para el almacenamiento del código fuente y el control de versiones del proyecto, bajo el repositorio “TAM-Laravel”. Mediante esta herramienta se registraron los cambios realizados, permitiendo llevar un seguimiento del avance a través del historial de commits.

Enlace del repositorio: https://github.com/DanielPinedaV/TAM-Laravel

4.1 Repositorio en GitHub

A continuación se presenta el repositorio del proyecto alojado en GitHub, donde se encuentra almacenado el código fuente y la estructura general del desarrollo.



Área Web. Repositorio del proyecto TAM-Laravel en GitHub.

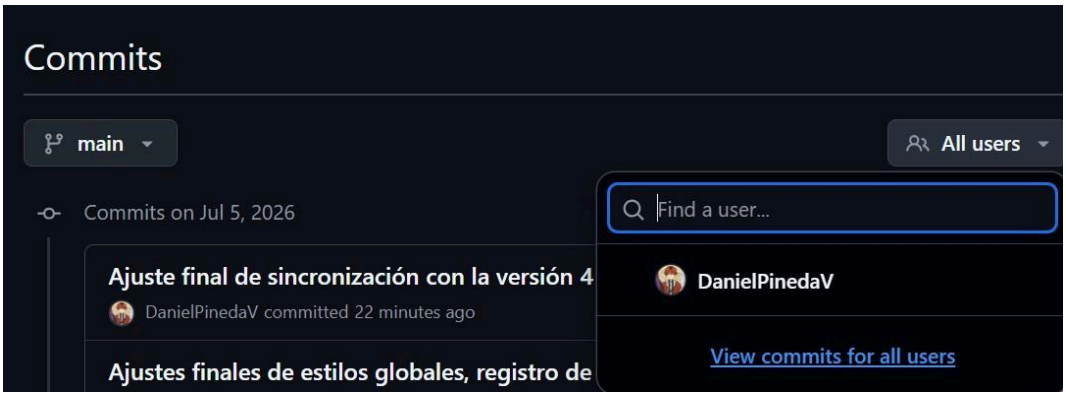
4.2 Historial de commits

El historial de commits permite visualizar la secuencia de cambios realizados durante el desarrollo del proyecto. Cada registro corresponde a una actualización del código o incorporación de nuevas funcionalidades.

Área Web. Historial de commits del proyecto TAM-Laravel.

4.3 Autores de los commits

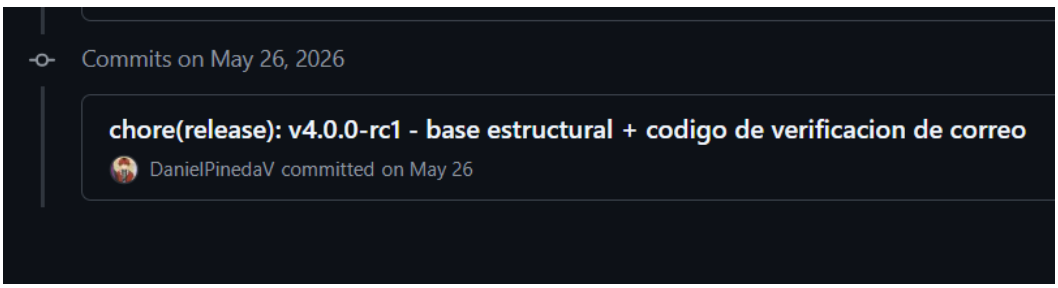
GitHub registra automáticamente el autor asociado a cada commit realizado en el repositorio, permitiendo identificar la cuenta desde la cual se efectuó cada actualización.



Área Web. Autor registrado en los commits del repositorio.

4.4 Fechas de los commits

Cada commit incluye la fecha y hora en que fue realizado, lo que permite llevar un registro cronológico de las actualizaciones efectuadas durante el desarrollo del proyecto.



Área Web. Fechas registradas en el historial de commits.

A diferencia del área Móvil, la evidencia recopilada para el área Web no incluye el uso de ramas (branches) ni Pull Requests, lo que sugiere un flujo de trabajo más simple basado en commits directos sobre la rama principal del repositorio.

5. Área Escritorio — Repositorio “Proyecto-escritorio-TAM”

Para el desarrollo del proyecto de escritorio, el equipo adoptó un modelo de integración centralizada. Debido a restricciones técnicas con herramientas de almacenamiento en la nube utilizadas previamente, se migró el control de versiones a GitHub, bajo el repositorio “Proyecto-escritorio-TAM”.

Enlace del repositorio: https://github.com/AlanJoesen/Proyecto-escritorio-TAM.git

5.1 Definición del flujo colaborativo

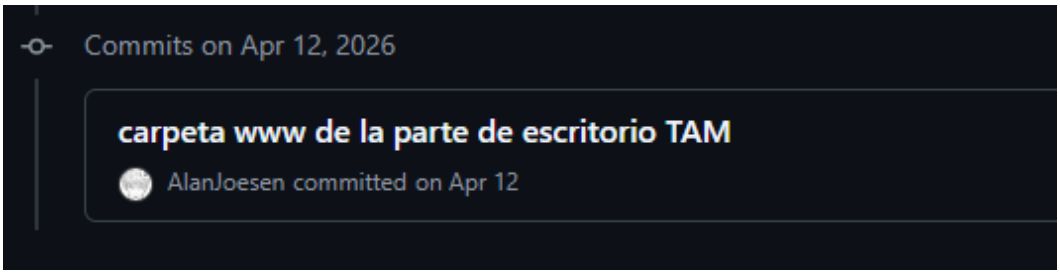
El flujo colaborativo del equipo de escritorio se gestionó de la siguiente manera:

1. Asignación de tareas: las funcionalidades se distribuyeron entre el equipo de desarrollo de escritorio.
2. Entrega al integrador: cada miembro del equipo enviaba sus módulos terminados al líder del proyecto.
3. Revisión del código: el líder realizaba la inspección manual, corrección de conflictos y pruebas de integración locales.
4. Despliegue a producción: una vez validada la estabilidad del código, el líder realizaba el despliegue directo a la rama principal en GitHub.

Este modelo, a diferencia del flujo por ramas y Pull Requests aplicado en el área Móvil, concentra la revisión e integración del código en una sola persona (el líder de proyecto), en lugar de distribuir esa responsabilidad mediante herramientas nativas de revisión de GitHub.

5.2 Historial de commits

Primer commit: se subió por primera vez la carpeta raíz del proyecto (el proyecto ya contaba con la mayoría de sus módulos funcionales).

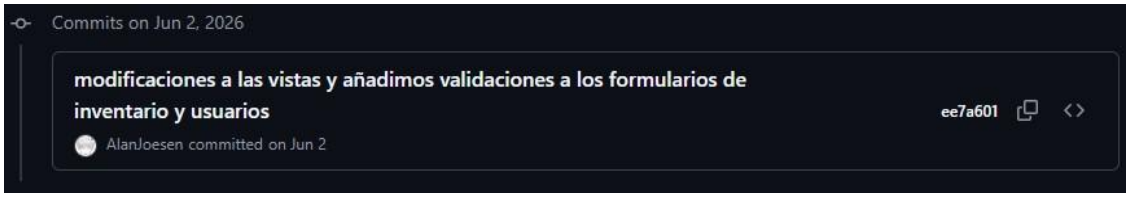


Área Escritorio. Primer commit — carga inicial de la carpeta raíz del proyecto.

Segundo commit: se reorganizó toda la estructura organizacional, aplicando separación por responsabilidades.

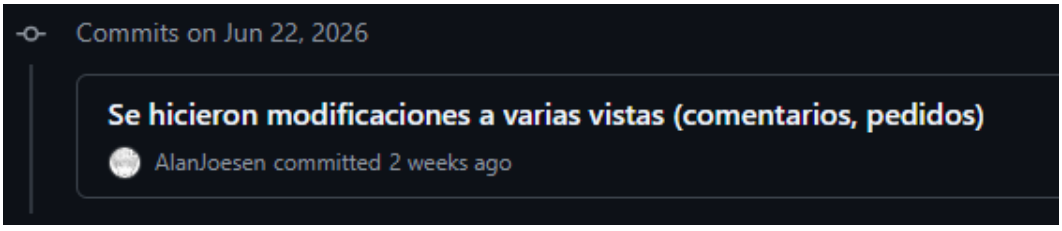
Área Escritorio. Segundo commit — reorganización de la estructura del proyecto.

Tercer commit: se modificaron las vistas y se añadieron validaciones a los formularios de inventario (agregar y editar producto) y de usuarios (crear y editar usuarios).



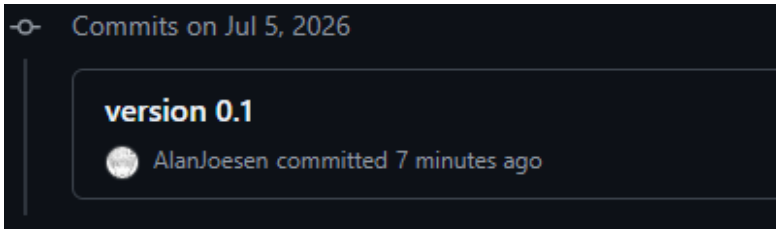
Área Escritorio. Tercer commit — validaciones en formularios de inventario y usuarios.

Cuarto commit: se modificaron y agregaron funcionalidades al módulo de Pedidos (el módulo de Comentarios quedó en fase de pruebas).



Área Escritorio. Cuarto commit — funcionalidades del módulo de Pedidos.

Quinto commit: versión de escritorio funcional con la mayoría de módulos operativos; el módulo de Comentarios quedó temporalmente deshabilitado, previsto para una próxima actualización.

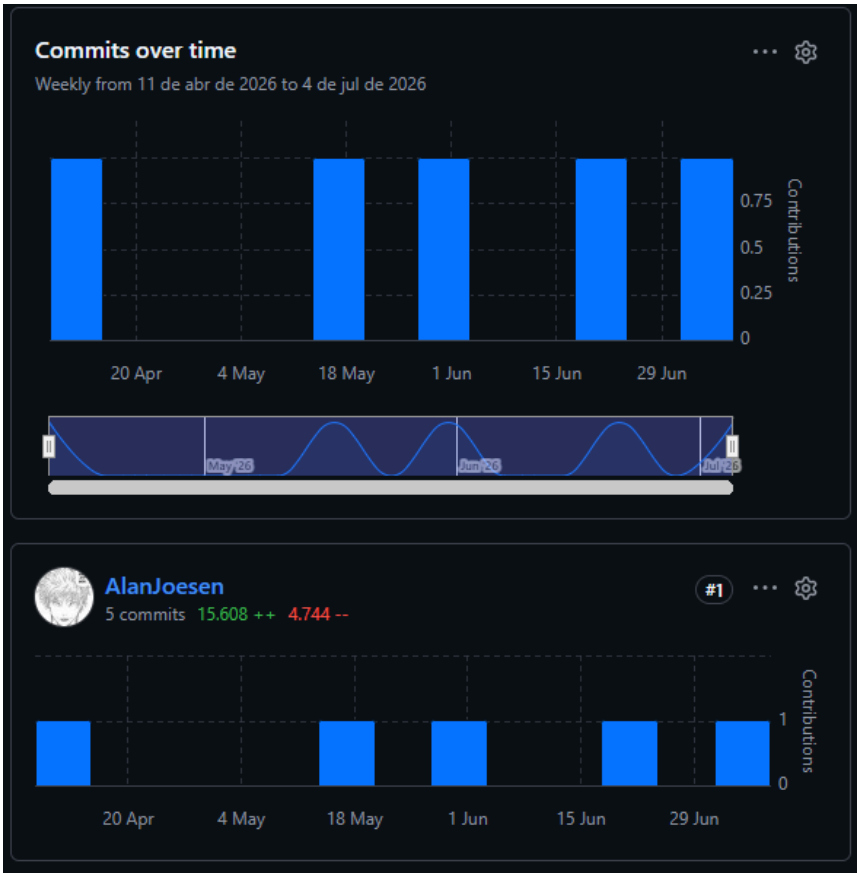


Área Escritorio. Quinto commit — versión funcional consolidada.

Esta versión se catalogó como Versión 0.1, debido a que, aunque cumple con los requisitos y la integración con la API central, el sistema se encuentra en una arquitectura modular escalable, con una posible actualización futura para integrar la funcionalidad de gestión de comentarios.

5.3 Gráfico de contribuciones

A continuación se presenta la gráfica de control obtenida directamente del repositorio central, la cual evidencia un historial de evolución constante desde abril. Las barras representan los ciclos de integración en los que el líder de proyecto unificó el código tras cada fase de asignación de tareas.



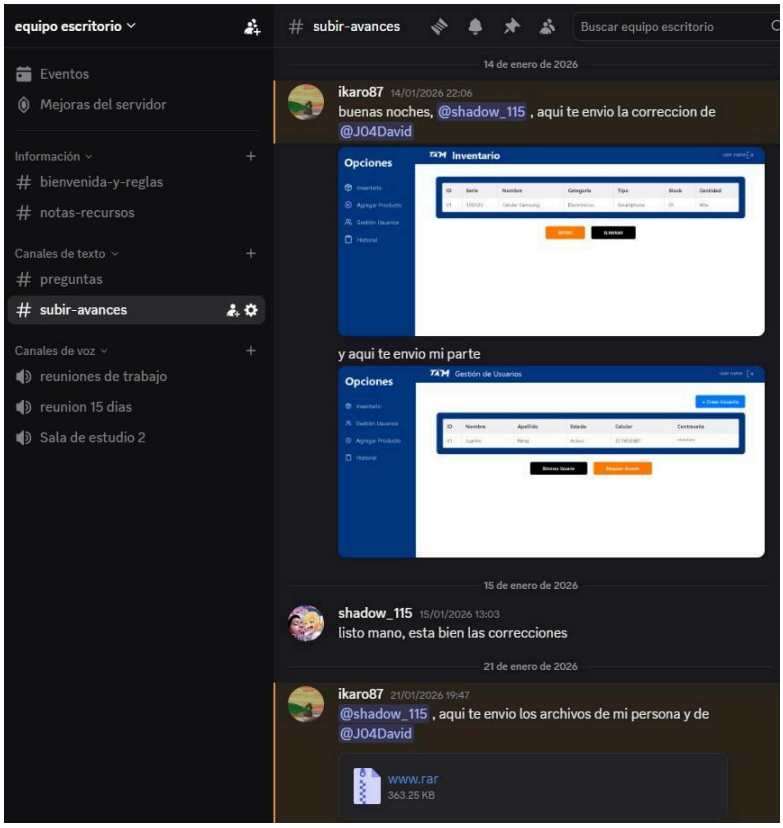
Área Escritorio. Gráfico de contribuciones del repositorio Proyecto-escritorio-TAM.

- **Ciclo inicial (abril):** Se integró el archivo raíz del proyecto con la mayoría de módulos funcionales.
- **Ciclos intermedios (mayo-junio):** Reorganización de la estructura organizacional, implementación de nuevas vistas y funciones, y validaciones en los formularios.

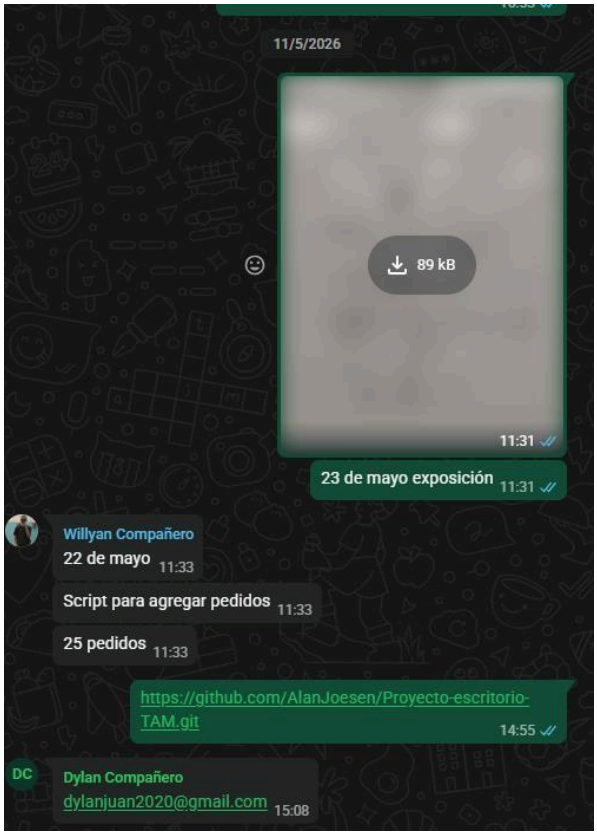
- **Ciclo de cierre (julio):** Consolidación de la versión más completa y estable.

5.4 Evidencias de gestión externa

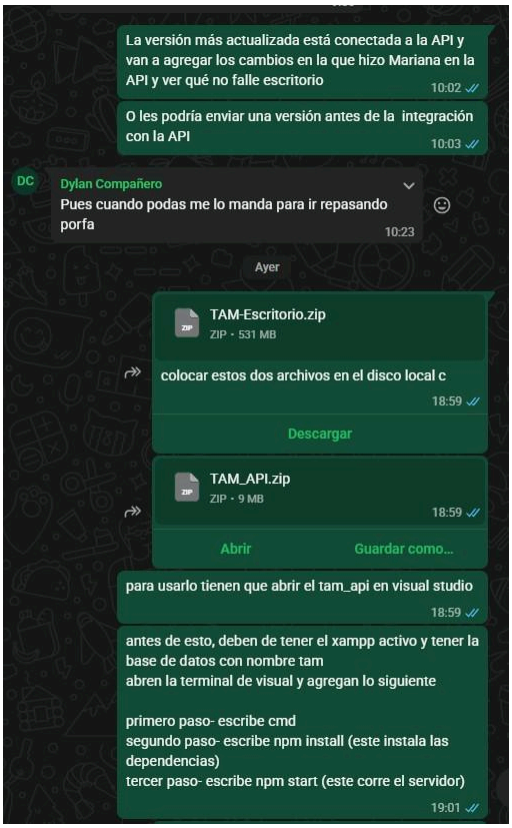
Debido a que el modelo de integración centralizada no se apoya en Pull Requests para la revisión de código, la coordinación entre los integrantes del equipo de escritorio se gestionó a través de canales de comunicación externos a GitHub, como Discord y WhatsApp, según se evidencia a continuación.



Área Escritorio. Evidencia de coordinación del equipo mediante Discord.



Área Escritorio. Evidencia de coordinación del equipo mediante WhatsApp (1).



Área Escritorio. Evidencia de coordinación del equipo mediante WhatsApp (2).

6. Análisis Comparativo de los Flujos de Trabajo

Las tres áreas del proyecto TAM utilizaron GitHub como herramienta de control de versiones, pero con niveles distintos de madurez en la aplicación de prácticas colaborativas de Git:

- **Área Móvil:** Aplicó un flujo de ramas por funcionalidad (feature branches) integradas mediante Pull Requests, con 32 ramas y al menos 3 Pull Requests documentadas. Es el flujo más cercano a las buenas prácticas estándar de trabajo colaborativo en Git (Feature Branch Workflow).
- **Área Web:** Empleó un flujo de commits directos sobre la rama principal, sin evidencia de uso de ramas ni Pull Requests. Es un flujo funcional pero más básico en cuanto a organización del trabajo en paralelo.
- **Área Escritorio:** Adoptó un modelo de integración centralizada, en el que un único integrante (el líder de proyecto) revisa e integra manualmente el código de todo el equipo directamente a la rama principal, apoyándose en canales externos (Discord, WhatsApp) para la coordinación, en lugar de las herramientas nativas de revisión de GitHub.

Esta comparación evidencia que, si bien las tres áreas lograron mantener un control de versiones funcional del código fuente, existe una oportunidad de estandarizar el flujo de trabajo colaborativo entre los equipos, adoptando de forma generalizada el modelo de ramas y Pull Requests aplicado en el área Móvil.

7. Conclusiones Generales

- Las tres áreas del proyecto TAM (Móvil, Web y Escritorio) utilizaron GitHub como sistema de control de versiones a lo largo del desarrollo, manteniendo repositorios independientes para cada frente tecnológico.
- El área Móvil presenta el flujo de trabajo colaborativo más estructurado, con un modelo de ramas por funcionalidad y Pull Requests documentadas, reflejado en 67 commits y 32 ramas.
- El área Web mantuvo un flujo de control de versiones más simple, basado en commits directos, que permitió trazar el avance del proyecto, aunque sin evidencia de segmentación del trabajo mediante ramas.
- El área Escritorio implementó un modelo de integración centralizada liderado por una persona, complementado con canales de comunicación externos (Discord, WhatsApp) para coordinar al equipo, alcanzando una versión funcional (0.1) del sistema.
- En conjunto, el proyecto TAM evidencia el uso efectivo de Git y GitHub como herramientas de control de versiones en sus tres áreas, con oportunidades claras de estandarización del flujo de trabajo colaborativo entre los equipos.

8. Recomendaciones Generales

- **Estandarización del flujo de trabajo:** Adoptar de forma estandarizada, en las áreas Web y Escritorio, un flujo de trabajo basado en ramas por funcionalidad y Pull Requests, similar al aplicado en el área Móvil, para distribuir la revisión de código y reducir la dependencia de un único integrador.

- **Área Móvil:** Complementar la evidencia del área Móvil con capturas de terminal mostrando la ejecución real de los comandos git init, git add, git commit, git push y git pull, y con el detalle de una Pull Request específica (diff y comentarios de revisión).
- **Área Web:** Incorporar evidencia del uso de ramas y Pull Requests si el equipo llega a adoptarlas, así como capturas de la pestaña “Insights > Contributors” para documentar la participación de cada integrante en los commits.
- **Área Escritorio:** Migrar progresivamente del modelo de integración centralizada hacia un flujo de ramas y Pull Requests dentro de GitHub, lo que permitiría documentar la revisión de código directamente en la plataforma y reducir la dependencia de canales de comunicación externos como Discord o WhatsApp.
- **General:** Documentar, en futuras versiones de este informe, la participación individual de cada integrante del equipo en los commits de las tres áreas, con el fin de evidenciar de forma más clara el trabajo colaborativo distribuido entre los miembros de cada equipo.