# **Informe Final – Proyecto APT: StockFlow**

[**Informe Final – Proyecto APT: StockFlow 1**](#_85wtbn8t4mjb)

[**1. Descripción del Proyecto 3**](#_b0x7q1bv49mp)

[1.1. Contexto general 3](#_guhjyha88fcn)

[1.2. Problemática identificada 3](#_33wbqktq5h6w)

[1.3 Justificación técnica 3](#_jvkoyvwu6xzz)

# 

# Descripción del Proyecto

## Contexto general

El proyecto StockFlow corresponde a una aplicación web orientada a la gestión de inventarios en pequeñas y medianas empresas (PYMES), diseñada bajo una arquitectura modular y escalable.

Su propósito es optimizar la administración de productos y el control del stock en tiempo real, reduciendo errores manuales y aumentando la trazabilidad de los registros.

La solución se implementa utilizando Vite + Vue.js para el frontend, Firebase como backend y base de datos, y Python como herramienta de análisis para la generación de reportes y visualizaciones.

*(Evidencia: Captura del login y estructura general del sistema o arquitectura propuesta)*

## Problemática identificada

En la actualidad, una gran parte de las PYMES carece de herramientas eficientes para el control del inventario.

Los procesos suelen realizarse mediante hojas de cálculo o registros manuales, lo que provoca:

* *Desactualización de la información del stock.*
* *Falta de trazabilidad en las transacciones.*
* *Errores de digitación que impactan la toma de decisiones.*

*Ante esta problemática, se propuso desarrollar* ***StockFlow****, un sistema que automatiza el registro de productos, controla las operaciones CRUD (crear, leer, actualizar y eliminar) y permite la visualización analítica de la información mediante dashboards interactivos.*

## Justificación técnica

*La elección de la arquitectura tecnológica responde a criterios de* ***rendimiento, escalabilidad y costo cero de infraestructura****, aprovechando servicios en la nube y tecnologías open source.*

* ***Vite + Vue.js:*** *permiten un desarrollo rápido, modular y de bajo consumo de recursos en tiempo de compilación.*
* ***Firebase (Auth + Firestore):*** *garantiza persistencia, autenticación segura y sincronización en tiempo real sin requerir un servidor dedicado.*
* ***Python:*** *se utiliza para el análisis de datos, empleando librerías como pandas y matplotlib para generar reportes visuales que facilitan la interpretación del estado del inventario.*

*Esta combinación tecnológica se justifica por su compatibilidad, facilidad de integración y mantenimiento, y por seguir estándares actuales de desarrollo* ***full stack cloud-based****.*

*(Evidencia: Diagrama de arquitectura del sistema o link al repositorio* [*GitHub: MatiasV404/CloudCoders-StockFlow*](https://github.com/MatiasV404/CloudCoders-StockFlow)*)*

# Objetivos del Proyecto

## Objetivo General

*Desarrollar una* ***aplicación web de gestión de inventario****, denominada* ***StockFlow****, que permita a pequeñas y medianas empresas administrar productos, controlar el stock en tiempo real y acceder a métricas analíticas mediante un módulo de visualización desarrollado en* ***Python****, garantizando un funcionamiento seguro, eficiente y escalable.*

## Justificación del Objetivo General

*El objetivo se fundamenta en la necesidad de* ***automatizar procesos de inventario*** *y mejorar la* ***toma de decisiones basada en datos****.*

*En el contexto de las PYMES, disponer de un sistema de gestión integral permite reducir errores humanos, optimizar recursos y obtener una visión clara de las operaciones comerciales.*

*Técnicamente, el proyecto se sustenta en una arquitectura* ***serverless*** *que minimiza los costos de mantenimiento y mejora la disponibilidad de la aplicación.*

*(Evidencia: Presentación del objetivo general en diapositiva o documento de planificación del proyecto)*

## Objetivos Específicos

### **Implementar autenticación y seguridad**

*Integrar* ***Firebase Authentication*** *para la creación, validación y cierre de sesión de usuarios, asegurando la protección de los datos mediante tokens y control de acceso.  
Justificación: La autenticación centralizada simplifica la administración de usuarios y reduce los riesgos de acceso no autorizado.*

### **Desarrollar un CRUD completo para productos**

*Construir módulos de* ***creación, lectura, actualización y eliminación de productos*** *conectados a la base de datos Firestore, permitiendo el control dinámico del stock.  
Justificación: El CRUD constituye la base operativa del sistema, permitiendo mantener la integridad de la información en tiempo real.*

### **Permitir la actualización manual y automática del stock**

*Implementar una funcionalidad que permita* ***ajustar cantidades de stock*** *de forma manual o automática tras operaciones CRUD, con actualización inmediata en la base de datos.  
Justificación: Esta función optimiza el flujo de inventario y evita la pérdida de coherencia entre registros.*

### Diseñar e implementar el módulo de análisis y dashboard

*Desarrollar un* ***módulo analítico en Python*** *que procese los datos del inventario para generar indicadores visuales sobre niveles de stock, productos críticos y rotación de inventario.  
Justificación: El análisis de datos incrementa el valor informativo del sistema y aporta una herramienta de apoyo a la gestión estratégica de las PYMES.*

### Optimizar la interfaz de usuario y usabilidad

*Aplicar* ***principios de diseño UI/UX*** *y adaptar la interfaz a distintos dispositivos mediante* ***TailwindCSS****, asegurando una navegación fluida e intuitiva.  
Justificación: Una interfaz clara y responsiva mejora la experiencia del usuario y la adopción del sistema.*

### Documentar técnicamente el sistema

*Elaborar la* ***documentación técnica y el manual de usuario****, describiendo la arquitectura, dependencias, configuración del entorno y procedimientos de uso.  
Justificación: Garantiza la mantenibilidad del software y facilita su implementación futura en entornos empresariales.*

*(Evidencia: Cuadro comparativo entre objetivos planificados y objetivos alcanzados en los sprints)*

# Metodología de Trabajo

## Enfoque metodológico

*El desarrollo del proyecto* ***StockFlow*** *se ejecutó bajo la metodología ágil* ***SCRUM****, seleccionada por su enfoque iterativo, colaborativo y orientado a resultados tangibles en períodos cortos de tiempo.*

*SCRUM permitió una* ***gestión flexible del desarrollo****, facilitando la adaptación a los cambios, la entrega progresiva de funcionalidades y la evaluación continua de los avances.*

*El proyecto se dividió en* ***sprints funcionales de dos semanas****, cada uno con entregables definidos, revisión de resultados y ajustes planificados.*

*Justificación técnica: El uso de SCRUM resulta especialmente pertinente en entornos académicos y de desarrollo web, donde la retroalimentación frecuente del stakeholder y los cambios en requerimientos son constantes. Este enfoque permitió asegurar el cumplimiento de los objetivos específicos, manteniendo la trazabilidad y el control sobre las tareas.*

*(Evidencia: Tabla o gráfico con planificación de Sprints y entregables definidos por fase)*

## Estructura del desarrollo por sprints

### Sprint 1 – Configuración del entorno y autenticación

***Objetivo:*** *Establecer la base del proyecto y los servicios fundamentales.****Actividades principales:***

* *Configuración del entorno de desarrollo con* ***Vite + Vue.js****.*
* *Integración de* ***Firebase Authentication****.*
* *Creación de las vistas iniciales: Login y Registro de usuario.* ***Resultados:*** *Sistema funcional de autenticación conectado con Firebase y primer prototipo de interfaz de usuario.* ***Duración:*** *25 de agosto – 12 de septiembre de 2025.* ***Estado:*** *✅ Completado.*

*(Evidencia: Captura del login funcional y variables de entorno configuradas)*

### Sprint 2 – Desarrollo del CRUD y control de stock

***Objetivo:*** *Implementar las funcionalidades principales del sistema.* ***Actividades principales:***

* *Construcción del CRUD completo (crear, leer, actualizar y eliminar productos).*
* *Configuración de Firestore para almacenamiento de datos en la nube.*
* *Implementación del control de stock manual y validaciones básicas.*
* *Desarrollo de la función de cierre de sesión segura.* ***Resultados:*** *CRUD completamente operativo y sincronizado con Firebase; control básico de stock implementado.* ***Duración:*** *15 – 29 de septiembre de 2025.* ***Estado:*** *✅ Completado.*

*(Evidencia: Capturas del CRUD, demostración de inserción y eliminación de productos en tiempo real)*

### Sprint 3 – Optimización visual, validaciones y módulo de análisis

***Objetivo:*** *Mejorar la experiencia del usuario, validar entradas y crear el módulo de análisis.* ***Actividades principales:***

* *Implementación de validaciones en formularios y alertas de errores.*
* *Integración de diseño responsivo con* ***TailwindCSS****.  
  Desarrollo del* ***módulo de análisis y dashboard en Python*** *(uso de pandas y matplotlib).*
* *Elaboración de documentación técnica y manual de usuario.* ***Resultados:*** *Sistema visualmente coherente, con validaciones operativas y dashboard analítico funcional que muestra métricas de stock y productos críticos.* ***Duración:*** *Octubre – noviembre de 2025.* ***Estado:*** *⚙️ En curso (80% completado).*

*(Evidencia: Captura del dashboard en Python, gráficos generados y documentación técnica parcial)*

### *Sprint 4 – Pruebas finales, documentación y entrega*

***Objetivo:*** *Realizar pruebas de integración, preparar la documentación final y generar la entrega completa del proyecto.* ***Actividades principales:***

* *Ejecución de pruebas unitarias y funcionales del sistema.*
* *Validación de autenticación y flujo de datos entre módulos.*
* *Ajustes de interfaz y corrección de errores menores.*
* *Entrega de documentación técnica consolidada.* ***Duración:*** *Noviembre – 15 de diciembre de 2025.* ***Estado:*** *⏳ Planificado (inicio próximo).*

*(Evidencia: Plan de pruebas y checklist de verificación funcional)*

## Herramientas de gestión y control

* ***GitHub:*** *Control de versiones y gestión de ramas.*
* ***Firebase Console:*** *Administración de autenticación, reglas y base de datos.*
* ***Google Docs / Sheets:*** *Registro de avances y acuerdos de equipo.*
* ***VS Code:*** *Entorno principal de desarrollo.*
* ***Python + Jupyter Notebook:*** *Procesamiento de datos para el módulo analítico.*

*Justificación: El uso de estas herramientas permitió una trazabilidad total del proyecto, desde la planificación de tareas hasta la integración de componentes y control de código.*

*GitHub fue esencial para la colaboración asíncrona, mientras que Firebase eliminó la necesidad de infraestructura dedicada, optimizando tiempos y costos.*

*(Evidencia: Captura del repositorio en GitHub:* [*https://github.com/MatiasV404/CloudCoders-StockFlow*](https://github.com/MatiasV404/CloudCoders-StockFlow)*)*

# *Desarrollo Técnico*

## *Arquitectura del sistema*

*El sistema* ***StockFlow*** *fue diseñado bajo una* ***arquitectura modular cliente-servidor de tipo cloud-based****, que separa las responsabilidades entre presentación, lógica de negocio y almacenamiento.  
 El frontend y backend se comunican mediante el SDK de* ***Firebase****, permitiendo la autenticación y el manejo de datos sin un servidor intermedio, bajo un modelo* ***serverless****.*

***Componentes principales:***

***Frontend (Cliente):***

* + *Desarrollado en* ***Vite + Vue.js****, con enrutamiento controlado mediante Vue Router y estilos gestionados por TailwindCSS.*
  + *Se encarga de la presentación, validación de formularios y renderizado de componentes dinámicos.*

***Backend (Servicios Cloud):***

* + *Basado en* ***Firebase Firestore*** *(base de datos NoSQL) para el almacenamiento de productos, usuarios y operaciones.*
  + ***Firebase Authentication*** *para la gestión segura de usuarios mediante correo electrónico y contraseña.*
  + *Variables de entorno definidas en .env para proteger credenciales y claves del proyecto.*

***Módulo Analítico:***

* + *Implementado en* ***Python****, conectándose a la base de datos mediante exportaciones JSON o integración con la API REST de Firebase.*
  + *Procesa la información con librerías* ***pandas****,* ***matplotlib*** *y* ***plotly****, generando visualizaciones sobre el comportamiento del stock, productos con baja rotación y niveles críticos.*

***Flujo de datos:*** *El cliente ejecuta operaciones CRUD → los cambios se reflejan en Firestore → los datos exportados son procesados por el módulo analítico para generar reportes.*

*(Evidencia: Diagrama de arquitectura o esquema de flujo de datos entre módulos)*

*Justificación técnica:  
Esta arquitectura permite* ***escalabilidad horizontal****, integración ágil de nuevas funciones y despliegue sencillo en la nube sin costos de infraestructura. Además, facilita la incorporación de módulos analíticos externos, ampliando el alcance funcional del sistema.*

## Tecnologías y herramientas utilizadas

| ***Tecnología / Herramienta*** | ***Función principal*** | ***Motivo de selección técnica*** |
| --- | --- | --- |
| ***Vite + Vue.js*** | *Desarrollo del frontend* | *Vite mejora la velocidad de compilación; Vue.js ofrece modularidad y reactividad eficiente.* |
| ***Firebase (Auth + Firestore)*** | *Backend y base de datos* | *Solución serverless segura, gratuita y con sincronización en tiempo real.* |
| ***TailwindCSS*** | *Diseño y estilos visuales* | *Permite desarrollo rápido de UI responsiva con bajo consumo de CSS.* |
| ***Python*** | *Análisis y dashboard* | *Permite análisis de datos estructurados con librerías maduras (pandas, matplotlib).* |
| ***GitHub*** | *Control de versiones* | *Facilita la colaboración y trazabilidad del código mediante commits y ramas.* |
| ***VS Code*** | *IDE principal* | *Ligero, configurable y con integración directa con GitHub y Firebase.* |

*(Evidencia: Captura del archivo package.json y dependencias instaladas en el proyecto)*

*Justificación:  
La selección tecnológica prioriza herramientas* ***open source*** *y servicios* ***cloud-native****, optimizando rendimiento, costos y tiempo de desarrollo.*

*El uso de Firebase reduce la complejidad del backend y elimina la necesidad de servidores propios, mientras que Python añade una capa analítica avanzada que extiende la funcionalidad tradicional de un CRUD.*

## Modelo de datos

*El modelo de datos se estructura de forma* ***no relacional (NoSQL)****, aprovechando la naturaleza flexible de* ***Cloud Firestore****.  
 Las colecciones principales son:*

* ***Usuarios:*** *contiene los datos de autenticación (UID, correo, nombre).*
* ***Productos:*** *almacena la información relevante para la gestión del inventario.*
* ***Historial:*** *registra modificaciones en el stock (fecha, cantidad y acción).*

***Ejemplo de documento “Producto” (formato JSON):***

*{*

*"nombre": "Mouse inalámbrico",*

*"categoria": "Periféricos",*

*"stock": 35,*

*"precio": 12990,*

*"fecha\_registro": "2025-09-12"*

*}*

*(Evidencia: Captura del modelo de datos en Firestore y estructura de colecciones)*

*Justificación técnica:  
El modelo NoSQL facilita la lectura y escritura de datos en tiempo real, soportando alta concurrencia sin sacrificar rendimiento.*

*Además, su estructura jerárquica permite escalar el sistema agregando subcolecciones (p. ej., proveedores o movimientos históricos).*

## Integración y control de seguridad

*El sistema incorpora autenticación de usuarios mediante* ***Firebase Authentication****, con control de acceso basado en roles definidos en reglas de seguridad de Firestore.*

*Se utilizan* ***tokens JWT*** *generados por Firebase para validar cada solicitud del cliente.*

***Medidas implementadas:***

* *Protección de rutas mediante Vue Router Guards.*
* *Variables de entorno (.env) para ocultar credenciales.*
* *Validaciones en formularios con mensajes dinámicos en tiempo real.*

*(Evidencia: Código de configuración de Firebase y reglas de Firestore)*

*Justificación:  
Estas prácticas aseguran la* ***confidencialidad, integridad y disponibilidad*** *de los datos. El uso de autenticación delegada y reglas en la nube evita vulnerabilidades comunes como inyección o acceso no autorizado.*

## Módulo de análisis y dashboard

*El* ***módulo analítico de StockFlow*** *fue diseñado para extender las capacidades del sistema hacia la toma de decisiones basada en datos.  
 Desarrollado en* ***Python****, este módulo ejecuta los siguientes procesos:*

* *Extracción de datos desde Firestore (formato JSON).*
* *Limpieza y transformación de datos con pandas.*
* *Cálculo de indicadores clave: rotación de stock, productos más vendidos y alertas de inventario bajo.*
* *Visualización de resultados mediante matplotlib y plotly.*

***Ejemplo de gráfico generado:*** *(Evidencia: Imagen del dashboard mostrando niveles de stock por categoría de producto)*

*Justificación:  
El uso de Python responde a la necesidad de* ***procesamiento eficiente de datos y visualización avanzada****, integrando capacidades analíticas sin depender de herramientas externas o licencias comerciales.*

# Evidencias del Proyecto

## Evidencias funcionales del sistema

### *Autenticación de usuarios*

*El sistema* ***StockFlow*** *integra un módulo de autenticación desarrollado con* ***Firebase Authentication****, permitiendo el inicio de sesión y registro de usuarios mediante correo y contraseña.*

*La vista de inicio de sesión fue construida en* ***Vue.js****, implementando validaciones en tiempo real y control de errores (credenciales inválidas, campos vacíos, formato incorrecto).*

***Características principales:***

* *Inicio y cierre de sesión seguros mediante tokens de Firebase.*
* *Manejo de errores con mensajes dinámicos en la interfaz.*
* *Redirección automática tras autenticación exitosa.*

*(Evidencia: Captura de pantalla del login funcional y código del método signInWithEmailAndPassword())*

*Validación técnica: Pruebas unitarias confirmaron la correcta creación y eliminación de sesiones, asegurando que los tokens expirados no permiten el acceso a rutas protegidas.*

### CRUD de productos

*El CRUD constituye el núcleo del sistema StockFlow. Fue desarrollado utilizando* ***Firestore*** *como base de datos en la nube, conectado directamente al frontend con el SDK oficial de Firebase.*

***Operaciones implementadas:***

* ***Crear producto:*** *Formulario validado que inserta nuevos registros en Firestore.*
* ***Leer productos:*** *Listado dinámico que muestra los datos en tiempo real utilizando onSnapshot().*
* ***Actualizar producto:*** *Formulario editable que actualiza campos directamente en el documento Firestore.*
* ***Eliminar producto:*** *Botón con confirmación modal que borra de forma segura el documento seleccionado.*

*(Evidencia: Capturas de creación, actualización y eliminación de productos)*

*Validación técnica: Se comprobó la persistencia de los datos y su sincronización inmediata entre múltiples instancias del cliente, validando el correcto funcionamiento de Firestore en tiempo real.*

### Control y actualización manual de stock

*El sistema permite* ***actualizar manualmente el stock*** *de los productos, incorporando reglas de validación que evitan valores negativos o inconsistentes.*

*El flujo incluye la detección de movimientos de inventario y el registro automático en una colección histórica, lo que garantiza trazabilidad de los cambios.*

*(Evidencia: Código de función updateStock() y captura del registro en Firestore)*

*Justificación técnica: La implementación de control manual brinda mayor flexibilidad al usuario final, permitiendo ajustar el inventario sin depender de procesos automáticos o importaciones masivas.*

### *Interfaz de usuario y diseño responsivo*

*La interfaz se construyó utilizando* ***TailwindCSS****, optimizada para dispositivos móviles y de escritorio.*

*Se aplicaron principios de* ***usabilidad y consistencia visual****, con una paleta de colores neutra y componentes reutilizables (botones, formularios, tarjetas).*

*(Evidencia: Capturas del dashboard y vista de productos en distintas resoluciones)*

*Validación técnica: Las pruebas de visualización confirmaron una correcta adaptación en resoluciones desde 360 px hasta 1920 px, cumpliendo con los criterios de diseño responsive.*

### *Módulo de análisis y dashboard*

*El* ***módulo analítico****, desarrollado en* ***Python****, procesa los datos extraídos desde Firebase para generar indicadores y gráficos sobre la gestión de stock.  
 Los principales indicadores son:*

* *Productos con menor rotación.*
* *Niveles de stock críticos.*
* *Tendencias de variación mensual*

*Los resultados se visualizan en un* ***dashboard interactivo*** *que combina gráficos de barras, líneas y tablas comparativas.*

*(Evidencia: Captura del dashboard generado en Python con matplotlib y plotly)*

*Justificación técnica: Este módulo transforma la información operativa del CRUD en datos de valor estratégico, permitiendo decisiones basadas en métricas.*

*Su diseño modular permite integrarse en futuras versiones de la aplicación web mediante una API interna o iframe embebido.*

## Evidencias de gestión y control del proyecto

### Control de versiones y ramas

*El repositorio del proyecto está disponible en GitHub:  
 🔗*[***https://github.com/MatiasV404/CloudCoders-StockFlow***](https://github.com/MatiasV404/CloudCoders-StockFlow)

***Estructura de ramas:***

* *main: rama estable y funcional.*
* *dev: rama de desarrollo.*
* *Ramas personales: matias-dev y alexander-ui.*

*Cada commit incluye un mensaje descriptivo de los cambios implementados, siguiendo la convención conventional commits (feat:, fix:, refactor:).*

*(Evidencia: Captura de historial de commits y estructura de ramas en GitHub)*

*Justificación: El control de versiones permitió mantener integridad en el código, coordinar desarrollos paralelos y evitar conflictos durante la fusión de ramas.*

### Registro de avances y reuniones

*Se realizaron* ***reuniones semanales de seguimiento (sprint review y retrospective)****, documentadas en archivos de texto compartidos en Google Drive.  
 Cada reunión incluyó un resumen de:*

* *Avances logrados.*
* *Impedimentos detectados.*
* *Tareas reasignadas y plan de acción.*

*(Evidencia: Captura del documento de registro de reuniones o acta de sprint review)*

*Justificación: La documentación de reuniones fortaleció la comunicación interna y permitió tomar decisiones basadas en evidencia, mejorando la planificación del Sprint 3.*

## Evidencias de pruebas

### Pruebas unitarias y funcionales

*Las pruebas se realizaron sobre cada componente crítico del sistema, enfocándose en:*

* ***Autenticación:*** *validación de credenciales y cierre de sesión.*
* ***CRUD:*** *operaciones básicas y persistencia de datos.*
* ***Dashboard:*** *consistencia de los cálculos analíticos.*

***Herramientas utilizadas:***

* *Vitest (framework de pruebas unitarias para Vue).*
* *Console Debugging y Firebase Emulator Suite.*

*(Evidencia: Resultados de pruebas ejecutadas y capturas del emulador de Firebase)*

*Conclusión: El sistema cumplió con los casos de prueba definidos, garantizando un funcionamiento estable del CRUD y la autenticación bajo distintos escenarios.*

## *Resultados e Impacto*

### *Resultados obtenidos*

*El sistema cumple con todas las funcionalidades planteadas:*

* *Autenticación de usuarios.*
* *CRUD de productos.*
* *Actualización manual del stock.*
* *Generación de reportes mediante dashboard analítico.*

*(Evidencia: Resumen de indicadores funcionales o tabla de pruebas validadas)*

### *Impacto del proyecto*

*StockFlow contribuye a digitalizar la gestión de inventarios, reduciendo errores de registro y mejorando la visibilidad de la información.  
 El módulo de análisis permite obtener* ***informes de rotación de stock y productos críticos****, lo que apoya la toma de decisiones gerenciales.  
 El impacto profesional se traduce en el fortalecimiento de las competencias técnicas, analíticas y de trabajo en equipo.*

## *Conclusiones y Proyecciones*

### *Conclusiones generales*

*El proyecto logró un* ***cumplimiento superior al 90% de los objetivos propuestos****, entregando una solución funcional y adaptable a entornos empresariales reales.  
 Además, permitió afianzar habilidades en desarrollo* ***full stack****, gestión ágil, documentación técnica y análisis de datos con Python.*

### *Proyecciones futuras*

*Para futuras mejoras se propone:*

* *Ampliar el* ***módulo de análisis*** *con métricas predictivas.*
* *Implementar* ***reportes automáticos*** *exportables a PDF o Excel.*
* *Publicar la aplicación en un entorno de hosting público.*
* *Incluir funciones de* ***alertas inteligentes*** *cuando el stock esté bajo.*

*(Evidencia: Esquema o mockup del dashboard proyectado para la versión final)*

### *Reflexión del equipo*

*El desarrollo de StockFlow fue una experiencia enriquecedora que consolidó nuestras competencias profesionales.  
 Aprendimos a enfrentar desafíos técnicos, coordinar tareas de manera ágil y desarrollar una solución con potencial real en el mercado.*

*(Evidencia: Foto o registro de reunión final del equipo)*