



UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Córdoba

Ingeniería en Sistemas de Información



ÓRBITA

PROYECTO FINAL — 5K3 — 2026

Grupo 14

INTEGRANTES

95119 De La Quintana, Segundo Javier

94490 Cufre, Ángel Hugo

94935 Madrid, Marcelo Miguel

99535 Malnis, María Sol

67369 Romaniello, Lucas Daniel

DOCENTES

Ing. Quinteros, Sergio Ramón

Ing. Santos, Virginia Inés

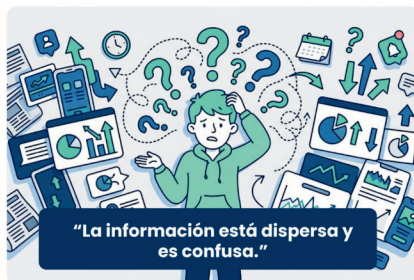


Órbita

Aprendé a invertir simulando, comparando y conversando con Orbi.

LA NECESIDAD · ANTECEDENTES

Invertir por primera vez abruma



OBJETIVO DEL PROYECTO

Órbita reúne todo en un solo lugar: simulá escenarios, compará instrumentos y decidí acompañado por Orbi e IA.

Simulá sin riesgo

Aprendé con Orbi

Decidí con datos e IA

DESCRIPCIÓN DEL TRABAJO

Así funciona Órbita

EL PROCESO

De los datos a tu decisión

01 Miramos el mercado

Precios, noticias y series históricas en tiempo real

02 Ordenamos los datos

Limpieza y preparación automática

03 La IA aprende patrones

Modelo predictivo entrenado con XGBoost y HAR-RV

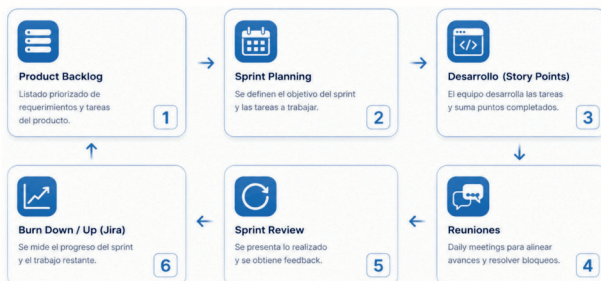
04 Te mostramos la información de manera sencilla

Proyecciones y gráficos fáciles de leer, con información descriptiva de cada indicador desplegado

05 Orbi te lo explica en simple

Asistente conversacional 24/7

MEDIO DE TRABAJO



RESULTADOS OBTENIDOS

- Entorno unificado para análisis y simulaciones concurrentes de inversiones cruzadas, con persistencia y seguimiento.
- Persistencia robusta en PostgreSQL
- Despliegue continuo vía GitLab CI/CD, reduciendo el tiempo de puesta en producción.
- Proyecciones de XGBoost clasificador y otros modelos regresores como HAR-RV para predecir volatilidad.

Tecnologías



Universidad Tecnológica
Nacional
Facultad Regional Córdoba
Ingeniería en Sistemas de Información
Proyecto Final · 2026

DOCENTES
Ing. Sergio Quinteros
Mg. Ing. Virginia Santos

AUTORES
Cufre, Ángel Hugo
cufreangel@gmail.com
De la Quintana, Segundo
segudlg@gmail.com
Madrid, Marcelo Miguel
marcelo.madrid03@gmail.com

Malnis, Maria Sol
solmalnis@gmail.com
Romaniello, Lucas
ldromaniello@gmail.com

CATEGORIZACIÓN
Producto



App Órbita "Plataforma de simulación y análisis financiero con forecasting predictivo"

Cufre, Ángel Hugo - de la Quintana, Segundo Javier - Madrid, Marcelo Miguel - Malnis, Maria Sol - Romaniello, Lucas Daniel

Universidad Tecnológica Nacional, Facultad Regional Córdoba

Abstract

Se presentó el desarrollo de App Órbita, Una plataforma web orientada a mitigar la desventaja de acceso, procesamiento e interpretación de información financiera que enfrentan los inversores principiantes. El sistema permitió simular y comparar y analizar múltiples instrumentos financieros mediante una arquitectura desacoplada multilenguaje, gestionada bajo el marco ágil Scrum. La capa analítica se desarrolló en Python con Flask, NumPy y Pandas, para el forecasting de series temporales con distintos targets financieros. También, se incorporaron indicadores financieros tradicionales, un módulo de análisis de sentimiento y resumen de noticias, modelos dedicados para determinados activos, como XGBoost y HAR-RV, y un asistente conversacional construido sobre la API de OpenAI. Los resultados demostraron la viabilidad técnica de la integración de modelos de series temporales en plataformas web tradicionales, con una tasa de colisiones nula en las pruebas de estrés de la base de datos PostgreSQL y tiempos de despliegue optimizados mediante GitLab CI/CD. Se concluyó que el sistema cumplió los objetivos de simulación, análisis y forecasting financiero, aunque se identificaron limitaciones del modelo XGBoost ante la elección de targets que no presentaban señal y una dependencia de la API externa de OpenAI en términos de latencia y costos.

Palabras Clave

Forecasting financiero, arquitectura multilenguaje, Scrum, XGBoost, HAR-RV, Carry+Roll-Down. Clasificación: Producto.

Introducción

En el escenario macroeconómico contemporáneo, la volatilidad de los mercados financieros plantea un desafío crítico para los jóvenes inversores individuales, tanto para quienes poseen conocimientos financieros básicos como para quienes carecen de ellos. Este grupo dispone de pocas herramientas accesibles para evaluar alternativas de inversión de manera integrada. En consecuencia, la toma de decisiones financieras suele basarse en

análisis fragmentados o en datos históricos estáticos que no reflejan las tendencias proyectadas a mediano y largo plazo. El objetivo de esta investigación es desarrollar una plataforma web integral que permita simular, comparar y analizar múltiples instrumentos financieros como plazos fijos en pesos y dólares, acciones, bonos y rendimiento de billeteras virtuales. El sistema mitiga la asimetría informativa mediante la incorporación de un módulo predictivo avanzado y un asistente automatizado enfocado en el perfil del usuario. El artículo se estructura detallando la metodología ágil aplicada, la arquitectura tecnológica multi-lenguaje implementada, los resultados de rendimiento del modelo predictivo y la discusión sobre las limitaciones y proyecciones del sistema.

Elementos del Trabajo y metodología

Para el desarrollo del proyecto se aplicó el marco de trabajo ágil SCRUM [1], parametrizado mediante reuniones bisemanales de sincronización. El ciclo de vida del software se gestionó a través de historias de usuario, estimadas en Story Points para establecer la velocidad y capacidad del equipo, monitoreadas mediante gráficos Burn Down y Burn Up en la plataforma Jira. La arquitectura del sistema se diseñó bajo un enfoque desacoplado, utilizando contenedores Docker para aislar los servicios. El control de versiones y el despliegue automatizado se estructuraron sobre pipelines de GitLab CI/CD.

La solución se organizó en tres capas principales e independientes: presentación, negocio y análisis. La capa de presentación frontend se construyó sobre Vite con TypeScript, React y Tailwind CSS,

priorizando un renderizado eficiente y responsivo. La capa de negocio backend se implementó en TypeScript bajo una arquitectura de microservicios, encargada de la persistencia de datos en un motor relacional PostgreSQL[2], la gestión de perfiles de usuario, la autenticación de doble factor y el procesamiento de las reglas de negocio de las simulaciones. La capa analítica del módulo científico se desarrolló en Python, utilizando el microframework Flask para exponer servicios REST que consumen las librerías NumPy y Pandas para la manipulación algorítmica de datos financieros extraídos en tiempo real. El núcleo de investigación y desarrollo predictivo se fundamentó en el algoritmo XGBoost[3]. Ante limitaciones identificadas en este modelo, se lo complementó con HAR-RV[4] y con la estrategia de carry+roll-down, incorporando además variables financieras como rendimientos, duraciones modificadas y tasas de fondos federales. De esta manera se entrenaron modelos predictivos optimizados para series temporales no lineales, lo que permitió proyectar indicadores financieros futuros. Adicionalmente, se integró la API de OpenAI[5] para el procesamiento de lenguaje natural del ChatBot de asistencia al inversor.

Resultados

Se construyó un entorno unificado que permitió realizar análisis concurrentes de inversiones, analizar y comparar métricas de distintos activos y exportar el análisis en un momento dado en formato PDF o persistiéndolos en la aplicación para darle seguimiento futuro. Además se integró un resumen de noticias con un sentimiento fácil de leer para que el usuario tenga noción sobre lo que se dice en el mercado sobre los activos que está analizando. La persistencia y el procesamiento en la base de datos relacional PostgreSQL demostraron una tasa de colisiones nula durante las pruebas de estrés de múltiples perfiles simulados. El pipeline de integración y despliegue continuo en

GitLab redujo el tiempo de puesta en producción de nuevos módulos analíticos a rangos óptimos. En el aspecto metodológico, los gráficos Burn Down reflejaron una estabilización de la velocidad del equipo a partir de las iteraciones intermedias, cumpliendo con la planificación de los Story Points establecidos.

En el módulo predictivo, la descomposición carry + roll-down explicó entre el 50% y el 91% del retorno realizado según instrumento y horizonte, con un residuo cuya relación ruido-síñal creció con la duración del instrumento y decreció con el horizonte. HAR-RV superó consistentemente al camino aleatorio sobre las series de rendimiento a 2 y 10 años.

El componente de TradingView[6] renderizó los gráficos vela con baja latencia tras la petición del cliente. Por último, el ChatBot basado en la API de OpenAI resolvió de manera fluida las consultas conceptuales sobre guías de inversión y brokers autorizados.

Discusión

Los resultados obtenidos demuestran que la integración de modelos matemáticos de series temporales dentro de plataformas web tradicionales es técnicamente viable y proporciona una ventaja competitiva medible para el usuario. La arquitectura multi-lenguaje adoptada permite separar las responsabilidades de los componentes del sistema. TypeScript aporta robustez a la gestión transaccional y a los mecanismos de seguridad, mientras que Python facilita el procesamiento numérico y estadístico.

Sin embargo, el desempeño de XGBoost depende en gran medida de la calidad de las variables utilizadas durante el entrenamiento. Su capacidad predictiva puede disminuir cuando las características temporales no representan adecuadamente la estacionalidad, las tendencias o los cambios de régimen presentes en las series financieras. Asimismo, el modelo puede presentar dificultades ante eventos abruptos

o condiciones de mercado que no hayan sido observadas previamente durante el entrenamiento.

Conclusión

Se concluye que el sistema desarrollado cumple con los objetivos de simulación y análisis financiero, mientras que el módulo predictivo se evalúa mediante un protocolo temporal diseñado para reducir la filtración de información y medir su capacidad de generalización fuera de muestra: La evaluación sigue un esquema de validación walk-forward validation con ventana expansiva para evitar filtración de información, los hiper parámetros se reajustan por fold mediante búsqueda en grilla con validación cruzada temporal, la significancia estadística se evalúa mediante el test de Wilcoxon y el test de Diebold-Mariano con corrección HACLas métricas reportadas incluyen AUC, MCC, Brier Score y Log-Loss. Todo esto sumado a que XGBoost incluye en su definición estrategias para evitar overfitting y subfitting.

La modularización de los servicios mediante Docker mitiga los riesgos de acoplamiento y facilita el escalado horizontal de la aplicación.

Como trabajo futuro, se proyecta evaluar la utilización entre NestJS[8] y Express[9] para optimizar el backend, así como la incorporación de bases de datos no relacionales complementarias. Asimismo, se prevé la profundización en la factibilidad de web scraping para la recolección autónoma de datos financieros en tiempo real, independizando parcialmente a la plataforma de las APIs comerciales externas. Finalmente, se plantea ampliar las funcionalidades de la plataforma mediante la incorporación de nuevos objetos de predicción, instrumentos financieros y activos, de acuerdo con las necesidades de los usuarios y la evolución del mercado.

Agradecimientos

Queremos agradecer a todas las personas involucradas en el proyecto Órbita.A nuestras

familias y amigos por el apoyo incondicional a cada miembro del equipo. Agradecer también a nuestros profesores de la cátedra de Proyecto Final por ser parte de este último tramo de la carrera, con una mención especial a Virginia Santos, nuestra profesora adjunta en el proyecto. Por último, queremos agradecer a la UTN FRC por darnos la posibilidad de formarnos como profesionales. Queremos hacer una mención especial a Claude.ia por ayudarnos en la redacción de este paper.

Referencias

- [1] SCRUM – Guía SCRUM – <https://www.scrum.org/resources/scrum-guide> – Julio 2026
- [2] PostgreSQL – API Documentation – <https://www.postgresql.org/docs/> – Julio 2026
- [3] XGBoost – Documentación oficial – <https://xgboost.readthedocs.io/en/stable/> – Julio 2026
- [4] Corsi, F. – A Simple Approximate Long-Memory Model of Realized Volatility – Journal of Financial Econometrics – 2009.
- [5] OpenAI – API Documentation – <https://platform.openai.com/docs> – Julio 2026.
- [6] TradingView – Documentación oficial – <https://es.tradingview.com/free-charting-libraries/> – Agosto 2026.
- [7] Walk-Forward Validation – <https://otexts.com/fpp3/> – Agosto 2026.
- [8] NestJS – API Documentation – <https://docs.nestjs.com/> – Julio 20206
- [9] Express – API Documentation – <https://expressjs.com/> - Julio 2026

Datos de Contacto

de la Quintana, Segundo Javier. Universidad Tecnológica Nacional, Facultad Regional Córdoba. E-mail: segudlq@gmail.com

Madrid Marcelo Miguel. Universidad

Tecnológica Nacional, Facultad Regional Córdoba. E-mail: marcelo.madrid03@gmail.com

Cufre Angel Hugo. Universidad Tecnológica Nacional, Facultad Regional Córdoba. E-mail: cufreangel@gmail.com

Romaniello Lucas Daniel. Universidad Tecnológica Nacional, Facultad Regional Córdoba. E-mail: ldromaniello@gmail.com

Malnis Maria Sol. Universidad Tecnológica Nacional, Facultad Regional Córdoba. E-mail: solmalnis@gmail.com

AÑO	2026	CURSO Y NRO. DE GRUPO	5K3 - G 14
NOMBRE DEL SISTEMA / PROYECTO			
Órbita			
CATEGORÍA (Solución De Negocio / Producto / Proyecto De Impacto Social)			
Producto			
HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS			
ÁMBITO DE APLICACIÓN		NOMBRE Y VERSIÓN	
ENTORNO DE DESARROLLO		Visual Studio Code (v1.128.0)	
REPOSITORIOS Y VERSIONADO		GitLab (v18.4.1-ee)	
PROGRAMACIÓN		Frontend: React (v19.2.5) TypeScript (v6.0.2) Backend: Express (v5.2.1) TypeScript (v6.0.2)	
BASE DE DATOS		PostgreSQL (v16.13)	
COMUNICACIÓN INTERNA		Discord y Whatsapp	
CAPACITACIÓN		Páginas web oficiales, Chat GPT, Claude, Youtube, documentación de tecnologías.	
PRUEBAS DE SISTEMA		Pruebas unitarias, pruebas de integración, pruebas end-to-end	
GESTIÓN DEL PROYECTO		Jira	
DOCUMENTACIÓN		Google Drive	
MODELOS		Miro	