Proyecto Integrado V - Linea de Énfasis (Entrega 1)

Jhon Alexis Machado Rodríguez
C.C: 1017143658

Julián José Martínez Camacho
C.C: 1081907956

Ingeniería de Software y Datos

Institución Universitaria Digital de Antioquia

Semestre 9°

Proyecto Integrado V - Línea de Énfasis

PREICA2501B020128

Andres Felipe Callejas

Domingo 11 de mayo del 2025

Tabla de Contenido

Automatización de la Recolección y Persistencia de Datos Históricos de Microsoft (MSFT) desde Yahoo Finanzas

1. Resumen

En un mundo donde los datos financieros son esenciales para el análisis cuantitativo y la toma de decisiones, en este proyecto presentamos una solución automatizada para recolectar, procesar y almacenar datos históricos de la acción de Microsoft Corporation (MSFT) desde Yahoo Finanzas. Desarrollada en Python con un diseño orientado a objetos, el sistema utiliza la librería yfinance para descargar cotizaciones diarias desde el debut bursátil de MSFT en 1986. La clase MSFTCollector gestiona la descarga y persistencia de datos, mientras que la clase Logger garantiza una trazabilidad completa mediante registros detallados. Los datos se almacenan incrementalmente en una base de datos SQLite (historical.db) y un archivo CSV (historical.csv), asegurando integridad y accesibilidad. Un flujo de trabajo de GitHub Actions automatiza actualizaciones diarias, y el proyecto se gestiona en un repositorio GitHub para control de versiones. Los resultados preliminares demuestran fiabilidad, escalabilidad y una base sólida para análisis estadísticos y predictivos futuros, con aplicaciones en inversión, investigación académica y machine learning.

Palabras clave: Automatización, datos históricos, MSFT, Python, GitHub Actions, SQLite, CSV, programación orientada a objetos, series temporales.

2. Introducción

El panorama financiero global depende en gran medida de datos históricos de valores bursátiles, que sirven como base para el análisis cuantitativo, la evaluación de estrategias de inversión y la investigación económica. Entre los indicadores más relevantes se encuentra Microsoft Corporation (MSFT), una de las empresas más influyentes del índice NASDAQ y un pilar de la industria tecnológica. Desde su oferta pública inicial el 13 de marzo de 1986, MSFT ha acumulado casi cuatro décadas de datos diarios, que incluyen precios de apertura, máximo, mínimo, cierre y volumen de transacciones. Esta serie histórica, disponible a través de plataformas como Yahoo Finanzas, es una fuente invaluable para

estudiar tendencias de mercado, volatilidad, correlaciones sectoriales y patrones de comportamiento financiero (Yahoo Finance, 2025).

Microsoft es reconocida mundialmente por su liderazgo en múltiples áreas: software (Windows, Microsoft Office), servicios en la nube (Azure), hardware (Surface, Xbox) y tecnologías emergentes como la inteligencia artificial. Su capitalización de mercado, que supera los dos billones de dólares, y su inclusión en índices como el S&P 500 y el Dow Jones Industrial Average reflejan su importancia en la economía global. La alta liquidez de MSFT, con millones de acciones negociadas diariamente, asegura que sus datos sean robustos y representativos, mientras que su estabilidad relativa frente a activos más volátiles la convierte en un caso de estudio ideal para desarrollar sistemas de recolección de datos escalables.

Sin embargo, la obtención manual de datos financieros presenta desafíos significativos. Descargar información de Yahoo Finanzas, procesarla para eliminar inconsistencias y almacenarla de manera estructurada es un proceso tedioso, propenso a errores y poco eficiente. Estas limitaciones reducen el tiempo disponible para tareas analíticas de mayor valor, como el desarrollo de modelos predictivos o la interpretación de tendencias. Para abordar esta problemática, en este proyecto proponemos una solución automatizada que combina herramientas modernas de desarrollo de software con prácticas avanzadas de gestión de datos.

El sistema utiliza Python como lenguaje principal, aprovechando la librería yfinance para acceder a los datos, pandas para su procesamiento y sqlite3 para su almacenamiento estructurado. La clase MSFTCollector encapsula la lógica de recolección y persistencia, mientras que la clase Logger proporciona trazabilidad mediante registros detallados. Un flujo de trabajo de GitHub Actions automatiza actualizaciones diarias, eliminando la necesidad de intervención manual, y el proyecto se gestiona en un repositorio GitHub para garantizar control de versiones, reproducibilidad y auditoría. Este enfoque no solo resuelve el problema inmediato de la recolección de datos, sino que también establece una base sólida para análisis avanzados, como el modelado de series temporales con statsmodels o la implementación de algoritmos de machine learning con scikit-learn.

3. Objetivo General

Desarrollar un sistema automatizado que respalde la recolección, procesamiento y almacenamiento de datos históricos de la acción de Microsoft Corporation (MSFT), garantizando su integridad, enriqueciendo la información disponible y facilitando el modelado de su comportamiento para apoyar la toma de decisiones acertadas en contextos financieros, académicos y tecnológicos.

4. Objetivos Específicos

1. **Recolectar datos históricos**: Obtener cotizaciones diarias de MSFT desde el 13 de marzo de 1986 hasta la fecha actual utilizando la librería yfinance, asegurando un histórico completo y actualizado.

2. **Preprocesar datos**: Estandarizar los datos en un formato claro y consistente (columnas en español, fechas como tipo date, separar la fecha en año, mes, día) y eliminar cualquier inconsistencia para facilitar su uso en análisis posteriores.

3. **Persistir datos incrementalmente**: Almacenar los datos en una base de datos SQLite (historical.db) y un archivo CSV (historical.csv), actualizando solo los registros nuevos para mantener la integridad del histórico sin duplicados.

4. **Garantizar trazabilidad**: Implementar un sistema de logging que registre cada paso del proceso, desde la descarga hasta el almacenamiento, permitiendo auditoría, diagnóstico de errores y transparencia operativa.

5. **Automatizar actualizaciones**: Configurar un flujo de trabajo en GitHub Actions para ejecutar actualizaciones diarias automáticas, integradas con control de versiones para asegurar consistencia y accesibilidad.

5. Justificación de la Selección de MSFT

La elección de Microsoft Corporation (MSFT) como indicador para este proyecto se basó en una evaluación de diversas alternativas, incluyendo índices de materias primas (e.g., petróleo, oro, esmeraldas), divisas y acciones de alta volatilidad como Tesla (TSLA). MSFT destacó como la opción más adecuada por varias razones fundamentales:

1. **Historia extensa y continua**: Desde su debut bursátil el 13 de marzo de 1986, MSFT ofrece casi 40 años de datos diarios, lo que proporciona una profundidad histórica ideal para análisis de largo plazo, estudios de tendencias y modelado de series temporales. Esta longevidad permite capturar ciclos económicos completos, eventos de mercado significativos y patrones estacionales.

2. **Alta liquidez y capitalización**: Como una de las empresas más valiosas del S&P 500, MSFT registra un volumen de transacciones diarias elevado, con millones de acciones negociadas. Esta liquidez asegura que los datos sean robustos, representativos y menos susceptibles a distorsiones causadas por baja actividad de mercado.

3. **Relevancia tecnológica**: Microsoft es un líder indiscutible en la industria tecnológica, con un portafolio que abarca software, servicios en la nube, hardware y tecnologías emergentes como la inteligencia artificial. Sus datos reflejan las dinámicas de un sector que impulsa la economía global, lo que los hace especialmente útiles para estudiar tendencias, volatilidad y correlaciones con otras empresas tecnológicas.

4. **Estabilidad relativa**: A diferencia de acciones más volátiles, como las de empresas emergentes o sectores cíclicos, MSFT presenta un comportamiento dinámico, pero con fluctuaciones moderadas. Esta estabilidad facilita la validación inicial del sistema, ya que los datos son menos propensos a anomalías que podrían complicar el desarrollo.

5. **Interés académico**: La serie histórica de MSFT es ampliamente utilizada en disciplinas como econometría, finanzas cuantitativas y machine learning. Desde investigaciones académicas que analizan la volatilidad del mercado hasta aplicaciones prácticas en estrategias de trading algorítmico, los datos de MSFT tienen un impacto significativo, asegurando que los resultados del proyecto sean relevantes para una amplia audiencia.

6. Metodología

El diseño del proyecto se basó en un enfoque modular, escalable y reproducible, dividido en varias etapas.

1. Diseño Orientado a Objetos

El núcleo del sistema es un paquete Python estructurado en programación orientada a objetos (OOP), lo que permite un código modular, fácil de mantener y extensible. Se desarrollaron dos clases principales para encapsular la lógica del sistema:

Clase MSFTCollector (collector.py)

Esta clase es el corazón operativo del sistema y se encarga de las siguientes tareas:

- **Descarga de datos**: Utiliza la librería yfinance para obtener cotizaciones diarias de MSFT desde el 13 de marzo de 1986 hasta el día siguiente a la fecha de ejecución. Este enfoque asegura que se incluya el cierre más reciente, incluso en días festivos o fines de semana, maximizando la actualidad de los datos.

- **Procesamiento de datos**: Transforma el DataFrame crudo de yfinance en un formato estandarizado y accesible. Las columnas se renombran a español (Fecha, Abrir, Máx., Mín., Cerrar, Volumen) para mejorar la legibilidad, y las fechas se convierten a tipo date asi también se separa la fecha en año, mes y día para facilitar consultas y comparaciones. Además, se eliminan datos nulos o inconsistentes, si los hubiera.

- **Persistencia**: Gestiona el almacenamiento de los datos en dos formatos complementarios: una base de datos SQLite (historical.db) y un archivo CSV (historical.csv). La clase implementa actualizaciones incrementales, comparando fechas nuevas con las existentes para evitar duplicados y garantizar un histórico completo.

Clase Logger (logger.py)

- **Formato profesional y consistente**: Registra eventos en formato YYYY-MM-DD HH:MM:SS - LEVEL - MESSAGE, facilitando la lectura y el análisis durante la ejecución o auditoría posterior.

- **Salida dual y persistente**: Los mensajes se imprimen en consola y se almacenan en un archivo permanente (msft_analytics.log) dentro de static/logs, permitiendo trazabilidad histórica.

- **Prevención de duplicados**: Evita múltiples registros del mismo mensaje al controlar que no se creen handlers duplicados, garantizando un log limpio y sin redundancias.

2. Estructura del Paquete

El código está organizado en un paquete Python bajo el directorio `src/msft_analytics/`:

Directorio/Archivo	Descripción
collector.py	Contiene la clase MSFTCollector para la descarga, procesamiento y persistencia de datos.
logger.py	Configura el sistema de logging personalizado, con salida a consola y archivo.
static/data/historical.db	Base de datos SQLite que almacena la tabla msft_data con los datos históricos de MSFT.
static/data/historical.csv	Archivo CSV con el histórico completo, ideal para revisiones manuales y auditorías.
models/	Directorio reservado para futuros modelos de análisis, evidencia 2.
__init__.py	Archivo vacío que marca el directorio como un paquete Python, habilitando la importación de módulos.
setup.py	Define la instalación del paquete, sus dependencias y un *entry point* (msft-collector) para ejecutar el sistema desde la consola.
requirements.txt	Lista las dependencias del proyecto (yfinance, pandas, sqlite3).

3. Descarga de Datos

La recolección de datos se realiza mediante la librería yfinance, una herramienta robusta y ampliamente utilizada para acceder a datos financieros de Yahoo Finanzas. El proceso incluye los siguientes pasos:

- **Período de datos**: Se descargan todas las cotizaciones diarias de MSFT desde el 13 de marzo de 1986 hasta el día siguiente a la fecha de ejecución. Este enfoque garantiza que los datos estén siempre actualizados, incluso en ejecuciones automáticas programadas.

- **Datos descargados**: Los datos incluyen precios de apertura (Open), máximo (High), mínimo (Low), cierre (Close) y volumen de transacciones (Volume). El parámetro auto_adjust=False asegura que los precios de cierre sean los originales, sin ajustes automáticos por dividendos o splits.

- **Procesamiento**:
  Las columnas se renombran a español para mayor claridad: Fecha, Abrir, Máx., Mín., Cerrar, Volumen.

  La columna Fecha se convierte a tipo date y se separa en año, mes y día utilizando pandas para facilitar consultas y comparaciones.

  Se verifican y eliminan datos nulos o inconsistentes, asegurando que el conjunto de datos sea limpio y confiable.

- **Validación**: Antes de procesar los datos, se verifica que el DataFrame contenga las columnas esperadas y que no haya errores en la conexión con Yahoo Finanzas.

4. Persistencia Incremental

El sistema almacena los datos en dos formatos complementarios para maximizar su utilidad y cumplir con los requisitos del proyecto:

Base de Datos SQLite (historical.db)

- **Conexión y creación**: El sistema se conecta a la base de datos historical.db ubicada en static/data/. Si la tabla msft_data no existe, se crea con las columnas correspondientes.

- **Actualización incremental**: Si la tabla ya existe, el sistema carga los datos existentes y los compara con los nuevos basándose en la columna Fecha. Solo se insertan los registros correspondientes a fechas no registradas, evitando duplicados y asegurando un histórico completo.

- **Integridad de datos**: Antes de cada actualización, se valida la presencia de la columna Fecha y se eliminan duplicados por fecha, si los hubiera. Esto garantiza que la base de datos sea consistente y libre de errores.

- **Ventajas**: SQLite es una base de datos ligera, portátil y sin servidor, ideal para aplicaciones que requieren consultas estructuradas. Su compatibilidad con SQL permite realizar análisis avanzados directamente en la base de datos, mientras que su formato de archivo único facilita la auditoría y el respaldo.

Archivo CSV (historical.csv)

- **Exportación**: Tras actualizar la base de datos SQLite, el histórico completo se exporta a historical.csv en el directorio static/data/.

- **Formato**: Los datos se guardan con un formato numérico estandarizado (dos decimales para precios, e.g., 123.45) y fechas en formato YYYY-MM-DD para consistencia.

- **Ventajas**: El formato CSV es universalmente accesible, lo que lo hace ideal para revisiones manuales, auditorías y usuarios no técnicos. También sirve como respaldo adicional del histórico, complementando la base de datos SQLite.

5. Trazabilidad y Logging

La trazabilidad es un componente crítico del sistema, ya que permite auditar cada ejecución, diagnosticar problemas y garantizar transparencia operativa. La clase Logger genera mensajes detallados en cada paso del proceso, con un formato que incluye la fecha, hora, nivel de severidad y descripción del evento. Los mensajes se registran simultáneamente en la consola y en un archivo (msft_analytics.log), proporcionando un registro permanente para auditoría.

ejemplo de salida:

2025-05-12 15:57:14,252 - INFO - Descargando Datos Desde Yahoo Finanzas...
2025-05-12 15:57:19,168 - INFO - Guardando Datos En SQLite...
2025-05-12 15:57:19,230 - INFO - Tabla msft_data Creada Con 9868 Registros.
2025-05-12 15:57:19,230 - INFO - Base De Datos SQLite Generada En: …\proyecto-integrado-v-datos-msft-analytics\src\msft_analytics\static\data\historical.db
2025-05-12 15:57:19,261 - INFO - Guardando Datos En CSV...
2025-05-12 15:57:19,336 - INFO - Archivo CSV Generado En: …\proyecto-integrado-v-datos-msft-analytics\src\msft_analytics\static\data\historical.csv
2025-05-12 15:57:19,336 - INFO - Proceso Completado Con Éxito.

Estos mensajes permiten a los usuarios:

- Confirmar que cada etapa del proceso (descarga, almacenamiento, exportación) se completó correctamente.

- Identificar el número de registros nuevos añadidos en cada ejecución, lo que facilita el seguimiento de los deltas de información.

- Localizar los archivos generados mediante sus rutas absolutas, útil para verificar la salida del sistema.

- Detectar advertencias o errores, como problemas de conexión con Yahoo Finanzas, datos faltantes o inconsistencias en la base de datos.

6. Automatización con GitHub Actions

Para garantizar que los datos se actualicen regularmente sin intervención manual, se configuró un flujo de trabajo de integración continua en GitHub Actions, definido en el archivo .github/workflows/update_data.yml. Este flujo incluye:

Disparadores:

- **Programado:** Se ejecuta diariamente mediante una expresión cron (cron: "0 0 * * *"), que corresponde a la medianoche (UTC).

- **Manual**: Puede activarse a través de la interfaz de GitHub utilizando el evento workflow_dispatch, útil para pruebas o actualizaciones fuera de horario.

- **Push**: Se dispara automáticamente al realizar un *push* a la rama principal (main), asegurando que los cambios en el código se reflejen en los datos.

Pasos:

- **Checkout**: Clona el repositorio en el entorno de GitHub Actions utilizando la acción actions/checkout@v3.

- **Configuración de Python**: Instala Python 3.9, la versión utilizada para el desarrollo, mediante actions/setup-python@v4.

- **Instalación de dependencias**: Ejecuta pip install -e . para instalar el paquete msft_analytics y sus dependencias (yfinance, pandas), asegurando que el entorno esté completamente configurado.

- **Ejecución del script**: Llama al comando msft-collector, definido como *entry point* en setup.py, para recolectar, procesar y persistir los datos.

- **Commit y push**: Actualiza los archivos historical.db y historical.csv en el repositorio, utilizando comandos de Git para registrar los cambios con un mensaje descriptivo (“Update MSFT historical data”). Si no hay cambios, el paso se omite para evitar commits vacíos.

Entorno: El flujo se ejecuta en un contenedor Ubuntu proporcionado por GitHub (ubuntu-latest), garantizando consistencia y portabilidad en las ejecuciones.

7. Control de Versiones

Todo el proyecto, incluyendo el código fuente, los datos generados y la documentación, se gestiona en un repositorio de GitHub:

- **Auditoría**: Cada cambio en el código, los datos o la documentación queda registrado, con metadatos que incluyen el autor, la fecha y el propósito del cambio.

- **Colaboración**: El repositorio permite que nosotros tabajemos en el proyecto simultáneamente.

- **Reproducibilidad**: Cualquier usuario puede clonar el repositorio y reproducir el sistema en su propio entorno, gracias a la documentación clara y las dependencias especificadas.

7. Resultados

La implementación inicial del sistema ha demostrado ser exitosa en varios aspectos clave, cumpliendo con los objetivos establecidos:

- **Fiabilidad**: Las ejecuciones locales y automáticas (vía GitHub Actions) actualizan el histórico de MSFT sin duplicados ni pérdidas de datos. Cada día a la medianoche, se añade un registro nuevo o varios, correspondiente a la última ejecución, dependiendo de los días festivos o cierres de mercado.

- **Escalabilidad**: El diseño modular basado en OOP permite incorporar nuevos indicadores financieros (e.g., otras acciones, índices o divisas) con modificaciones mínimas en la clase MSFTCollector. La estructura del paquete también facilita la integración con herramientas analíticas futuras.

- **Transparencia**: El sistema de logging proporciona una trazabilidad completa, con mensajes claros que detallan cada paso del proceso, desde la descarga hasta el almacenamiento. Los registros en msft_analytics.log y la consola permiten auditar ejecuciones y diagnosticar problemas rápidamente.

- **Eficiencia**: La automatización elimina la necesidad de tareas manuales, reduciendo significativamente el tiempo y esfuerzo requeridos para mantener el histórico actualizado. El flujo de GitHub Actions asegura que las actualizaciones se realicen de manera consistente y sin intervención humana.

- **Accesibilidad de datos**: Los datos almacenados en SQLite (historical.db) y CSV (historical.csv) son compatibles con una amplia gama de herramientas de análisis, desde consultas SQL en SQLite hasta importaciones en Excel, Python (pandas), o software de visualización como Tableau o Power BI.

- **Volumen de datos**: En una ejecución reciente, la tabla msft_data alcanzó aproximadamente 9,863 registros, correspondientes a los días hábiles desde 1986

hasta mayo de 2025. Este volumen refleja la profundidad histórica de MSFT y la capacidad del sistema para manejar grandes conjuntos de datos.

Un ejemplo concreto de los resultados es el crecimiento incremental del histórico. Durante una semana típica, el sistema añade nuevos registros correspondientes a los días hábiles (lunes a viernes), y los archivos historical.db y historical.csv se actualizan para reflejar estos cambios. La validación de los datos muestra que no hay duplicados por fecha, y los valores de precios y volumen son consistentes con los proporcionados por Yahoo Finanzas.

9. Conclusión

El proyecto de automatización de la recolección y persistencia de datos históricos de Microsoft (MSFT) representa un avance significativo en la gestión eficiente de datos financieros. Al combinar herramientas modernas como Python, yfinance, SQLite y GitHub Actions, el sistema ofrece una solución robusta, escalable y fácil de usar que elimina las barreras asociadas con la obtención manual de datos. Su diseño orientado a objetos, trazabilidad detallada y automatización diaria lo convierten en una herramienta poderosa para analistas, investigadores y desarrolladores interesados en el análisis financiero y el modelado predictivo.

Los resultados preliminares confirman la fiabilidad del sistema, con un histórico que crece de manera consistente y datos accesibles en formatos versátiles. La capacidad del sistema para manejar deltas de información, mantener la integridad del histórico y proporcionar trazabilidad completa cumple con los estándares profesionales de desarrollo de software y gestión de datos. Además, la elección de MSFT como indicador asegura que los resultados sean relevantes para aplicaciones prácticas, desde la evaluación de estrategias de inversión hasta el desarrollo de modelos de machine learning.

10. Referencias

Yahoo Finance. (2025). Microsoft Corporation (MSFT). *https://es.finance.yahoo.com/quote/MSFT/*

Yahoo Finance. (s/f). Yahoo Finance. https://finance.yahoo.com/

Python Software Foundation. (2023). Python 3 documentation. *https://docs.python.org/3*

Github.com Documentación de ayuda. (s/f). https://docs.github.com/es

GitHub, Inc. (2023). GitHub Actions documentation. https://docs.github.com/actions

Scikit-learn. (s/f). Scikit-learn.org. https://scikit-learn.org/

Pandas Development Team. (2023). Pandas documentation. *https://pandas.pydata.org/docs/*

SQLite Consortium. (2023). SQLite documentation.

Statsmodels Development Team. (2023). Statsmodels documentation. *https://www.statsmodels.org/stable*