Concurrent Web Scraping and Financial Analysis System

Project Report for Object Orientated Development

María Muñoz Nadales

Degree: Mathematical Engineering

December 2, 2025

Abstract

This project presents the development of a complete data-processing pipeline capable of scraping, storing, and analyzing financial information from the Spanish stock market. The system retrieves real-time market prices and financial summaries from the IGBM index (CincoDías, El País) and processes them using a modular, concurrency-oriented design.

Python's `threading` library is used extensively to implement multithreaded scraping and analysis workflows. Key concurrency mechanisms; threads, locks, critical sections, semaphores, and synchronization via `join()` are applied in a real-world context involving web requests, database interactions, and data processing.

The pipeline includes modules for price scraping, concurrent financial scraping, dataset merging, financial ratio computation, classification, and investment recommendation. The final system produces structured datasets, analysis results, and execution logs, delivering a functional and extensible tool aligned with the principles of this subject's Topic 7: Concurrency. The results are realistic given the nature of the companies analyzed, and the methodology demonstrates a practical, robust implementation of concurrent programming in Python.

Table of Contents

1 Introduction

The objective of this project is to develop a system capable of collecting, processing, and analyzing stock-market data from the Spanish IGBM index (CincoDías – El País). The system demonstrates the practical use of concurrency within a real-world data-processing pipeline.

Using Python, the system retrieves market data, financial summaries, processes information, conducts analysis, and generates a final investment recommendation.

2 Project Objectives

The main objective of this work is to build a Python application that performs real-time data scraping and financial analysis using concurrent programming. More specifically, the project aims to automate the retrieval of market data, implement concurrent scraping of company financial summaries using multiple threads, store all extracted information in persistent formats such as CSV and SQLite, and compute a set of financial indicators.

2.1 Specific Objectives

- Automate retrieval of stock data from the Internet.
- Implement concurrent scraping of financial summaries.
- Store scraped data in CSV and SQLite.
- Compute ratios and generate recommendations.
- Apply concurrency in scraping and analysis.

3 Theoretical Background

3.1 Web Scraping

Web scraping automates extraction of information from websites using:

- `requests` for HTTP
- `BeautifulSoup` for HTML parsing

3.2 Concurrency: Threads, Locks, Semaphores

This project applies threads for parallel tasks, locks for critical sections, semaphores for concurrency limits and synchronization via `join()`

3.3 Data Storage

Two formats are used:

- CSV for raw stock data
- SQLite for structured financial data

3.4 Financial Ratios

Key metrics:

- PER: measures how many times investors are paying for each unit of a company's annual earnings based on its share price.
- BPA: indicates the portion of a company's net profit allocated to each individual share.
- EBITDA: represents operating profit before interest, taxes, depreciation, and amortization, reflecting the company's real cash-generation ability.
- Net Debt: total financial debt minus cash and equivalents, showing the company's true indebtedness.
- Debt/EBITDA: expresses how many years the company would need to repay its net debt using its EBITDA, indicating its leverage level.

4 System Architecture

4.1 Pipeline Flow

1. Market scraping (CSV)
2. Multithreaded financial scraping (SQLite)
3. Multithreaded analysis (ratios, signals, recommendations)
4. Export to CSV and Excel

4.2 Folder Structure

```
proyecto_bolsa_concurrencia/

 src/
    scraping_cotizacion.py
    scraping_finanzas.py
    analisis.py
    main.py

 raw/
    cotizaciones/
    finanzas/

 processed/
    dataset_unido.csv

 analysis/
    analisis_resultados_YYYYMMDD.csv
    analisis_resultados.xlsx

 logs/
    cotizaciones.log
    finanzas.log
    analisis.log

 README.md
```

5 Detailed Implementation

5.1 scraping_cotizacion.py

Downloads IGBM data, parses HTML, extracts table values, converts Spanish numeric formats, saves CSV, logs execution.

5.2 scraping_finanzas.py

Uses multithreading:

- One thread per company
- Semaphore to limit simultaneous scraping
- Lock to protect SQLite writes

5.3 analisis.py

Concurrent tasks:

- Ratio computation
- Financial signal classification
- Recommendation generation
- Export to CSV and Excel

5.4 main.py

Orchestrates entire workflow.

6 Concurrency Diagram

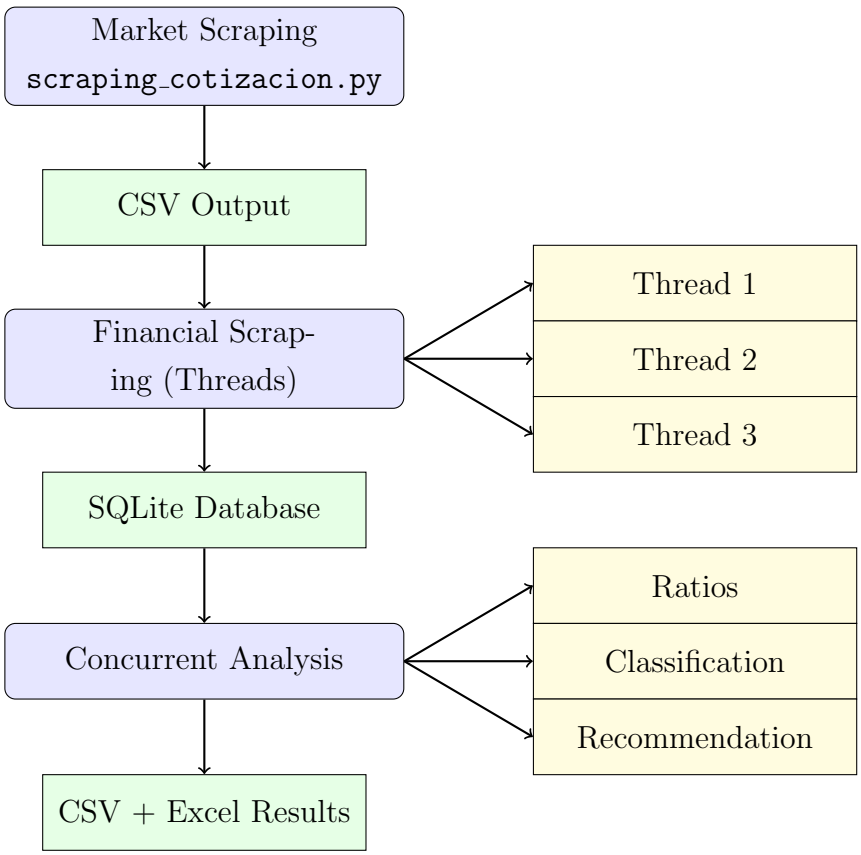

Figure 1: Concurrent Data Processing Pipeline

7 Results and Discussion

Financial ratios retrieved from IBEX companies fall into stable ranges, producing mostly "Hold" recommendations, consistent with real market analysis.

8 Limitations

- Dependence on webpage structure.
- Only daily snapshots (no historical series).
- Basic financial model.

9 Conclusions

This project demonstrates the successful integration of web scraping, structured data storage, and concurrent financial analysis in a unified software system. By leveraging Python's concurrency mechanisms, the system efficiently retrieves external data, manages shared resources safely, and performs complex computations in parallel. The results obtained are realistic and consistent with expectations for major Spanish companies, confirming both the validity of the pipeline and the practical usefulness of the methodology. Overall, the work illustrates how the concepts introduced in Topic 7: threads, locks, semaphores, and synchronization—can be effectively applied to real data-engineering challenges, producing a modular, extensible, and professionally structured solution.