Tarifflo System Design

My approach to the pricing tracker challenge

By: Jayanth Gowda Ramanna | jayanthgowda.ramanna@gmail.com

Created: July 24, 2025

My Understanding of the Problem

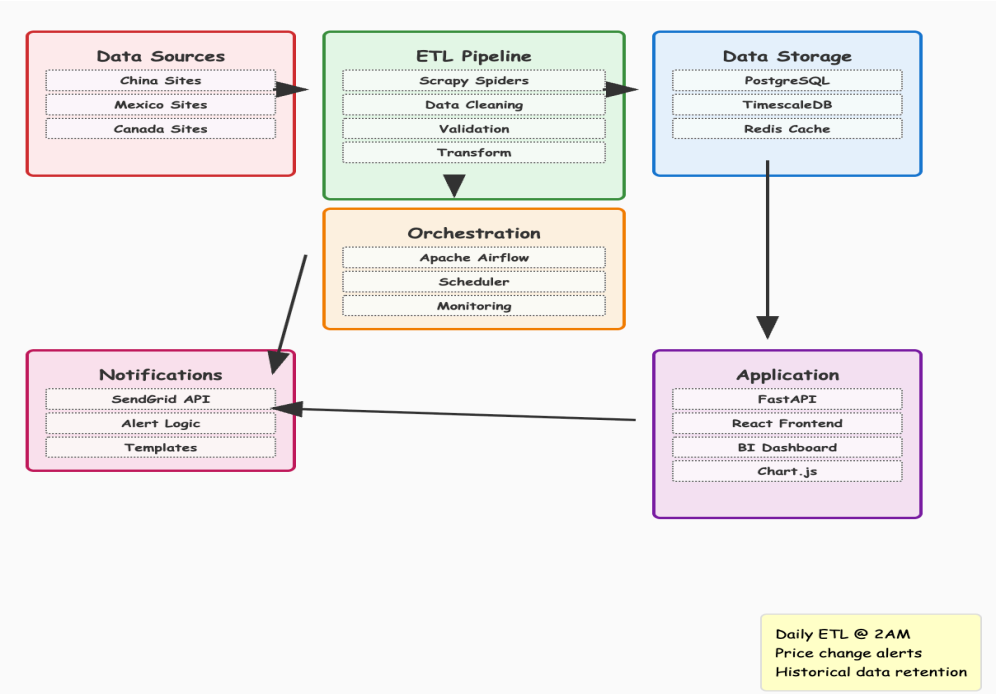
So basically, I need to build a system that:

- Scrapes pricing data from websites in China, Mexico, and Canada
- Runs this every day automatically
- Stores the data so we can see price changes over time
- Sends emails when prices change
- Shows graphs and charts on a dashboard

Note: I'm still learning about system design, so I tried to keep this practical and not over-engineer it. I focused on technologies I know or have used before.

System Architecture (My Version)

Tarifflo System Architecture



How Each Part Works

1. Web Scraping (ETL Pipeline)

I'd use Python with these libraries:

- **Requests/BeautifulSoup:** For simple HTML scraping
- **Selenium:** For JavaScript-heavy sites (if needed)
- **Pandas:** For data cleaning and processing

2. Database Design

I'd keep it simple with PostgreSQL.

3. Daily Scheduling

Two options I considered:

Simple Cron Jobs

- Easy to set up
- No extra dependencies
- Good for small scale

Apache Airflow

- Better monitoring
- Handles failures better
- More complex to set up

4. Web Application

For the dashboard, I'd build:

- **Backend:** FastAPI or Flask (I'm more familiar with Flask)
- **Frontend:** React with Chart.js for the graphs
- **API endpoints:** Get price history, current prices, etc.

My Technology Stack

Backend & Data Processing

- **Python 3.9+** - Main language (what I know best)
- **Flask or FastAPI** - Web framework
- **PostgreSQL** - Main database
- **Redis** - For caching frequently accessed data
- **Pandas** - Data processing
- **Requests/BeautifulSoup** - Web scraping

Frontend & Visualization

- **React** - Frontend framework
- **Chart.js** - For price graphs
- **Bootstrap** - Quick styling
- **Axios** - API calls

Infrastructure & Tools

- **Docker** - Containerization
- **SendGrid** - Email service
- **GitHub Actions** - CI/CD
- **DigitalOcean or AWS** – Hosting

How Data Flows Through the System

1. **Daily at 2AM:** Cron job triggers Python scrapers
2. **Scraping:** Scripts visit distributor websites and extract prices
3. **Data Cleaning:** Remove duplicates, fix formatting, validate prices
4. **Database Insert:** Store new price data in PostgreSQL
5. **Change Detection:** Compare with yesterday's prices
6. **Email Alerts:** If prices changed, send emails to subscribers
7. **Dashboard Update:** Users can see new data on the website

Problems I Expect & My Solutions

Challenge	My Solution	Why This Works
Websites blocking scrapers	Add delays, rotate user agents, respect robots.txt	Be respectful and look like a real user
Data quality issues	Validation checks, manual review for outliers	Catch bad data before it goes to users
System crashes	Try/catch blocks, logging, restart mechanisms	Keep the system running even with errors
Too many emails	Only send if price change > 5%, daily digest option	Don't spam users with tiny changes

Making It Scale (Future Thoughts)

Honest note: I don't have much experience with large-scale systems, but here's what I think would help if this gets big:

If we get more data:

- Split scraping across multiple servers
- Use database partitioning by date or country
- Add more Redis caching for common queries
- Maybe consider MongoDB for very large datasets

If we get more users:

- Load balancer in front of web servers
- CDN for static files
- Database read replicas

- Queue system for email sending

Handling Things When They Go Wrong

Scraping Errors

Database Errors:

- Daily database backups
- Connection pooling to handle multiple requests
- Rollback transactions if data insert fails

User-Facing Errors:

- Show friendly error messages instead of technical ones
- Cache some data so dashboard works even if database is slow
- Log all errors for debugging later

How I'd Build This (Step by Step)

Week	What I'd Build	Goal
Week 1-2	Basic scraper for one website, simple database	Prove the concept works
Week 3-4	Add more websites, improve data cleaning	Get reliable data from all countries
Week 5-6	Build basic web dashboard with charts	Users can see the data
Week 7-8	Email alerts, daily scheduling	Complete working system

What I'd Need to Learn/Research More

- Advanced web scraping techniques (handling CAPTCHAs, etc.)
- Time series databases (maybe TimescaleDB?)
- Better monitoring and alerting systems
- Database optimization for large datasets
- Security best practices for web scraping

My Honest Take

This is a pretty complex system for someone at my level, but I think I can build a working version. I'd definitely need to learn some new things along the way, especially around handling large amounts of data and making sure the scrapers don't get blocked.

The parts I'm most confident about: Python scripting, basic web development, database design. The parts I'd need help with: scaling, advanced error handling, and some of the DevOps stuff.