System Architecture Documentation

1 Overview

This document describes the architecture of the Pricing API system, detailing its components, data flow, and deployment environment. The system provides a REST API to calculate and retrieve the cost of database services over different billing periods (hourly, monthly, yearly). It is built using FastAPI, PostgreSQL, and scheduled jobs running within a Docker environment.

2 Architecture Diagram

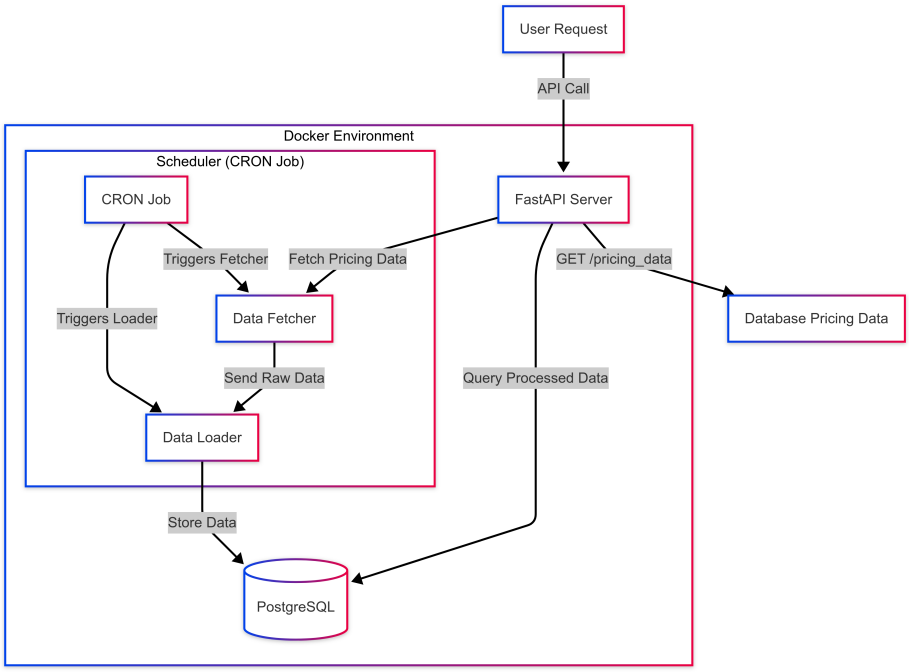

Figure 1: System Architecture

3 Components

3.1 FastAPI Server

- Handles user requests and API calls.
- Routes include:
  - `GET /pricing_data`: Retrieves processed pricing data.
  - `POST /skus/{sku}/terms/`: Creates a pricing term for a specific SKU.
  - `PUT /skus/{sku}/terms/{term_type}`: Updates an existing pricing term.
  - `DELETE /skus/{sku}/terms/{term_type}`: Deletes a pricing term.

3.2 PostgreSQL Database

- Stores raw and processed pricing data.
- Acts as the primary storage for SKU pricing terms.
- Interacts with FastAPI for data retrieval and modifications.

3.3 Data Fetcher

- Fetches pricing data from external sources.
- Sends raw data to the Data Loader.

3.4 Data Loader

- Processes and stores fetched pricing data into PostgreSQL.
- Ensures data consistency before storage.

3.5 Scheduler (CRON Job)

- Runs scheduled tasks to automate data fetching and loading.
- Ensures the system remains updated with the latest pricing information.

4 Deployment and Environment

- All services run within Docker containers.
- The database service (`PostgreSQL`) is defined in `docker-compose.yml`.
- The API server (`FastAPI`) depends on database migrations before launching.

- The Scheduler runs within Docker to trigger fetching and loading tasks at regular intervals.

5 Conclusion

This architecture ensures modularity, scalability, and efficient data processing, leveraging FastAPI, PostgreSQL, and scheduled jobs within a containerized environment.