Job Market Project

by Tom, Shaku, Manh and Sabine

Introduction of the Team

Sabine Felber

- Lives in Cologne
- Business Analyst
  Data Engineer
  (future)

Wilhelm Thomas Röder

- Lives in Berlin
- Localization Engineer
  Data Engineer (future)

Manh Cuong Nguyen

- Lives in Berlin
- Cloud Engineer
  (future)

Shakural Islam

- Lives in Munich
- Software Engineer
  Data Engineer (future)

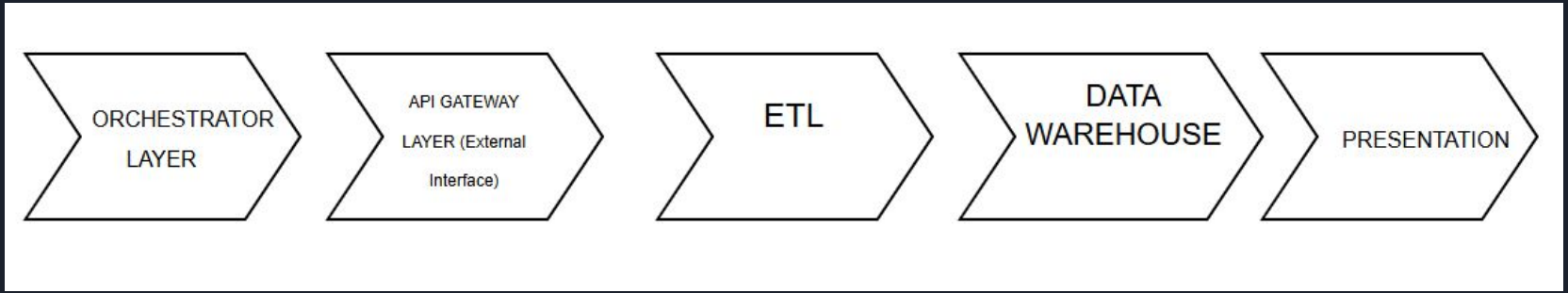
Project Overview

This project focuses on building a data-driven overview of the job market.

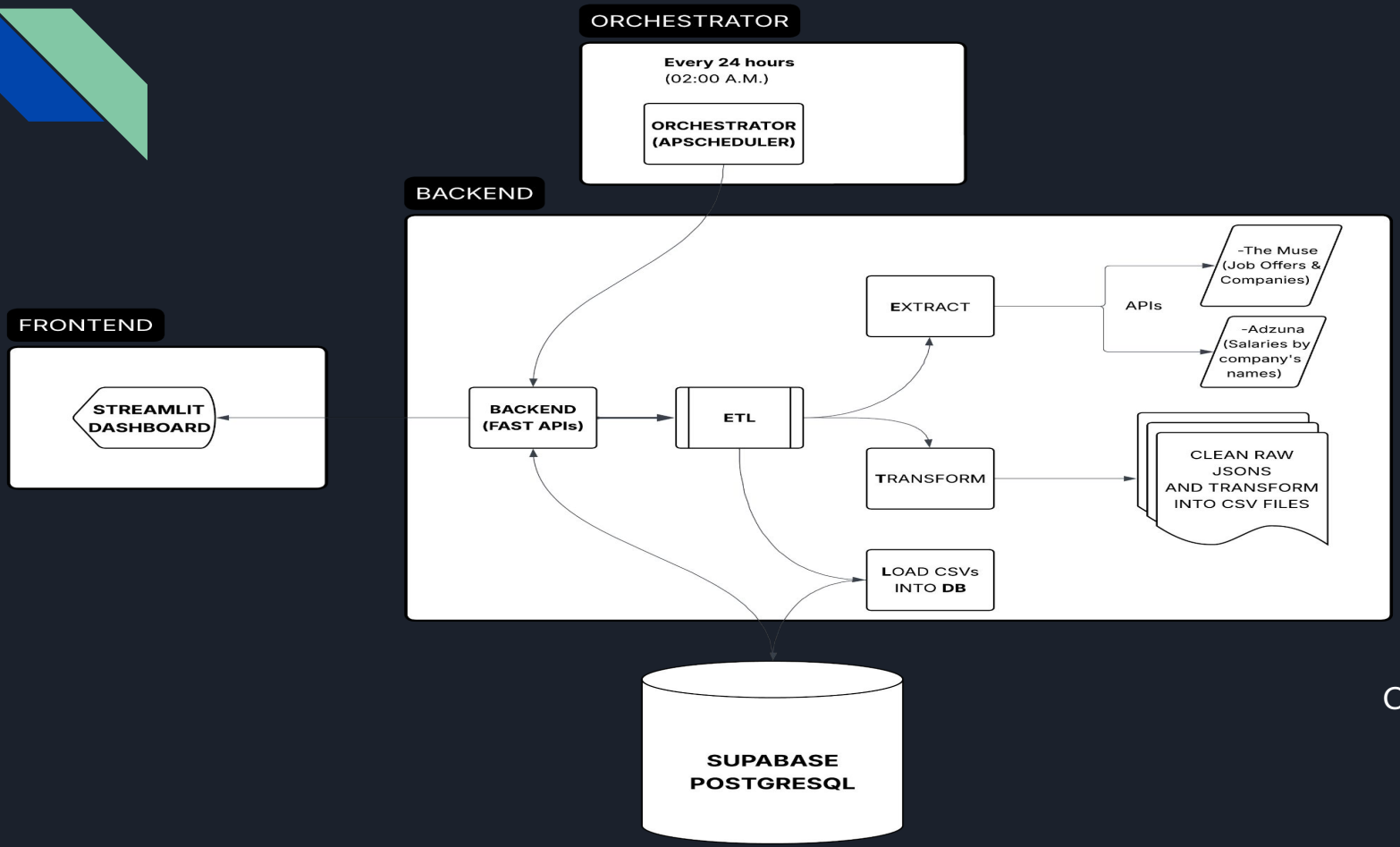
We design and implement a data pipeline that collects, harmonizes, and stores job-market data from different APIs in a structured database before analyzing hiring activity, experience-level demand, salary patterns, and regional trends.

The processed metrics are delivered through a unified API layer and an interactive dashboard for streamlined exploration of the job market.

Pipeline


ORCHESTRATOR LAYER
API GATEWAY LAYER (External Interface)
ETL
DATA WAREHOUSE
PRESENTATION

Architecture Diagram



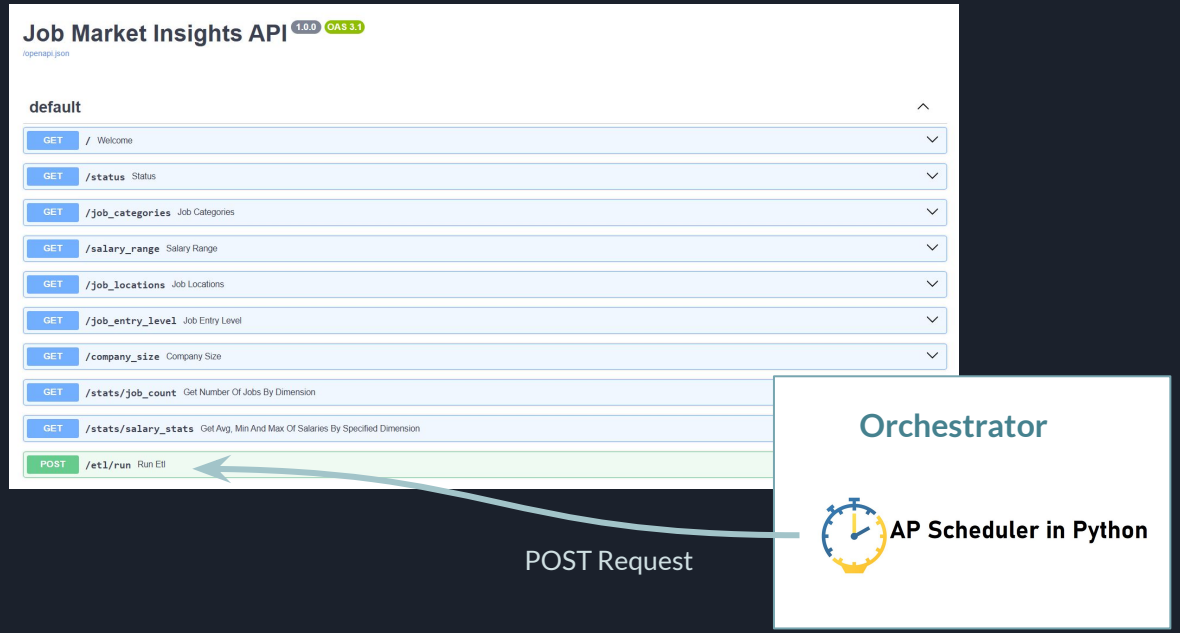
CI/CD:

- *Github Actions*
- *DockerHub*

Deep Dive Architecture

Orchestration & API Gateway

- Scheduled ETL-Trigger with APScheduler (Python-native integration, flexible, lightweight)
- Locking prevents overlapping executions
- FastAPI endpoints:
  - Run ETL pipeline
  - Statistics
  - Meta data



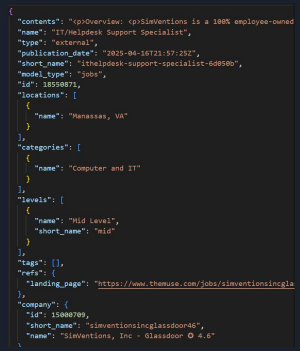
Deep Dive Architecture

ETL Pipeline

- Automated Batch process
- The Muse API delivers job and company data
- Adzuna API delivers additional salary data
- Transform flatten, clean and harmonizes data for merging in the final model
- Multi-layer loading: Raw → Norm → Star

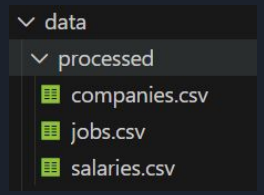
Extract

fetch semi-structured data
store raw JSON
archive historical raw data



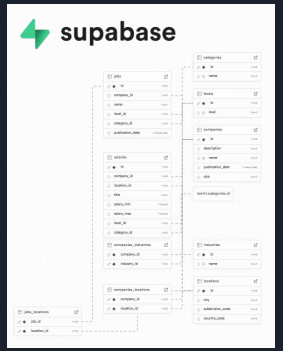
Transform

flatten, clean, normalize, enrich & deduplucate
structured CSVs for load



Load

load CSVs to Supabase
populate schemas
raw → norm → star



Deep Dive Architecture

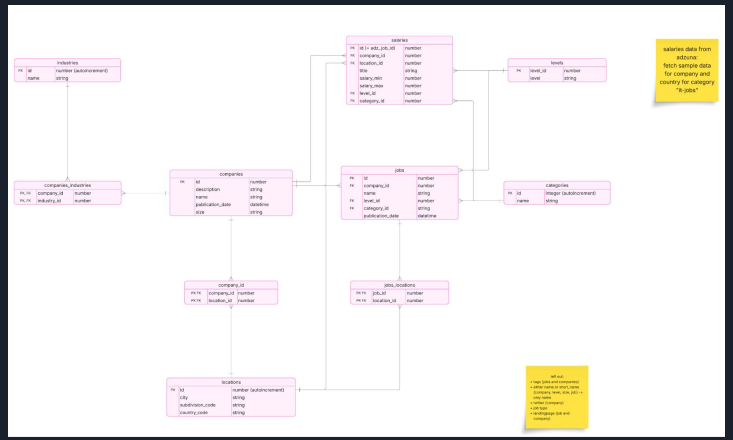
Database

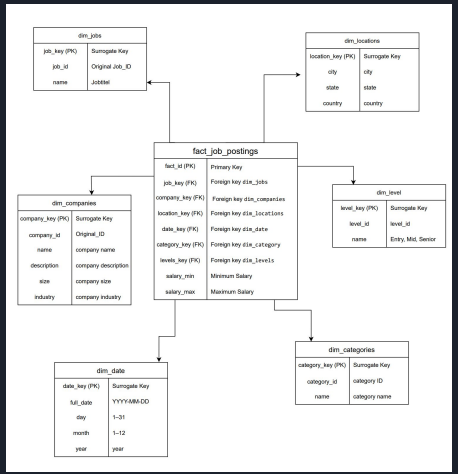
- 3 stages: Raw -> Norm -> Star
- CSV data is loaded first into Raw
- Data cleansing and normalization happen in the Normalized layer
- Star Schema is optimized for analytics and reporting
- Streamlit frontend and API endpoints read data from the Star layer
- Entire pipeline runs on Supabase (PostgreSQL)

Raw
|-raw.companies (Muse)
|-raw.jobs (Muse)
|-raw.salaries (Adzuna)

Norm
|-norm.categories
|-norm.companies
|-norm.companies_industries
|-norm.companies_locations
|-norm.industries
|-norm.jobs
|-norm.jobs_location
|-norm.levels
|-norm.locations
|-norm.salaries

Star
|-star.dim_categories
|-star.dim_companies
|-star.dim_date
|-star.dim_jobs
|-star.dim_levels
|-star.dim_locations
|-star.v_job_postings
|-star.v_job_salaries





Deep Dive Architecture

Devops Strategy

- **Tools used**
  - GitHub Actions for CI/CD
  - Docker for containerization
  - Docker Hub for image storage
- **Pipeline Stages**

  lint → test → build → deploy

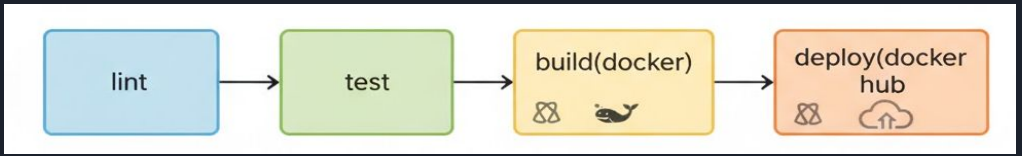
- **Services**
  - frontend
  - backend
  - orchestrator
- **Quality gates**
  - flake8 for code checks
  - pytest for automated tests

Docker Containerization

- All services containerized for consistent environments
- Custom Dockerfiles for efficient builds
- Reproducible runtime settings
- Faster local development and deployment

CI/CD Pipeline with GitHub Actions

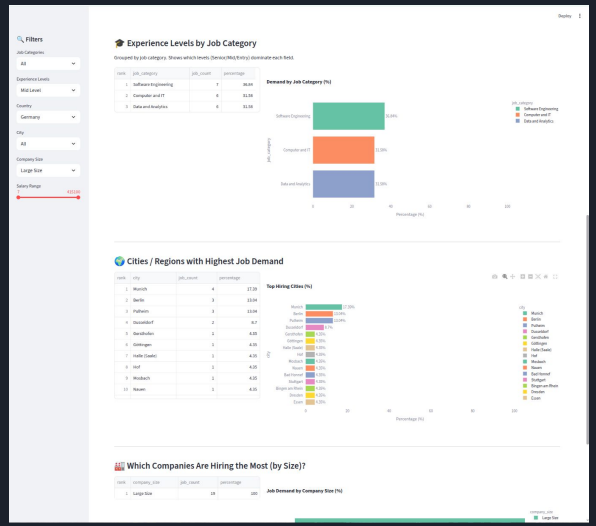
- Automated linting and testing
- Matrix builds for each service
- Automated Docker image creation
- Reliable, continuous integration and deployment



Deep Dive Architecture

Frontend

- Interactive Streamlit Dashboard
- Powered by Analytics API
- Insights: Top companies, salary insights, demand by experience level, category, etc.
- Filter: location, experience level, company size, etc.



1. Top 10 companies by average salary
2. Top 10 companies by number of job postings
3. Which experience levels are most in demand?
4. Which experience levels are most in demand by job category?
5. Which job categories have the most open positions?
6. Which cities have the highest number of open positions?
7. Which company sizes are hiring the most?
8. How does salary vary by company size?

Demo

Future Roadmap
(Move towards a cloud-native stack)

Orchestration & Scalability

- Migrate orchestration from APScheduler to Airflow (DAG-based)
- Implement distributed ETL processing with Apache Spark
- Deploy ETL workflows on Kubernetes for scalability

Cloud Data Infrastructure

- Build a Data Lake on AWS S3 for raw and staged data storage

Monitoring & Observability (Application Health)

- Replace local ETL logs with CloudWatch or OpenSearch for centralized logging and monitoring (alerting and run-metrics dashboards)

Dashboard & Machine Learning

- Develop ML-based enrichment, including job classification and salary prediction models
- Extend dashboard and analytics (i.e. trend analysis, job search capabilities, additional insights)

Thank you for your time!

Questions?

Introduction of the Team

Sabine Felber

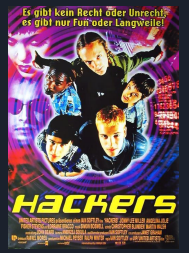
- Lives in Cologne
- Business Analyst
  Data Engineer
  (future)

Manh Cuong Nguyen

- Lives in Berlin
- Cloud Engineer
  (future)

Wilhelm Thomas Röder

- Lives in Berlin
- Localization Engineer
  Data Engineer (future)



Shakural Islam

- Lives in Munich
- Software Engineer
  Data Engineer (future)

Manh

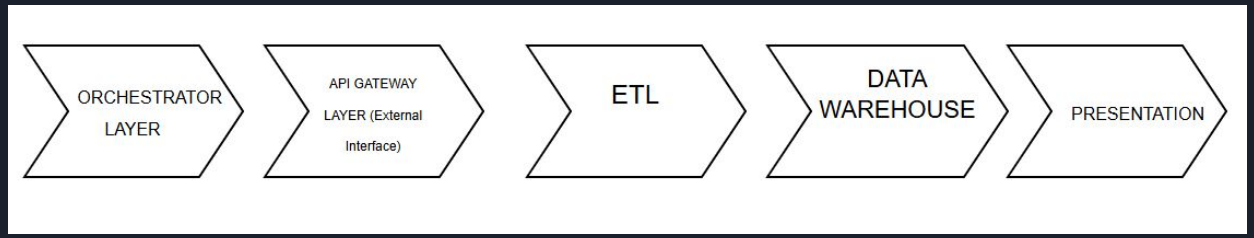
Conclusions

Sabine

- We built a fully automated ETL batch pipeline and a robust data foundation for flexible analytics.
- Throughout the project, we could apply and expand our knowledge in data engineering, orchestration, PostgreSQL modeling, API design, and dashboard development.
- We strengthened our team collaboration, communication, and project planning skills.
- Thank you for your attention!

Pipeline

Tom



Pipeline Overview 5 Steps

1. **Orchestrator (APScheduler)**
   Schedules and triggers the ETL pipeline automatically (daily at 2:00 AM) or on-demand.

2. **API Gateway (FastAPI)**
   Provides a **secure interface** to run ETL, expose analytics endpoints, and serve filter options to the dashboard.

3. **ETL Pipeline**
   Extracts job data from external APIs, cleans and normalizes it, enriches fields, loads raw/normalized/star-schema tables, and archives old data.

4. **Data Warehouse (Supabase PostgreSQL)**
   Stores structured analytics-ready data in raw, normalized, and star-schema layers, plus materialized views for reporting.

5. **Presentation Layer (Streamlit)**
   Interactive dashboards and analytics tools for end users, powered by the API and warehouse views.

Future Roadmap
(Move towards a cloud-native stack)

all

- Orchestration by Airflow
- Datalake (i.e. AWS S3) for raw and staged data storage
- Distributed etl processing (i.e. with Spark)
- Kubernetes for scalability
- ETL (currently) register logs manually -> CloudWatch/OpenSearch for monitoring

- ML-based enrichment: job classification, salary prediction
- Extend dashboard for job search