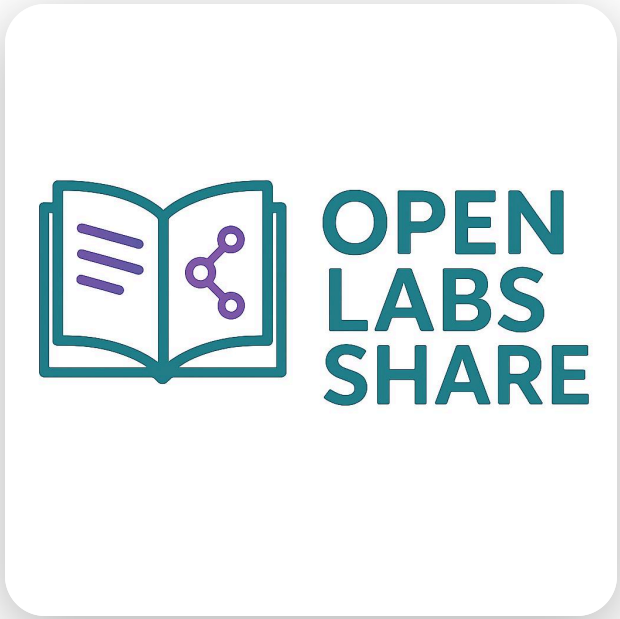


Open Labs Share

Next-Gen Learning Platform: Microservices Meets Education

Kirill Efimovich, Aleliya Turushkina, Mikhail Trifonov, Nikita Maksimenko, Timur Salakhov, Ravil Kazeev, Kirill Shumskiy

⚡ The Problem & Our Vision

Current challenges in tech education:

- 📚 **Fragmented Learning Resources**
- 🔄 **Weak Industry-Education Bridge**

Our solution - Open Labs Share:

> Peer-to-peer knowledge sharing platform, with comprehensive feedback system and interactive learning experience through labs and articles with AI-powered autograding.

🔰 Meet the team

- 🌉 **Kirill Efimovich (PM/DevOps)** - *Project Leadership & DevOps Engineer*
- 🔒 **Mikhail Trifonov** - *Backend Engineer*
- 🏗️ **Nikita Maksimenko** - *Backend Engineer*
- 📚 **Timur Salakhov** - *Backend Engineer*
- 💌 **Ravil Kazeev** - *Backend Engineer*
- 🤖 **Kirill Shumskiy** - *ML & Backend Engineer*
- 🎨 **Aleliya Turushkina** - *Designer & Frontend Engineer*

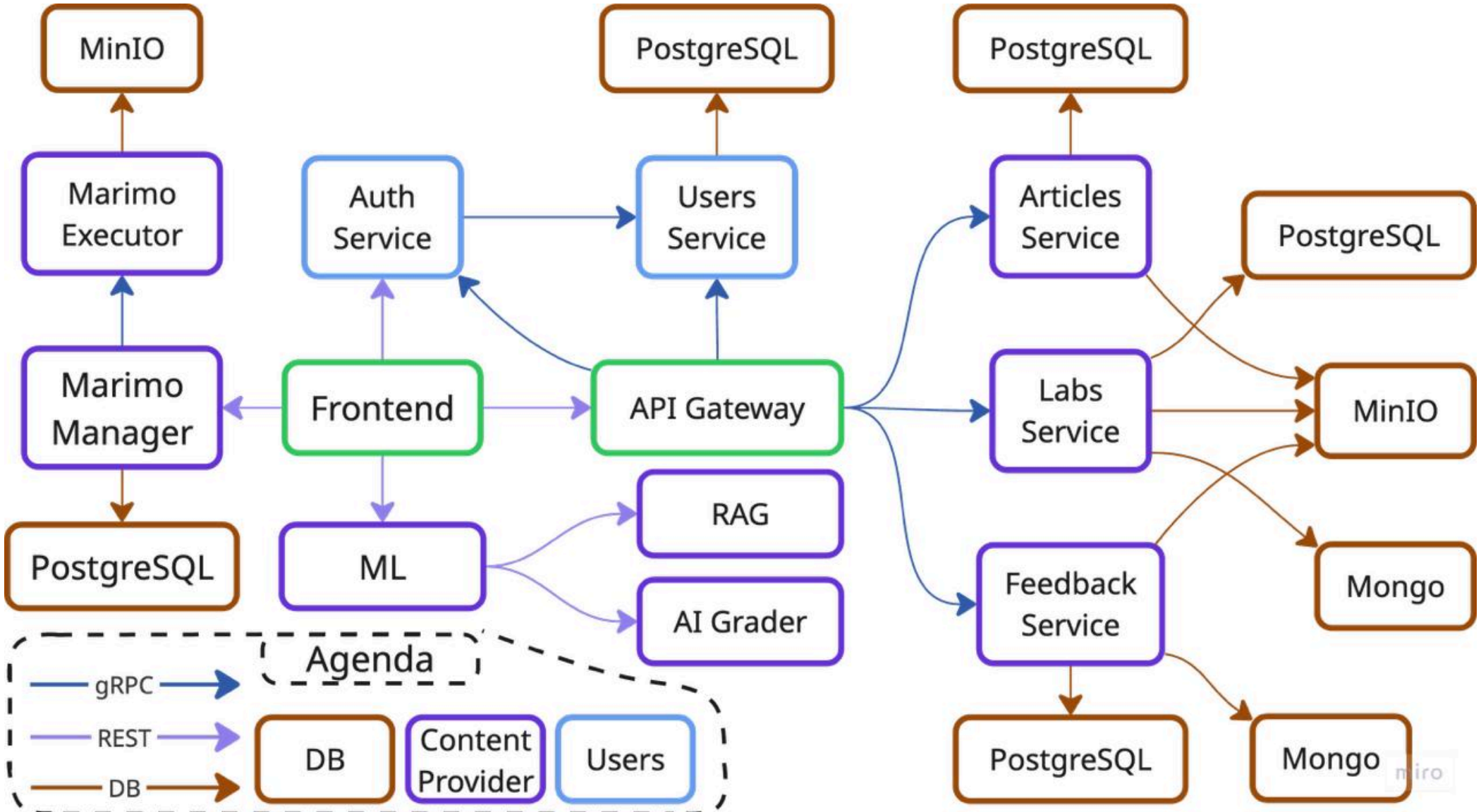
Live Technical Demo:

Live Demo Links:

- 🎥 Main Demo: Google Drive
- 🎬 Alternative: Yandex Disk

🌸 Frontend: Tech Stack & Connections

- ☀️ **Frontend:** React, Vite, Tailwind CSS, React Router
- 🌱 **Component Libraries:** React PDF Viewer, Markdown/KaTeX
- 🌍 **API Integration:**
  - Communicates with backend via REST API through the API Gateway
  - Auth, Labs, Articles, Submissions, Feedback, and ML services
  - Real-time and file download support from MinIO


MinIO
PostgreSQL
PostgreSQL
Marimo Executor
Auth Service
Users Service
Articles Service
PostgreSQL
Marimo Manager
Frontend
API Gateway
Labs Service
MinIO
PostgreSQL
ML
RAG
AI Grader
Feedback Service
Mongo
Agenda
gRPC
REST
DB
DB
Content Provider
Users
PostgreSQL
Mongo
miro

🔐 Authentication & Users Service 👥

Mikhail Trifonov (Backend Engineer)

🔐 Authentication Service

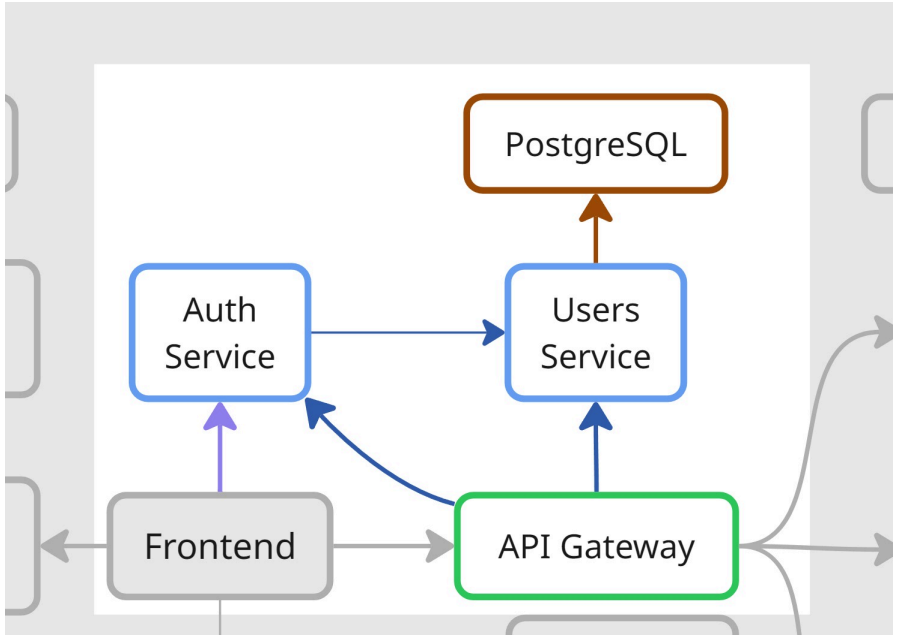
Handles all authentication flows and token lifecycle management for secure access control 🔑

👫 Users Service

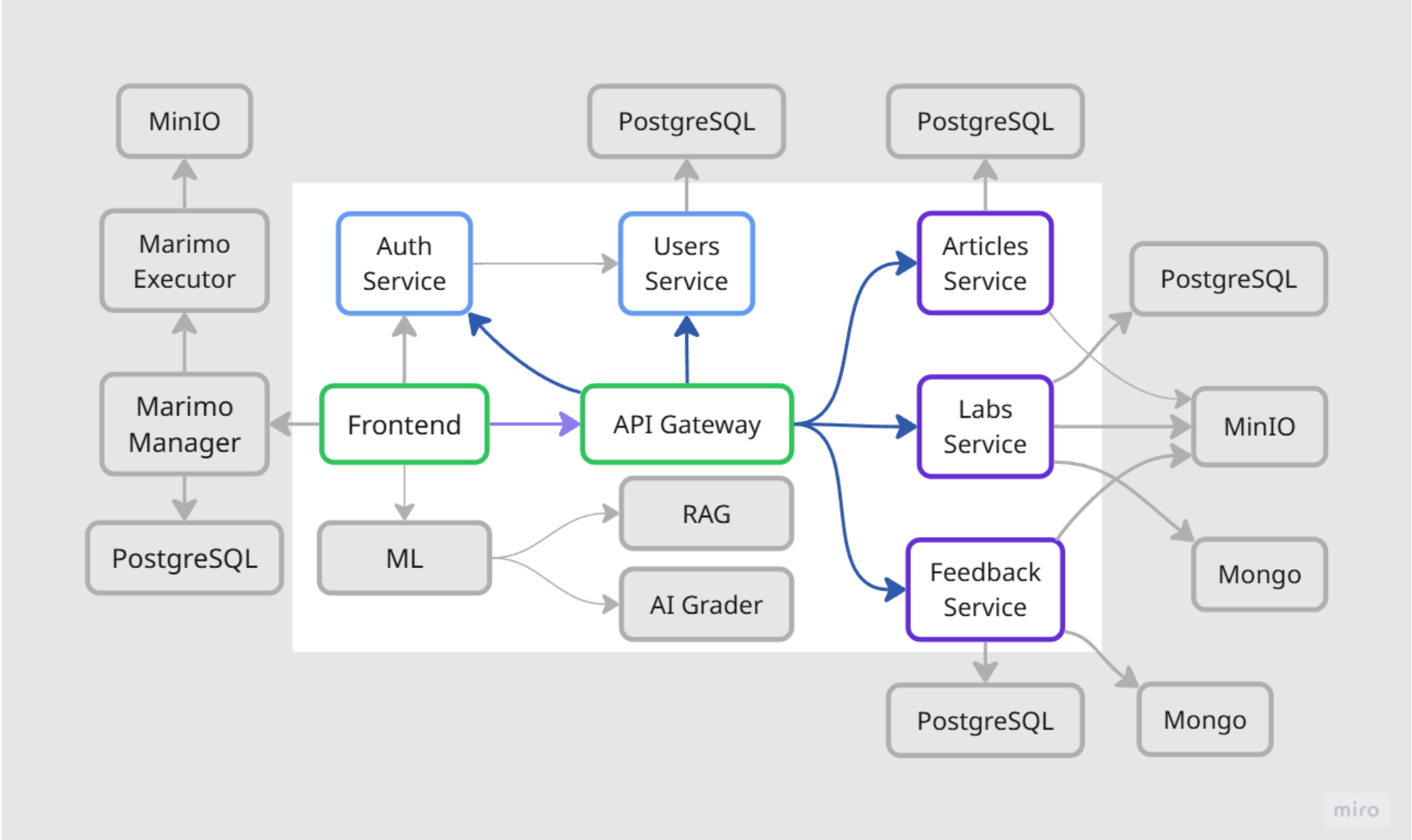
Manages all user data, credentials, and points for solving & reviewing labs 💸

☕ Tech Stack:

Java 21 + Spring Boot 3.5, Spring Security + JWT, Flyway



API Gateway


MinIO
PostgreSQL
PostgreSQL
Marimo Executor
Auth Service
Users Service
Articles Service
PostgreSQL
Marimo Manager
Frontend
API Gateway
Labs Service
MinIO
PostgreSQL
ML
RAG
AI Grader
Feedback Service
Mongo
PostgreSQL
Mongo
miro

API Gateway: Primary Use Case

Centralized entry point and request orchestration for all client interactions 🌐

- 🌐 **Centralized Entry Point:** Serves as the unified access layer for all client REST API requests
- 🔀 **Request Routing:** Directs incoming requests to the appropriate microservice (`auth` , `user` , `article` , `lab`) via gRPC
- 🔒 **Authentication & Security:** Validates JWT tokens and user's permissions
- 📝 **Cross-Cutting Concerns:** Handles logging, request tracing, and error handling for all API traffic
- 🧠 **Business Logic Execution:** Aggregating data and enforcing business rules beyond simple routing

API Gateway: Tech Stack & Connections

Java Spring Boot with REST-to-gRPC translation ☕🔄

- 👨‍💻 **Java 21 + Spring Framework:**

  ↳ REST API, gRPC, Jackson Validators, Spring AOP

- 📥 **REST API:**

  ↳ REST is the simplest and most widely supported method for web communication

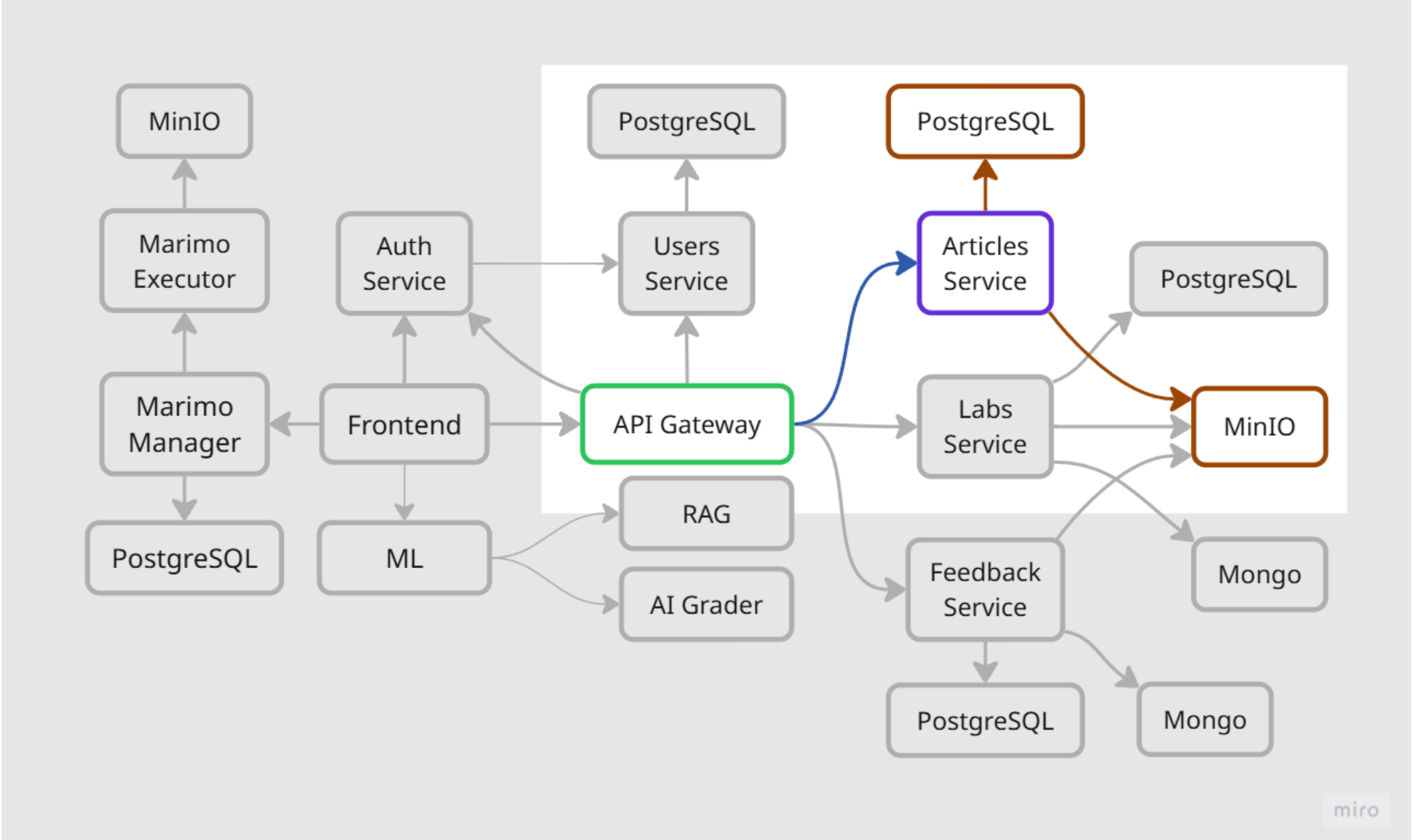
- 🛡️ **Security Layer:**

  ↳ Intercept incoming REST requests for authentication and authorization

- 🔀 **gRPC Client:**

  ↳ gRPC provides high-speed, type-safe, and scalable service-to-service communication

Articles Service



Articles Service: Primary Use Case

Manages all articles & assets metadata 🗄️

- 📝 **Articles Operations:** Provides CRUD for articles details
- 🗂️ **Content Management:** Handles articles assets in independent storage system
- ⚙️ **Metadata Management:** Organizes and updates metadata for articles and its assets
- 🔍 **Searching:** Provides articles searching based on its title and abstract

Articles Service: Tech Stack & Connections

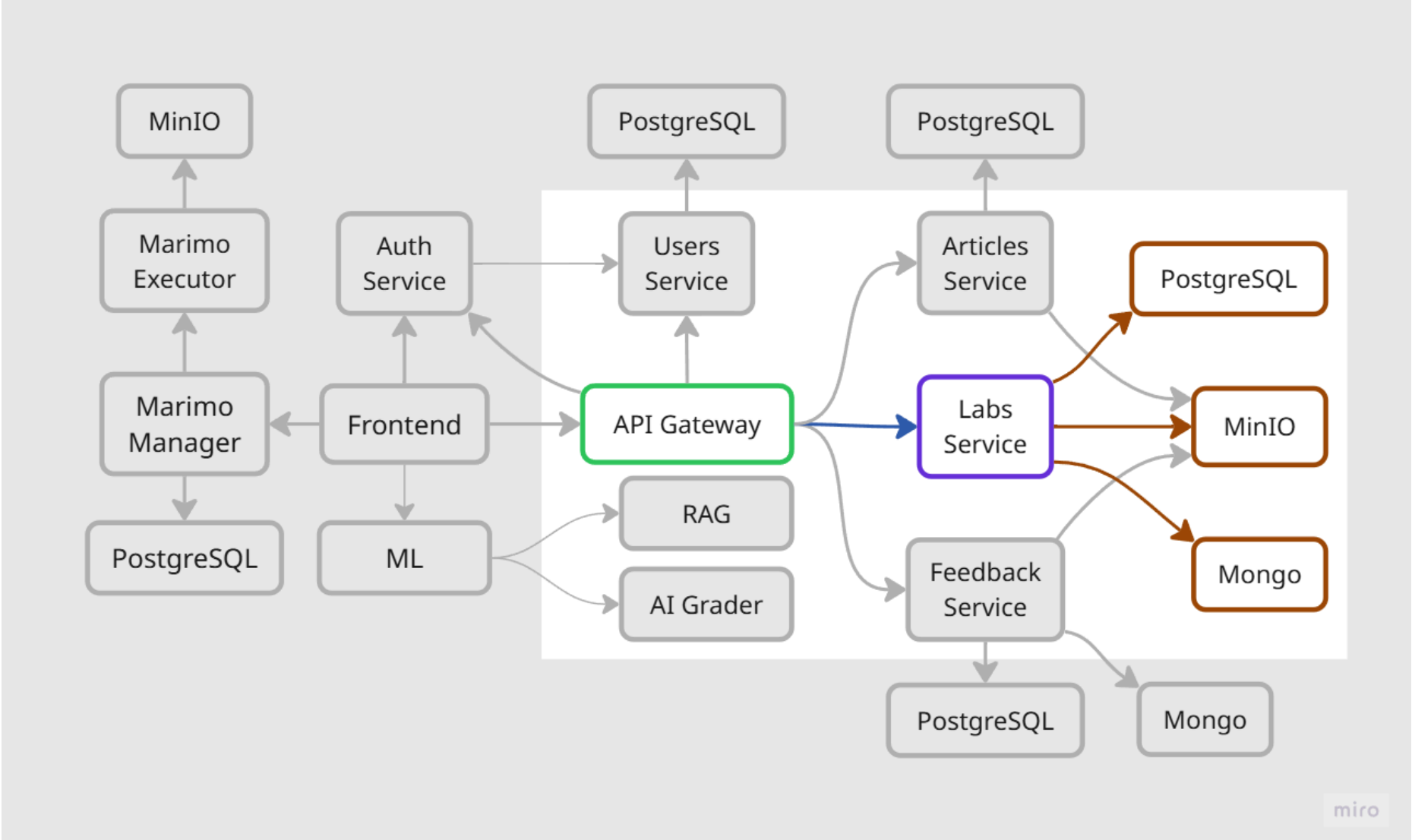
Python-based microservice with PostgreSQL and MinIO storage 🐍

- 🐍 **Programming Language:** Python 3.12
- 🔄 **Inter-service Communication:** gRPC

Service Integrations:

- 🚪 **API Gateway:** Receive and return data in gRPC format
- 🗄️ **PostgreSQL Database:** Store all articles and its assets metadata
- ☁️ **MinIO Storage System:** Store all articles assets

Labs Service



Labs Service: Primary Use Case

Manages all labs, submissions & educational content 🗄️

- 📚 **Labs Operations:** Provides CRUD for lab assignments with tags
- 📤 **Submissions Management:** Handles submissions with text content and file assets
- 🏷️ **Tag System:** Organizes labs with flexible tagging and search capabilities
- 📊 **Grading System:** Tracks submission status and grade workflow

Labs Service: Tech Stack & Connections

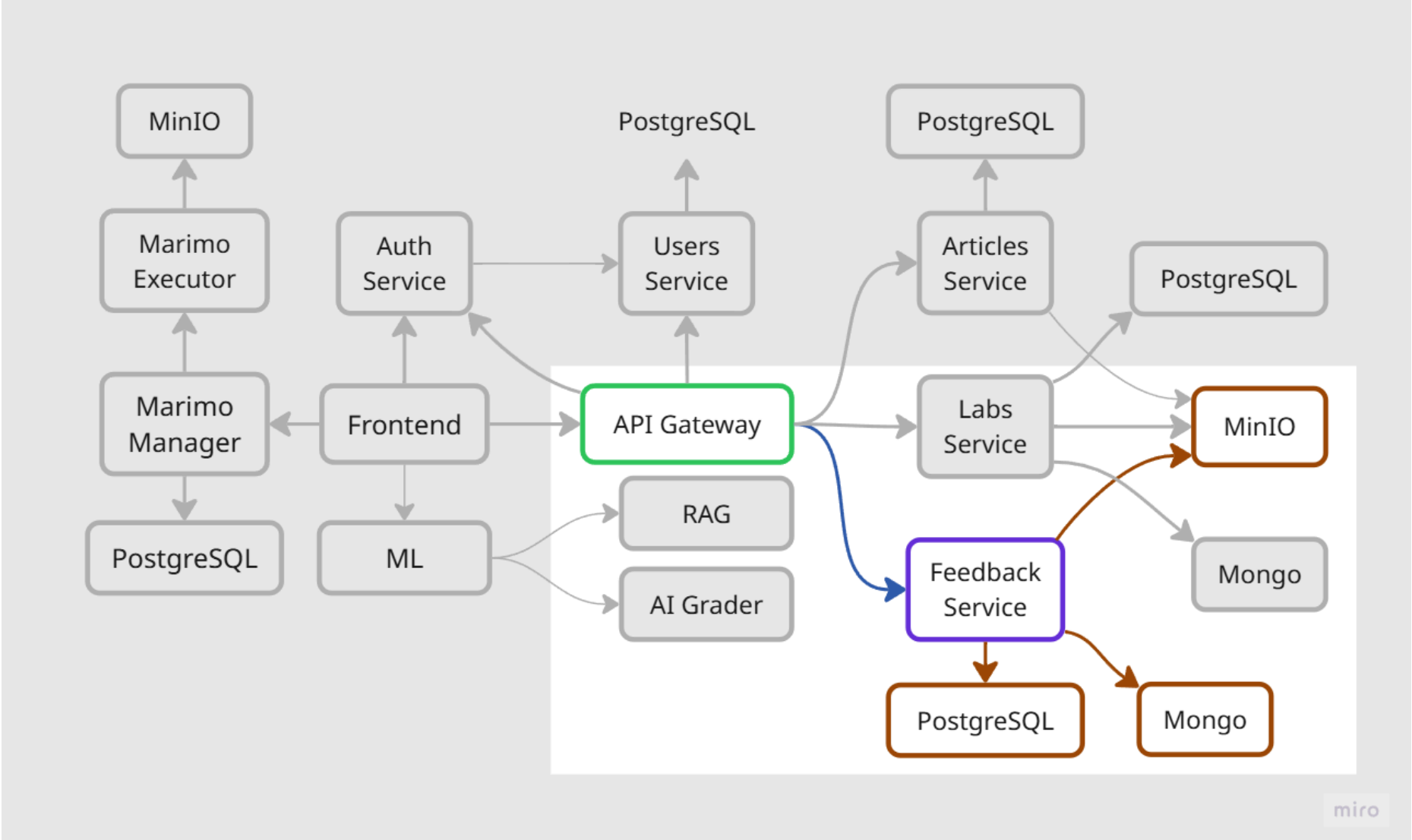
Python with hybrid database architecture and MinIO storage 🐍

- 🐍 **Programming Language:** Python 3.12
- 🔄 **Inter-service Communication:** gRPC

Service Integrations:

- 🚪 **API Gateway:** Single entry point for all requests
- 🗄️ **PostgreSQL Database:** Store labs, submissions, tags, and assets metadata
- 📄 **MongoDB Database:** Store submission text content for flexible storage
- ☁️ **MinIO Storage System:** Store lab and submission assets in organized buckets

💬 Feedback Service



Feedback Service: Primary Use Case

Comprehensive feedback and discussion management system 💬

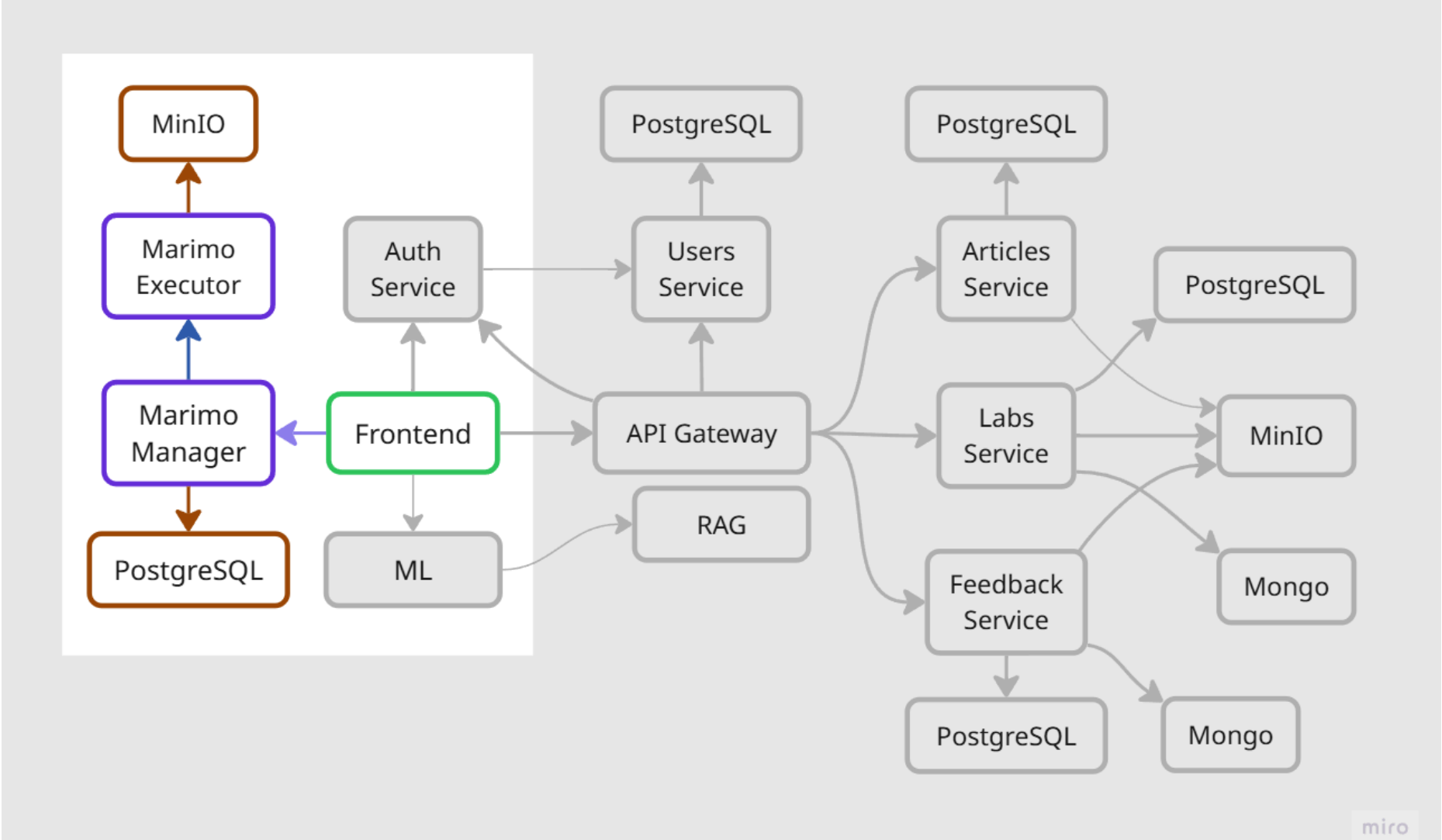
- 📝 **Comprehensive Feedback System:** Enables reviewers to create, update, and delete detailed feedback on submissions using Markdown for text and code formatting
- 💬 **Organized Discussion Section:** Powers a threaded commenting system for both labs and articles. Nested replies keep conversations structured and easy to follow
- 📎 **Attachment Handling:** Allows multiple file attachments per feedback entry, using efficient gRPC streaming to handle large uploads and downloads without high memory usage

Feedback Service: Tech Stack & Connections

Go with a multi-storage backend and gRPC API 🐹💾

- 🐹 **Go 1.24:**
  ↳ High-performance, concurrent service ideal for I/O-heavy tasks
- 🗣️ **gRPC Server:**
  ↳ Provides a typed API for feedback, comments, and file streaming
- 🗄️ **Multi-Storage Backend:**
  ↳ **PostgreSQL:** Stores structured feedback metadata
  ↳ **MongoDB:** Stores unstructured comments and feedback content
  ↳ **MinIO:** Object storage for all file attachments

Marimo Service



Marimo Service: Primary Use Case

- 👟 **Code Execution:** Real-time cell execution with output capture and error handling 🖐️
- 📊 **Asset Management:** Upload/download datasets and files for notebook use 🐪
- 🎛️ **Interactive Widgets:** Set of basic Marimo input widgets which value can be used in code (sliders, switchers, text fields, etc.) 📟
- 📂 **Cross-cells state memory:** Variables and modules from executed cells are available in other cells 📦

Marimo Service: Tech Stack & Connections

- 🔧 **Java Manager + Python Executor:**
  ↳ Java handles `REST API` and `metadata` while Python `executes` notebooks
- 🗄️ **PostgreSQL:**
  ↳ Tracks notebook metadata, user sessions, and execution trails with TTL cleanup
- 📦 **MinIO:**
  ↳ Object storage for notebook `files` and user-uploaded `assets`
- 🔗 **gRPC:**
  ↳ Java Manager ← `execute requests, session management` → Python Executor
- 🐍 **Marimo:** Interactive notebook execution with widgets
  ↳ Interactive notebook execution with ✨ `widgets` ✨

ML Service: Primary Use Case

Two powerful AI enhancements for the learning platform 🧠

- 🔍 **AI RAG Assistant:** Context-aware code and documentation helper, leveraging Retrieval-Augmented Generation (RAG) to deliver accurate, real-time support to students
- ✅ **Autograding:** Automated code assessment system for evaluating submissions instantly — ideal for learning platforms

ML Service: Tech Stack & Connections

FastAPI backend with specialized AI models and infrastructure 🐍🤖

🤖 AI RAG Assistant

- Qwen2.5-Coder-1.5B-Instruct (local inference)
- Qdrant vector store
- BAAI/bge-small-en-v1.5 embeddings

✅ Autograding

- deepseek-r1-distill-llama-70b (groq inference)
- Menagerie dataset: Graded CS1 Assignments for evaluation

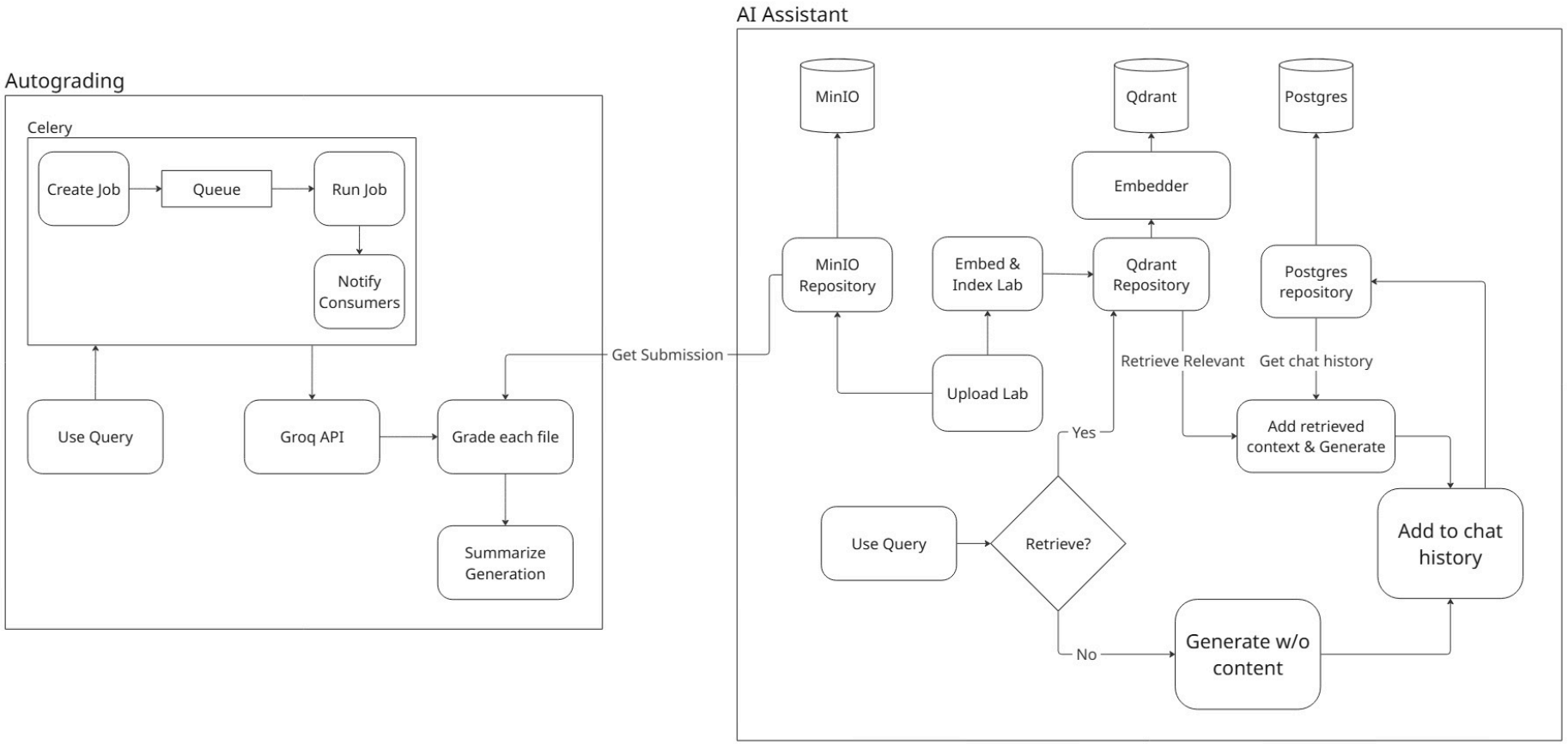
Core Architecture:

🐍 **FastAPI-based backend** with three-layer structure

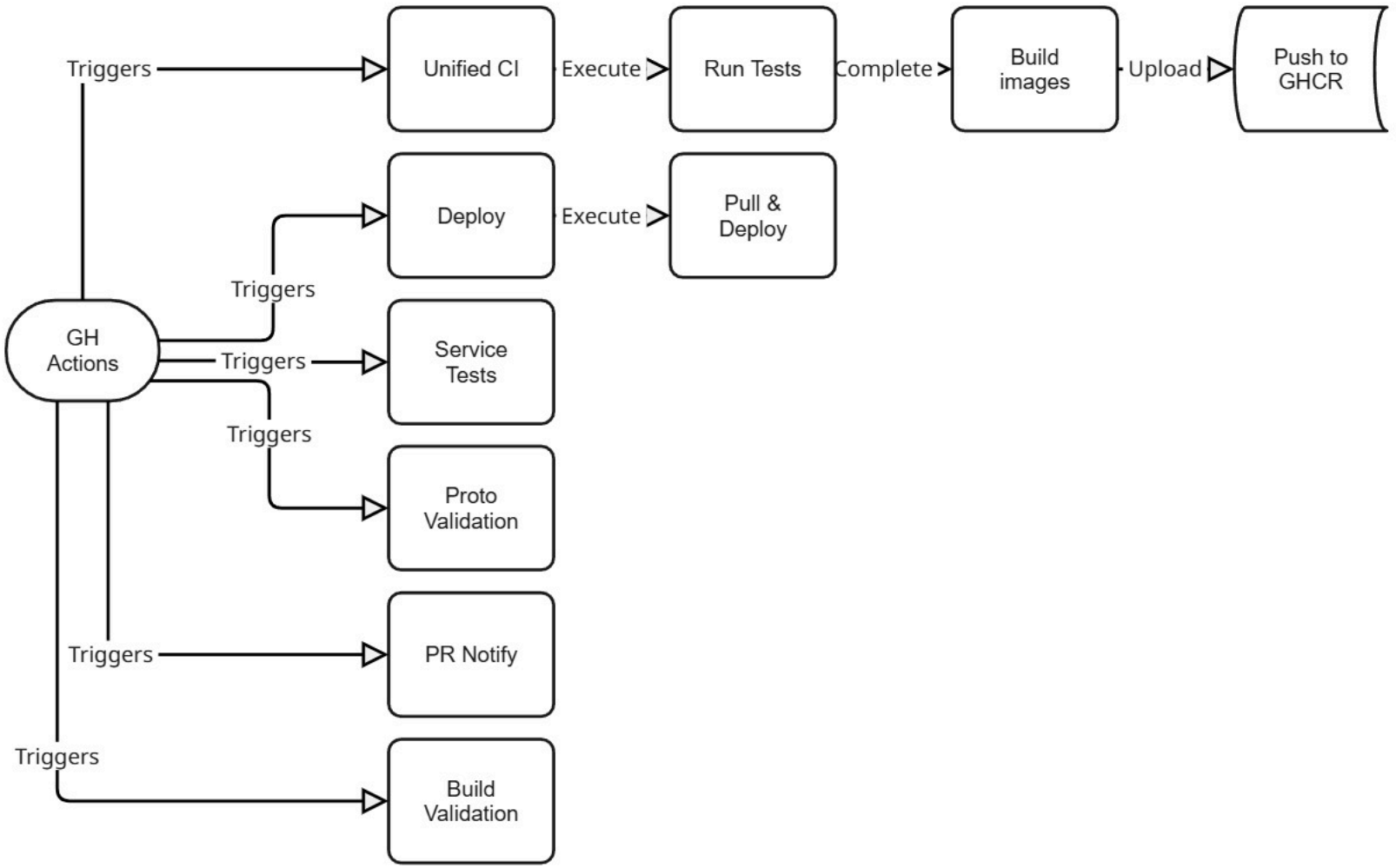
🥬 **Celery** for asynchronous tasks

🧵 **Redis** for caching and message broker

ML Service Architecture



DevOps & Infrastructure



🏛️ DevOps: Primary Use Case

Key GitHub Actions Workflows: 💫

- 🔧 **Compilation Validation:** Ensures all services compile
- 🪚 **Test Execution:** Runs unit & integration tests
- 🐳 **Docker Build Validation:** Buillds, validates and pushes images to GHCR
- ✈️ **Deployment Automation:** Handles the Blue-Green deployment logic
- 🔗 **Team help tools:** to automate issues managing and PR notifiers to keep the team perfectly synchronized

DevOps: Infrastructure

🔵 Green-Blue Strategy 🟢

- 0️⃣ **Zero Downtime:** Updates are seamless
- 🎞️ **Workflow:**
  i. Deploy new version (Green) alongside Production (Blue)
  ii. Test Green environment internally
  iii. Switch HAProxy to route traffic to Green
  iv. Keep Blue for instant rollback

🐧 Server & Networking

- **Host:** Self-managed server on Ubuntu 24.04
- **Specs:** 6-Core CPU, 16GB RAM, 240GB SSD
- **Proxy:** NGINX & HAProxy
- **Access:** CloudPub for public NAT traversal
- **Monitoring:** cAdvisor for container metrics

📺 Communication Problems

❌ Problems	✅ Solutions
❌ Problems in task setting and communication between people	✅ Create clear GitHub rules for issue creation, assignment workflows, and collaborative development processes
❌ Too many services that use the same data model	✅ Create scripts that automatically check data model consistency across all services

🏭 Implementation Problems

❌ Problems	✅ Solutions
❌ A single database was inefficient for managing varied data types.	✅ Used the best database for each job: PostgreSQL for metadata, MongoDB for comments, and MinIO for file attachments.
❌ University network NAT blocked access to self-hosted server.	✅ After issues with Cloudflare, we successfully used **CloudPub** to create a secure tunnel for public access.

🏃 Try it out!

Thank you!

We're glad to hear your questions! 🛒🤗🎸