EXPENSE TRACKER

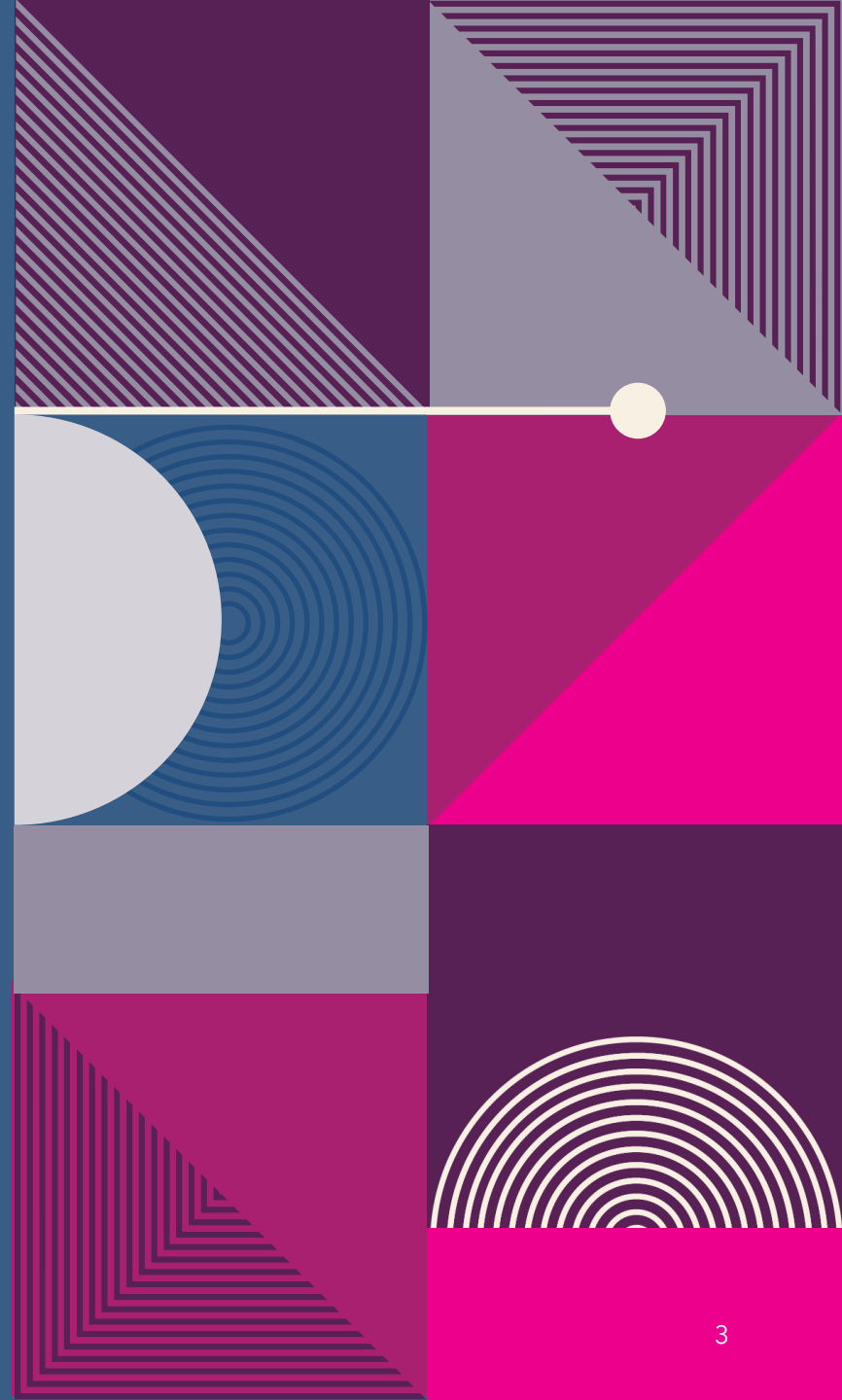
ANASTASIA TSATSOU

PROJECT OVERVIEW

- Developed a 2-tier microservices-based expense tracker
- Used Python (FastAPI) and MariaDB
- Managed via Git and Docker
- Automated CI/CD with GitHub Actions
- Application monitoring via a /metrics endpoint

SYSTEM ARCHITECTURE

- Frontend (API): FastAPI – CRUD for expenses
- Backend (Database): MariaDB – category & expense storage
- Diagram: 2-tier architecture with API ↔ DB



GIT & REPOSITORY STRUCTURE

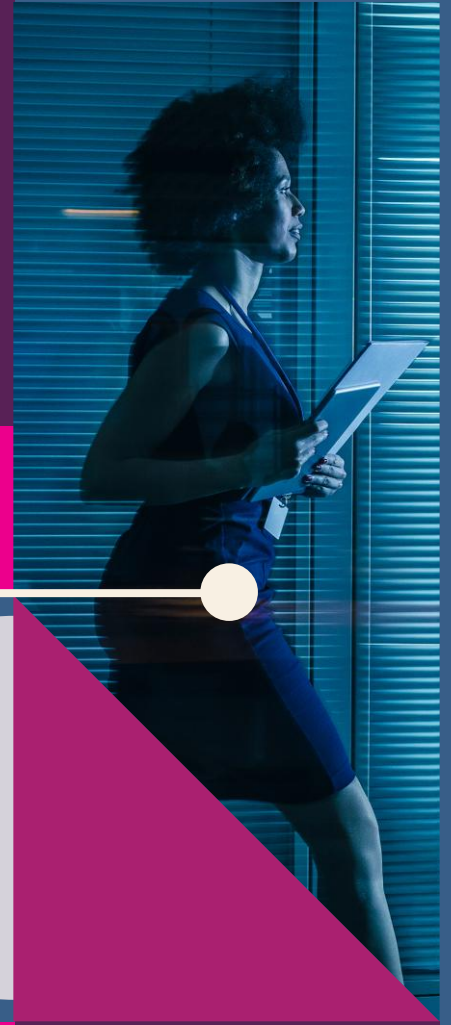
- GitHub repository with clear structure:
  - app/, database/, docker/, tests/
- Branching model: Git Flow
  - main, dev
- Commit history shows incremental development

DATABASE SCHEMA

- Single table: expenses
- Fields: id, amount, description, category, date, timestamps
- Schema stored in SQL (init.sql)

CONTAINERIZATION STRATEGY

- Dockerized FastAPI and MariaDB services
- docker-compose.yml used for orchestration
- Enables easy setup and deployment



CI/CD IMPLEMENTATION

- GitHub Actions used for automation
- CI: Code checkout, dependency install, pytest
- CD: Docker build and image testing
- Triggers: push, PRs, manual

TESTING STRATEGY

- Unit Tests: Using pytest
- Integration Tests: API and DB behavior
- API Tests: Validate endpoint behavior
- Tests included in CI workflow

MONITORING & OBSERVABILITY

- Exposed /metrics endpoint
- Request/response times, DB query counts, CPU/memory
- Helps track performance and health

KEY ACHIEVEMENTS

- Built a clean microservices app
- Git-based dev workflow with feature branching
- Full CI/CD automation with GitHub Actions
- Dockerized deployment
- Robust testing + monitoring

THANK YOU

Anastasia Tsatsou