Dokumentacja migracyjna
Aplikacja do korepetycji

Zespół Projektowy

- Jakub Kucharek
- Igor Osiakowski
- Mikołaj Lewandowski
- Mateusz Ługowski
- Michał Kruczyński

1 Wprowadzenie

Niniejszy dokument opisuje migracje aplikacji do korepetycji ze środowiska lokalnego do chmury Microsoft Azure. Poczatkowo była to aplikacja monolityczna uruchamiana lokalnie w kontenerze Docker przy użyciu `docker-compose`. W ramach modernizacji środowiska zdecydowano sie na migracje do architektury opartej na usługach chmurowych, co umożliwiło rozdzielenie komponentów, zwiekszenie elastyczności i lepsze zarzadzanie zasobami.

2 Środowisko poczatkowe

Pierwotna wersja aplikacji była zbudowana jako:

- Aplikacja o architekturze hybrydowej
- Zbudowana głównie w oparciu o framework Flask,
- Uruchamiana na jednej wirtualnej maszynie,
- Konteneryzowana i zarzadzana przy użyciu `docker-compose`,
- Rozdzielona na osobny serwis backendowy, frontendowy oraz usługe wysyłki e-maili, jednak wszystkie komponenty działały w ramach jednej wspólnej instancji środowiska.

3 Nowa architektura i środowisko docelowe

Po migracji do Microsoft Azure, aplikacja została podzielona na odrebne komponenty:

3.1 Wykorzystane technologie

- **Flask** — framework aplikacji webowej,
- **Flasgger** — generowanie dokumentacji API (Swagger),
- **SQLAlchemy** — mapowanie obiektowo-relacyjne (ORM),
- **Flask-JWT-Extended** — obsługa JWT,
- **APScheduler** — planowanie zadań cyklicznych.

3.2 Zasoby na platformie Azure

Wszystkie komponenty zostały wdrożone w jednej grupie zasobów Azure:

- **Backend** — kontenerowa aplikacja API (Azure Container App),
- **Email-service** — osobna aplikacja kontenerowa do obsługi e-maili,
- **Frontend** — aplikacja SPA hostowana jako Azure App Service,
- **projbezpieczenstwo_db** — zarzadzana baza danych PostgreSQL (Azure Database).

4 Powody i zalety migracji

Decyzja o migracji do Azure została podjeta ze wzgledu na kluczowe potrzeby projektu:

- **Skalowalność** — możliwość niezależnego skalowania poszczególnych usług,
- **Bezpieczeństwo** — lepsze zarzadzanie dostepem i danymi (np. Key Vault, App Service),
- **Modularność** — wyraźne rozdzielenie komponentów backendu ułatwiło rozwój i utrzymanie,
- **Niezawodność** — wysoka dostepność dzieki zarzadzanym usługom chmurowym,
- **Monitoring** — integracja z narzedziami Azure do logowania i diagnostyki.

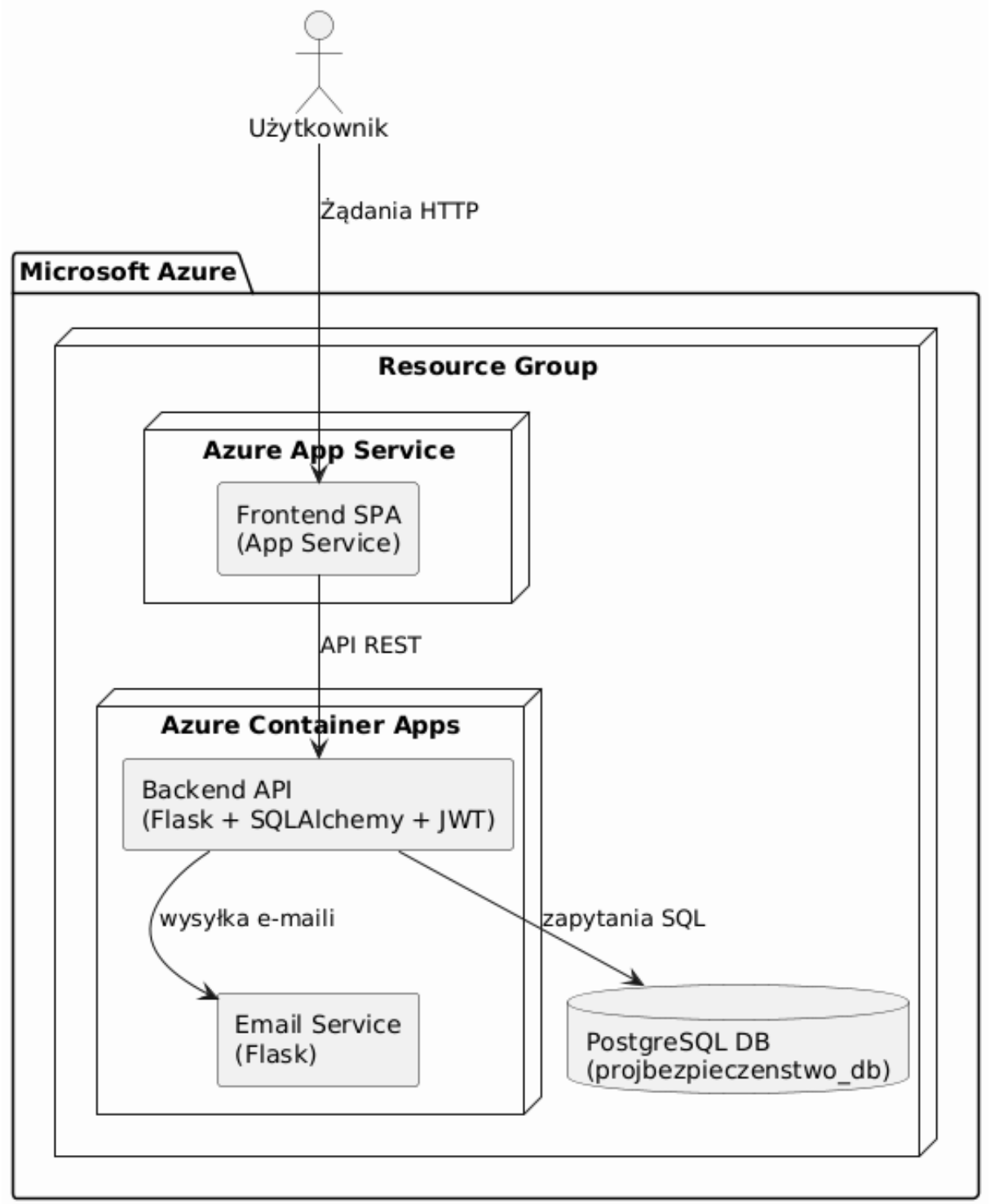
5 Proces migracji

Migracja przebiegła w kilku etapach:

1. Refaktoryzacja kodu i uporzadkowanie istniejacej architektury hybrydowej,
2. Stworzenie i skonfigurowanie kontenerów Dockera dla poszczególnych komponentów,
3. Utworzenie zasobów w Azure (baza danych, kontenery, App Service),
4. Konfiguracja zmiennych środowiskowych oraz integracja miedzy komponentami,
5. Wykonanie testów integracyjnych oraz wdrożenie środowiska produkcyjnego.

6 Diagram architektury systemu

Poniżej przedstawiono aktualna architekture aplikacji po migracji do platformy Azure.



Rysunek 1: Architektura aplikacji po migracji do Azure