Uniwersytet Gdański
Wydział Matematyki, Fizyki i Informatyki
Instytut Informatyki

Titanic App

Bartłomiej Wnuk

Projekt z przedmiotu technologie chmurowe na kierunku informatyka profil praktyczny na Uniwersytecie Gdańskim.

Gdańsk

27 czerwca 2024

Spis treści

1 Opis projektu

Australijski miliarder postanowił odbudować Titanica – nowy statek ma być oddany do użytku w 2027 roku. To właśnie wtedy bedzie można zapisać sie na rejs i przepłynać trase, która była zaplanowana na 1912 rok. Firma, która bedzie zajmować sie organizacja rejsu poprosiła o stworzenie aplikacji, która bedzie pozwalała na zapisanie sie na rejs oraz sprawdzenie szans na przeżycie gdyby okazało sie, że statek znowu uderzy w góre lodowa i zacznie tonać.

1.1 Opis architektury - 8 pkt

Architektura systemowa projektu opiera sie na podejściu mikroserwisów, co oznacza, że różne funkcjonalności aplikacji sa realizowane przez niezależne komponenty (mikroserwisy), które komunikuja sie ze soba. Aplikacja jest zbudowana z pieciu serwisów hostowanych na klastrze Kubernetes

1. Serwis frontendowy - interfejs użytkownika
2. Baza danych
3. Serwis backendowy - zarzadzanie baza danych
4. Serwis integracji z klientem uczenia maszynowego
5. Serwis autentykacji Keycloak

1.2 Opis infrastruktury - 6 pkt

1.2.1 Docker

Każdy mikroserwis jest zapakowany w osobny kontener Dockerowy. Kontenery zawieraja wszystkie zależności potrzebne do uruchomienia aplikacji.

1.2.2 Docker-compose

Służy do definiowania i uruchamiania wielokontenerowych aplikacji Dockerowych. Plik docker-compose.yml opisuje konfiguracje kontenerów, sieci i wolumenów dla lokalnego środowiska deweloperskiego.

1.2.3 Kubernetes

Aplikacja jest zaprojektowana do działania w środowisku Kubernetes, wykorzystujac Docker Desktop jako platforme deweloperska. Dzieki temu środowisku możliwa jest szybka orkiestracja kontenerów Docker, co jest idealne dla celów deweloperskich i testowych. Do zarzadzania klastrami Kubernetes używamy narzedzia ‘kubectl‘, co umożliwia efektywne wdrażanie, skalowanie oraz monitorowanie usług. Konfiguracja w Docker Desktop zapewnia izolowane sieci(każdy kontener ma swoja własna odseparowana sieć) i zarzadzanie pamiecia masowa za pomoca Persistent Volumes. Pozwalaja one na przechowywanie danych bez wzgledu na cykl życia poszczególnych podów.

1.3 Opis komponentów aplikacji - 8 pkt

1.3.1 Serwis frontendowy - interfejs użytkownika

Stworzony przy użyciu technologi React. Umożliwia logowanie istniejacych użytkowników. Zawiera również w sobie interfejs odpowiadajacy za wyświetlenie predykcji przeżycia. Konto administartora zawiera interfejs umożliwiajacy usuwanie użytkowników z bazy danych.

1.3.2 Baza danych

Stworzona przy użyciu Postgresql. Zawiera osoby jakie wzieły udział w wyprawie w 1912 roku oraz jest przygotowana na dodawanie nowych użytkowników, którzy chcieliby zapisać sie na rejs.

1.3.3 Serwis backendowy - zarzadzanie baza danych

Stworzony przy użyciu technologi FastAPI w Pythonie. Umożliwia wykonywanie operacji na członkach załogi przez integracje z baza danych - ich tworzenie, usuwanie oraz odczytywanie.

1.3.4 Serwis integracji z klientem uczenia maszynowego

Stworzone przy użyciu technologi FastAPI oraz Scikit Learn w pythonie. Zawiera wytrenowany model oraz API do odbierania danych wejściowych, które sa przekazywane do modelu w celu zwrócenia wyniku predykcji.

1.3.5 Serwis autentykacji Keycloak

Został użyty do zabezpieczenia frontendu i umożliwienia bezpiecznego logowania z uwzglednieniem rol użytkowników.

1.4 Konfiguracja i zarzadzanie - 4 pkt

1.4.1 Deploymenty

Każdy komponent aplikacji jest uruchamiany jako osobny Deployment w Kubernetes. Deploymenty pozwalaja na łatwe skalowanie i aktualizacje komponentów aplikacji.

1.4.2 ConfigMaps

Zarzadzanie w systemie odbywa sie za pomoca ConfigMap w Kubernetesie, co umożliwia wstrzykiwanie wartości per środowisko.

1.4.3 Persistent Volume Claims

Dane bazy danych sa przechowywane na Persistent Volume za pomoca Persistent Volume Claim - usługi, która jest żadaniem o przydzielenie zasobu przechowywania danych określonego w Persistent Volume.

1.5 Zarzadzanie błedami - 2 pkt

1.5.1 Aplikacja

Zarzadzanie błedami na poziomie aplikacji skupia sie na walidacji requestów HTTP, co pozwala na wczesne wykrycie i obsługe błednych danych.

1.5.2 Kubernetes

Dla deploymentu każda usługa korzysta z mechanizmu replikacji, który przekieruje do drugiego poda gdy pierwszy napotka problem.

1.6 Skalowalność - 4 pkt

Skalowanie horyzontalne - pozwala na automatyczne skalowanie liczby replik kontenerów w zależności od obciażenia.Liczba replik w deploymentach serwisów została zwiekszona do dwóch co pozwala na lepsze rozłożenie obciażenia dla każdego z komponentów aplikacji.

1.7 Wymagania dotyczace zasobów - 2 pkt

1.8 Architektura sieciowa - 4 pkt

1.8.1 Protokół http

- komunikacja frontendu z keycloakiem
- komunikacja frontendu z backendem

1.8.2 Protokół bazodanowy postgresql

- komunikacja backendu z baza danych

1.8.3 Protokół OAuth2 z keycloak

- zarzadzanie sesja użytkownika oraz jej autoryzacja

Literatura