Raport z negocjacji

Marek Kacprzak

11 października 2024

Contents

1 Podsumowanie **1**

2 Technologia **1**

3 Opis działania aplikacji **2**
3.1 Główne widoki 2
3.2 Logika działania 3
3.3 Mechanizmy 3
3.4 Możliwe wyzwania i rozwiązania 3
3.5 Pytania 4

4 Sekcja techniczna **4**
4.1 Podział projektu 4
4.1.1 Scraper 4
4.1.2 Baza danych 4
4.1.3 API 4
4.1.4 Frontend 5
4.1.5 Dokumentacja 5
4.1.6 Testowanie 5

1 Podsumowanie

Zadaniem zespołu jest opracowanie aplikacji, która pozwala na wgląd w osiągnięcia naukowe pracowników akademickich. Aplikacja będzie prezentować różnorodne dane, takie jak obszary badawcze, publikacje oraz wskaźniki bibliometryczne poszczególnych naukowców. Dodatkowo, system umożliwi porównywanie dorobku między naukowcami.

2 Technologia

Aplikacja będzie działać w przeglądarce internetowej i zostanie oparta na następujących technologiach:

- **Golang** - Język programowania do tworzenia API.
- **React** - Biblioteka JavaScript do tworzenia interfejsów użytkownika.
- **Python** - Język programowania do tworzenia skryptów scrapujących.
- **PostgreSQL** - System zarządzania relacyjnymi bazami danych.
- **DigitalOcean** - Dostawca usług chmurowych.
- **Docker** - Narzędzie do virtualizacji aplikacji.

3 Opis działania aplikacji

3.1 Główne widoki

- **Strona główna** - Zawiera wyszukiwarkę pozwalającą na znalezienie pracowników naukowych według różnych kryteriów. Na stronie tej wyświetlani będą też naukowcy o najwyższej liczbie publikacji oraz najlepiej wypadający pod względem wskaźników bibliometrycznych.
- **Widok pracownika** - Prezentuje szczegółowe informacje o naukowcu, w tym jego obszary badań, wskaźniki bibliometryczne oraz publikacje. Użytkownik może również przejść do porównania z innymi naukowcami.
- **Widok porównania** - Umożliwia zestawienie dorobku naukowców według różnych filtrów, takich jak stopień naukowy czy dziedzina badań. Aplikacja wygeneruje graficzną wizualizację wyników porównania.
- **Widok dziedzin** - Prezentuje listę dziedzin naukowych oraz naukowców z danej dziedziny. Użytkownik może wybrać dziedzinę, aby zobaczyć listę naukowców.
- **Widok publikacji** - Wyświetla listę publikacji naukowca wraz z informacjami takimi jak liczba cytowań czy czasopismo naukowe.
- **Widok statystyk** - Prezentuje ogólne statystyki dotyczące naukowców, takie jak średnia liczba publikacji czy średni wskaźnik cytowań.
- **Konto przeglądającego** - Umożliwia zalogowanie się do aplikacji, co pozwala na zapisywanie ulubionych naukowców oraz dostęp do dodatkowych funkcji.
- **Widok ustawień** - Pozwala na dostosowanie ustawień aplikacji, takich jak język czy powiadomienia.
- **Widok pomocy** - Zawiera informacje o aplikacji, w tym instrukcje obsługi i odpowiedzi na najczęściej zadawane pytania.
- **Widok o aplikacji** - Zawiera informacje o autorach aplikacji, źródłach danych oraz sposobie działania aplikacji.

- **Widok kontaktu** - Pozwala na kontakt z zespołem odpowiedzialnym za aplikację.

3.2 Logika działania

- Aplikacja będzie regularnie zdzierać dane ze strony `bw.sggw.edu.pl` , oczyszczać je, a natępnie aktualizować swoją bazę danych.
- Po wpisaniu odpowiednich informacji i ustawieniu filtrów, użytkownik będzie mógł wyszukać pracownika SGGW, a dane zostaną pobrane z bazy i wyświetlone w widoku naukowca.
- Aplikacja pozwoli także na porównanie dorobku kilku naukowców, z odpowiednią wizualizacją wyników.
- Użytkownik będzie mógł zalogować się do aplikacji, co pozwoli na zapisywanie ulubionych naukowców oraz dostęp do dodatkowych funkcji.

3.3 Mechanizmy

- **Mechanizm pobierania danych** - Odpowiada za regularne zbieranie danych ze strony `bw.sggw.edu.pl`.
- **Mechanizm aktualizacji danych** - Umożliwia aktualizację bazy danych w chmurze po każdej operacji pobierania danych.
- **Mechanizm wyszukiwania** - Pozwala na wyszukiwanie pracowników według filtrów z tolerancją na błędy użytkownika.
- **Mechanizm wizualizacji** - Przetwarza dane z bazy i przedstawia je w formie graficznych podsumowań i porównań.

3.4 Możliwe wyzwania i rozwiązania

- **Problemy z pobieraniem danych** - Zmiana struktury strony może spowodować, że skrypty przestaną działać. Możliwe rozwiązanie to wprowadzenie systemu powiadomień dla osób odpowiedzialnych za utrzymanie aplikacji.
- **Trudności w porównywaniu dorobku** - Porównanie naukowców z różnych dziedzin może być trudne ze względu na różnice w ich wskaźnikach bibliometrycznych. Rozwiązaniem może być dodanie sekcji wyjaśniającej metodę obliczania wskaźników i specyfikę różnych dziedzin naukowych.
- **Wydajność przy dużych zbiorach danych** - Aplikacja może działać wolno przy dużej liczbie danych. Konieczne jest wdrożenie optymalnych algorytmów wyszukiwania i sortowania.

3.5 Pytania

1. Jak najlepiej porównywać naukowców z różnych dziedzin?
2. Kim będą główni użytkownicy aplikacji?
3. Czy aplikacja będzie dostępna dla ogółu, czy tylko dla klienta?
4. Czy interfejs powinien obsługiwać różne wersje językowe?
5. Czy aplikacja powinna zbierać dane o wyszukiwaniach i generować statystyki?
6. Czy interfejs powinien posiadać określoną paletę kolorów?
7. Jakie wskaźniki bibliometryczne są najistotniejsze?
8. Czy znaczenie wskaźników zależy od dziedziny naukowca?
9. Czy aplikacja powinna umożliwiać porównanie więcej niż dwóch naukowców jednocześnie?
10. Jak często baza danych powinna być aktualizowana?

4 Sekcja techniczna

4.1 Podział projektu

Na podstawie wymagań projektowych podzielono projekt na poniższe komponenty:

4.1.1 Scraper

- **Python: Beautiful Soup** lub **Scrapy**
- **JavaScript: Puppeteer**
- Scraper będzie odpowiadał za przeglądanie strony i zbieranie danych, które następnie zostaną zapisane w bazie danych.

4.1.2 Baza danych

- **PostgreSQL** - Brak dodatkowych uwag.

4.1.3 API

- API będzie pośredniczyć między bazą danych a frontendem użytkownika.
- Możemy napisać API w Pythonie, Go, Rust, lub, ewentualnie, w C#, choć ostatnia opcja nie jest preferowana.

4.1.4 Frontend

- Możliwe są dwa podejścia: tradycyjne z użyciem HTML, CSS i JS lub nowoczesne z wykorzystaniem frameworka, np. **React**, **Angular** czy **Vue**.
- Frontend będzie odpowiedzialny za prezentację danych z bazy i umożliwi użytkownikowi filtrowanie informacji.

4.1.5 Dokumentacja

- Dokumentację można generować automatycznie tam, gdzie jest to obsługiwane, np. za pomocą **OpenAPI**, lub przygotować ręcznie w plikach `.yml`.

4.1.6 Testowanie

- **Testowanie API** - Testy jednostkowe.
- **Testowanie frontendu** - Testy integracyjne.
- **Testowanie scrapera** - Sposób testowania będzie ustalony później.