

Zastosowania informatyki w gospodarce (projekt)

Termin zajęć: czwartek, 13:15-15:00

Prowadzący: dr inż. Tomasz Walkowiak

Mapa literacka (geolokalizacja na podstawie danych z Linera)
dla korpusów (zmienność w czasie, np. rozdziały) (PL) -
dokumentacja projektu

Michał Bańka
235051

Przemysław Jedlikowski
234927

Witold Marciniak
226194

Wojciech Mielczarek
218393

Marcin Pawluś
235555

czerwiec 2020

Spis treści

1	Wprowadzenie	4
2	Wymagania projektowe	5
2.1	Aplikacja webowa	5
2.2	Architektura projektu	5
3	Technologie	6
3.1	Liner	6
3.2	Spring Framework	6
3.3	Vue.js	7
3.4	Vuetify	7
3.5	MongoDB	7
3.6	Nominatim	7
3.7	OpenLayer	8
3.8	Docker i docker-compose	8
4	Implementacja	9
5	Wygląd i działanie aplikacji	11
6	Napotkane problemy	15
7	Podsumowanie	16

Spis rysunków

1	Diagram sekwencji czynności - tworzenie nowego projektu.	10
2	Menu główne aplikacji.	11
3	Otwieranie istniejącego projektu.	11
4	Tworzenie nowego projektu.	12
5	Proces tworzenia nowego projektu w trakcie przetwarzania.	12
6	Punkty naniesione na mapę.	13
7	Okno zmiany współrzędnych dla lokalizacji.	13
8	Skrócony opis projektu i autorzy.	14

1 Wprowadzenie

Niniejszy projekt powstał w ramach przedmiotu „Zastosowania informatyki w gospodarce” prowadzonego przez dr. inż. Tomasza Walkowiaka. Realizuje temat „Mapa literacka (geolokalizacja na podstawie danych z Linera) dla korpusów (zmiennosc w czasie, np. rozdziały) (PL)”.

Projekt ma za zadanie ułatwić analizę tekstów, a jego głównymi użytkownikami będą naukowcy nauk humanistycznych. Pozwala on na tworzenie nowych projektów (analiz tekstów) oraz przechowywanie ich do późniejszych analiz przez użytkownika, wizualizuje przetworzone informacje poprzez naniesienie punktów na mapę, pozwala na modyfikację przetworzonych danych przez użytkownika oraz na rozszerzenie istniejącego słownika poprzez dodanie nowych nazw lokalizacji.

Przetwarzanie wprowadzonych danych może trwać kilka minut lub kilka godzin, a cały proces nie jest responsywny. Przykładowe użycie aplikacji powinno wyglądać następująco:

- użytkownik wgrywa korpus tekstów - archiwum zip z plikami tekstowymi zawierającymi kolejne rozdziały (fragmenty) dzieła literackiego,
- następuje przetworzenie korpusu przy użyciu narzędzia *Liner* - rozpoznawane są (m. in.) nazwy lokalizacji geograficznych,
- pozyskiwane są współrzędne geograficzne rozpoznanych miejsc przy użyciu usługi *Nominatim*,
- użytkownik może dodać nową nazwę geograficzną do listy rozpoznanych miejsc,
- zebrane dane są wizualizowane na mapie.

Projekt został stworzony w języku Java z użyciem frameworka *Spring* (część serwerowa) oraz frameworka *Vue.js* (patrz: rozdział 3). Dane z Linera są przechowywane w nierelacyjnej bazie danych *MongoDB*. Wykorzystano narzędzie *Liner* służące do analizy tekstu (lub tekstów) oraz uzyskania nazw własnych lokalizacji znajdujących się w tekście, jak również usługę *Nominatim* do uzyskania współrzędnych geograficznych lokalizacji w postaci pliku JSON na podstawie ich nazw własnych.

2 Wymagania projektowe

W niniejszym rozdziale znajduje się spis wymagań stawianych poszczególnym elementom tworzonej aplikacji oraz całemu projektowi.

2.1 Aplikacja webowa

- a) Ułatwia użytkownikom analizę tekstów; głównymi użytkownikami aplikacji będą naukowcy nauk humanistycznych.
- b) Pozwala na tworzenie nowych projektów (analiz tekstów) oraz przechowywanie ich do późniejszej analizy przez użytkownika.
- c) Wizualizuje przetworzone informacje poprzez naniesienie punktów na mapę.
- d) Pozwala na wizualizację danych dla użytkownika w zadanym przedziale czasowym.
- e) Może przetwarzać wprowadzone dane kilka minut lub kilka godzin, a proces ten nie musi być responsywny (nie mniej powinien działać się w tle, nie wymagając wciąż otwartego okna aplikacji).
- f) Powinna być responsywna w procesie modyfikacji danych oraz ich wizualizacji.
- g) Powinna pozwalać na dodawanie, usuwanie i edytowanie przez użytkownika słownika z danymi lokalizacji.

2.2 Architektura projektu

- a) Aplikacja powinna zostać zbudowana w sposób umożliwiający jej konteneryzację.
- b) Każdy projekt stworzony przez użytkownika musi posiadać unikalny identyfikator.
- c) W bazie danych nie mogą być przechowywane całe teksty, lecz tylko tokeny lokalizacji zwrócone przez Linera.
- d) Implementując projekt, należy przygotować możliwość wprowadzenia uwierzytelniania i autoryzacji użytkowników w przyszłości.

3 Technologie

Realizując niniejszy projekt, użyto wiele różnego rodzaju technologii. W tym rozdziale opisano najważniejsze z nich i wskazano ich użycie w implementowanej aplikacji.

3.1 Liner

Liner jest usługą powstałą w ramach projektu CLARIN-PL, służącą do rozpoznawania nazw własnych i wyrażeń temporalnych. Użyte w jej ramach narzędzia to:

- a) konwerter plików do tekstu *Apache Tika*,
- b) analizator morfologiczny *Morfeusz 2* ze słownikiem SGJP,
- c) tager *WCRFT2* (DEMO),
- d) narzędzie do wyznaczania nazw własnych *Liner2*.

3.2 Spring Framework

Spring Framework jest szkieletem tworzenia aplikacji w języku Java dla platformy Java Platform. Powstał na bazie kodu opublikowanego w książce Roda Johnsona pt. "Design and Development" jako alternatywa dla programowania aplikacji z użyciem Enterprise JavaBeans. Programowanie z użyciem EJB narzucało wiele ograniczeń – wymagając między innymi przyjęcia określonego modelu tworzenia oprogramowania. Funkcjonalność EJB okazała się także „za ciężka” do wszystkich zastosowań (w małych projektach wykorzystywano tylko niewielką część oferowanej przez EJB funkcjonalności), a stworzenie małej aplikacji w środowisku EJB wymagało nakładu pracy jak przy aplikacji dużej. Odmienna koncepcja Springa – lekkiego szablonu, który nie wymusza specyficznego modelu programowania, stała się bardzo popularna wśród programistów Javy. Spring Framework oferuje dużą swobodę w tworzeniu rozwiązań, a jednocześnie jest dobrze udokumentowany i zawiera rozwiązania wielu, często występujących w programowaniu problemów. Podczas gdy bazowe komponenty Springa mogą być używane praktycznie w każdej aplikacji, istnieje w nim wiele rozszerzeń, które pozwalają budować aplikacje webowe na bazie Java EE [4].

Spring Framework został użyty w celu implementacji aplikacji serwerowej.

3.3 Vue.js

Vue.js to biblioteka pozwalająca na tworzenie prostych, składających się z komponentów, aplikacji webowych opartych o architekturę MVVM (ModelView View Model). Wśród dostępnych w niej mechanizmów można wskazać template’owanie kodu czy powiązywanie danych z elementami DOMu. Ogromnym plusem Vue.js jest to, że jest bardzo kompaktowy i intuicyjny [6].

Vue.js został użyty do stworzenia aplikacji klienckiej.

3.4 Vuetify

Vuetify jest frameworkiem dla Vue.js, który ma na celu zapewnienie czystych i semantycznych komponentów wielokrotnego użytku. Obsługuje wszystkie nowoczesne przeglądarki i jest kompatybilny z *Vue CLI-3*. Oferuje również podstawowe szablony dla *Simple HTML*, *Webpack*, *Nuxt*, *PWA*, *Electron*, *A La Carte*, *Apache Cordova* [7].

Framework Vuetify do stworzenia komponentów wyświetlanych w aplikacji klienckiej.

3.5 MongoDB

MongoDB jest otwartym, nierelacyjnym systemem zarządzania bazą danych napisanym w języku C++. Charakteryzuje się dużą skalowalnością, wydajnością oraz brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego dane składowane są jako dokumenty w stylu JSON, co umożliwia aplikacjom bardziej naturalne ich przetwarzanie, przy zachowaniu możliwości tworzenia hierarchii oraz indeksowania. Interfejsy programistyczne pozwalające obsługiwać bazy MongoDB powstały dla wszystkich wiodących języków programowania, w tym dla C, C++, Javy, PHP, Perla, Pythona i Ruby. Wśród zalet i możliwości MongoDB warto wymienić jednoodrodne wsparcie dla systemu Unicode, dużą liczbę obsługiwanych typów danych, indeksowanie, wsparcie dla agregacji danych, możliwość składowania plików w bazie czy architekturę zaprojektowaną z myślą o łatwej replikacji [2].

MongoDB zostało użyte jako silnik bazy danych w celu przechowywania informacji związanych z projektem: rozpoznanych słów czy ich współrzędnych geograficznych.

3.6 Nominatim

Nominatim (z łaciny „według nazwy”) to narzędzie do wyszukiwania danych OpenStreetMap (OSM) według nazwy i adresu (geocoding) oraz do generowania syntetycznych adresów punktów OSM (reverse geocoding). Można go znaleźć na stronie nominatim.openstreetmap.org [1].

Usługa Nominatim została użyta w tym właśnie celu. W podsumowaniu projektu umieszczono informację o problemach wynikających z jej zastosowania.

3.7 OpenLayer

OpenLayer jest biblioteką napisaną w języku JavaScript, ułatwiającą dodawanie dynamicznych map na stronach internetowych. Udostępniana jest na otwartej licencji BSD. Obsługuje formaty danych KML, GML, GeoJSON; implementuje standardy *OGC Web Map Service* i *Web Feature Service* [3].

Biblioteka ta została użyta w celu prezentowania mapy.

3.8 Docker i docker-compose

Docker jest oprogramowaniem tworzącym bardzo użyteczny „wrapper” na specyficznej cesze jądra systemu Linux – przestrzeniach nazw, tworząc kontenery przeznaczone do uruchamiania konkretnych aplikacji. Tworzone w Dockerze kontenery opierają się na szablonach stanowiących podstawę ich utworzenia. Uruchamiając kontener, tworzona jest zapisywalna warstwa systemu plików, nie powodując modyfikacji plików szablonu. Gotowe szablony (zarówno „czystych” systemów, jak i całych aplikacji skonfigurowanych do użycia w filozofii *plug and play*) można pobrać z publicznego repozytorium *Docker Hub* [5].

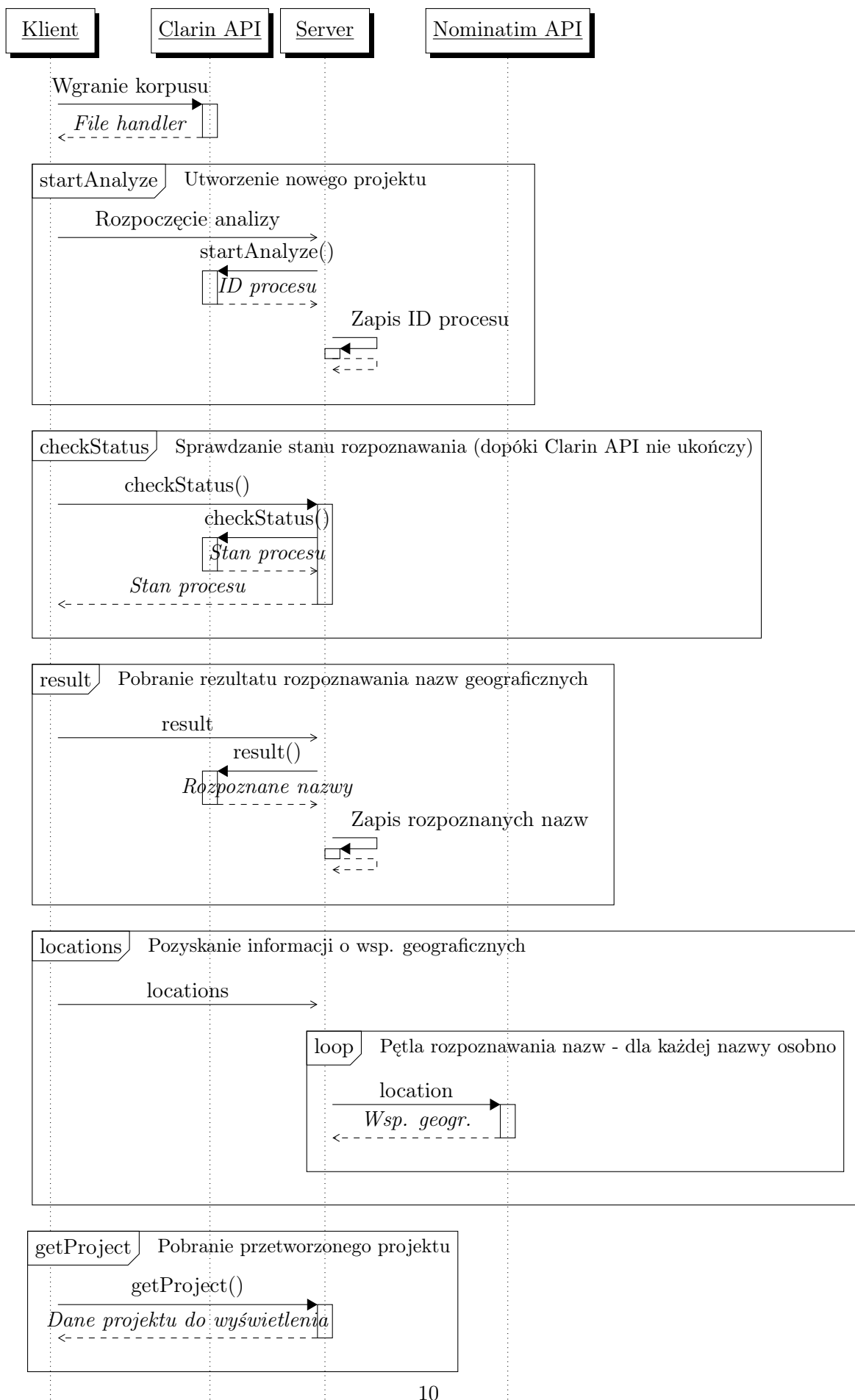
W projekcie użyto Dockera do utworzenia osobnych kontenerów dla aplikacji klienckiej, aplikacji serwerowej oraz bazy danych. Do określenia routingu między nimi oraz sposobu ich budowania użyto narzędzia docker-compose.

4 Implementacja

Opracowany kod dzieli się na część wykonywaną po stronie klienta i na część wykonywaną po stronie serwera. Jego działanie opiera się na wykonaniu pięciu żądań (zapytań):

- a) `Upload zip` - wgranie na serwer korpusu tekstów,
- b) `startAnalyze` - rozpoczęcie analizy,
- c) `ckeckStatus` - weryfikacja stanu procesu przetwarzania korpusu na nazwy geograficzne,
- d) `result` - pobranie rezultatu przetwarzania,
- e) `locations` - pozyskanie współrzędnych geograficznych na podstawie rozpoznanych nazw geograficznych.

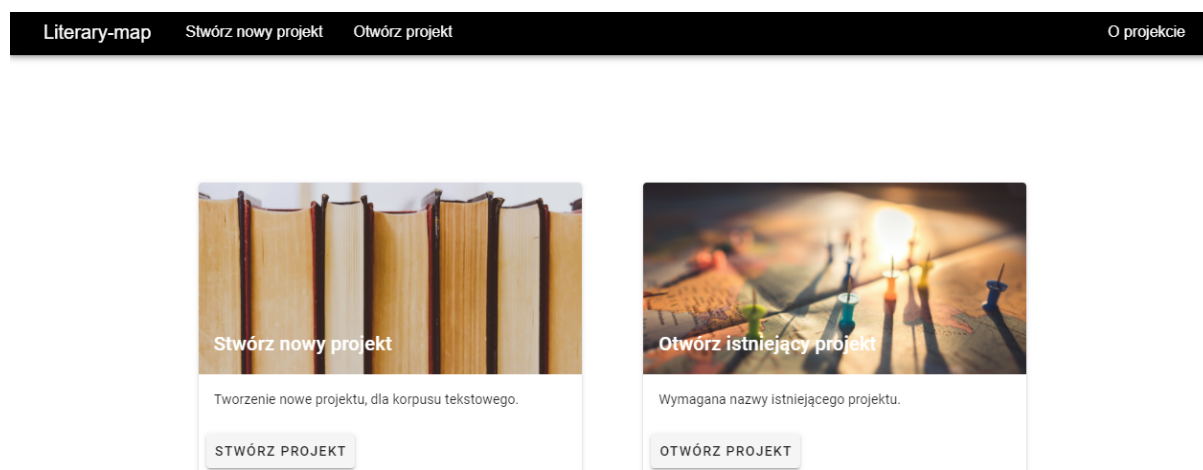
Zapytaniem wieńczącym ten proces jest `getProject`, który pobiera z serwera wszystkie dane przetworzonego projektu. Na rysunku 1 można zapoznać się z diagramem sekwencji, obrazującym kolejne czynności wykonywane przez aplikację.



Rysunek 1: Diagram sekwencji czynności - tworzenie nowego projektu.

5 Wygląd i działanie aplikacji

W tej części ukazano wygląd i działanie zaimplementowanej aplikacji. Na rysunku 2 przedstawiono menu główne, z którego możemy wybrać opcję stworzenia nowego projektu lub otworzyć już istniejący. W prawym górnym rogu znajduje się link pozwalający na wyświetlenie podstawowych informacji na temat projektu oraz autorów.

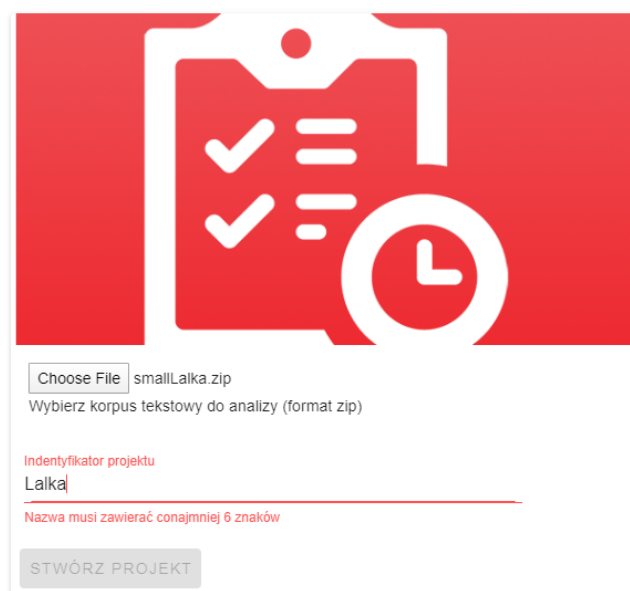


Rysunek 2: Menu główne aplikacji.

Na rysunku 3 przedstawiono, jak wygląda otwieranie istniejącego projektu. Wymagane jest podanie jego nazwy w odpowiednim polu.

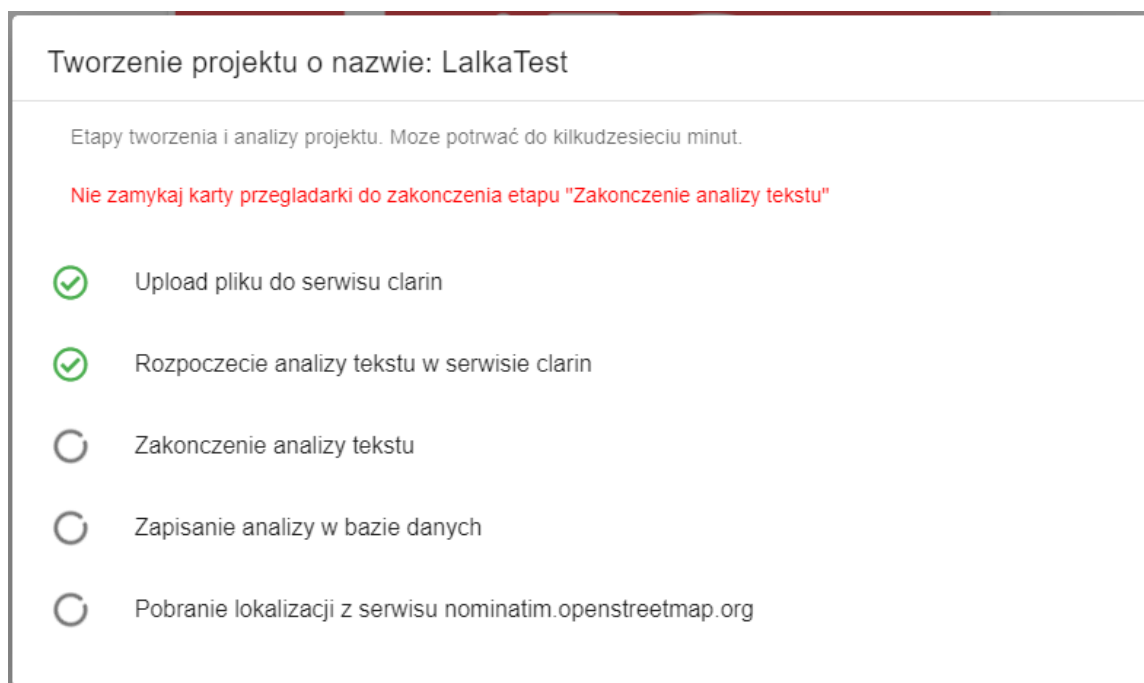
Rysunek 3: Otwieranie istniejącego projektu.

Na rysunkach 4 i 5 przedstawiono proces tworzenia nowego projektu. Najpierw musimy użyć przycisku *Choose file* (*Wybierz plik*) i wskazać plik w odpowiednim formacie, następnie zaś podać nazwę projektu składającą się z co najmniej sześciu znaków. Zatwierdzenie procesu rozpoczyna etap przetwarzania.



The screenshot shows a web form for creating a new project. At the top is a red header with a white icon of a clipboard and a clock. Below the header, there is a file selection area with a 'Choose File' button and the text 'smallLalka.zip'. Below this is a label 'Wybierz korpus tekstowy do analizy (format zip)'. Then, there is a label 'Identyfikator projektu' in red, followed by a text input field containing 'Lalka'. Below the input field is a red error message: 'Nazwa musi zawierać co najmniej 6 znaków'. At the bottom is a grey button labeled 'STWÓRZ PROJEKT'.

Rysunek 4: Tworzenie nowego projektu.

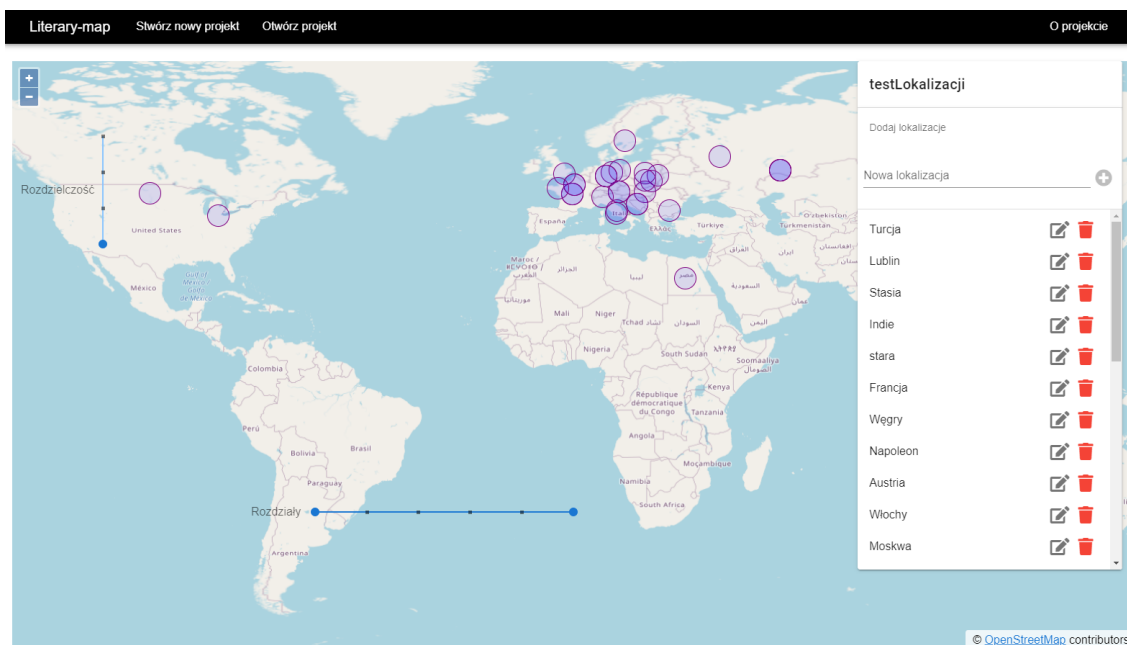


The screenshot shows a progress bar for creating a project named 'LalkaTest'. The title is 'Tworzenie projektu o nazwie: LalkaTest'. Below the title is a message: 'Etapy tworzenia i analizy projektu. Może potrwać do kilkadziesiąt minut.' followed by a red warning: 'Nie zamykaj karty przeglądarki do zakończenia etapu "Zakończenie analizy tekstu"'. The progress bar consists of five steps, each with a circular icon and a description:

- ✓ Upload pliku do serwisu clarin
- ✓ Rozpoczęcie analizy tekstu w serwisie clarin
- Zakonczenie analizy tekstu
- Zapisanie analizy w bazie danych
- Pobranie lokalizacji z serwisu nominatim.openstreetmap.org

Rysunek 5: Proces tworzenia nowego projektu w trakcie przetwarzania.

Na rysunku 6 przedstawiono mapę z naniesionymi na nią różnymi punktami odzwierciedlającymi pewne lokalacje. Po prawej stronie użytkownik ma możliwość dodania lokalizacji, a także usunięcia i edycji słów znajdujących się aktualnie w słowniku. Widzimy również suwaki pozwalające zmienić rozdzielczość oraz przewijać rozdziały.



Rysunek 6: Punkty naniesione na mapę.

Na rysunku 7 pokazane jest okienko umożliwiające użytkownikowi zmianę współrzędnych danej lokalizacji.

Zmień współrzędne dla lokalizacji: Turcja

Szerokość geograficzna

Długość geograficzna

ZATWIERDZ

Rysunek 7: Okno zmiany współrzędnych dla lokalizacji.

Na rysunku 8 przedstawiono wspomnianą na początku zakładkę zawierającą informacje o autorach i krótki opis projektu.

Autorzy:

Michał Bańka, nr indeksu: 235051

Przemysław Jedlikowski, nr indeksu: 234927

Witold Marciniak, nr indeksu: 226194

Wojciech Mielczarek, nr indeksu: 218393

Marcin Pawluś, nr indeksu: 235555

O projekcie:

Projekt ten powstał w ramach przedmiotu "Zastosowania informatyki w gospodarce" prowadzonego przez dr inż. Tomasza Walkowiaka. Realizuje on temat (2.) "Mapa literacka (geolokalizacja na podstawie danych z Linera) dla korpusów (zmienność w czasie, np. rozdziały) (PL)". Projekt został stworzony w języku Java z użyciem Springa oraz frameworku Vue. Dane z Linera są przechowywane w nierelacyjnej bazie danych MongoDB. Wykorzystano również narzędzie Linera służące do analizy tekstu lub tekstów oraz uzyskania nazw własnych lokalizacji znajdujących się w tekście, a także Nominatim do uzyskania współrzędnych w postaci pliku JSON na podstawie nazwy własnej.

Projekt ma za zadanie ułatwić analizę tekstów, a jego głównymi użytkownikami będą naukowcy nauk humanistycznych. Pozwala on na tworzenie nowych projektów (analiz tekstów) oraz przechowywanie ich do późniejszych analiz przez użytkownika, wizualizuje przetworzone informacje poprzez naniesienie punktów na mapę, pozwala na modyfikację przetworzonych danych przez użytkownika oraz pozwala użytkownikowi na rozszerzenie istniejącego słownika poprzez dodanie swoich zwrotów. Przetwarzanie wprowadzonych danych może trwać kilka minut lub kilka godzin, a cały proces nie jest responsywny. Przykładowe użycie aplikacji powinno wyglądać następująco: najpierw następuje wgranie korpusu tekstów, potem użytkownik ma możliwość poszerzenia słownika o własne zwroty, kolejno następuje przetwarzanie danych, po którym istnieje możliwość przetwarzania parametrów przez użytkownika, a na koniec wizualizacja i naniesienie wyników na mapę.

Rysunek 8: Skrócony opis projektu i autorzy.

6 Napotkane problemy

Podczas wykonywania projektu natrafiłszy na pewne problemy. Jednym z nich było ograniczenie serwisu Nominatim zezwalające na wysyłanie do API jedynie jednego żądania na sekundę - w przeciwnym razie dostęp do usługi jest blokowany. Aby temu zaradzić, użyto w kodzie funkcji `Thread.sleep()`, jednak rozwiązuje to problem jedynie w przypadku jednego użytkownika korzystającego z aplikacji jednocześnie. Rozwiązaniem tego problemu byłby tzw. *broker*. Jest to program pośredniczący, który tłumaczy wiadomości z formalnego protokołu przesyłania komunikatów wydawcy na formalny protokół przesyłania komunikatów odbiorcy. Z jego pomocą wiadomości można buforować w kolejce w przypadku, gdy odbiorca nie może nadążyć za przetwarzaniem wiadomości przychodzących. Jednym z popularniejszych brokerów jest np. **Apache ActiveMQ** - napisany w Javie, wykorzystujący interfejs API *Java Message Service* do tworzenia, wysyłania i odbierania wiadomości. **Apache ActiveMQ** jest obecnie najpopularniejszym, otwartoźródłowym serwerem do przesyłania wiadomości. Obsługuje standardowe protokoły branżowe, dzięki czemu użytkownicy mogą czerpać korzyści z wyborów klientów w szerokim zakresie języków i platform.

Innym rozwiązaniem byłoby wykorzystanie kontenera Nominatim w miejsce zewnętrznej usługi (np. <https://github.com/mediagis/nominatim-docker>). W takim przypadku proces rozpoznawania nie wymaga zachowania co najmniej sekundowego opóźnienia między zapytaniami o współrzędne geograficzne zadanej nazwy. Projekt można bardzo łatwo zmodyfikować, podając nowy adres, pod którym taki kontener znajdowałby się. Należy jednak zwrócić uwagę, że rozmiar kontenera tej usługi może sięgać nawet 1 GB, a dane map - nawet 50-100 GB.

7 Podsumowanie

Udało się zrealizować główne założenia projektu. Po uruchomieniu aplikacji użytkownik ma możliwość wgrać wybrany przez siebie korpus tekstów, a następnie rozszerzyć słownik o nowe zwroty. Nowo dodane słowa mogą być zarówno edytowane, jak i usuwane. Na koniec następuje analiza, a wyniki wizualizowane są w postaci punktów nanoszonych na mapę.

Implementację aplikacji serwerowej i klienckiej wykonano z myślą o jej prostej rekonfiguracji. Dzięki temu można niskim kosztem wdrożyć zmiany opisane w rozdziale 6. Wskazać należy, że powstała aplikacja jest prototypem, w szczególności wymaga rozwiązania problemu ograniczenia częstości używania zewnętrznego serwisu *Nominatim*, który został opisany we wskazanym rozdziale.

Proces przetwarzania korpusu tekstów wymaga od użytkownika pozostawienia witryny otwartej przynajmniej do momentu zakończenia przetwarzania przez usługę *Liner*. Wynika to z postawionego założenia nieprzechowywania na serwerze przesyłanych tekstów ze względu na przypuszczalny konflikt z prawem autorskim i prawami pokrewnymi. Dopuszczenie możliwości wgrania korpusu na serwer w celu przesłania go do zewnętrznej usługi (pod warunkiem natychmiastowego zniszczenia przesłanego pliku na serwerze po potwierdzeniu poprawności przesłania do usługi zewnętrznej) pozwoliłoby na „uwolnienie” użytkownika od procesu oczekiwania.

Do zalet procesu realizacji projektu nie można zaliczyć terminowości rozliczeń z kamieniami milowych. Ze względu na całkowitą zmianę sposobu realizacji zajęć uczelnianych oraz życia codziennego, praca nad projektem uległa znacznemu zaburzeniu.

Źródła

- [1] OpenStreetMap Wiki contributors. *Nominatim*. 15 maj. 2020. URL: <https://wiki.openstreetmap.org/w/index.php?title=Nominatim&oldid=1991279> (term. wiz. 09.06.2020).
- [2] Wikipedia contributors. *MongoDB*. 5 wrz. 2019. URL: <https://pl.wikipedia.org/w/index.php?title=MongoDB&oldid=57418794> (term. wiz. 09.06.2020).
- [3] Wikipedia contributors. *OpenLayers*. 5 lut. 2019. URL: <https://pl.wikipedia.org/w/index.php?title=OpenLayers&oldid=55815665> (term. wiz. 09.06.2020).
- [4] Wikipedia contributors. *Spring Framework*. 2 czer. 2020. URL: https://pl.wikipedia.org/w/index.php?title=Spring_Framework&oldid=59976706 (term. wiz. 09.06.2020).
- [5] Przemysław Jedlikowski. *Aplikacja sklepu internetowego jako przykład środowiska ciągłej integracji i ciągłego dostarczania (CI/CD)*. Praca inżynierska wykonana na Wydziale Elektroniki Politechniki Wrocławskiej, promotor: dr inż. Piotr Patronik. Grud. 2019.
- [6] Andrzej - jsdn.pl. *Vue.js – wprowadzenie. Podstawowe informacje*. 9 wrz. 2016. URL: <http://jsdn.pl/vue-js-wprowadzenie/> (term. wiz. 09.06.2020).
- [7] Andi Pavllo. *Choosing the Right Front-End Framework for Your Vue App*. 19 wrz. 2018. URL: <https://medium.com/the-web-tub/choosing-the-right-front-end-framework-for-your-vue-app-4448bac12ce7> (term. wiz. 09.06.2020).