



**Politechnika
Śląska**

PROJEKT INŻYNIERSKI

System do monitorowania położenia geoprzestrzennego obiektów

Jakub SIBIK

Nr albumu: 300420

Kierunek: Informatyka

Specjalność: Informatyczne Systemy Mobilne i Przemysłowe

PROWADZĄCY PRACĘ

dr inż. Łukasz Wyciślik

KATEDRA Informatyki Stosowanej

Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2025

Tytuł pracy

System do monitorowania położenia geoprzestrzennego obiektów

Streszczenie

(Streszczenie pracy – odpowiednie pole w systemie APD powinno zawierać kopię tego streszczenia.)

Słowa kluczowe

lokalizator, lokalizacja, pojazd, kierowca

Thesis title

Thesis title in English

Abstract

(Thesis abstract – to be copied into an appropriate field during an electronic submission – in English.)

Key words

(2-5 keywords, separated by commas)

Spis treści

1	Wstęp	1
1.1	Wprowadzenie w problem	1
1.2	Cel pracy	2
1.3	Zakres pracy	2
2	Analiza tematu	3
3	Wymagania i narzędzia	5
3.1	Wymagania funkcjonalne	5
3.2	Wymagania niefunkcjonalne	6
3.3	Narzędzia	7
3.4	Stos technologiczny	9
3.4.1	Java	9
3.4.2	Spring Boot	9
3.4.3	JavaScript	10
3.4.4	TypeScript	10
3.4.5	React	10
3.4.6	PostgreSQL	10
3.4.7	PostGIS	10
3.4.8	Hibernate	11
4	Specyfikacja zewnętrzna	13
4.0.1	Wymagania sprzętowe i programowe	13
4.0.2	Kategorie użytkowników	13
4.0.3	Sposób obsługi	14
5	Specyfikacja wewnętrzna	29
5.1	Przedstawienie idei	29
5.2	Opis struktur danych	29
5.3	Szczegóły implementacji wybranych fragmentów	30

6	Weryfikacja i walidacja	37
6.1	Sposób testowania	37
6.2	Wykryte i usunięte błędy	37
7	Podsumowanie i wnioski	39
7.1	Uzyskane wyniki w świetle postawionych celów i zdefiniowanych wymagań	39
7.2	Kierunek ewentualnych prac	39
	Bibliografia	41
	Spis skrótów i symboli	45
	Źródła	47
	Lista dodatkowych plików, uzupełniających tekst pracy	49
	Spis rysunków	52
	Spis tabel	53

Rozdział 1

Wstęp

1.1 Wprowadzenie w problem

Początki technologii do określania pozycji sięgają lat 60. XX wieku. Powstał wtedy system NAVSAT (ang. Navigation Satellite System) - będący pierwszym satelitarnym systemem nawigacyjnym. Został on opracowany przez Stany Zjednoczone oraz wykorzystywany był przez tamtejszą marynarkę wojenną. W latach 70. XX wieku postanowiono wprowadzić międzynarodowy standard, dzięki czemu powstał system GPS (ang. Global Positioning System), który jest używany po dziś dzień.

Długi czas obecności tego systemu na rynku zaowocował jego rozwojem, jak również dostępnością dla przeciętnych użytkowników. W efekcie tego, GPS jest wsparciem dla ludzi w wielu dziedzinach. W obecnych czasach ponad połowa światowej populacji posiada smartfony, które to mają wbudowane systemy GPS. Pozwala to przede wszystkim na sprawną nawigację do celu czy dokładne ustalenie pozycji danej osoby. Samochody również są w posiadaniu znacznej części populacji, a ponadto stanowią dosyć znaczną część budżetu domowego. Z tego względu ludzie zaopatrują się w lokalizatory samochodowe. Podczas kupna takiego urządzenia, klient otrzymuje zazwyczaj dostęp do strony internetowej, na której jest w stanie sprawdzić położenie swojego samochodu, w którym został umieszczony lokalizator. Takie rozwiązania są dosyć proste, niewystarczające dla wielu użytkowników, wygląd interfejsu również pozostawia wiele do życzenia. Wraz z rozwojem komputerów oraz smartfonów, zwiększają się możliwości do stworzenia aplikacji do zarządzania lokalizatorami w pojazdach, która oferowałaby większą ilość funkcjonalności, oraz która byłaby atrakcyjniejsza niż podstawowe odpowiedniki, będące obecnie na rynku.

1.2 Cel pracy

Celem niniejszej pracy inżynierskiej jest stworzenie aplikacji przeglądarkowej pozwalającej na zarządzanie lokalizatorami, pojazdami i kierowcami, która usprawniłaby obsługę tego typu urządzeń w obrębie rodziny lub firmy posiadającej flotę samochodów i ułatwiłaby przeglądanie tras przebytych przez poszczególne osoby, auta czy lokalizatory.

1.3 Zakres pracy

Praca obejmuje proces i sposób tworzenia oprogramowania, specyfikacje: zewnętrzną i wewnętrzną, testowanie, jak również efekty i wnioski.

Rozdział 2

Analiza tematu

Lokalizatorów samochodowych na rynku jest wiele. W zależności od ceny, możemy otrzymać dodatkowe funkcjonalności, mniej lub bardziej znaczące, są to między innymi czujnik wstrząsu, monitorowanie prędkości czy podsłuch. Producenci zazwyczaj posiadają własne strony internetowe, do których klient otrzymuje bezpłatny dostęp po zakupie produktu. Pozwalają one zwykle powiązać lokalizator z kontem użytkownika i śledzić na bieżąco jego lokalizację.

W celu stworzenia aplikacji do monitorowania lokalizatorów dla jak największej liczby ludzi, korzystających z tego typu urządzeń, warto przeanalizować kilka kwestii. Niezbędną funkcją lokalizatora jest możliwość jego konfiguracji, aby wysyłał dane na konkretny adres IP oraz port. Umożliwi to dostęp aplikacji do lokalizacji urządzenia. Z tego powodu odrzucono produkty o najniższej cenie na rynku, gdyż nie oferują one wymaganej do działania programu funkcji. Kolejnym znaczącym aspektem wyboru lokalizatora jest jego cena w kontekście klienta. Należy wykluczyć najdroższe opcje, aby nie ograniczyć ilości potencjalnych użytkowników aplikacji. Uwzględniając powyższe wymagania, wybrano model Mking MK07A. Dodatkowym atutem jest jego akumulator, którego pojemność wynosi 10000 mAh, dzięki czemu nie wymaga częstego ładowania.

Podczas przeprowadzania analizy tematu, należało również wybrać stos technologiczny, w którym aplikacja będzie tworzona. Ze względu na liczne biblioteki usprawniające proces rozwijania oprogramowania, część funkcjonalną programu postanowiono napisać w języku Java. W celu konfiguracji aplikacji przeglądarkowej została wybrana platforma programistyczna, jaką jest Spring Boot - jedna z bardziej popularnych opcji przy tego typu programach pisanych w języku Java. Drugą częścią, która zostanie stworzona, jest część interfejsu. W tym przypadku wybrano bibliotekę React, będącą powszechnym narzędziem, oferującym wiele przydatnych funkcji. Biblioteka oparta będzie na języku TypeScript - jest to rozszerzenie języka JavaScript, które wymaga określania typów zmiennych, dzięki czemu można uniknąć dużej liczby błędów w przyszłości, które pojawiają się w JavaScript

przy braku typowania. Należy się również zastanowić nad przechowywaniem danych. Służą do tego bazy danych. W niniejszej aplikacji w celu tworzenia i obsługi takiej bazy zostanie użyty język PostgreSQL. Jest to uwarunkowane danymi, które pojawiają się w systemie, a mianowicie dane geograficzne - długość i szerokość geograficzna. PostgreSQL posiada rozszerzenie - PostGIS - umożliwiające zapis długości i szerokości geograficznej w jednej kolumnie, jako punkt na Ziemi.

Rozdział 3

Wymagania i narzędzia

Tworzona aplikacja ma ułatwić użytkownikowi zarządzanie lokalizatorami, kierowcami i pojazdami jednocześnie oraz pozwalać na wyświetlanie tras poszczególnych kierowców i pojazdów. Wymagania aplikacji należy podzielić na funkcjonalne i нефункционалне. Przyjmy się w pierwszej kolejności wymaganiom funkcjonalnym. Poniżej wypisano przewidziane przypadki użycia programu.

3.1 Wymagania funkcjonalne

Dla niezalogowanego użytkownika (przedstawia je również diagram widoczny na rys 3.1):

- Rejestracja - gdy użytkownik wchodzi na stronę aplikacji, może się zarejestrować, czyli stworzyć bezpłatne konto, które będzie przypisane do podanego przez niego w formularzu rejestracji adresu email
- Logowanie - w przypadku uprzedniej rejestracji w systemie, użytkownik może zalogować się na istniejące konto

Dla zalogowanego użytkownika (przedstawia je również diagram widoczny na rys 3.2):

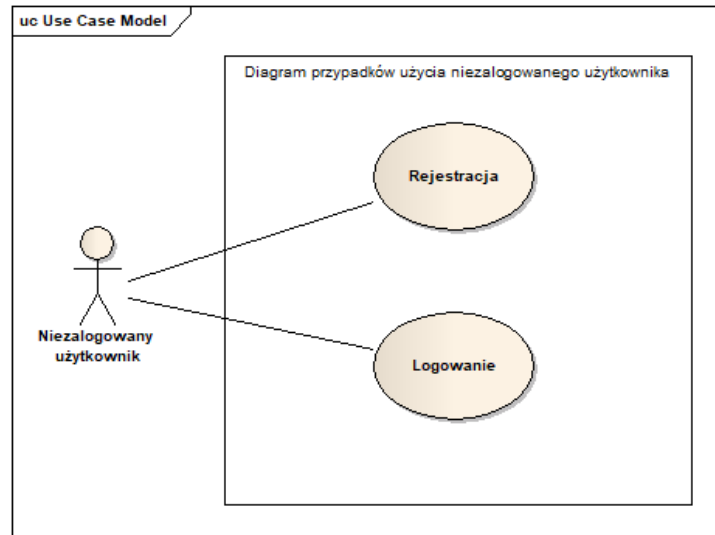
- Dodanie obiektu
 - Dodanie kierowcy - użytkownik ma możliwość dodania kierowcy do swojego konta, podając ich imię oraz nazwisko
 - Dodanie pojazdu - pozwala dodać pojazd do konta poprzez podanie jego marki, modelu, numeru rejestracyjnego oraz numeru VIN
 - Dodanie lokalizatora - w ten sposób użytkownik może dodać posiadane lokalizatory TK108 do swojego konta, podając jego nazwę, numer seryjny oraz typ
- Dodanie powiązania

- Dodanie kierowcy pojazdu - ten przypadek umożliwia powiązanie kierowcy z pojazdem w podanym przez użytkownika okresie czasu
- Dodanie lokalizatora w pojeździe - działa na takiej zasadzie, jak powyższy podpunkt, umożliwia powiązanie lokalizatora z pojazdem w danym okresie czasu
- Wyświetlenie trasy
 - Wyświetlenie trasy kierowcy - użytkownik może wyświetlić na mapie trasę przebytą przez danego kierowcę w wybranym przez siebie okresie czasu
 - Wyświetlenie trasy pojazdu - analogicznie do powyższego przypadku, możliwość wyświetlenia trasy pokonanej przez pojazd w danym czasie
- Usunięcie obiektu
 - Usunięcie kierowcy - powoduje usunięcie istniejącego kierowcy z systemu
 - Usunięcie pojazdu - daje możliwość usunięcia wybranego pojazdu z aplikacji
 - Usunięcie lokalizatora - podobnie do powyższych, skutkuje usunięciem danego lokalizatora
- Usunięcie powiązania
 - Usunięcie kierowcy pojazdu - usuwa czasowe powiązanie kierowcy oraz pojazdu
 - Usunięcie lokalizatora w pojeździe - analogicznie, użytkownik może się w ten sposób pozbyć powiązania lokalizatora z pojazdem z danego okresu czasu
- Usunięcie trasy z mapy
 - Usunięcie trasy kierowcy - powoduje zniknięcie z mapy trasy kierowcy z danego okresu czasu
 - Usunięcie trasy pojazdu - pozwala usunąć z mapy wcześniej wyświetloną trasę pojazdu

3.2 Wymagania нефункциональные

Przejdźmy teraz do wymagań нефункциональных:

- Niskie wymagania systemowe - aplikacja powinna być dostępna dla każdego urządzenia posiadającego przeglądarkę internetową oraz dostęp do internetu. W tym celu wymagane jest, aby program był aplikacją przeglądarką.

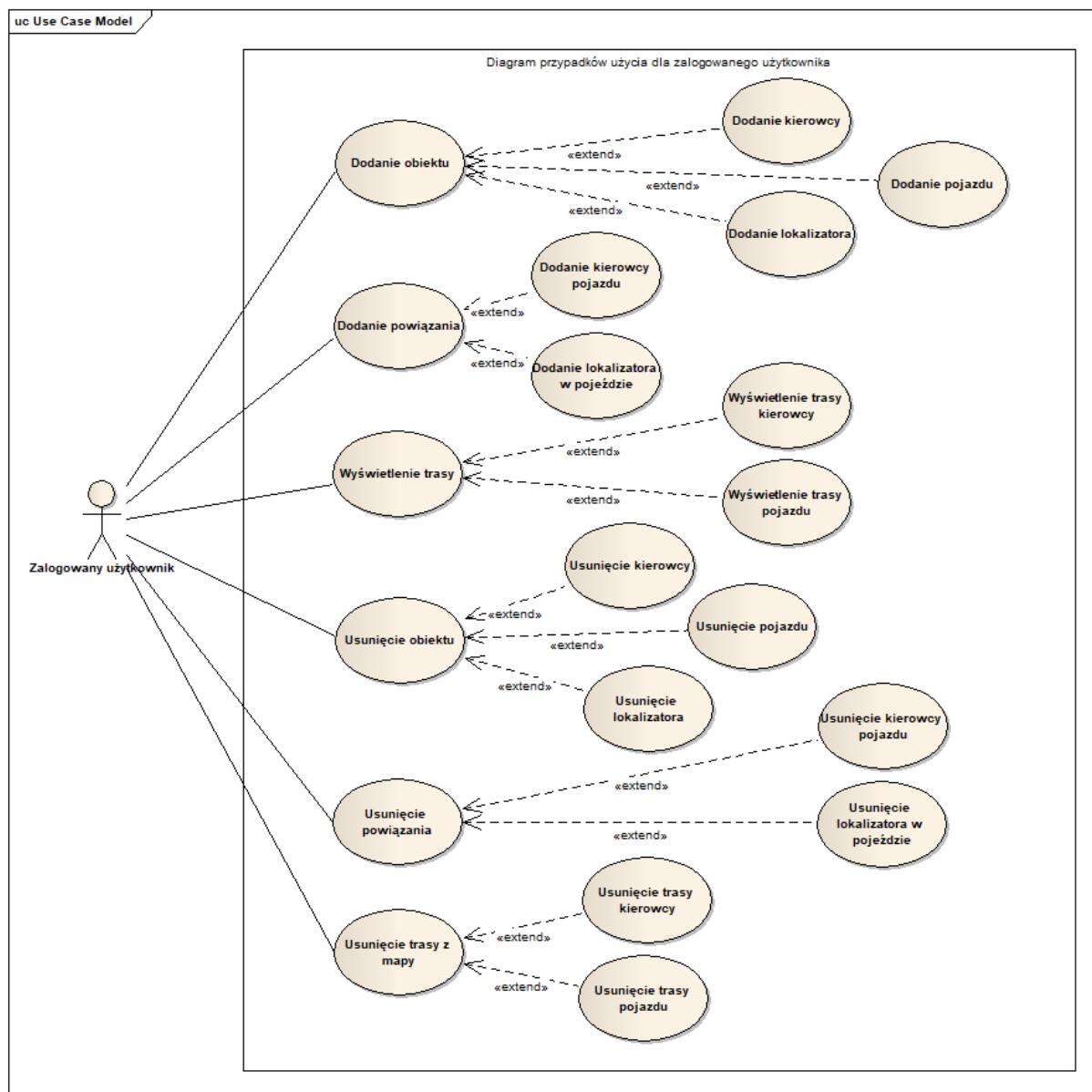


Rysunek 3.1: Diagram przypadków użycia niezalogowanego użytkownika.

- Skalowalność interfejsu użytkownika - wymagane jest, aby program był dostosowany do różnej wielkości oraz rozdzielczości ekranów. Użytkowanie aplikacji na smartfonie powinno być równie sprawne i nie sprawiające problemów, jak podczas korzystania z komputera stacjonarnego lub laptopa.
- Prosta obsługa - ze względu na duże i zróżnicowane grono potencjalnych użytkowników, założono, iż aplikacja będzie możliwie jak najmniej skomplikowana w obsłudze. Dostęp do każdej funkcji programu powinien być możliwy przy wykonaniu minimalnej ilości kliknięć myszy (lub ekranu w przypadku urządzeń dotykowych), nieprzekraczającej trzech.

3.3 Narzędzia

Aby przystąpić do tworzenia aplikacji, należało przygotować odpowiednie narzędzia. W pierwszej kolejności zakupiono lokalizator Mking MK07A oraz kartę SIM, aby możliwe było uruchomienie i skonfigurowanie urządzenia. Kolejnym etapem było zainstalowanie niezbędnego oprogramowania do edycji kodu źródłowego aplikacji. Do części funkcjonalnej został wykorzystany IntelliJ Idea wydawcy JetBrains. Ten program jest zaawansowanym, wieloplatformowym środowiskiem programistycznym, posiadającym dużą ilość wbudowanych narzędzi, jak również obsługującym wiele bibliotek i platform programistycznych, takich jak Spring Boot. Hibernate oraz JPA również są wspierane przez IntelliJ. Są to narzędzia, które potrafią powiązać kod aplikacji z bazą danych, co znacznie uprości rozwijanie programu. W kontekście bazy danych, potrzebne było oprogramowanie, które będzie w stanie obsługiwać zapytania kierowane do bazy danych - na początku do stworzenia bazy, poszczególnych tabel, a w przyszłości do eliminowania ewentualnych błędów



Rysunek 3.2: Diagram przypadków użycia zalogowanego użytkownika.

oraz do testowania aplikacji. Przechodząc do interfejsu użytkownika, a zatem do kodu w języku TypeScript z zastosowaniem biblioteki React, wybrano Microsoft Visual Studio Code, który jest powszechnie używany przez programistów w tym celu, ze względu na odpowiednie wsparcie języków i bibliotek typowych do tworzenia wizualnej części aplikacji. Ostatnim elementem jest narzędzie do systemu kontroli wersji. Dzięki temu postępy w pracy będą zabezpieczone. Niemniej jednak, to nie jedyny atut. W przypadku błędów aplikacji spowodowanych zmianami w kodzie źródłowym, zlokalizowanie ich przyczyny będzie znacznie ułatwione. Wybrano system Git, gdyż jest on najpopularniejszą opcją, a co za tym idzie, jest obsługiwany przez niemal każde oprogramowanie programistyczne, lecz to nie wszystko. Należy również dokonać wyboru platformy hostingowej, która będzie przechowywała repozytorium Git. W tym celu skorzystano z GitHuba, będącego jedną z najbardziej powszechnych opcji. Jest on także wspierany przez większość środowisk programistycznych. Poszczególne elementy stosu technologicznego zostały szczegółowo opisane poniżej.

3.4 Stos technologiczny

3.4.1 Java

Jeden z najpowszechniejszych wysokopoziomowych języków programowania. Stworzony w 1995 roku, jest nadal rozwijany i powszechnie używany. Jego cechami są przede wszystkim: obiektowość, dobrze rozwinięta biblioteka standardowa, niezliczona ilość dodatkowych bibliotek. Posiada on również Garbage Collector, odpowiadający za zarządzanie pamięcią, co zwiększa bezpieczeństwo języka. Jest to częsty wybór w przypadku tworzenia aplikacji przeglądarek, ze względu na istniejące platformy programistyczne oraz biblioteki, będące idealnym rozwiązaniem do tego typu programów.

3.4.2 Spring Boot

Popularna platforma programistyczna dla języka Java, która w znaczny sposób ułatwia konfigurację aplikacji przeglądarkowej, jak również późniejsze jej rozwijanie pod względem tworzenia REST API czy połączenie programu z bazą danych. Konfiguracja odbywa się w dużej mierze automatycznie, wkład programisty opiera się przede wszystkim na dodaniu odpowiednich zależności. Charakterystyczne dla Spring Boot są adnotacje, które są poprzedzone znakiem "@". To dzięki nim Spring Boot potrafi interpretować napisany kod, aby następnie wykonywać pewne operacje automatycznie.

3.4.3 JavaScript

To również bardzo powszechny język programowania, natomiast w odróżnieniu do Javy, odpowiada on przede wszystkim za interfejs użytkownika. Ważną jego cechą, wykorzystywaną w aplikacjach przeglądarkowych, jest obsługa asynchroniczności. Jest to spowodowane wysyłaniem zapytań HTTP, które trafiają do Javy, a odpowiedzi nie zawsze są natychmiastowe.

3.4.4 TypeScript

Nie jest uznawany za osobny język, a za nadzbiór JavaScriptu. Rozszerza jego możliwości, przede wszystkim dzięki typowaniu, które polega na tym, iż zmiennym można przypisywać konkretny, zdefiniowany typ. Dotyczy to również danych zwracane przez funkcje. Znacznie zwiększa to czytelność kodu oraz pozwala uniknąć przypisywania niewłaściwych typów do zmiennych.

3.4.5 React

React jest biblioteką do tworzenia interfejsu użytkownika. Działa z językiem JavaScript, jak również TypeScript. Został on stworzony, aby budowane aplikacje przeglądarkowe były dynamiczne - to znaczy aby nie potrzebowały odświeżania strony, aby aktualizować jej widok. React polega na tworzeniu komponentów, które odpowiadają za renderowanie określonej części interfejsu. Charakterystycznymi elementami tej biblioteki są Hooki, czyli funkcje umożliwiające zarządzanie stanem komponentów.

3.4.6 PostgreSQL

Zaawansowany system do zarządzania bazą danych, zgodny ze standardem SQL, pozwalający również na dostosowanie do indywidualnych potrzeb, gdyż posiada na przykład możliwość tworzenia własnych typów danych. Często wykorzystywany w aplikacjach przeglądarkowych, ze względu na jego wysoką wydajność oraz skalowalność.

3.4.7 PostGIS

Jest to rozszerzenie do PostgreSQL, które wspiera obsługę danych geograficznych i przestrzennych. Daje możliwość przechowywania danych przede wszystkim o punktach, ale również o liniach, wielokątach. Rozszerzenie to jest używane w przypadku programów związanych z obsługą GPS oraz map, także idealnie pasuje do niniejszej pracy.

3.4.8 Hibernate

Bardzo często wybierana platforma programistyczna w celu automatyzacji mapowania między obiektami w języku Java, a tabelami w bazie danych. Umożliwia to pracę nad obiektami, bez konieczności wykonywania skomplikowanych zapytań SQL.

Rozdział 4

Specyfikacja zewnętrzna

4.0.1 Wymagania sprzętowe i programowe

Aplikacja do działania potrzebuje serwera - komputera, na którym skompilowany z kodu źródłowego program, jak również baza danych, będą uruchomione. Serwer musi posiadać dostęp do internetu oraz jego adres musi być publicznie dostępny. Dzięki temu ma on możliwość obsługiwania żądań HTTP wysyłanych przez użytkowników.

W przypadku działającego serwera, można do użytkowania aplikacji. W celu uzyskania do niej dostępu, użytkownik musi mieć zainstalowaną przeglądarkę na urządzeniu (np. komputerze stacjonarnym, laptopie, tablecie czy smartfonie) oraz połączenia z internetem. Po spełnieniu powyższych wymagań, w pasek adresu URL w przeglądarce należy wpisać adres serwera oraz port, na którym uruchomiona jest aplikacja. Można również uprościć użytkownikom uruchomienie aplikacji poprzez uruchomienie serwera na konkretnej domenie DNS, którą wystarczy wpisać w przeglądarce, natomiast nie zostało to zaimplementowane w trakcie procesu tworzenia systemu.

4.0.2 Kategorie użytkowników

Wyróżniamy dwa rodzaje użytkowników:

- niezalogowany użytkownik
- zalogowany użytkownik

Z niezalogowanym użytkownikiem mamy do czynienia wówczas, gdy nastąpi włączenie aplikacji lub po wylogowaniu się ze swojego konta podczas użytkowania. Funkcje, które ma on do dyspozycji, są mocno ograniczone. Użytkownik ma wtedy dostęp do:

- strony głównej - tylko wyświetla tekst
- formularza rejestracji - pozwala utworzyć konto użytkownika

- formularza logowania - umożliwia zalogowanie się na istniejące w systemie konto

Zalogowany użytkownik uzyskuje dostęp do właściwych funkcji systemu, dzięki którym może zarządzać lokalizatorami, jak również kierowcami i pojazdami.

4.0.3 Sposób obsługi

Po włączeniu aplikacji ukazuje się strona główna. Widoczny na niej tekst zachęca do skorzystania z wszystkich funkcji systemu. Jej wygląd jest widoczny na rys. 4.1. Na górze strony znajduje się pasek nawigacji, który towarzyszy użytkownikowi przez cały czas użytkowania programu, lecz różni się w zależności od tego, czy użytkownik jest niezalogowany (pasek wtedy przybiera formę jak na rys. 4.1), czy zalogowany (w tym przypadku pasek wygląda jak na rys. 4.4). Po kliknięciu w poszczególną opcję na pasku, odpowiadająca jej zakładka jest otwierana. Aby przejść do formularzy logowania i rejestracji, należy wybrać opcję "Zaloguj się".

W pierwszej kolejności ukazuje się formularz logowania (rys. 4.2). Jest to celowy zabieg, ponieważ użytkownik, po założeniu konta, z każdym kolejnym uruchomieniem aplikacji będzie się logował. Wynika z tego, iż liczba logowań będzie wyższa niż liczba rejestracji, a zatem będzie to częstsza operacja wykonywana przez użytkowników. W celu otwarcia formularza rejestracji, należy kliknąć w tekst w lewym dolnym rogu formularza logowania. Jego treść brzmi: „Nie masz jeszcze konta? Zarejestruj się” - rys. 4.2. Jest to link, który umożliwia zmianę formularza z opcji do zalogowania się, na opcję do utworzenia konta. Aby stworzyć konto użytkownika, należy wpisać adres email, z którym będzie owe konto powiązane. Kolejnym etapem jest ustalenie hasła, którym następnie wypełnia się drugie pole. W celach bezpieczeństwa, hasło powinno być trudne do rozszyfrowania przez innych ludzi. Niemniej jednak, nie ma możliwości odzyskania zapomnianego hasła, o czym warto pamiętać podczas korzystania z aplikacji. Aby dokończyć proces tworzenia konta, należy nacisnąć niebieski przycisk w prawym dolnym rogu. Widok formularza rejestracji znajduje się na rys. 4.3.

Do formularza logowania można dostać się na dwa sposoby. Jeden został wymieniony powyżej, drugim jest kliknięcie w tekst znajdujący się w lewym dolnym narożniku formularza rejestracji, brzmiący: „Masz już konto? Zaloguj się”. Dzięki temu w prosty sposób można przejść do zalogowania się, bezpośrednio po stworzeniu konta. Formularz logowania zawiera dwa pola, jedno służy do wpisania adresu email, z którym powiązane jest konto. W drugie pole należy wpisać hasło, które zostało ustalone podczas rejestracji. Zamiast znaków wpisywanych w to pole, pojawiają się kropki. Jest to mechanizm służący podniesieniu bezpieczeństwa, aby żadna dodatkowa osoba będąca w bliskiej okolicy użytkownika

nie była w stanie przeczytać hasła, które jest wprowadzane. Po wypełnieniu obydwu pól wystarczy kliknąć niebieski przycisk pod formularzem.

W przypadku poprawnego wprowadzenia emailu oraz hasła, na ekranie ukazuje się panel użytkownika (rys. 4.4). Służy on przede wszystkim do dodawania powiązań między kierowcami, pojazdami i lokalizatorami, natomiast w pierwszej kolejności trzeba dodać poszczególne obiekty, jakimi są kierowca, pojazd czy lokalizator. Niemniej jednak, założono, że w trakcie używania aplikacji, użytkownicy nie często będą dodawać np. nowe pojazdy, stąd pierwszą zakładką, jaka pokazuje się po zalogowaniu, jest panel użytkownika.

Aby dodać kierowcę, w pierwszej kolejności należy przejść do zakładki nazwanej „Kierowcy”. Podstawowym elementem tej części aplikacji jest lista kierowców. W przypadku gdy lista jest pusta, wyświetlany jest tekst „Brak kierowców.”, widoczny na rys. 4.5. Pod listą lub napisem świadczącym o pustej liście znajduje się przycisk służący do dodania nowego kierowcy do listy. Po naciśnięciu przycisku otworzy się formularz - jego wygląd jest dostępny na rys. 4.6. Należy wypełnić jego pola, którymi są imię oraz nazwisko kierowcy. Przycisk oznaczony napisem „Dodaj kierowcę” pozwala dokończyć proces i zapisać wprowadzone dane w systemie. Formularz zostanie zamknięty, a na liście pojawi się nowa pozycja - przykład widoku z przykładowym kierowcą przedstawia rys. 4.7. W przypadku kliknięcia czerwonego przycisku „Anuluj”, proces dodawania kierowcy zostanie wycofany, a aplikacja wróci do widoku zakładki z listą kierowców. Będąc tej w zakładce, użytkownik ma również możliwość usunięcia wybranego kierowcy. Służy do tego przycisk „Usuń”, znajdujący się po prawej stronie od każdej pozycji na liście.

Zakładką o analogicznym działaniu jest zakładka „Pojazdy”. Posiada listę samochodów zalogowanego użytkownika oraz przycisk pozwalający dodać do niej nową pozycję - wygląd z pustą listą znajduje się na rys. 4.8. Formularz w tym przypadku posiada cztery pola: marka, model, tablica rejestracyjna oraz VIN. Pierwsze dwa służą przede wszystkim zwiększeniu przejrzystości listy dla użytkownika, natomiast trzecie i czwarte pole umożliwiają rozróżnienie poszczególnych samochodów, podczas gdy są one tej samej marki oraz tego samego modelu. Rys. 4.9 przedstawia formularz dodawania pojazdu. Usuwanie elementów z listy jest dostępne również za pomocą przycisku znajdującego się po prawej stronie od każdego pojazdu.

„Lokalizatory” to trzecia zakładka o podobnej zasadzie działania. Zrzut ekranu z jej wyglądem jest dostępny na rys. 4.10. Lista zawiera lokalizatory dodane przez użytkownika. Po kliknięciu w przycisk zatytułowany „Dodaj lokalizator”, otworzy się formularz, tym razem posiadający trzy pola do wypełnienia (rys. 4.11). Nazwa i typ są polami dowolnymi, które mają na celu ułatwić użytkownikowi znalezienie pożądanego lokalizatora

na liście, natomiast numer seryjny jest polem identyfikacyjnym, które jest niepowtarzalne dla każdego urządzenia. Usuwanie odbywa się za pomocą przycisku „Usuń”, widocznego na rys. 4.10.

Po dodaniu kierowców, pojazdów i lokalizatorów można przejść do tworzenia powiązań między nimi. W tym celu należy otworzyć zakładkę „Moje konto”, która przedstawia panel użytkownika. Aplikacja pozwala na stworzenie dwóch rodzajów powiązań:

- kierowca pojazdu
- lokalizator w pojeździe

Kierowcy pojazdów, będący jednocześnie tytułem pierwszej listy w panelu użytkownika, definiują w jakim okresie czasu dany kierowca korzysta z danego pojazdu. Będzie to miało szczególne znaczenie podczas wyświetlania tras przebytych przez poszczególną osobę. W celu dodania powiązania, należy kliknąć przycisk „Dodaj kierowcę” (rys. 4.4). Aplikacja wyświetli formularz składający się z czterech pól (rys. 4.12). Pierwsze dwa są polami rozwijanymi, elementami pierwszego będą wszystkie pojazdy użytkownika, natomiast w drugim znajdować się będą jego kierowcy. Kolejne dwa pola to początek i koniec okresu użytkowania wybranego samochodu przez wybraną osobę. W celu wyznaczenia dat należy wybrać odpowiedni dzień z wyświetlonego kalendarza. Aby dokończyć proces tworzenia powiązania, należy kliknąć przycisk z napisem „Dodaj powiązanie”. Powiązanie zostanie zapisane do bazy danych i wyświetli się jako nowy element listy. Przycisk „Anuluj” czyści pola formularza jednocześnie go zamykając.

Drugą opcją powiązań są lokalizatory w pojazdach. Ich lista znajduje się pod listą kierowców pojazdów. Podobnie jak w przypadku pierwszego powiązania, formularz, który pozwala na dodanie nowego elementu od listy, ma cztery pola. Pierwsze pole jest niezmiennione, co można zobaczyć na rys. 4.13. Jest to spowodowane faktem, iż to powiązanie również wykorzystuje pojazdy, zatem można w tym miejscu dokonać wyboru samochodu. Drugie pole uległo zmianie, jest to rozwijana lista lokalizatorów, z których należy wybrać jeden, który w danym okresie użytkownik planuje umieścić w pojeździe. Okres powiązania definiują kolejne dwa pola formularza, które również nie różnią się od pól wypełnianych podczas dodawania powiązania kierowcy z pojazdem. Przyciski oraz ich działanie jest także analogiczne do powyżej opisanego powiązania.

Usuwanie powiązań dokonuje się w identyczny sposób, w jaki zostało opisane usuwanie pojedynczych obiektów. Niemniej jednak, ważnym aspektem usuwania kierowców, pojazdów i lokalizatorów jest fakt, iż w przypadku uczestnictwa danego obiektu w jakimkolwiek

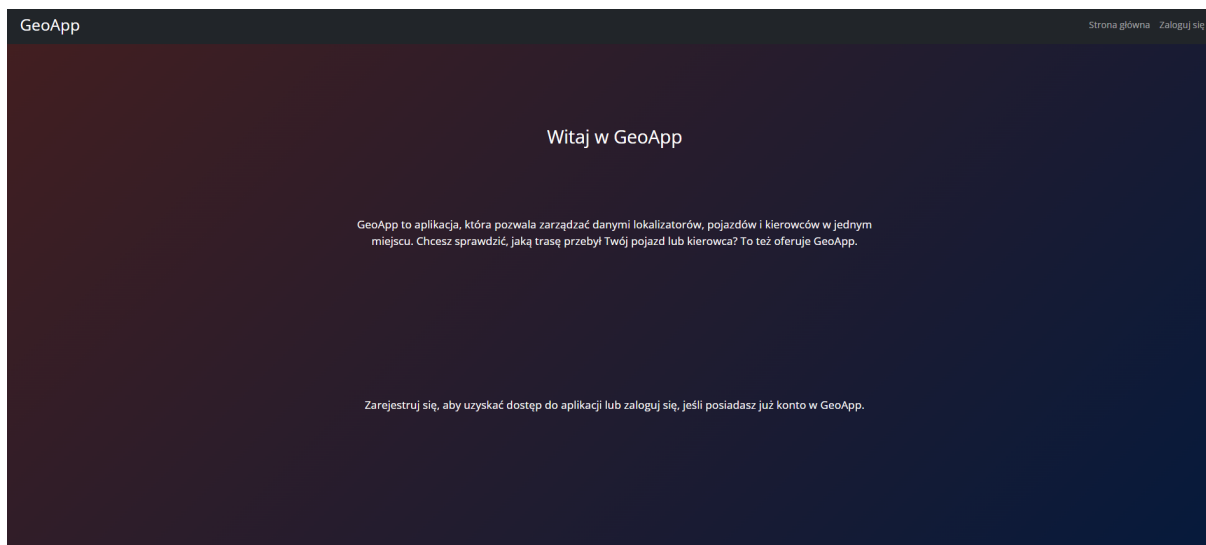
powiązaniu, niedozwolone jest jego usunięcie. Pierwszym krokiem jest wtedy wyeliminowanie powiązania, z którym związany jest wybrany przez użytkownika np. kierowca, a następnie przeprowadzenie procesu usunięcia kierowcy.

Zakładka pod tytułem „Mapa” pozwala na wyświetlenie tras w danym okresie czasu. Po jej otwarciu oczom użytkownika ukazuje się widok z rys. 4.14. Mapę można oddalać oraz przybliżać, korzystając z kółka myszy, a także przesuwając, poruszając myszą przy wciśniętym lewym przycisku. Pod mapą znajdziemy dwie listy z trasami, kierowców oraz pojazdów. Aby wyświetlić trasę kierowcy, należy kliknąć przycisk „Dodaj kierowcę”. Aplikacja wyświetli formularz posiadający trzy pola (rys. 4.15). Pierwsze jest rozwijaną listą, która umożliwia wybór kierowcy. Drugie to data będąca początkiem okresu, z którego trasa ma zostać wyświetlona, natomiast trzecie pole oznacza datę końca tego okresu (rys. 4.16). Podobnie jak w przypadku poprzednich formularzy: niebieski przycisk służy do zapisania danych wprowadzonych do formularza, podczas gdy czerwony powoduje anulowanie procesu i powrót do widoku zakładki. Dodanie trasy pojazdu następuje w analogiczny sposób, z różnicą w pierwszym polu formularza. W tym wypadku pole służy do wyboru samochodu z listy rozwijanej. Po poprawnym dodaniu trasy, do listy zostaje dołączony nowy element. Na mapie również pojawia się wtedy nowo dodana trasa. Jej kolor można sprawdzić na próbce koloru - posiada ją każda trasa na liście. Widok z jedną trasą przedstawia rys. 4.17. Pozycje z list można usuwać, klikając przycisk „Usuń” po prawej stronie każdej z tras.

Na pasku nawigacyjnym zalogowanego użytkownika widnieje jeszcze jedna zakładka - „Wyloguj się” (rys. 4.4). Jest to tak naprawdę przycisk umożliwiający wylogowanie się z konta użytkownika. Po jego kliknięciu, aplikacja nas wylogowuje i przenosi do strony głównej. Pojawia się wtedy pasek nawigacji widoczny na rys. 4.1, co sprawia, iż w celu dostępu do funkcji zalogowanego użytkownika, należy ponownie się zalogować zgodnie z wyżej opisanymi krokami.

Jeśli „Specyfikacja zewnętrzna”:

- wymagania sprzętowe i programowe
- sposób instalacji
- sposób aktywacji
- kategorie użytkowników
- sposób obsługi
- administracja systemem
- kwestie bezpieczeństwa



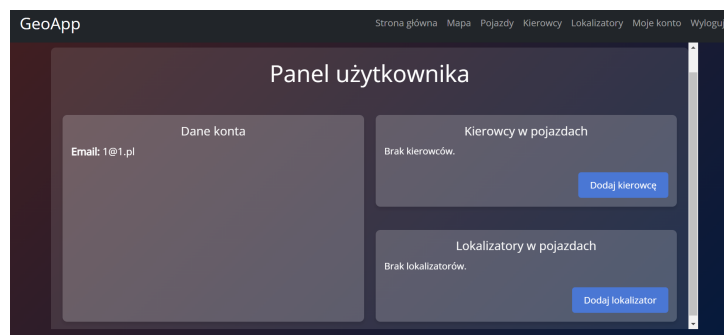
Rysunek 4.1: Zrzut ekranu przedstawiający stronę główną.

This is a screenshot of the login form. It has a dark background with a central light gray box. The title 'Zaloguj się na swoje konto' is at the top. Below it are two input fields labeled 'Email' and 'Hasło'. At the bottom left of the box is a link 'Nie masz jeszcze konta? Zarejestruj się' and at the bottom right is a blue button labeled 'Zaloguj'.

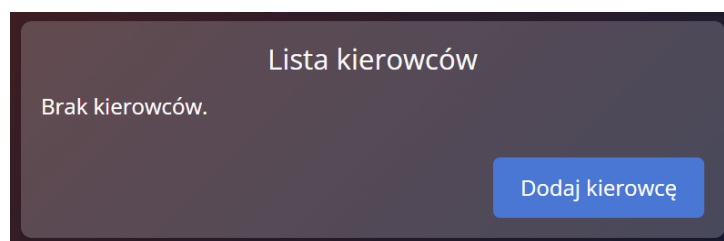
Rysunek 4.2: Zrzut ekranu przedstawiający formularz logowania.

This is a screenshot of the registration form. It has a dark background with a central light gray box. The title 'Utwórz nowe konto' is at the top. Below it are two input fields labeled 'Email' and 'Hasło'. At the bottom left of the box is a link 'Masz już konto? Zaloguj się' and at the bottom right is a blue button labeled 'Zarejestruj się'.

Rysunek 4.3: Zrzut ekranu przedstawiający formularz rejestracji.



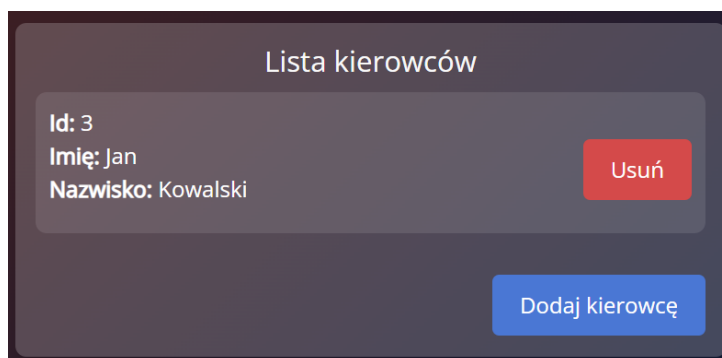
Rysunek 4.4: Zrzut ekranu przedstawiający zakładkę Moje konto.



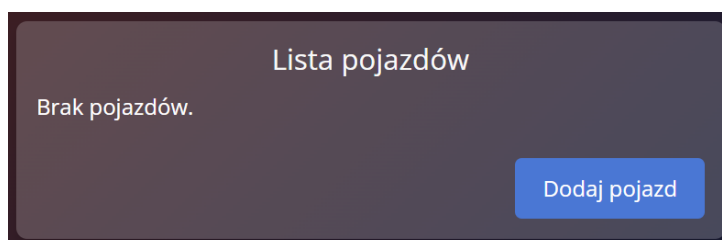
Rysunek 4.5: Zrzut ekranu przedstawiający zakładkę Kierowcy.

The screenshot shows the 'Dodaj nowego kierowcę' (Add new driver) form. It has a dark background with a light gray form container. The title 'Dodaj nowego kierowcę' is at the top. Below it are two input fields: 'Imię' (First name) and 'Nazwisko' (Last name). At the bottom of the form, there are two buttons: a blue 'Dodaj kierowcę' (Add driver) button and a red 'Anuluj' (Cancel) button.

Rysunek 4.6: Zrzut ekranu przedstawiający formularz dodawania kierowcy.



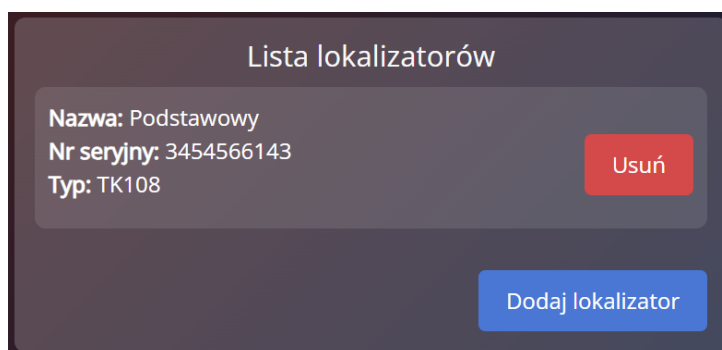
Rysunek 4.7: Zrzut ekranu przedstawiający zakładkę Kierowcy z jednym kierowcą na liście.



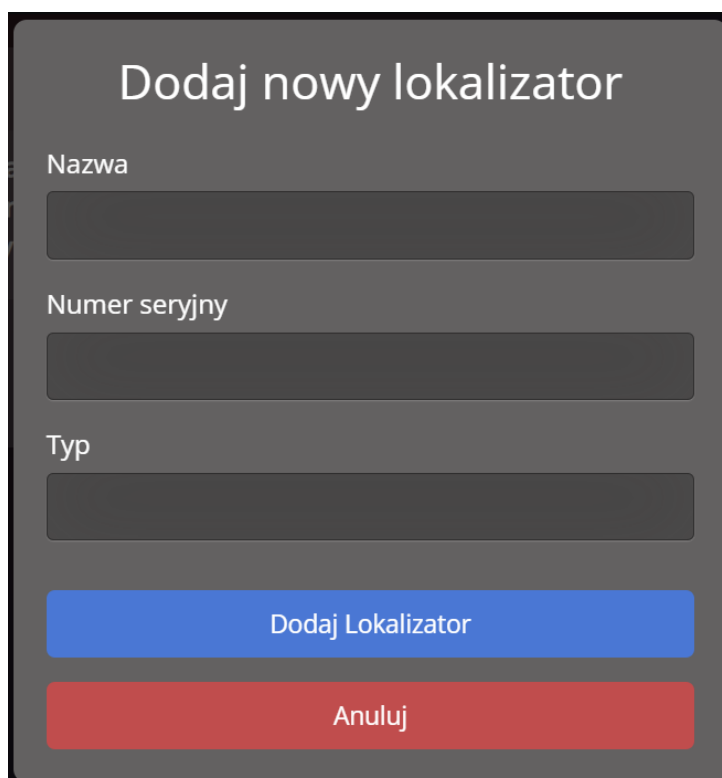
Rysunek 4.8: Zrzut ekranu przedstawiający zakładkę Pojazdy.

A screenshot of a mobile application interface titled "Dodaj nowy pojazd". It contains four input fields with labels: "Marka", "Model", "Tablica rejestracyjna", and "VIN". At the bottom of the form are two buttons: a blue button labeled "Dodaj pojazd" and a red button labeled "Anuluj".

Rysunek 4.9: Zrzut ekranu przedstawiający formularz dodawania pojazdu.



Rysunek 4.10: Zrzut ekranu przedstawiający zakładkę Lokalizatory.



Rysunek 4.11: Zrzut ekranu przedstawiający formularz dodawania lokalizatora.

Powiąz pojazd z kierowcą

Wybierz pojazd

-- Wybierz pojazd --

Wybierz kierowcę

-- Wybierz kierowcę --

Data od

dd.mm.rrrr

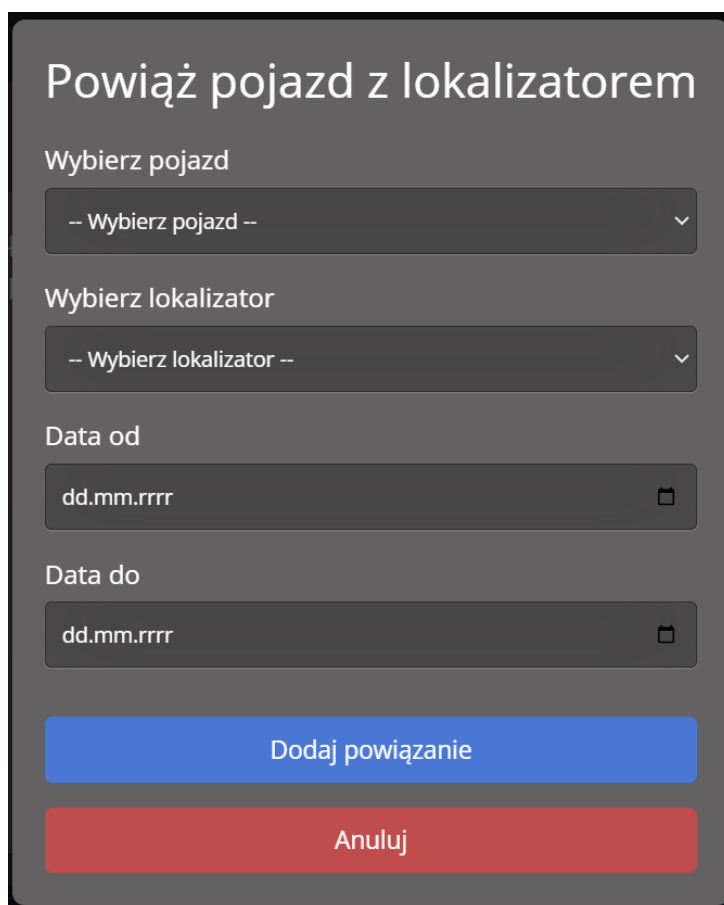
Data do

dd.mm.rrrr

Dodaj powiązanie

Anuluj

Rysunek 4.12: Zrzut ekranu przedstawiający formularz dodawania powiązania kierowcy z pojazdem.



Powiązanie pojazdu z lokalizatorem

Wybierz pojazd

-- Wybierz pojazd --

Wybierz lokalizator

-- Wybierz lokalizator --

Data od

dd.mm.rrrr

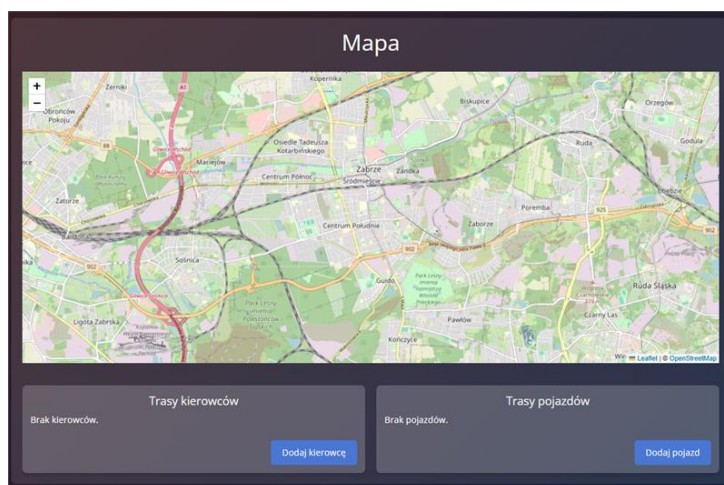
Data do

dd.mm.rrrr

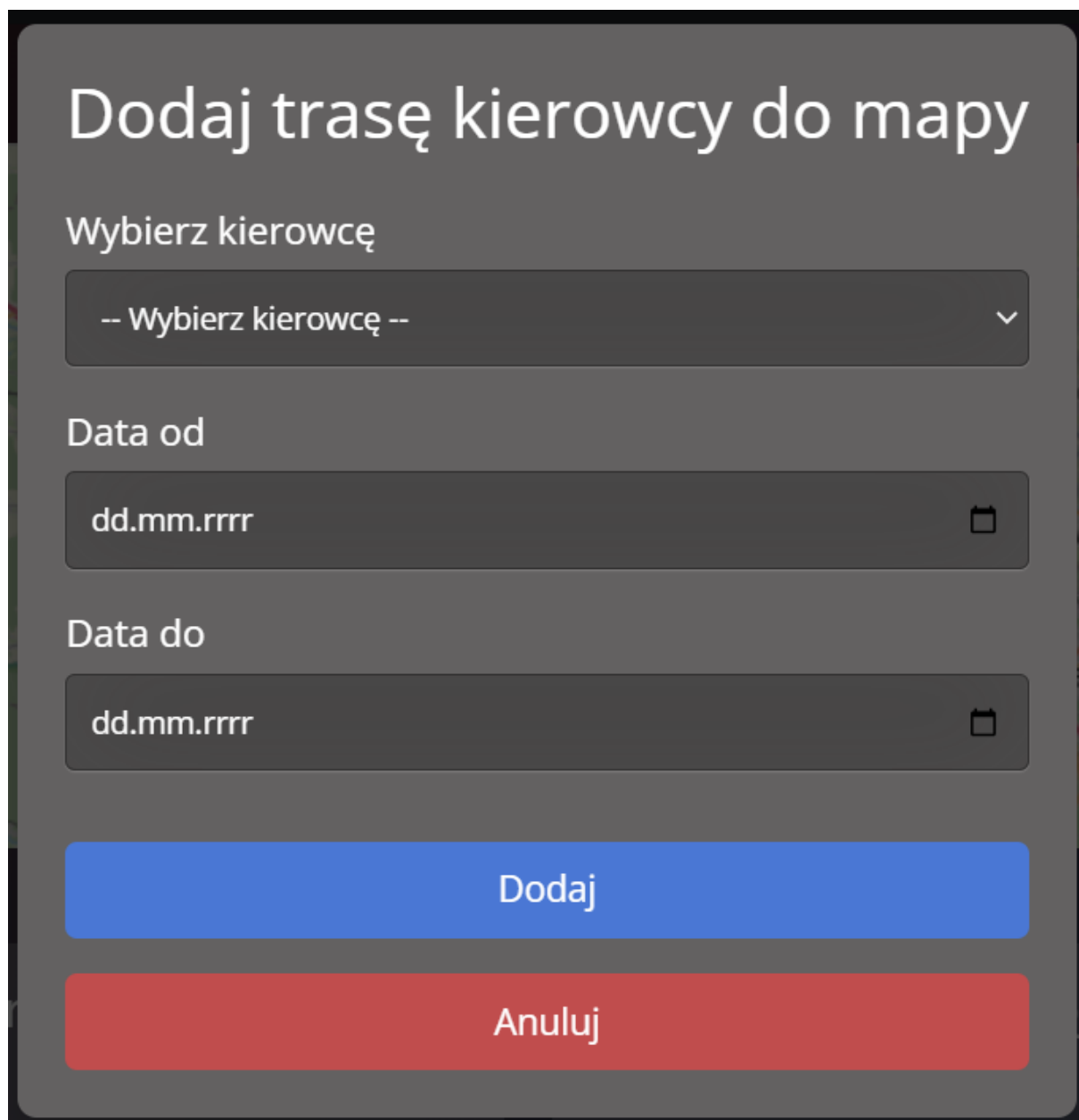
Dodaj powiązanie

Anuluj

Rysunek 4.13: Zrzut ekranu przedstawiający formularz dodawania powiązania lokalizatora z pojazdem.



Rysunek 4.14: Zrzut ekranu przedstawiający zakładkę Mapa.



Dodaj trasę kierowcy do mapy

Wybierz kierowcę

-- Wybierz kierowcę --

Data od

dd.mm.rrrr

Data do

dd.mm.rrrr

Dodaj

Anuluj

Rysunek 4.15: Zrzut ekranu przedstawiający formularz dodawania trasy kierowcy.

Dodaj trasę pojazdu do mapy

Wybierz pojazd

-- Wybierz pojazd --

Data od

dd.mm.rrrr

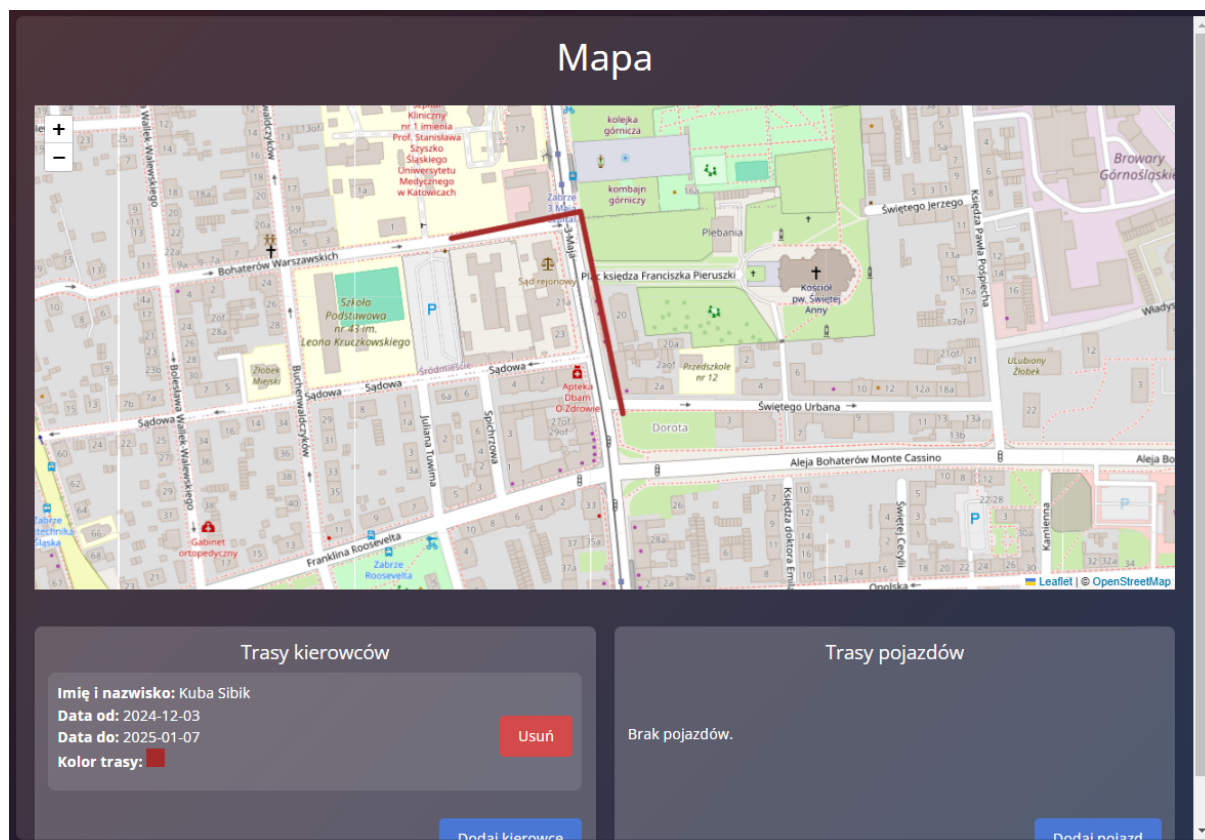
Data do

dd.mm.rrrr

Dodaj

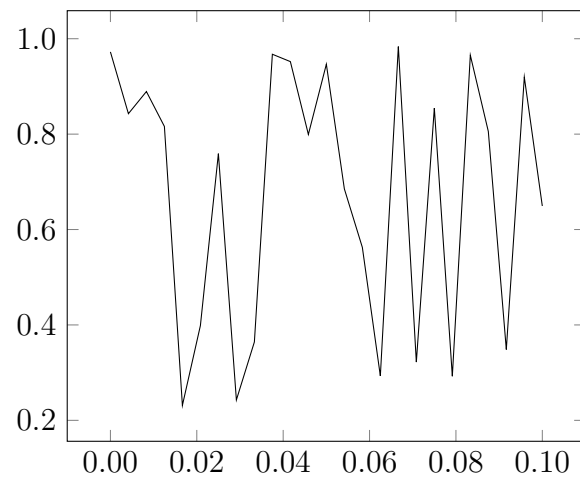
Anuluj

Rysunek 4.16: Zrzut ekranu przedstawiający formularz dodawania trasy pojazdu.



Rysunek 4.17: Zrzut ekranu przedstawiający mapę z jedną trasą.

- przykład działania
- scenariusze korzystania z systemu (ilustrowane zrzutami z ekranu lub generowanymi dokumentami)



Rysunek 4.18: Podpis rysunku po rysunkiem.

Rozdział 5

Specyfikacja wewnętrzna

5.1 Przedstawienie idei

Widok użytkownika napisany został w języku TypeScript, bazuje na komponentach biblioteki React. Podłoże aplikacji jest oparte na języku Java. Znanym wzorcem projektowym jest MVC (ang. Model View Controller), co przetłumaczone na język polski brzmi: model, widok, kontroler. Model to część odpowiadająca za przechowywanie danych. Widok to część dostępna dla użytkownika aplikacji, natomiast kontroler wykonuje operacje i jest odpowiedzialny za logikę programu. Do stworzenia programu wykorzystano architekturę warstwową, która bazuje na MVC oraz skorzystano z wzorca projektowego o nazwie repozytorium. Repozytorium oznacza oddzielenie logiki biznesowej od bazy danych, natomiast architektura warstwowa zakłada istnienie takich warstw jak: encje - będące reprezentacją danych, kontrolery - obsługujące żądania HTTP, serwisy - odpowiedzialne za logikę biznesową i będące wywoływanymi przez kontrolery, repozytoria - mające dostęp do bazy danych, wykonujące operacje zapisu lub pobrania danych. Mapowanie encji na tabele w bazie danych, jak i odwrotnie, jest dostępne dzięki zastosowaniu adnotacji Hibernate.

5.2 Opis struktur danych

Dane w kodzie języka Java są reprezentowane przez encje. Są to klasy posiadające pola - zmienne odzwierciedlające kolumny w tabelach w bazie danych. Ich obiekty są odwzorowaniem bazodanowych rekordów. Przykład kodu można zobaczyć na rys. 5.1. Każda zmienna posiada adnotację (poprzedzoną znakiem "@"), jest to składania Hibernate. Dzięki temu jest możliwe mapowanie między obiektem encji, a rekordem w tabeli. Encje posiadają również funkcje nazywane getterami - do pozyskiwania danych z konkretnych pól oraz setterami - do wpisywania wartości do pól.

Aby wysyłać i odbierać dane od tzw. frontendu, czyli części aplikacji dotyczącej interfejsu użytkownika, niezbędne są klasy reprezentujące żądania (ang. request) - odbierane są przez kontrolery oraz odpowiedzi (ang. response) - wysyłane dane do części z językiem TypeScript. Zazwyczaj mają pola analogiczne do encji. Posiadają również funkcje pobierające wartości z konkretnych zmiennych oraz zapisujące wartości do nich, tak jak jest to w przypadku encji. Odpowiedzi są również wyposażone w funkcje konwertujące obiekt encji na obiekt odpowiedzi. Przykład klasy żądania znajduje się na rys. 5.2, a metody do konwersji, będącej w klasie odpowiedzi, na rys. 5.4

Analogiczne typy danych zostały stworzone w języku TypeScript w formie interfejsów, pozwalają one w prosty sposób wysyłać żądania i odbierać odpowiedzi. Posiadają one zmienne o identycznych nazwach co klasy w języku Java oraz typy, które pozwalają na automatyczne ich konwertowanie między językami podczas wysyłania i odbierania żądań HTTP. Wygląd interfejsów przedstawia rys. 5.3.

5.3 Szczegóły implementacji wybranych fragmentów

Do stworzenia interfejsu użytkownika wykorzystano komponenty funkcyjne z biblioteki React. Każda zakładka aplikacji posiada oddzielny komponent - każdy w osobnym pliku. Zwracają one JSX, czyli kod zbliżony do HTML, w którym można zagnieźdzać inne elementy. Niemniej jednak, należy pamiętać o zasadzie, iż zwrócić można tylko jeden element nadrzędny - to w nim można umieścić kolejne. Rys.5.5 pokazuje zwracany JSX przez zakładkę strony głównej (tekst z elementów `<p>` został wycięty).

Ważnym elementem stworzonej aplikacji są formularze, to dzięki nim użytkownik dodaje obiekty czy powiązania. W tym celu została wykorzystana biblioteka React Hook Form, dostępna w React. Stworzono typy danych, którego przykład można zobaczyć na rys. 5.6. Aby skorzystać z biblioteki, wykorzystano Hook do destrukuryzacji niezbędnych stałych (rys. 5.7). Aby skorzystać z obsługi formularza, zarejestrowano pola oraz dodano walidacje - przykład na rys. 5.8.

Jeśli „Specyfikacja wewnętrzna”:

- przedstawienie idei
- architektura systemu
- opis struktur danych (i organizacji baz danych)
- komponenty, moduły, biblioteki, przegląd ważniejszych klas (jeśli występują)
- przegląd ważniejszych algorytmów (jeśli występują)
- szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe

- diagramy UML

Krótką wstawka kodu w linii tekstu jest możliwa, np. **int a;** (biblioteka `listings`). Dłuższe fragmenty lepiej jest umieszczać jako rysunek, np. kod na rys 5.10, a naprawdę długie fragmenty – w załączniku.

```
1 @Entity
2 @Table(name = "user", schema = "adm")
3 public class UserEntity {
4     @Id
5     @Column
6     private String email;
7
8     @Column
9     private String password;
10
11     @OneToMany(mappedBy = "user", fetch = FetchType.LAZY)
12     private Set<DriverEntity> driver;
13
14     public String getEmail() {
15         return email;
16     }
17
18     public void setEmail(String username) {
19         this.email = username;
20     }
21
22     public String getPassword() {
23         return password;
24     }
25
26     public void setPassword(String password) {
27         this.password = password;
28     }
29
30     public Set<DriverEntity> getDriver() {
31         return driver;
32     }
33
34     public void setDriver(Set<DriverEntity> driver) {
35         this.driver = driver;
36     }
37 }
```

Rysunek 5.1: Kod encji użytkownika - Java.

```

1 public class UserRequest {
2
3     @NotNull
4     private String email;
5
6     @NotNull
7     private String password;
8
9     public String getEmail() {
10         return email;
11     }
12
13     public void setEmail(String email) {
14         this.email = email;
15     }
16
17     public String getPassword() {
18         return password;
19     }
20
21     public void setPassword(String password) {
22         this.password = password;
23     }
24 }

```

Rysunek 5.2: Kod żądania użytkownika.

```

1 export interface UserRequest {
2     email: string | null;
3     password: string | null;
4 }

```

Rysunek 5.3: Kod żądania użytkownika - TypeScript.

```

1 public static TrackerResponse fromEntity(TrackerEntity entity){
2     TrackerResponse dto = new TrackerResponse();
3     dto.setSerialNumber(entity.getSerialNumber());
4     dto.setName(entity.getName());
5     dto.setType(entity.getType());
6     return dto;
7 }

```

Rysunek 5.4: Funkcja konwertująca encję na odpowiedź.

```

1 return (
2   <div className="main-container">
3     <h1 className="welcome-title">Witaj w GeoApp</h1>
4     <p className="text_description1">
5       </p>
6     <p className="text_description2">
7       </p>
8   </div>
9 );

```

Rysunek 5.5: Zwracany JSX przez zakładkę strony głównej.

```

1 type FormState = {
2   name: string;
3   surname: string;
4 };

```

Rysunek 5.6: Typ danych do formularza.

```

1 const [formState, setFormState] = useState<string>("addDriver");
2 const {
3   register,
4   handleSubmit,
5   formState: { errors },
6   reset,
7 } = useForm<FormState>();

```

Rysunek 5.7: Destrukturyzacja stałych z biblioteki React Hook Form.

```

1 <label className="form-label" htmlFor="name">Imie</label>
2 <input
3   className="form-input"
4   id="name"
5   {...register("name", { required: "Imie_jest_wymagane" })}
6 />
7 {errors.name && <p className="form-error_form-validation-text">{
   errors.name.message}</p>}

```

Rysunek 5.8: Rejestracja i walidacja pól.

```
1 class test : public basic
2 {
3     public:
4         test (int a);
5         friend std::ostream operator<<(std::ostream & s,
6                                         const test & t);
7     protected:
8         int _a;
9
10 };
```

Rysunek 5.9: Pseudokod w listings.

```
1 class test : public basic
2 {
3     public:
4         test (int a);
5         friend std::ostream operator<<(std::ostream & s,
6                                         const test & t);
7     protected:
8         int _a;
9
10 };
```

Rysunek 5.10: Pseudokod w listings.

Rozdział 6

Weryfikacja i walidacja

6.1 Sposób testowania

Aby przetestować działanie serwera oraz prawidłowość w konfiguracji lokalizatora wykorzystano maszynę wirtualną z systemem Linux Ubuntu, która posiadała publiczny adres IP. Zakupiono lokalizator Mking MK07A oraz kartę SIM z pakietem internetu oraz wysłką wiadomości SMS. Następnie urządzenie zostało skonfigurowany poprzez SMS-y tak, aby wysyłało dane na adres maszyny wirtualnej z użyciem portu 8080. Jednak serwer w języku Java był dostępny na lokalnym komputerze stacjonarnym z systemem Windows. Aby przekierować pakiety na serwer, skorzystano z programu MobaXTerm, który umożliwia tunelowanie portów TCP. Kolejnym korkiem było zatrzymywanie serwera w odpowiednich miejscach w programie IntelliJ Idea, dzięki czemu następnie eliminowano błędy wynikające z niewłaściwej interpretacji pakietów oraz poprawiano kod serwera.

W celu sprawdzenia aplikacji pod kątem błędów przechodzono przez jej zakładki, a w nich przez formularze. Testowano ich pola oraz czy zostają one przesłane, podczas gdy brakuje wymaganych pól.

6.2 Wykryte i usunięte błędy

Podczas testów aplikacji wykryto, że formularze pozwalają wpisać w pola dat zakończenia daty wcześniejsze niż te będące już w polach dat rozpoczęcia. Rozwiązano ten błąd poprzez wprowadzenie walidacji na pola z datami, aby data do była musiała być późniejszą datą niż data od.

- sposób testowania w ramach pracy (np. odniesienie do modelu V)
- organizacja eksperymentów
- przypadki testowe zakres testowania (pełny/niepełny)

Tabela 6.1: Nagłówek tabeli jest nad tabelą.

ζ	metoda						
	alg. 1	alg. 2	alg. 3			alg. 4, $\gamma = 2$	
			$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

- wykryte i usunięte błędy
- opcjonalnie wyniki badań eksperymentalnych

Rozdział 7

Podsumowanie i wnioski

7.1 Uzyskane wyniki w świetle postawionych celów i zdefiniowanych wymagań

Aplikacja została napisana zgodnie z wymaganiami, które zostały przedstawione. Odpowiednio skonfigurowany lokalizator - co ułatwia instrukcja dostępna w programie - poprawnie współpracuje z serwerem, w wyniku czego można korzystać z aplikacji w sposób zgodny z początkowymi celami. Prosty interfejs pozwala użytkownikowi na sprawne i przyjemne obsługiwanie programu. Widok został dostosowany do różnej wielkości ekranów, co wskazuje na sukces podczas wprowadzania skalowalności. Wszystkie operacje, które miały być dostępne do wykonania przez użytkownika, i które miały na celu osiągnięcie użytecznej aplikacji, zostały wprowadzone.

7.2 Kierunek ewentualnych prac

Dalszym etapem rozwijania aplikacji byłoby przede wszystkim zwiększenie dostępnych typów lokalizatorów, wspieranych przez program. W tym celu należy rozbudować kod serwera, odbierającego dane geograficzne od urządzeń. Przykładem innego lokalizatora jest Coban TK108, który również umożliwia wysyłanie pakietów TCP na niestandardowy adres IP oraz port. Takie działania zwiększyłyby grupę potencjalnych użytkowników aplikacji, gdyż nie byłiby oni ograniczeni do zakupu jednego modelu lokalizatora.

- uzyskane wyniki w świetle postawionych celów i zdefiniowanych wyżej wymagań
- kierunki ewentualnych danyh prac (rozbudowa funkcjonalna ...)
- problemy napotkane w trakcie pracy

Bibliografia

- [1] Imię Nazwisko i Imię Nazwisko. *Tytuł strony internetowej*. 2021. URL: <http://gdzies/w/internecie/internet.html> (term. wiz. 30.09.2021).
- [2] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. „Tytuł artykułu konferencyjnego”. W: *Nazwa konferencji*. 2006, s. 5346–5349.
- [3] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. „Tytuł artykułu w czasopiśmie”. W: *Tytuł czasopisma* 157.8 (2016), s. 1092–1113.
- [4] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. *Tytuł książki*. Warszawa: Wydawnictwo, 2017. ISBN: 83-204-3229-9-434.

Dodatki

Spis skrótów i symboli

DNA kwas deoksyrybonukleinowy (ang. *deoxyribonucleic acid*)

MVC model – widok – kontroler (ang. *model-view-controller*)

N liczebność zbioru danych

μ stopnień przyleżności do zbioru

\mathbb{E} zbiór krawędzi grafu

\mathcal{L} transformata Laplace’a

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2     throw std::string ("unknown_number_of_clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4     throw std::string ("You should set a maximal number of
        iteration or minimal difference — epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6     throw std::string ("Both number of iterations and minimal
        epsilon set — you should set either number of iterations
        or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

3.1	Diagram przypadków użycia niezalogowanego użytkownika.	7
3.2	Diagram przypadków użycia zalogowanego użytkownika.	8
4.1	Zrzut ekranu przedstawiający stronę główną.	18
4.2	Zrzut ekranu przedstawiający formularz logowania.	18
4.3	Zrzut ekranu przedstawiający formularz rejestracji.	18
4.4	Zrzut ekranu przedstawiający zakładkę Moje konto.	19
4.5	Zrzut ekranu przedstawiający zakładkę Kierowcy.	19
4.6	Zrzut ekranu przedstawiający formularz dodawania kierowcy.	19
4.7	Zrzut ekranu przedstawiający zakładkę Kierowcy z jednym kierowcą na liście.	20
4.8	Zrzut ekranu przedstawiający zakładkę Pojazdy.	20
4.9	Zrzut ekranu przedstawiający formularz dodawania pojazdu.	20
4.10	Zrzut ekranu przedstawiający zakładkę Lokalizatory.	21
4.11	Zrzut ekranu przedstawiający formularz dodawania lokalizatora.	21
4.12	Zrzut ekranu przedstawiający formularz dodawania powiązania kierowcy z pojazdem.	22
4.13	Zrzut ekranu przedstawiający formularz dodawania powiązania lokalizatora z pojazdem.	23
4.14	Zrzut ekranu przedstawiający zakładkę Mapa.	23
4.15	Zrzut ekranu przedstawiający formularz dodawania trasy kierowcy.	24
4.16	Zrzut ekranu przedstawiający formularz dodawania trasy pojazdu.	25
4.17	Zrzut ekranu przedstawiający mapę z jedną trasą.	26
4.18	Podpis rysunku po rysunkiem.	27
5.1	Kod encji użytkownika - Java.	32
5.2	Kod żądania użytkownika.	33
5.3	Kod żądania użytkownika - TypeScript.	33
5.4	Funkcja konwertująca encję na odpowiedź.	33
5.5	Zwracany JSX przez zakładkę strony głównej.	34
5.6	Typ danych do formularza.	34
5.7	Destrukturyzacja stałych z biblioteki React Hook Form.	34

5.8	Rejestracja i walidacja pól.	34
5.9	Pseudokod w <code>listings</code>	35
5.10	Pseudokod w <code>listings</code>	35

Spis tabel

6.1	Nagłówek tabeli jest nad tabelą.	38
-----	--	----