POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Mateusz Deleżuch
Nr albumu s19082

Inteligentny system zarządzania zbiórką odpadów

Praca magisterska

Dr inż. Mariusz Trzaska

miejsce, miesiąc, rok obrony

Streszczenie

Niniejsza praca dotyczy problemu zarządzania zbiórką odpadów. Autor analizuje dostępne na rynku systemy informatyczne usprawniające proces. Przedstawiony zostaje pomysł systemu optymalizującego trasy przejazdu pojazdów zbierających odpady w oparciu o dane napływające z symulowanych czujników. Mimo że proponowane rozwiązanie nie wymaga zastosowania fizycznych urządzeń pomiarowych to wpisuje się w koncepcję Internetu Rzeczy.

Rezultatem pracy jest prototyp systemu oparty na ogólnie dostępnych technologiach oraz zgodny z najnowszymi trendami tworzenia aplikacji webowych. Autor starał się dołożyć wszelkich starań aby aplikacja była łatwa w obsłudze. Zastosowane zostały interaktywne mapy oraz stworzono czytelny interfejs użytkownika.

Spis treści

1. Wstęp

Wytwarzanie, zbieranie, transport przetwarzanie odpadów oraz nadzór nad tymi działaniami składają się na pojęcie gospodarki odpadami. Racjonalna strategia zarządzania tym procesem to wyzwanie dla współczesnego społeczeństwa. Rozwój technologii w dziedzinach elektroniki, telekomunikacji oraz informatyki umożliwiły wsparcie systemów gospodarki odpadami. Niniejsza praca skupia się na wykorzystaniu technologii internetowych wspomagających ten proces.

1.1. Informatyzacja procesu zbiórki odpadów

Współczesne metody zarządzania zbiórką odpadów coraz częściej skupiają się na wykorzystaniu technologii internetowych. W dużej mierze nadzieje pokładane są w ciągle rozwijanym Internecie Rzeczy. Połączone sieci czujników kontrolujących zapełnienie pojemników na odpady coraz częściej znajdują zastosowanie we współczesnych miastach. Ze względu na zainteresowanie branżą Internetu Rzeczy autor pracy postanowił przeanalizować dostępne na rynku rozwiązania oraz zaproponować własne.

1.2. Cel pracy

Niniejsza praca skupia się na propozycji usprawnienia procesu zbiórki odpadów poprzez ograniczanie liczby wysyłanych samochodów zbierających odpady. W wyniku przeprowadzonych analiz przedstawione zostają wymagania systemu zarządzania zbiórką odpadów. Rozwiązanie proponowane przez autora pracy ma za zadanie optymalizację tras przejazdu śmieciarek. Założeniem była promocja ochrony środowiska, ekologicznego transportu oraz usprawnienie procesu zarządzania zbiórką odpadów.

1.3. Rozwiązania przyjęte w pracy

Realizacja tematu pracy opierała się na wykorzystaniu ogólnodostępnych, darmowych rozwiązań technologicznych. Stworzony prototyp aplikacji jest zgodny z architekturą REST. Autor zdecydował się na użycie *framework'ów* Angular oraz Grails. Wykorzystana została baza danych PostgreSql oraz serwis mapowy Bing Maps. Wymiana danych odbywa się asynchronicznie dzięki zastosowaniu biblioteki RxJS oraz standardu JSON.

1.4. Rezultaty pracy

Rezultatem pracy jest aplikacja webowa optymalizująca trasy przejazdu pojazdów zbierających odpady uwzględniając informację o ich pojemności oraz zapełnieniu pojemników. Użytkownik symuluje informacje o zapełnieniu pojemników oraz pojemności śmieciarek poprzez udostępnione formularze. Aplikacja umożliwia badanie zmian parametrów wejściowych systemu na realizację zakładanych tras.

Aplikacja jest intuicyjna w obsłudze. Na interaktywnej mapie przedstawiane są dodane przez użytkownika pojemniki oraz pojazdy. Po uruchomieniu odpowiedniej funkcji na mapie prezentowane są wyznaczone trasy przejazdu. Dane szczegółowe zoptymalizowanych tras umieszczone zostają w dynamicznych tabelach.

1.5. Organizacja pracy

Niniejsza praca przedstawia problem zarządzania zbiórką odpadów przechodząc przez zasady postępowania z nimi. Autor przybliża istniejące systemy gospodarki odpadami oraz wykorzystywane w nich metody unieszkodliwiania śmieci.

Kolejno przedstawione zostają istniejące rozwiązania informatyczne wspomagające zarządzanie procesem zbiórki odpadów. Autor ocenia wady i zalety dostępnych rozwiązań.

Następnie opisując ogólne zasady działania Internetu Rzeczy autor pracy przedstawia swój pomysł na wsparcie instytucji zajmujących się gospodarką odpadami. Wymienione zostają wymagania jakim sprostać musi zakładany prototyp systemu.

W kolejnym etapie przedstawione zostają narzędzia oraz technologie zastosowane przy realizacji tematu pracy. Autor uzasadnia wybór konkretnych rozwiązań.

Następnie bardziej szczegółowo zostaje przedstawiony zrealizowany prototyp. Opisane zostają konkretne rozwiązania implementacyjne, które umożliwiły realizację założonych wymagań. Zaprezentowany zostaje ostateczny wygląd stworzonego systemu wraz z opisem jego funkcjonalności. Autor pracy przedstawia jak zmiana parametrów systemu wpływa na działanie prototypu aplikacji. Przybliżając warunki świata rzeczywistego na przykładzie wybranego miasta wykonana zostaje symulacja procesu zbiórki odpadów.

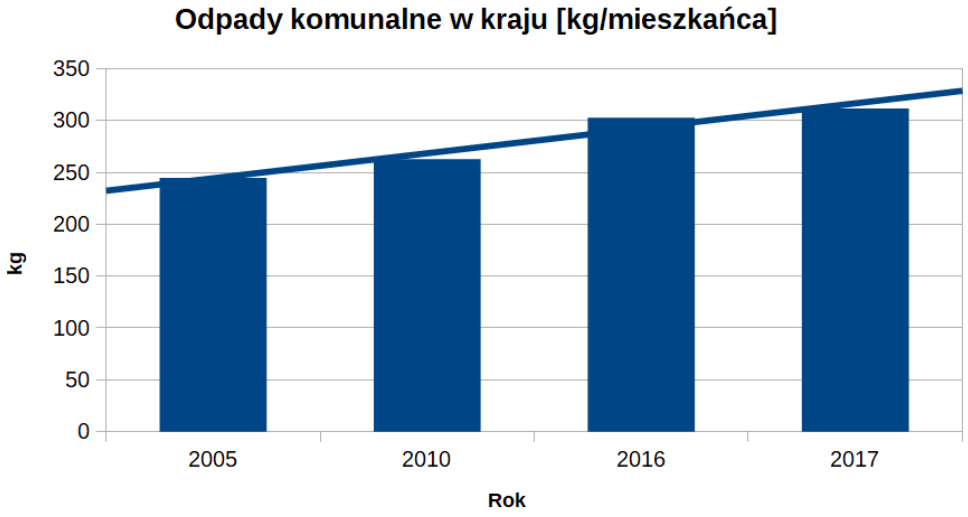
Ostatni etap skupia się na przedstawieniu wad i zalet przyjętego rozwiązania. Określone zostają konieczne kroki jakie należy podjąć w celu wdrożenia rezultatu pracy w warunkach rzeczywistych.

2. Problem zbiórki odpadów

Racjonalna gospodarka odpadami jest jednym z większych problemów, z którym borykają się współczesne miasta. Brak zrównoważonej strategii zarządzania odbija się na sprawności funkcjonowania danego obszaru oraz jego prezencji. Informatyzacja procesu zbiórki odpadów wpływa na zyski ekonomiczne oraz udogodnienia dla osób w nią uwikłanych. Rozdział przedstawia problem zbiórki odpadów uwzględniając podstawy prawne, podejście logistyczne oraz używane środki techniczne.

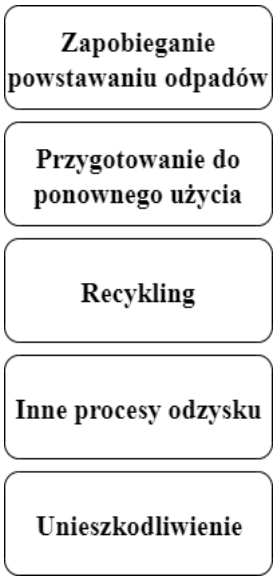
2.1. Opis natury problemu

Wzrost gospodarczy wraz z rosnącą konsumpcją niosą za sobą coraz większa produkcję odpadów. Widoczna na wykresie (patrz rysunek 1) linia trendu obrazuje zmianę ilości generowanych odpadów przez mieszkańców Polski na przestrzeni lat. Progresja wskaźnika skłania do przemyśleń odnośnie konieczności wprowadzania środków zapobiegawczych. Metodą walki z takim stanem może być przyjęcie innej strategii planowania gospodarki odpadami, uwzględniając nowe rozwiązania technologiczne oraz logistyczne.



Rysunek 1. Wykres przedstawiający ilość zebranych odpadów w danych latach w kg/mieszkańca. Źródło: [28]

Zrównoważona strategia gospodarki odpadami opiera się na ponownym wykorzystaniu zużytych produktów. Polityka ta ogranicza konsumpcję zasobów naturalnych oraz produkcje nowych śmieci. Odzyskane w ten sposób surowce mogą stać się cenne dla gospodarki. Główny cel gospodarki odpadami to ochrona życia i zdrowia ludzi oraz ochrona środowiska. Podmioty za nią odpowiedzialne powinny w szczególności unikać zanieczyszczenia wody, powietrza i gleby. Równie istotnym aspektem jest niestwarzanie zagrożenia dla roślin oraz zwierząt. Gospodarka odpadami nie powinna powodować uciążliwości zapachowych oraz generować nadmiernego hałasu. Niedozwolona jest również ingerencja w tereny wiejskie oraz miejsca o znaczeniu kulturowym bądź przyrodniczym. W Polsce obowiązuje hierarchia sposobów postępowania z opadami przedstawiona na rysunku 2.



Rysunek 2. Hierarchia sposobów postępowania z odpadami. Źródło: [1]

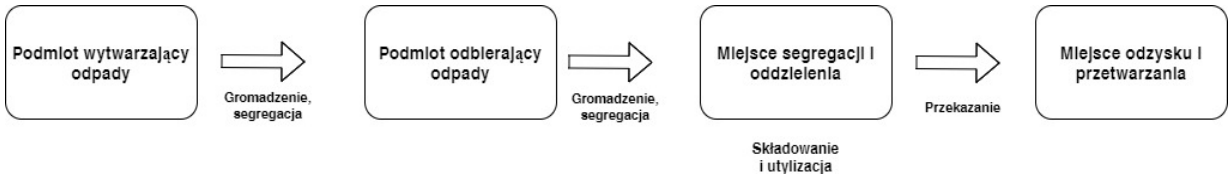
Podmioty generujące odpady powinny kierować się hierarchią postępowania przedstawioną na rysunku 2, aby zminimalizować ich negatywne skutki oddziaływania na człowieka oraz środowisko. Posiadacz śmieci ma prawo do ich unieszkodliwienia wyłącznie jeżeli nie mogą one ulec procesowi odzysku. Zgodnie z zasadą bliskości [1] odpady należy w pierwszej kolejności poddawać przetworzeniu w miejscu ich powstania. Wytwórca odpadów bądź ich obecny posiadacz ponosi koszty ich unieszkodliwiania. Zbiórka śmieci przebiega w sposób selektywny. Transport prowadzony jest mając na uwadze ochronę środowiska oraz bezpieczeństwo życia i zdrowia ludzi. Uwzględnione być muszą przy tym wszystkie zagrożenia jakie mogą powodować odpady, rozpatrując ich stan skupienia oraz właściwości fizyczne i chemiczne. Podmiot zlecający usługę transportową ma obowiązek wskazania miejsca przeznaczenia śmieci. Transportujący po umieszczeniu indywidualnego numeru rejestrowego na dokumentach usługi dostarcza odpady do miejsca przeznaczenia przekazując je posiadaczowi. Efektywne wykorzystanie zasobów może ograniczyć a czasem nawet zapobiec powstawaniu śmieci. Środek ten należy uwzględnić już podczas planowania produkcji, wykorzystując najnowszą technologię oraz inwestując w jej rozwój. Egzekwowanie przyjętych wskaźników przez władze lokalne oraz krajowe może znacząco podnieść świadomość ekologiczną posiadaczy odpadów wpływając korzystnie na środowisko. Konsument wybierający produkty z opakowań biodegradowalnych, nadających się do recyklingu bądź ponownego użycia zapobiega powstawaniu nowych śmieci. W poprawie statystyk wykorzystania tego typu opakowań pomagają między innymi kampanie informacyjne bądź nałożenie opłat za produkty z materiałów szkodliwych dla środowiska. Przyjęte w Polsce plany gospodarki odpadami uwzględniają poniżej przedstawioną kolejność działania.

1. Analiza stanu gospodarki odpadami danego obszaru, w tym:
   - istniejących środków zapobiegawczych powstawaniu odpadów,
   - rodzajów, ilości i źródeł powstawania odpadów,
   - funkcjonujących systemów gospodarki odpadami,
   - uwzględnienie rodzajów i ilości odpadów z wyszczególnieniem metod odzysku lub unieszkodliwiania,

- ocena funkcjonowania systemu gospodarki odpadami, weryfikacja stosowności inwestycji w rozwój obecnych rozwiązań.

2. Przewidywanie zmian związanych z rozwojem gospodarczym regionu.
3. Określenie terminów realizacji założonych celów gospodarki odpadami.
4. Ustalenie harmonogramu oraz wykonawców, zapewnienie środków finansowania
5. Określenie sposobu oceny wdrażania planu [1]

Racjonalna gospodarka odpadami wymaga zwrócenia uwagi na logistykę zwrotną, czyli dziedzinę zajmującą się badaniem prawidłowości związanych z przepływami produktów, których cykl życia zakończył się [2]. Zastosowana, usprawnia przepływ odpadów w przestrzeniach miejskich. Zapewnia najmniej ingerujące w środowisko naturalne sposoby zbiórki śmieci, dążąc przy tym do minimalizacji jej kosztów. Zbudowanie wydajnego systemu zarządzania zbiórką odpadów wymaga nakładów poczynionych na infrastrukturę do zbierania, segregacji, transportu, ponownego wykorzystania oraz ich unieszkodliwiania. Rysunek 3. przedstawia łańcuch przepływu odpadów zaczynając od ich kreatora a kończąc na miejscu przeznaczenia.



Rysunek 3. Łańcuch przepływu odpadów. Źródło: [29]

Poprawnie działający system można wyróżnić po:

- sposobie gromadzenia odpadów,
- doborze pod względem technologicznym, lokalizacji i wielkości obiektów,
- skuteczności funkcjonowania obiektów,
- doborze środków transportu oraz dróg wywozu [9]

Planując transport odpadów należy wybrać odpowiednie pojazdy. Do najważniejszych parametrów, które należy uwzględnić zalicza się:

- ciężar użytkowy pojazdu,
- odległość od miejsca unieszkodliwienia lub stacji przeładunkowej,
- zastosowany system pojemników,
- topografię, zakłócenia lub ograniczenia w ruchu,
- szerokość ulic na trasach przejazdu,
- dzienny czas pracy, przerwy i przyzwyczajenia obsługi,
- liczebność załogi tych pojazdów [5].

Zbiórka odpadów wymaga doboru odpowiednich urządzeń technicznych biorących udział w realizacji konkretnych metod. Głównym ograniczeniem stosowania wybranych środków stanowią aspekty ekonomiczne, jednak oprócz nich należy mieć na uwadze:

- aspekty prawne,
- warunki techniczne w obiektach zagospodarowania odpadów,
- poziom urbanizacji,
- częstotliwość odbioru,
- bezpieczeństwo pracowników.

Stworzenie logistycznego systemu gospodarki odpadami wymaga spełnienia szeregu kryteriów ekonomicznych, społecznych i środowiskowych. Odpowiednio wcześniej przyjęty plan uwzględniać musi lokalizację punktów składowania, przetwarzania oraz przeładowania śmieci. Od wydajności pracy tych punktów zależy funkcjonowanie całego systemu. Skuteczny plan zawiera wytyczne dla kierowców, na które składają się kolejne miejsca odbioru odpadów, bądź ulice które należy odwiedzić w określonej kolejności. Istotna w planowaniu jest również optymalizacja częstotliwości odbioru śmieci. Dobry plan uwzględnia warunki ruchu drogowego, odległość pomiędzy elementami systemu oraz możliwości przewozowe pojazdów. Logistyka odzysku zyskuje na znaczeniu zarówno dla makro jak i mikroprzedsiębiorstw. Instytucje kojarzone jako proekologiczne zyskują większe zaufanie potencjalnych klientów. Korzyści ekologiczne oraz ekonomiczne pomaga osiągnąć ponowne wykorzystanie surowców bądź ich sprzedaż. Wyczerpujące się zasoby naturalne napędzają rozwój efektywnych metod wykorzystywania zasobów.

2.2. Przedstawienie istniejących rozwiązań

Niniejszy rozdział przedstawia obecnie istniejące systemy zbiórki odpadów, omawiając metody zbiórki oraz wymieniając kilka informatycznych systemów usprawniających cały proces.

2.2.1. Rodzaje systemów gospodarki odpadami

Wśród systemów gospodarki odpadami wyróżniają się oparte na modelach dynamicznym oraz statycznym. Pierwszy z nich opiera się na śledzeniu parametrów wejściowych w czasie, uwzględniając:

- ciągłe planowanie,
- możliwość lokalizacji obiektów systemu,
- śledzenie dynamiki zmian powstawania odpadów,
- dostępność terenu pod lokalizację nowych obiektów systemu.

Podejście statyczne zakłada analizę wyłącznie określonego momentu czasowego.Poprawnie stworzony projektsystemu zbiórki odpadów powinien uwzględniać ich właściwości fizykochemiczne. Dopiero na tej podstawie określić można:

- częstotliwość gromadzenia (zwózki) odpadów,
- rodzaj środków transportu wykorzystywanych do zwózki odpadów,
- rodzaj użytych pojemników i systemu zbierania odpadów,
- lokalizacje punktów przeładunkowych odpadów,
- sposoby utylizacji odpadów,
- sposoby unieszkodliwiania odpadów. [3]

2.2.2. Metody zbiórki odpadów

Metody zbiórki odpadów można kategoryzować pod względem wykorzystywanych pojemników w procesie zbiórki. Najczęściej stosowanymi rozwiązaniami są:

- zbiórka bezsystemowa (odpady wielkogarabytowe),
- metoda pojemników lub opakowań jednorazowych,
- metoda przeładunku (pojemniki niewymienne) – kontenery po opróżnieniu pozostają na miejscu (patrz rysunek 4),
- metoda pojemników wymiennych – puste kontenery ustawiane w miejsce zapełnionych (patrz rysunek 5),

- zbiórka bazująca na zasadzie hydraulicznej i pneumatycznej – w skład systemu wchodzą rury łączące zbiornik z punktami zrzutu odpadów. Odpady zasysane są do zbiornika przy użyciu podciśnienia. Istnieje również mobilna odmiana systemu, gdzie zamiast zbiornika przejściowego stosuje się pojazdy specjalnie przystosowane do próżniowej zbiórki odpadów.

Rysunek 4. Kontener niewymienny typu dzwon. Źródło: [11]

Rysunek 5. Kontenery wymienne rolkowe. Źródło: [12]

Według [2] stosowanie mniejszych objętościowo pojemników prowadzi do ograniczenia produkcji odpadów przez konsumentów. Metoda ta wymusza na użytkownikach stosowania zasad recyklingu. Konsument zaczyna lepiej wykorzystywać miejsce przeznaczone na odpady, jednocześnie więcej z nich segregując.

Transportem odpadów zajmują się śmieciarki wyposażone w podwozia samochodów ciężarowych oraz urządzenia do załadunku kontenerów. Wyróżnia się następujące rozwiązania:

- śmieciarki do usuwania odpadów z prasą zgniatającą,
- pojazdy do transportu kontenerów,
- pojazdy z urządzeniem hakowym,
- pojazdy i urządzenia do zbiórki odpadów segregowanych,
- samochody asenizacyjne – przewóz nieczystości płynnych [4]

2.3. Informatyczne systemy wspomagające zbiórkę odpadów

Wykorzystanie technologii informatycznych w gospodarce odpadami może znacząco usprawnić cały proces. Poniżej przedstawiono rozwiązania kilku producentów systemów informatycznych.

2.3.1. Ewisel

System przygotowany przez firmę GEM wykorzystywany przy ewidencji przebiegu zbiórki odpadów. Na workach lub pojemnikach umieszczany jest unikalny kod kreskowy identyfikujący podmiot od którego odbierane są odpady. Obsługa pojazdu wyposażona jest w skaner kodów kreskowych. Dane przesłane zostają następnie przez sieć GSM na serwer. Analiza otrzymanych danych odbywa się z wykorzystaniem środowiska MS SQL Server, po czym przygotowywane są czytelne dla użytkownika raporty. Na system składają się:

- EWISEL-Desktop – uruchamiany na komputerach,
- EWISEL-Mobile – działający na terminalu mobilnym,
- EWISEL-Skan – obsługa skanerów.

Zastosowanie systemu Ewisel:

- służby miejskie,
- firmy wywozowe,
- właściciele nieruchomości

Zalety:

- relatywnie tani,
- kojarzenie odpadów z ich właścicielem

Wady:

- system nie zmniejsza kosztów transportu odpadów,
- brak analizy tras pojazdów[6].

2.3.2. NaviCar

System stworzony przez firmę NaviSoft do zarządzania flotą pojazdów oraz maszyn. Aplikacja dokumentuje miejsca odbioru i zwrotu odpadów. Narzędzie zapewnia monitoring pojazdów w czasie rzeczywistym z wykorzystaniem GPS oraz GSM, w tym:

- prędkość,
- kierunek jazdy,
- ilość zużywanego paliwa,
- ilość przejechanych kilometrów.

Producent udostępnia aplikację mobilną oraz desktopową (patrz rysunek 6). Koszty przewozu oraz planowanie optymalnych tras przejazdu realizowane jest przy użyciu modułu TC eMap firmy TimoCom.

Zastosowanie systemu NaviCar:

- służby miejskie,
- firmy wywozowe,
- transport międzynarodowy,
- przewóz osób.

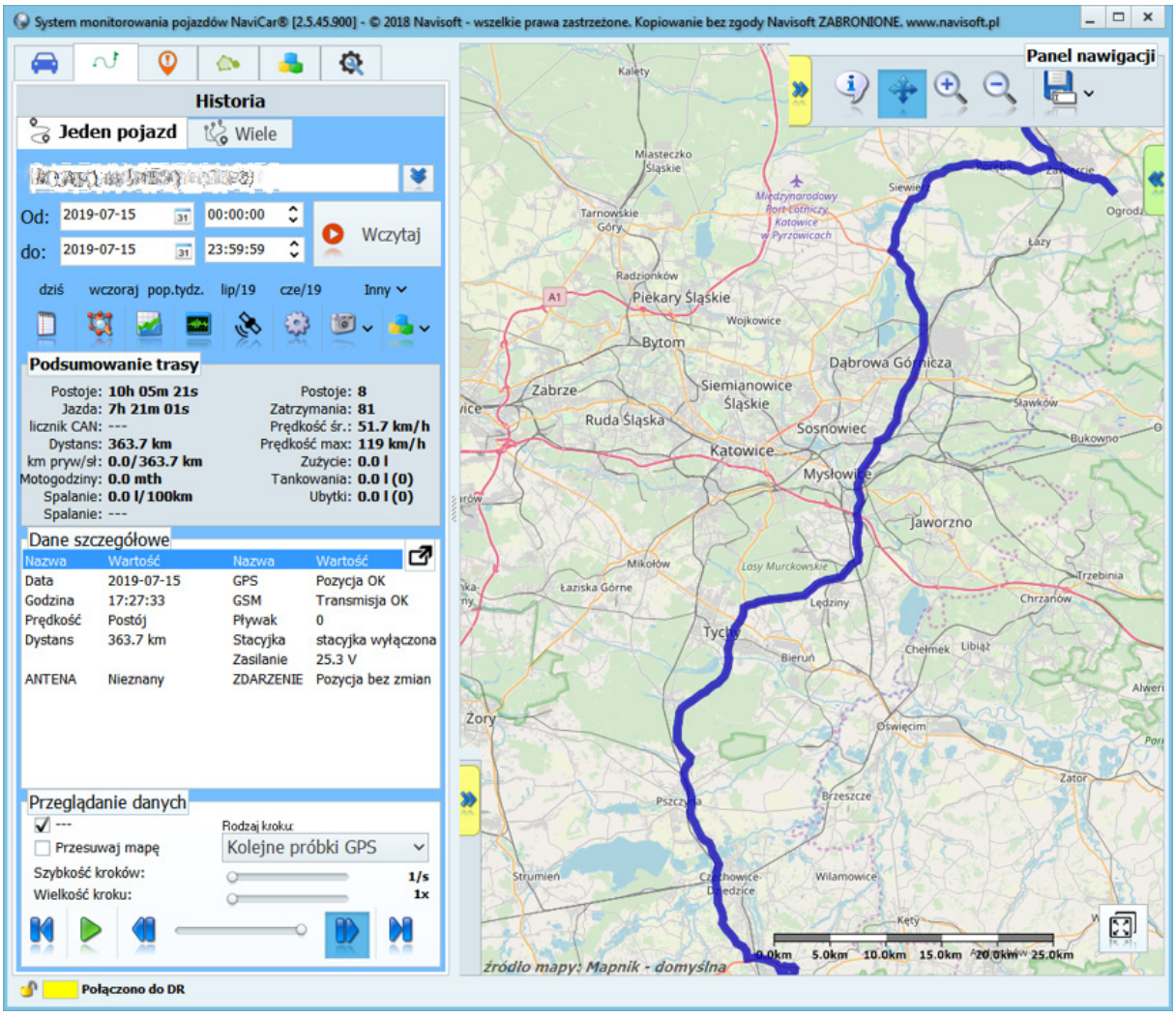
Zalety:

- nadzór nad pracą kierowców,

- zabezpieczenie pojazdów,
- optymalizacja tras, zmniejszenie kosztów prowadzenia działalności.

Wady:

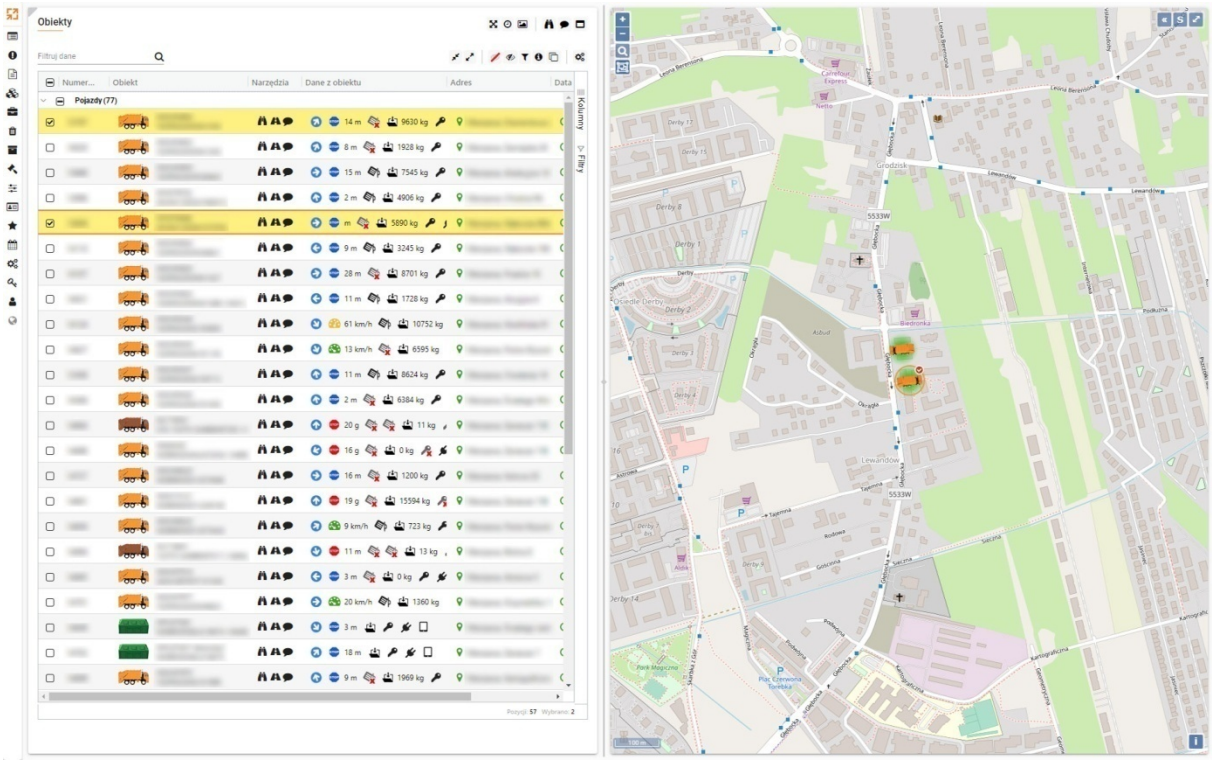
- system nie jest skierowany docelowo dla przedsiębiorstw zajmujących się zbiórką odpadów,
- brak możliwości śledzenia zebranych pojemników (brak ich identyfikacji)



Rysunek 6. Interfejs aplikacji NaviCar. Źródło: [15]

2.3.3. ELTE GPS

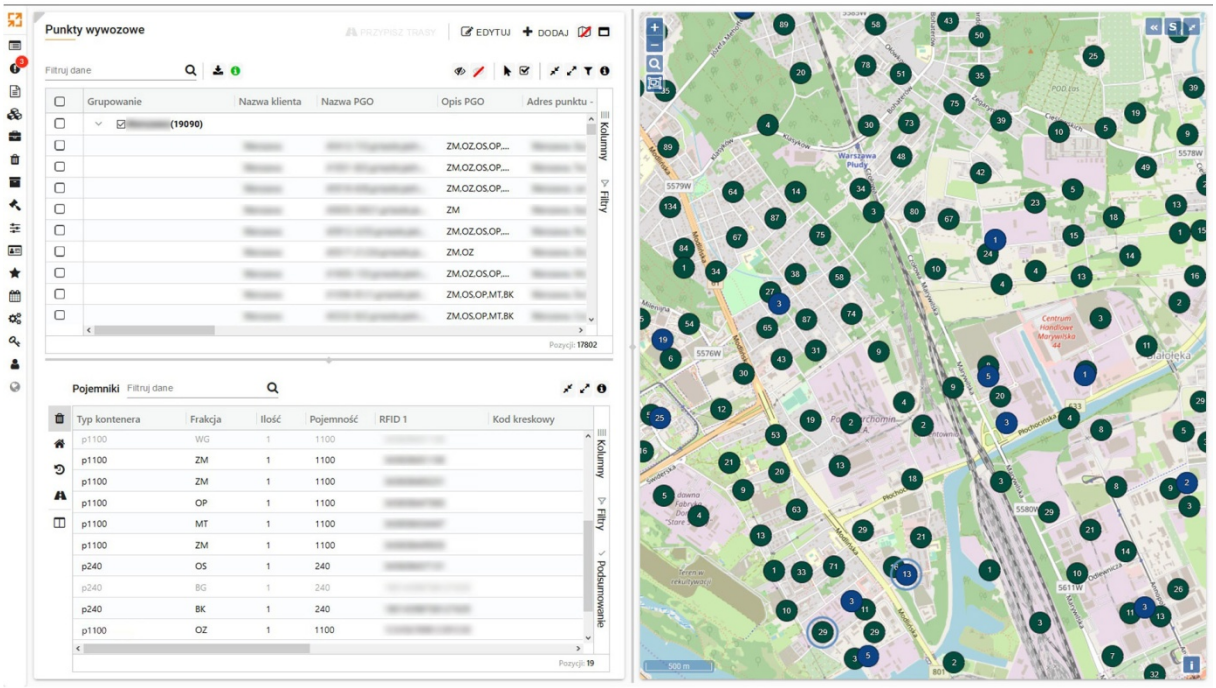
System ELTE GPS jest oparty na architekturze modułowej. Wdrożenie odbywa się mając na względzie zapotrzebowanie klienta. Wybierane są konkretne funkcjonalności systemu, które można następnie przystosować do pracy w określonym środowisku. Udostępnionym przez producenta systemem nadrzędnym jest Sepan. Klient otrzymuje intuicyjny interfejs webowy do zarządzania swoim przedsiębiorstwem. Podstawowym zadaniem aplikacji jest lokalizacja pojazdów z wykorzystaniem systemu GPS (patrz rysunek 7).



Rysunek 7. Interfejs aplikacji Sepan – wykaz pojazdów. Źródło: [14]

W oparciu o dane zapisane w pamięci lokalizatora generowane są raporty z tras przejazdu. Wdrożenie systemu w branży komunalnej można rozszerzyć o dodatkowe funkcjonalności. Dołączając system identyfikacji pojemników (patrz rysunek 8) ułatwić można zarządzanie bazą pojemników zwiększając jednocześnie efektywność wykonanej pracy. Nieprawidłowości w pracy jak opróżnienie niewłaściwego pojemnika rejestrowane są w systemie**.**Transmisja międzysystemowa odbywa się z wykorzystaniem technologii GSM/GPRS. Dostępne są poniższe konfiguracje:

- system automatycznej identyfikacji RFID – zestaw anten, czujników oraz transponderów RFID montowanych na pojazdach oraz pojemnikach. Umożliwia automatyczną rejestrację odbieranego pojemnika,
- system manulanej identyfikacji RFID – manualny czytnik RFID oraz transpondery zamontowane na pojemnikach. Możliwość zastosowania na różnego rodzaju pojemnikach,
- system identyfikacji za pomocą kodów kreskowych– manualny czytnik kodów kreskowych oraz etykiety z kodami kreskowymi umieszczane na pojemnikach, kontenerach oraz workach. Wydruk etykiet odbywa się poprzez dedykowaną aplikację.

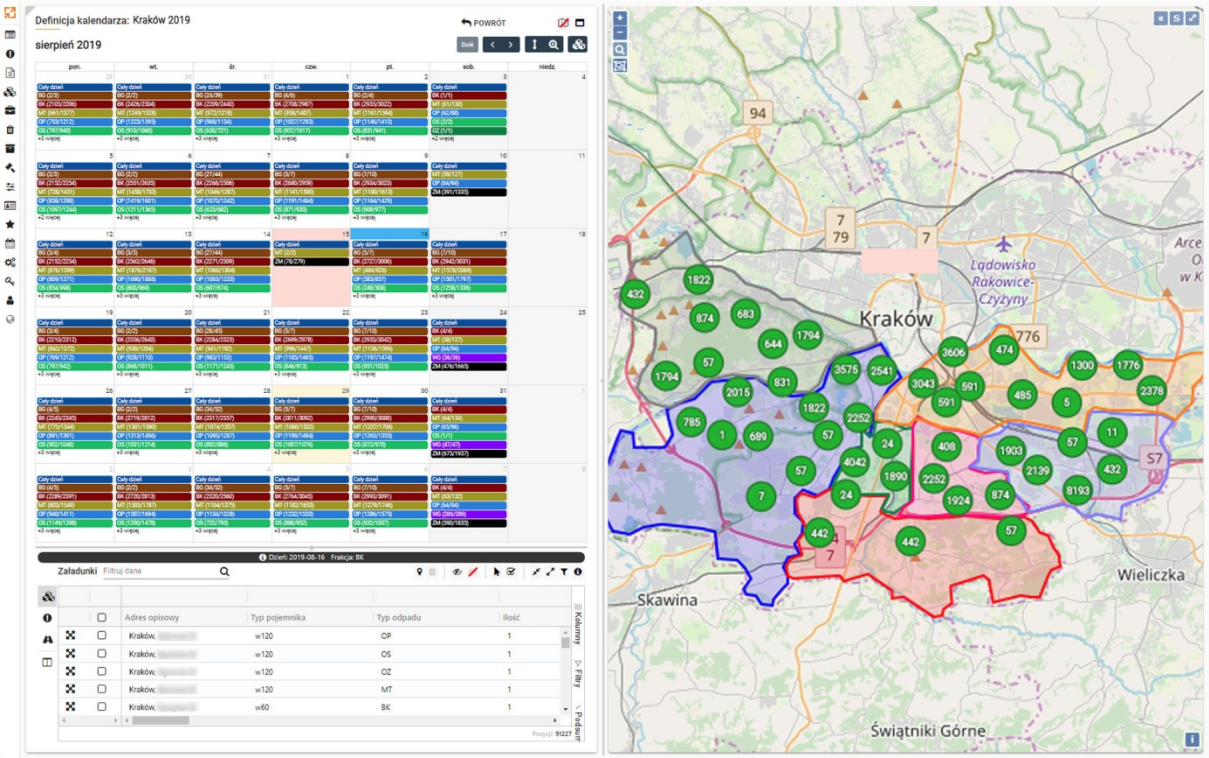


Rysunek 8. Interfejs aplikacji Sepan – wykaz pojemników. Źródło: [14]

Kolejnym modułem rozszerzającym funkcjonalność systemu jest system lokalizacji kontenerów ET Container. Wykorzystany lokalizator GPS wraz z modułami zasilającym, GSM oraz czujnikiem załadunku i wyładunku kontenera przesyła raz na dobę, oraz po obsłudze informacje do systemu zarządzania. Stosowanie systemów ważenia odpadów zapewnia możliwość automatycznego analizowania ciężaru pojemnika. Rozwiązanie usprawnia proces rozliczania mieszkańców i przedsiębiorców. Informacje przesyłane do bazy danych umożliwiają zdalny odczyt ważeń i sporządzania raportów. Producent udostępnia dwa rozwiązania:

- ET Dynamic – zautomatyzowany system nie wymagający zatrzymania urządzenia wrzutowego. Wykorzystuje komputer wagowy, akcelerometr oraz zestaw tensometrów,
- ET Static – system wymagający zatrzymania procesu opróżniania pojemnika. Działa w oparciu o komputer wagowy oraz zestaw tensometrów.

Usprawnienie procesu komunikacji z kierowcami zapewnia system ET Connect. Umożliwia diagnostykę pojazdowych elementów systemu oraz zgłaszanie usterek przez obsługę pojazdu. Zapewnia podgląd listy pojemników do odbioru na danej trasie wraz z informacją o ich specyfice. Kierowca może korzystać z nawigacji do wybranego punktu gromadzenia odpadów bez wpisywania jego adresu. Poprawę dokumentacji przebiegu procesu zbiórki odpadów może zapewnić system rejestracji obrazu ET Pics. Wykorzystując kamery, nośniki danych oraz sieć GSM umożliwia zapis poszczególnych punktów trasy w formie zdjęć bądź filmów. Dane są synchronizowane z lokalizacją na cyfrowej mapie. Wypracowane przez system materiały są pomocne w procesie weryfikacji wykonania zadań oraz zgłaszanych reklamacji. Planowanie tras oraz harmonogramów zapewnia system ET Plan (patrz rysunek 9). Danymi wejściowymi dla systemu są cykliczność odbiorów, rodzaj odpadów i ilość pojemników. W oparciu o przekazane informacje automatycznie tworzone są harmonogramy na kolejne dni.



Rysunek 9. Interfejs aplikacji Sepan – harmonogram. Źródło: [14]

Weryfikacja realizacji zaplanowanych tras oraz harmonogramów możliwa jest przy użyciu systemu ET Control.Zawiera informacje o pracy obsługi pojazdu wraz z informacją o zadaniach wykonanych poprawnie oraz tych które nie zostały zrealizowane. Zaawansowane możliwości ewidencji pojazdów i pracowników zapewnia narzędzie ET Register. Użytkownik ma wgląd w historię kosztów eksploatacyjnych oraz dostęp do terminarza, który przypomina o nadchodzących przeglądach bądź naprawach. Wykorzystanie ET CAN umożliwia kontrolę parametrów eksploatacyjnych pojazdu w czasie rzeczywistym. Warunkiem instalacji systemu jest obecność w pojeździe szyby CAN-BUS. Podstawowo system jest w stanie odczytać:

- poziom paliwa,
- stan licznika,
- ciśnienie w obwodzie hamulcowym,
- zużycie paliwa,
- aktualne obroty,
- temperaturę płynu chłodzącego,
- parametry zabudowy pojazdu[11].

Szczegółowe rozliczenie pracowników z wykonanej pracy zapewnia system ET ID.Zawiera on informacje o pojazdach, które obsługuje dana osoba oraz raporty z przebiegu trasy. System wykorzystuje technologię RFID. W celu optymalizacji tras przejazdu producent proponuje swój system ET Optimal (patrz rysunek 10) uwzględniający pojemności pojazdów i pojemników, częstotliwość ich odbioru oraz lokalizację[13]. Głównymi zadaniami narzędzia są:

- zwiększenie wydajności procesu zbiórki,
- minimalizacja czasu pracy oraz pokonywanego dystansu,
- lepsze dostosowanie pojazdów do zadań,

- obniżenie kosztów zbiórki odpadów.

Rysunek 10. Przykład działania systemu ET Optimal. Źródło: [13]

Niezwykle istotnym modułem systemu z punktu widzenia gospodarki odpadami jest ET Bins. Stworzony w oparciu o czujniki Bin Box (patrz rysunek 11) wykorzystujące fale ultradźwiękowe udostępnia informację o aktualnym stanie zapełnienia pojemników. Zebrane dane wraz z lokalizacją pojemnika przetwarza platforma analityczna. Użytkownik obserwuje zapełnienie pojemników na mapie cyfrowej będącej częścią systemu. Dodatkowymi atutami modułu jest przekazywanie informacji odnośnie:

- przewrócenia pojemnika,
- pożaru wewnątrz pojemnika,
- kradzieży odpadów.

Czujniki dopasowane są do różnego typu pojemników. Cechuje je kilkuletnia wytrzymałość baterii oraz wyposażenie w czujnik zapełnienia, temperatury oraz akcelerometr.

Rysunek 11. Urządzenie Bin Box. Źródło: [14]

Przykładowe miejsce montażu urządzenia Bin Box przedstawiają rysunki 12. oraz 13. Sposób instalacji uwzględniać musi zasadę działania fal ultradźwiękowych oraz łatwość dostępu do czujnika w celach serwisowych.

Rysunek 12. Montaż urządzenia Bin Box w pojemniku typu dzwon. Źródło: [13]

Rysunek 13. Montaż urządzenia Bin Box w kontenerze. Źródło: [13]

Zastosowanie systemu ELTE GPS:

- firmy wywozowe,
- zarząd gmin.

Zalety:

- modułowość – dostosowanie systemu do potrzeb klienta,

- szeroki wachlarz produktów.

Wady:

- koszty wdrożenia.

2.3.4. EcoBins

System EcoBins City działa w oparciu o Internet Rzeczy. Montowane w pojemnikach na odpady czujniki poziomu wypełnienia przesyłają informacje do platformy analitycznej. Transmisja wykorzystuje sieć GSM. Czujniki (patrz rysunek 14) działają w oparciu o fale ultradźwiękowe. Odczyt zapełnienia następuje o ustalonej porze bądź po przekroczeniu zadanego poziomu wypełnienia. Urządzenie jest również wyposażone w:

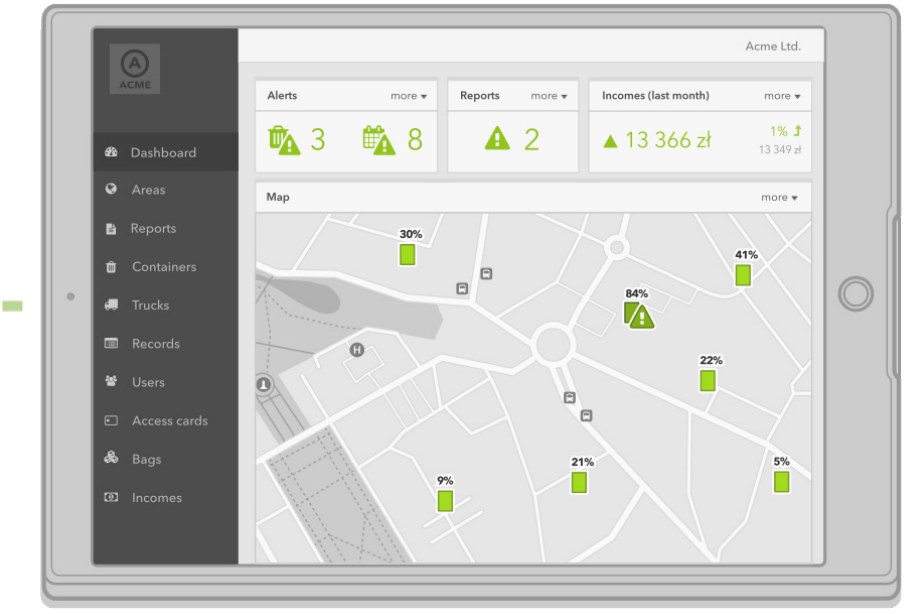
- czujnik przechylenia i wstrząsów,
- monitoring napięcia baterii,
- lokalizator GSM.

Rysunek 14. Czujnik EcoBins X2. Źródło: [8]

Dane z czujników po trafieniu na serwer są udostępnione użytkownikowi w portalu webowym oraz w aplikacji mobilnej. Mieszkańcy jako użytkownicy aplikacji mobilnej (patrz rysunek 15) mogą zgłaszać problemy z pojemnikami.

Rysunek 15. Aplikacja mobilna EcoBins City. Źródło: Opracowanie własne.

Docelowy klient dysponuje aplikacją webową (patrz rysunek 16), która optymalizuje trasy śmieciarek z uwzględnieniem rzeczywistego zapełnienia pojemników.



Rysunek 16. Aplikacja webowa EcoBins City. Źródło: [8]

Dane pomiędzy platformą analityczną a aplikacją webowąprzesyłane są za pomocą formatu JSON (patrz listing 1).

Listing 1. Obiekt JSON - dane odpowiedzi przykładowego czujnika. Źródło: [8]


```
{
        "alert": 33036,
        "dateEmptied": null,
        "dateLevel": "2019-11-09T13:39:32Z",
                "dateLevelPrev": "2019-11-09T06:56:45Z",
                "datePosition": "2017-11-28T12:17:51Z",
                "dateTemperature": "2017-11-28T12:17:51Z",
                "dateVoltage": "2019-11-09T13:39:32Z",
                "level": 3,
                "levelPerc": 100,
                "levelPercAvg": 86,
                "levelPercPrev": 100,
                "levelPrev": 3,
                "levelPercDailyAvgInc": 50.0,
                "daysToFull": 0,
                "locationId": 2358,
                "modelId": 30,
                "number": "1644",
                "position": {
                "lon": 5.272635519504547,
                "lat": 60.31712299510827,
                }
}
```


Producent udostępnia również moduły kontroli dostępu do pojemników podziemnych i półpodziemnych za pomocą kart RFID. Każda próba otwarcia pojemnika jest przesyłana na serwer i udostępniana w aplikacji klienckiej.

Zastosowanie systemu EcoBins City:

- służby miejskie,
- firmy wywozowe,
- właściciele nieruchomości.

Zalety:

- optymalizacja doboru opróżnianych pojemników,
- oszczędności z racji optymalizacji tras wywozu,
- kontrola dostępów do pojemników,
- niedopuszczanie do przepełnień pojemników,
- zmniejszenie korków w miastach,
- zmniejszenie oddziaływania na środowisko procesu zbiórki odpadów,
- możliwość integracji systemu EcoBins z funkcjonującymi systemami gospodarki odpadów

Wady:

- cena systemu,
- konieczność wymiany baterii w czujnikach,

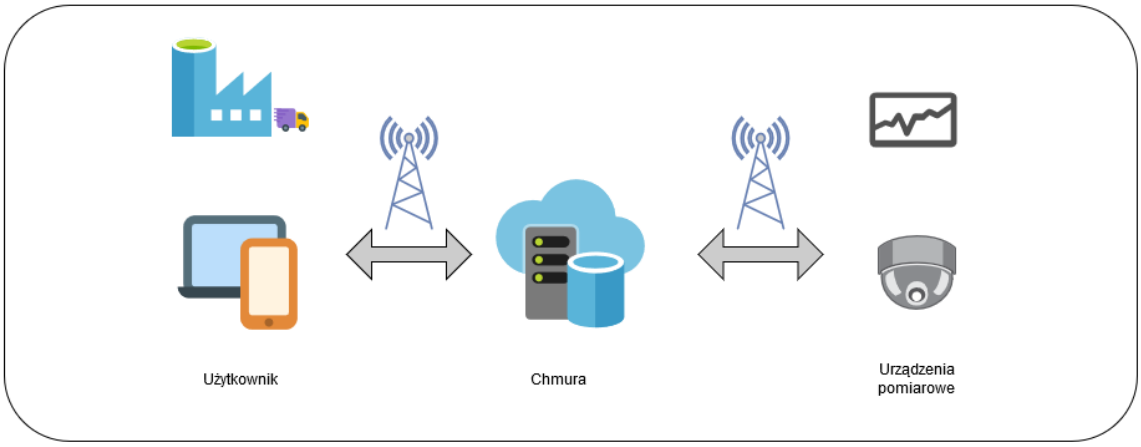
- błędy pomiarowe czujników przy niekonwencjonalnych rozmiarach odpadów.

3. Propozycja systemu zarządzania zbiórką odpadów

Niniejszy rozdział przechodzi od filarów działania Internetu Rzeczy po przedstawienie pomysłu zastosowania go w aplikacji.

3.1. Przedstawienie ogólnych założeń systemów inteligentnych miast

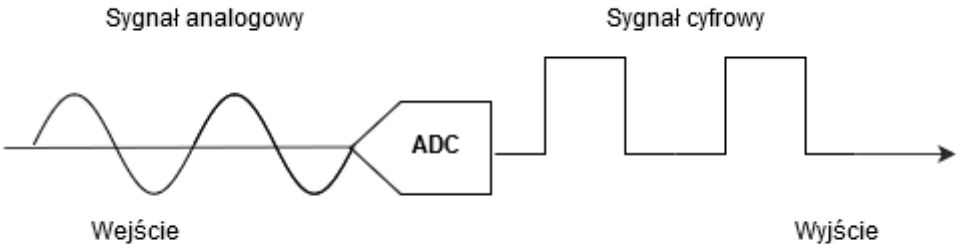
Rozwój technologii niesie za sobą rosnące zapotrzebowanie klientów na bardziej nowoczesne, ekonomiczne oraz przyjazne środowisku rozwiązania. W odpowiedzi na popyt, na rynku pojawiają się kolejne pomysły ułatwiające człowiekowi rutynowe czynności. Powstała zupełnie nowa gałąź technologii nazywana Internetem Rzeczy (*IoT – Internet of Things*). Szybko zyskując zwolenników stała się jedną z najbardziej rozwojowych w branży. Ogólnie IoT można przedstawić jako urządzenia pomiarowe podłączone do Internetu oraz do urządzeń z interfejsem użytkownika. Serwer, na którym uruchomiona jest aplikacja przetwarzająca dane potocznie nazwano chmurą. Rysunek 17 przedstawia schemat działania Internetu Rzeczy. Urządzenie pomiarowe wysyła do serwera dane, które przetworzone przez aplikację udostępnione zostają użytkownikowi końcowemu. Internet Rzeczy można podzielić na dwa obszary - użytkowników biznesowych oraz domowych. Ze względu na charakter niniejszej pracy autor zajmuje się głównie analizą sektora branży skierowanego do firm oraz zarządów miast.



Rysunek 17. Schemat IoT. Źródło: Opracowanie własne.

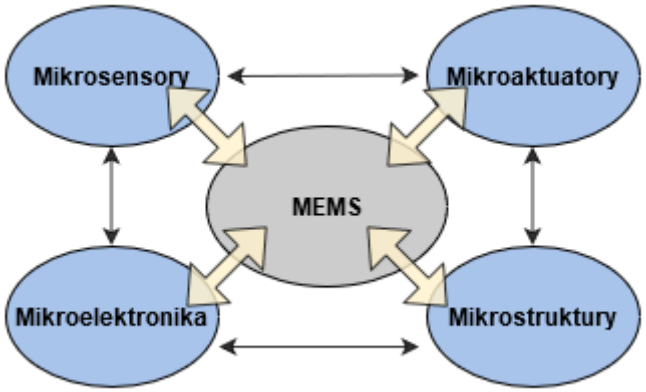
3.1.1 Elementy pomiarowe systemów IoT

Przeznaczeniem czujników w systemach Internetu Rzeczy jest pomiar specyficznych parametrów w danym środowisku. Zazwyczaj urządzenie wykorzystuje do tego sygnał analogowy, który musi następnie zamienić na postać przystępną dla mikroprocesora. Rozwiązaniem są tutaj konwertery analogowo cyfrowe ADC (patrz rysunek 18). Podany na wejściu układu sygnał analogowy przetwarzany jest na sygnał cyfrowy z uwzględnieniem wielkości wartości sygnału wejściowego. Konwertery ADC w rozwiązaniach IoT stosowane są między innymi do odczytu wyjścia napięciowego czujników.



Rysunek 18. Zasada działania konwertera analogowo cyfrowego. Źródło: Opracowanie własne.

Moduł czujnika musi dodatkowo zawierać układy MEMS będące zestawem miniaturowych elementów elektroniczno-mechanicznych. Pozwalają one na wykonywanie rzeczywistego pomiaru. Popularność układów MEMS w rozwiązaniach Internetu Rzeczy wywodzi się głównie z ich niewielkich rozmiarów oraz oszczędnej konsumpcji energii. Rysunek 19 przedstawia ogólną budowę układów MEMS. Mikrosensory po wykryciu zmiany badanego otoczenia przekazują sygnał do mikroaktuatorów uruchamiających kolejne elementy systemu.



Rysunek 19. Układy MEMS. Źródło: [31]

Kolejnym elementem są układy RF służące komunikacji. Czujniki IoT komunikują się bezprzewodowo, najczęściej poprzez fale radiowe. Urządzenia powinny być przystosowane do kilku trybów pracy. Nie zawsze jest wykorzystywana pełna ich moc. Zazwyczaj system pobiera informacje z czujników kilka razy dziennie. Pozostały czas urządzenie powinno optymalnie zarządzać energią. Czujniki należy projektować mając na względzie warunki panujące w środowisku ich pracy. Dobór odpowiednich komponentów jest zadaniem trudnym a jednocześnie bardzo ważnym. Zbyt krótki czas pracy bądź słaby zasięg systemu jest niedopuszczalny na etapie produkcji. Ze względu na umożliwienie serwisowania urządzeń w przyszłości kluczowy jest łatwy do nich dostęp.

Kluczowym etapem przygotowania rozwiązania IoT jest faza testów. Producent powinien sprawdzić wszystkie dostępne tryby pracy urządzenia. Szczególną uwagę w przypadku zasilania bateryjnego należy poświęcić kwestiom związanym z konsumpcją energii elektrycznej. Dodatkowo

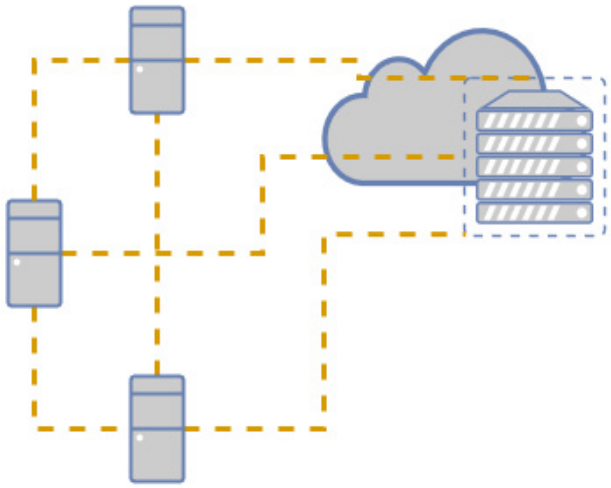
dostawca rozwiązania powinien zweryfikować zasięg komunikacji urządzenia oraz czas reakcji na określone zdarzenia.

3.1.2 Łączność i wymiana danych

Budowa sieci urządzeń IoT wymaga uwzględnienia szeregu czynników. Do podstawowych należy zaliczyć:

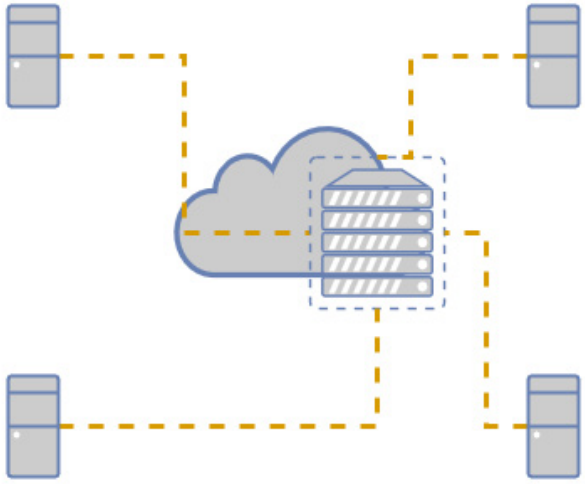
- zasięg sieci,
- zużycie energii przez urządzenie pomiarowe,
- szybkość przesyłania danych w sieci.

Wszystkie z wymienionych uwarunkowań zależą od pozostałych. Zwiększając zasięg bądź ilość przesyłanych danych w sieci należy zapewnić odpowiednio większe źródło energii. Parametry urządzenia muszą być optymalnie dobrane do specyficznych wymagań danego rozwiązania. Analizę należy przeprowadzić już na etapie projektowania, gdyż późniejsze zmiany generują nakłady finansowe. Najczęściej spotykanymi architekturami sieciowymi w projektach IoT są topologia siatki oraz topologia gwiazdy. Pierwsza z nich przedstawiona na Rysunku 20 zapewnia połączenie wszystkim urządzeniom. Połączenie realizowane jest jako każdy z każdym w obrębie zasięgu urządzeń. Rozwiązanie cechuje wysoka niezawodność, jednak jego budowa jest skomplikowana co niesie za sobą wysokie koszty. Wdrożenie topologii siatki ma sens jeśli planuje się w przyszłości rozszerzać zasięg sieci dodając kolejne urządzenia. Do zalet należy również zaliczyć ciągłość pracy systemu w przypadku awarii pojedynczych węzłów.



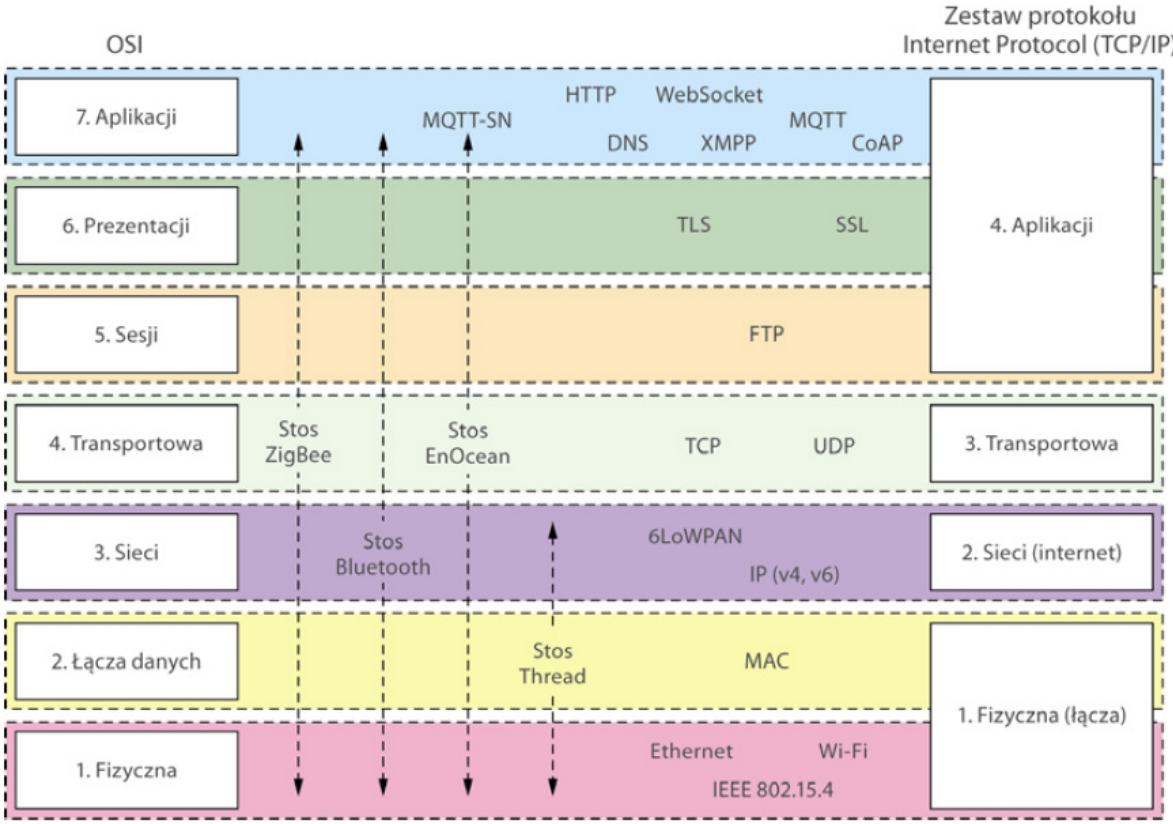
Rysunek 20. Topologia siatki. Źródło: Opracowanie własne.

Łącząc urządzenia wyłącznie do centralnego węzła sieci otrzymujemy topologię gwiazdy (patrz rysunek 21). Przykładowa konfiguracja Internetu Rzeczy łączy czujniki pomiarowe z chmurą przetwarzającą otrzymane wyniki. Rozwiązanie cechuje prostota oraz niższe koszty wdrożenia w porównaniu do struktury siatki. Zwiększa się również poziom bezpieczeństwa – atak na jedno urządzenie nie wpływa na sąsiednie. Czujniki mogą przechodzić w stan uśpienia, gdyż struktura sieci nie wymaga od nich ciągłego nasłuchiwania sąsiedniego punktu. Słabą stroną topologii gwiazdy jest ograniczony zasięg połączenia. Wybór rozwiązania zależy od wielu czynników. Wybór konkretnego może ułatwić jego dostępność w danym środowisku pracy.



Rysunek 21. Topologia gwiazdy. Źródło: Opracowanie własne.

Nawiązanie komunikacji z czujnikami pracującymi w ramach Internetu Rzeczy umożliwia wykorzystanie protokołu sieciowego TCP/IP. Połączone ze sobą protokoły z różnych warstw tworzą stosy protokołów wykorzystywane szeroko w branży IoT. Rysunek 22 przedstawia porównanie modeli OSI z TCP/IP oraz przykłady najpopularniejszych protokołów i stosów protokołów Internetu Rzeczy.



Rysunek 22. Porównanie modelu OSI z TCP/IP oraz przykłady protokołów i stosów protokołów Internetu Rzeczy. Źródło: [30]

Wybór protokołów sieciowych zależy od zasięgu działania tworzonej sieci. Ograniczone możliwości zasilania czujników wykorzystywanych przy budowie systemu uniemożliwiają stosowanie jednego standardu. Przy urządzeniach zasilanych bateryjnie należy uzyskać kompromis pomiędzy ilością przesyłanych danych oraz odległością na jakiej prowadzona jest transmisja. Tabela 1 przedstawia przykładowe protokoły wraz z ich zasięgiem.

Tabela 1. Zasięgi przestrzenne oraz odległości pomiędzy węzłami w różnych protokołach Internetu Rzeczy. Źródło: [30]

Zasięg przestrzenny	Typowy zakres	Przykłady
Zbliżeniowy (NFC)	poniżej 10 cm	NFC Forum
Sieć osobista (PAN)	1 – 50 m	Bluetooth, ZigBee, Thread, IEEE 802.4.15
Sieć lokalna (LAN)	50 m – 1 km	Wi-Fi, Ethernet
Sieć rozległa (WAN)	1 – 50 km	SigFox, LoRa, 5G, 4G, GSM

Systemy Internetu Rzeczy opisywane w niniejszej pracy skupiają się w głównej mierze na wykorzystaniu sieci rozległych (WAN). Obecnie na rynku dostępnych jest kilka rozwiązań stosowanych w branży IoT (patrz tabela 2). Producenci często wdrażają rozwiązania oparte na protokole GPRS ze względu na jego dostępność. Większe możliwości rozwoju niewątpliwie otworzy rozpowszechnienie sieci 5G dedykowanej dla Internetu Rzeczy. Ciągle testowany standard 5G w momencie pisania niniejszej pracy nie posiadał jeszcze ogólnie dostępnej specyfikacji.

Tabela 2. Porównanie najpopularniejszych protokołów WAN. Źródło: [30]

Nazwa	Zużycie baterii	Maksymalny zasięg	Dane zwrotne	Otwartość	Pokrycie
Weightless	Bardzo niskie	20+ km	Ograniczony	Średnia	Średnie
SigFox	Bardzo niskie	30+ km	Ograniczony	Niska	Średnie i wysokie
LoRa	Bardzo niskie	30+ km	Ograniczony	Średnia	Średnie
GPRS/3G/4G	Wysokie	50+ km	Tak	Wysoka	Wysokie
5G	Bardzo niskie	?	? (możliwe)	Wysoka	Brak wdrożenia

Najczęściej spotykaną obecnie architekturą aplikacji Internetu Rzeczy jest REST. Korzystając z protokołu HTTP zasoby systemu są identyfikowane przez unikalny adres URL. W świecie IoT komunikacja z urządzeniami pomiarowymi odbywa się zazwyczaj asynchronicznie. Podstawowym wymogiem jest reakcja systemu na zdarzenia. Na ogólny schemat architektury aplikacji Internetu Rzeczy składa się 7 warstw współpracujących ze sobą:

- urządzeń fizycznych,
- komunikacji,
- przetwarzania danych,
- przechowywania danych,
- dostępu do przetworzonych danych,
- aplikacji,
- zarządzania [32].

3.2. Propozycja systemu zarządzania zbiórką odpadów

Niniejsza praca skupia się na ukazaniu pomysłu zastosowania Internetu Rzeczy w celu usprawnienia procesu zbiórki odpadów. Autor chce zrealizować zadanie tworząc aplikację webową śledzącą ruch śmieciarek w oparciu o zapełnienia pojemników napływające z symulowanych czujników. Rozwiązanie nie wymaga stosowania fizycznych urządzeń pomiarowych. Implementacja systemu z wykorzystaniem rzeczywistych czujników zapełnienia umożliwi stworzenie aplikacji docelowej.

3.2.1. Menu aplikacji

Aplikacja powinna udostępniać panel do zarządzania parametrami systemu. Poprzez odpowiednie formularze użytkownik ma mieć możliwość dodawania pojemników oraz pojazdów. Panel udostępnia opcję ukrycia, nie może przesłaniać pozostałych elementów aplikacji. Dla pojemnika formularz powinien zawierać pola:

- nazwy,
- pozycji na mapie,
- typu pojemnika,
- aktualnego zapełnienia,
- zwiększania czasu obsługi.

Dla śmieciarek odpowiednio:

- nazwy,
- pozycji na mapie,
- typu pojazdu,
- pojemności,
- czasu pracy,
- dostępności.

Użytkownik ma mieć możliwość aktualizacji parametrów bądź usunięcia wybranych obiektów. Obsługa formularzy powinna być intuicyjna. Panel administracyjny musi zawierać dodatkowo przyciski do interakcji z pozostałymi komponentami aplikacji:

- odświeżania mapy,
- wyszukiwania tras,
- realizacji zbiórki odpadów,
- symulacji zapełnienia pojemników,
- przedstawienia sytuacji drogowej.

3.2.2. Mapa

System ma być zintegrowany z mapą udostępniającą widok ulic. Użytkownik powinien mieć możliwość podglądu natężenia ruchu ulicznego. Komponent ma zapewnić interakcję z użytkownikiem poprzez między innymi:

- podgląd szczegółów wybranego obiektu,
- przedstawienie dostępności pojazdów,
- wizualizację zaplanowanych tras przejazdu,
- wyodrębnienie trasy aktualnie oglądanej,
- wyświetlenie aktualnego zapełnienia pojemników,
- wizualizację wydłużonego czasu obsługi w danym punkcie,
- zapewnienie możliwości ustalania lokalizacji pojemników oraz pojazdów.

3.2.3. Tabele szczegółów tras

Aplikacja powinna udostępnić tabelaryczne przedstawienie szczegółów zaplanowanych tras przejazdu. Komponent musi być czytelny oraz łatwy w obsłudze. Użytkownik ma mieć wgląd w porównanie tras konkretnych pojazdów oraz móc wyświetlić ich szczegółowe informacje. Odnalezienie odpowiednich tras na mapie powinno być intuicyjne. Komponent powinien być zintegrowany z panelem administracyjnym oraz z mapą. W tabelach należy zawrzeć dla pojazdów:

- nazwy,
- czas rozpoczęcia pracy,
- czas trwania zbiórki,
- długości pokonanych tras,
- lokalizacje na mapie,
- pojemności pojazdów,

Szczegóły konkretnych tras powinny zawierać:

- nazwy pojemników,
- czas rozpoczęcia obsługi pojemników,
- całkowity czas obsługi pojemników,
- lokalizacje pojemników,
- zapełnienie pojemników.

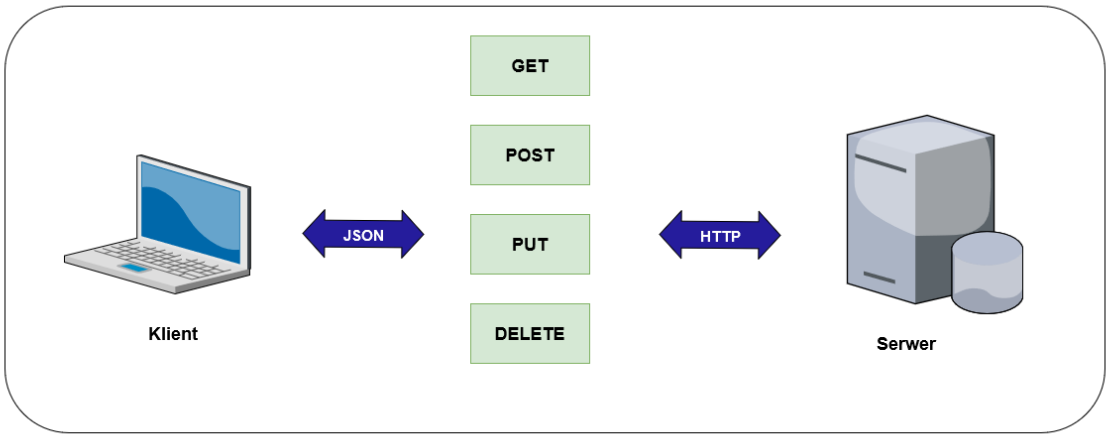
4. Opis technologii i narzędzi zastosowanych w pracy

Poniższy rozdział przedstawia technologie oraz narzędzia wykorzystane do stworzenia prototypu aplikacji.

4.1. REST API

Styl architektury REST został stworzony przez Roy'a Fielding'a w 2000r w odpowiedzi na potrzeby rozwijającego się Internetu [16]. Standard określa zasady projektowania API. Twórca oparł fundamenty jego działania na sześciu regułach:

- buforowanie danych występuje po stronie klienta,
- niezależność poszczególnych połączeń,
- jednolity interfejs,
- kod na żądanie (warunek opcjonalny),
- warstwowość systemu – warstwy współpracują ze sobą,
- odizolowanie interfejsu użytkownika od warstwy przechowywania danych. Implementacja aplikacji w architekturze klient-serwer (patrz rysunek 23).



Rysunek 23. Architektura REST. Źródło: [17]

4.1.1. JSON

JSON jest tekstowym formatem wymiany danych opartym na języku JavaScript (patrz listing 2). Popularność zyskał głównie dzięki:

- niezależności od technologii,
- przenośności,
- niezastrzeżonej licencji [18].

Nazwy właściwości JSON otoczone są cudzysłowami. Format umożliwia zagnieżdżanie. Dane kodowane są domyślnie w systemie UTF-8.

Za pomocą JSON możliwy jest zapis:

- napisów,

- liczb,
- tablic,
- wartości boolean oraz null.

Listing 2. Komunikat JSON. Źródło: Opracowanie własne

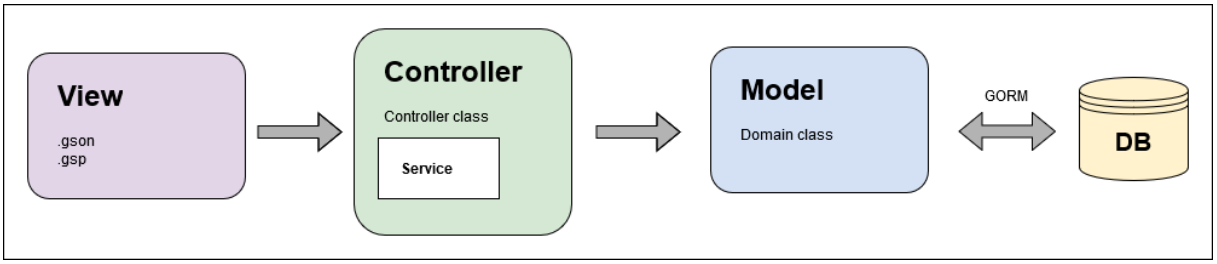
```
{
    "id": 1,
    "name": "Stefan",
    "birthday": "13-03-1994",
    "favouriteMovies": [
        "Kiler",
        "Rush"
    ]
}
```

4.1.2. RxJS

Rozwój technologii internetowej wymusza ciągłą pracę deweloperów, aby ich oprogramowanie mogło sprostać rosnącym zapotrzebowaniom. Na przestrzeni lat stworzono wiele interesujących rozwiązań usprawniających pracę programistów jak i samego oprogramowania. Do jednego z nich należy programowanie reaktywne, które skupia się na obsłudze asynchronicznych strumieni. Rozwiązanie rozszerza możliwości programowania funkcyjnego. Wprowadza pojęcia Obserwowanego rozsyłającego komunikaty oraz Obserwatora, który pobiera wybrane przez siebie wiadomości. RxJS stanowi bibliotekę JavaScript umożliwiającą programowanie reaktywne [19].

4.2. Grails

Grails to *open source'owy* framework oparty na technologiach Spring oraz Hibernate. Dzięki swojej nowoczesnej, nieskomplikowanej architekturze symplifikuje proces tworzenia aplikacji webowych. Framework wyróżnia szybkość konfiguracji projektu. Posiada własną implementację mapowania obiektowo-relacyjnego GORM [20]. Używa popularnej i sprawdzonej w świecie aplikacji webowych architektury MVC przedstawionej na rysunku 24.



Rysunek 24. Architektura MVC w Grails. Źródło: Opracowanie własne.

Framework udostępnia swój własny CLI, wykorzystanie którego odciąża programistę od wykonywania rutynowych czynności. Przykładowo stworzenie nowego kontrolera o nazwie `Book` sprowadza się do wydania polecenia `grails create-controller Book`. Grails korzysta z trzech poniżej opisanych technologii.

4.2.1. Gradle

Gradle jest narzędziem wykorzystywanym przy budowaniu projektu. Automatyzuje szereg zadań, które wykonać miałby programista. Znacznie ułatwia zarządzanie projektem oraz przyspiesza proces jego realizacji. Narzędzie udostępnione na licencji Apache jest *open source'owe*, oparte o języki z rodziny DSL – Groovy lub Kotlin. Wyróżniają je łatwa konfigurowalność, szybkość wykonywania zadań oraz duże wsparcie społeczności ze względu na jego popularność. Poza współpracą z najpopularniejszymi środowiskami programistycznymi Gradle posiada swój własny interfejs CLI, dzięki czemu możemy budować projekt z poziomu terminala. Narzędzie separuje logikę aplikacji od testów, co ułatwia wdrożenie aplikacji w środowisku produkcyjnym.

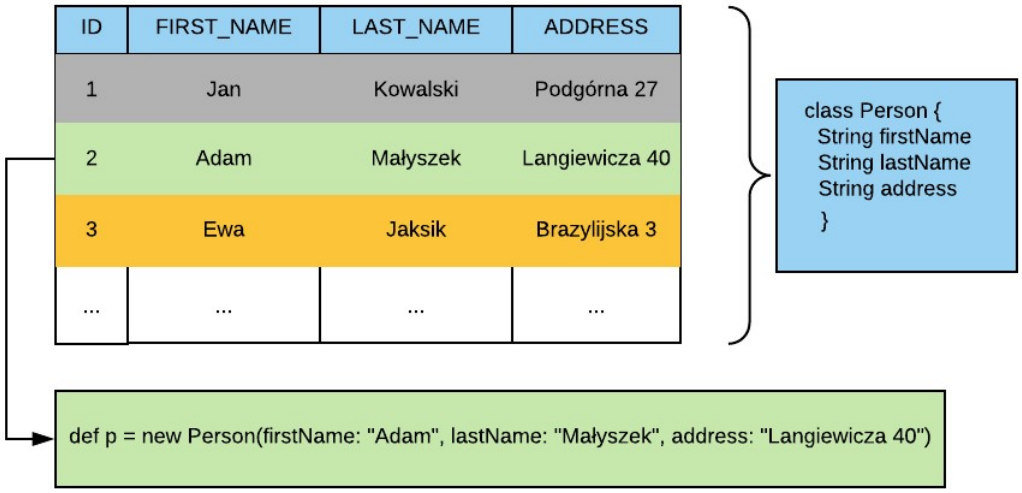
4.2.2. Groovy

Zbudowany na platformę Javy obiektowy język skryptowy. Posiada bardziej zwięzłą i czytelniejszą składnię od Javy. Zapewnia dynamiczne jak i statyczne typowanie. Wyróżnia się łatwością integracji z aplikacjami napisanymi w technologii Java. Groovy udostępnia programiście wbudowaną konsolę pomocną przy testowaniu fragmentów kodu. Atutami języka są między innymi:

- możliwość pisania skryptów,
- programowanie funkcyjne,
- domknięcia,
- wnioskowanie typów,
- łatwość tworzenia testów automatycznych,
- meta programowanie w czasie kompilacji oraz uruchomienia kodu [21].

4.2.3. Hibernate

Framework zapewniający funkcjonalność mapowania obiektowo-relacyjnego. Wykorzystanie narzędzia w pracy znacznie przyspiesza realizację zadań związanych z przetwarzaniem bazy danych. W dużym stopniu upraszcza kod aplikacji, nie wymagając od programisty pisania zapytań do bazy danych. Hibernate jest implementacją *Java Persistence API (JPA)*, może być stosowany w środowiskach wspierających *JPA*. Nie wymaga tabel ani pól w bazie danych, większość kodu SQL generuje podczas inicjalizacji systemu. Przykład mapowania obiektowo relacyjnego przedstawia rysunek 25.



Rysunek 25. Mapowanie obiektowo relacyjne. Źródło: Opracowanie własne.

4.3. Angular

Angular to platforma programistyczna do tworzenia aplikacji interfejsu użytkowników. Rozwijana przez firmę Google stała się obecnie jednym z najpopularniejszych rozwiązań.

4.3.1. Architektura komponentowa

Framework oparty jest na architekturze komponentowej. Jedną z podstawowych zalet stosowania komponentów jest możliwość ich wielokrotnego użycia. Tego typu styl programowania znacznie skraca czas tworzenia aplikacji oraz ułatwia zarządzanie projektem. Każdy komponent składa się podstawowo z szablonu oraz kontrolera. Cykl życia komponentu przedstawia rysunek 26. Platforma zbudowana jest w oparciu o wzorzec projektowy MVC, który zakłada dzielenie kodu programu na powiązane ze sobą części:

- Model (Model) – jako magazyn treści,
- Widok (View) – wyświetlający dane treści,
- Kontroler (Controller) – obsługujący zdarzenia wprowadzania danych przez użytkownika.

CLI Angulara jest zestawem narzędzi stworzonym do budowania aplikacji. Znacznie ułatwia zarządzanie projektem poprzez poniższe funkcjonalności:

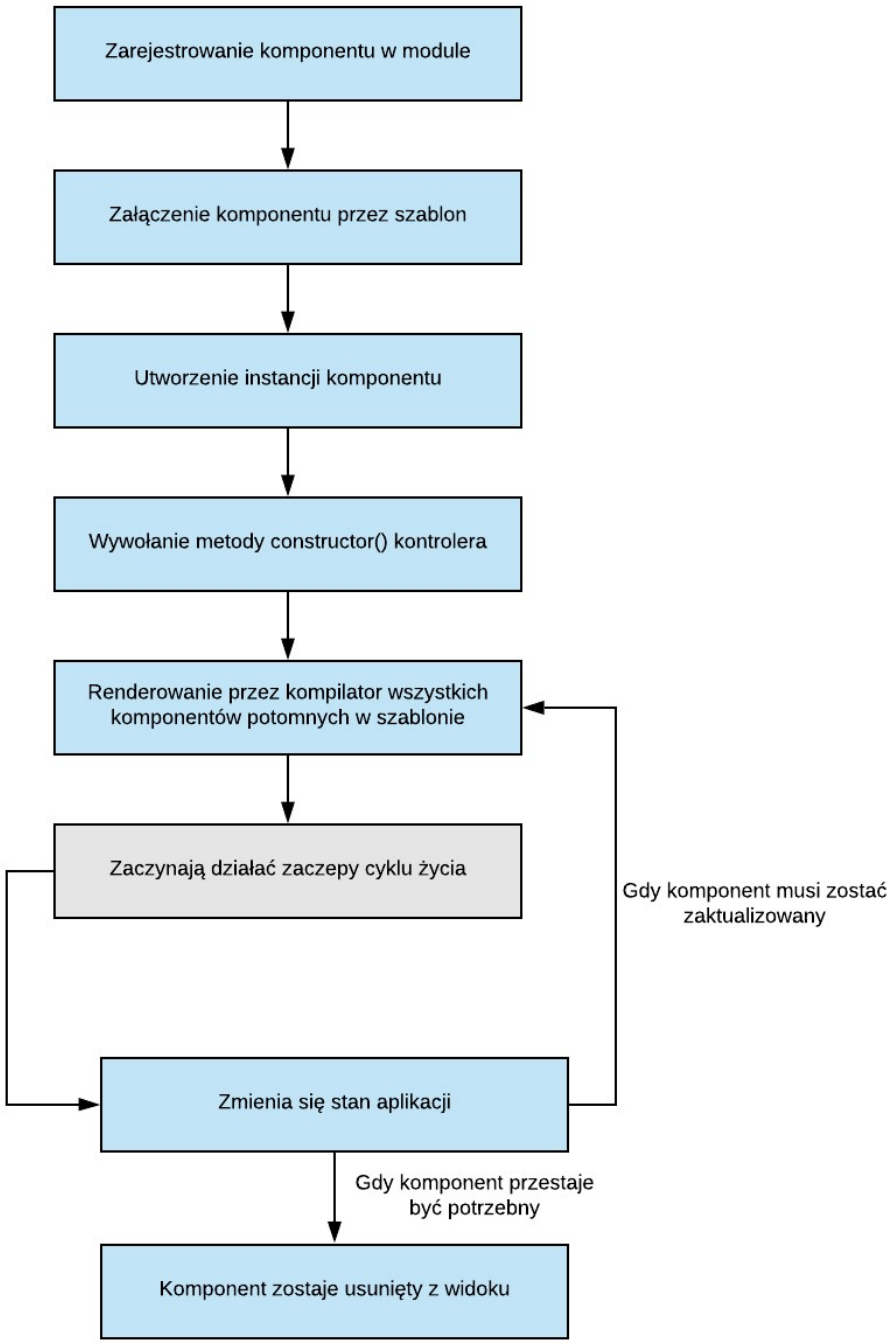
- generacja bazowego projektu,
- dodawanie nowych elementów,
- zarządzanie narzędziami do kompilacji,
- analiza i automatyczne formatowanie kodu,
- uruchamianie testów,
- obsługa serwera programistycznego lokalnego hosta.

4.3.2. Typescript

Angular został napisany w TypeScript'cie będącym rozszerzeniem języka JavaScript. Język dziedziczy wszystkie jego funkcjonalności oraz wprowadza możliwość statycznego typowania zmiennych. Jego składnia zapewnia większą klarowność kodu, wyłapuje błędy przed uruchomieniem oraz umożliwia pomijanie typowania w miejscach gdzie jest to zbędne. Kod Typescript jest kompilowany do JavaScript'u. Przewagę nad swoim przodkiem zyskuje silniej zorientowaną na klasy obiektowością oraz lepszym wsparciem w edytorach dzięki statycznemu typowaniu [22].

4.3.3. Npm

Angular działa w oparciu o środowisko Node.js, zaś do zarządzania pakietami wykorzystuje npm. Manager npm jest open source'owym narzędziem wspieranym przez wielu deweloperów. Praca z nim odbywa się poprzez interfejs CLI. Podstawowo npm zapewnia integrację z najnowszymi wersjami używanych w aplikacji modułów, sprawdzając czy ich twórcy nie wprowadzili zmian. Znacznie ułatwia ponowne wykorzystanie kodu w projektach. Bazą danych zawierającą informację o udostępnianych w npm pakietach jest rejestr, z którym deweloper komunikuje się za pomocą CLI.



Rysunek 26. Cykl życia komponentu. Źródło: [22]

4.4. PostgreSQL

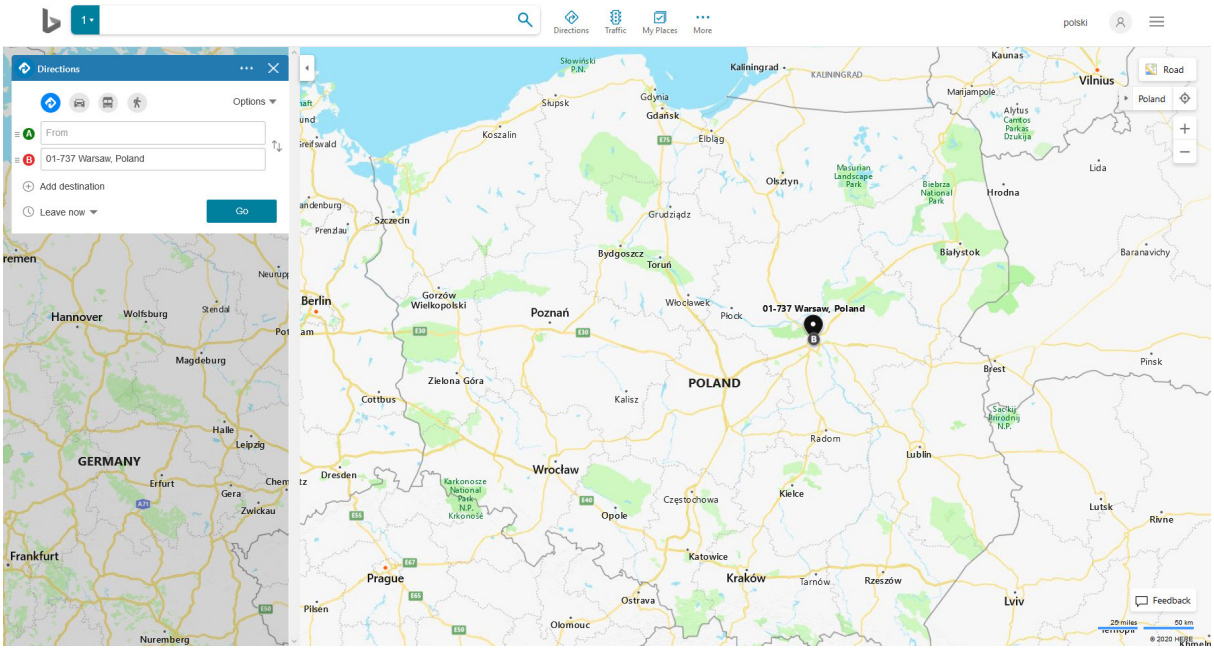
Obiektowo-relacyjny system baz danych udostępniony na licencji *open source*. Rozszerza tradycyjny język zapytań SQL o dodatki wspomagające bezpieczny zapis skomplikowanych struktur danych. Zyskał uznanie dzięki:

- sprawdzonej architekturze,

- niezawodności,
- integralności danych,
- bogatemu zestawowi funkcji,
- rozszerzalności,
- wsparciu społeczności open source,
- zapewnianiu wszystkich właściwości ACID [24].

4.5. Bing Maps

Serwis mapowy stworzony przez firmę Microsoft. Posiada interfejs webowy przedstawiony na rysunku 27. Aplikacja zintegrowana jest z wyszukiwarką internetową Bing. Użytkownik ma dostęp do map drogowych, zdjęć satelitarnych oraz widoku z lotu ptaka. Serwis umożliwia wyszukiwanie tras pomiędzy punktami oraz przedstawienie sytuacji na drogach. W celu skorzystania z oferty map Bing deweloper musi wygenerować klucz uwierzytelniający poprzez stronę internetową producenta.



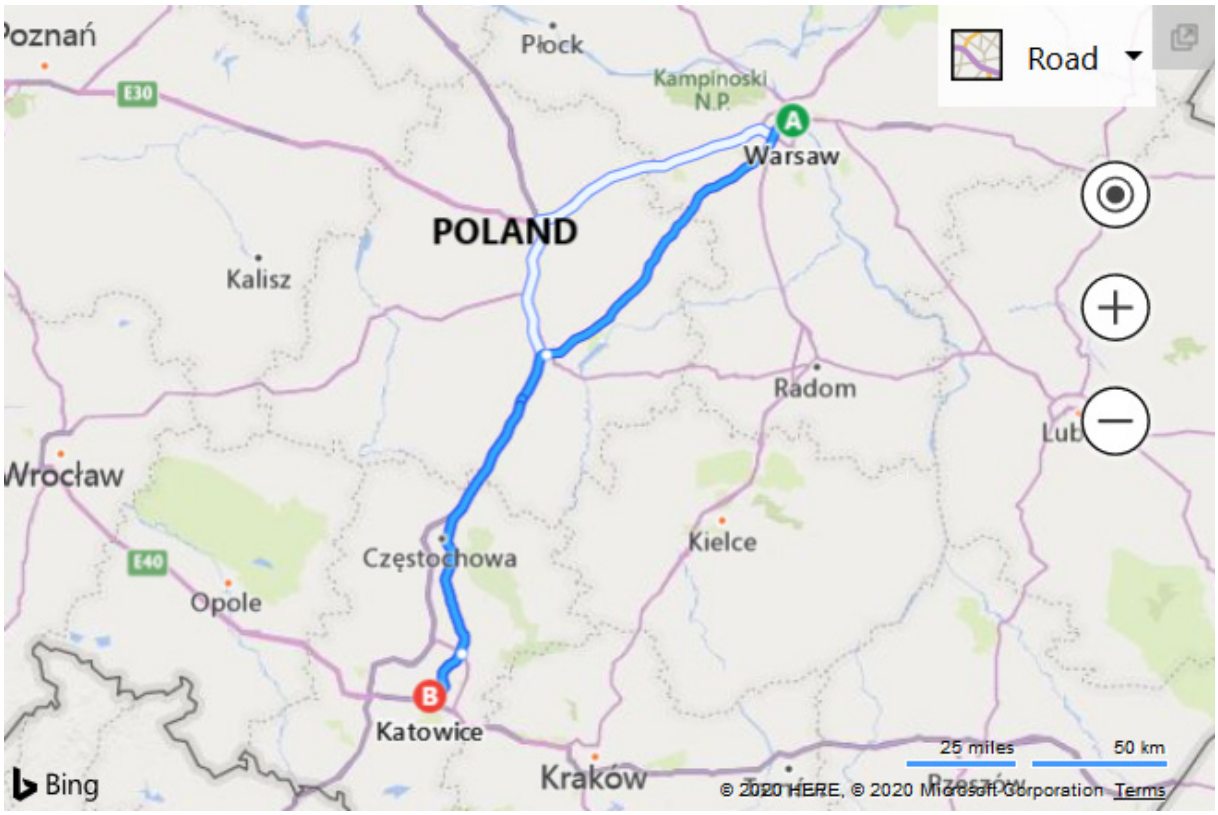
Rysunek 27. Aplikacja Bing Maps. Źródło: [25]

Wdrażanie funkcjonalności systemu ułatwia udostępnione programistom interaktywne webowe środowisko Bing Maps z możliwością kompilowania kodu JavaScript, HTML oraz TypeScript. Listing 3 oraz rysunek 28 przedstawiają test funkcjonalności znajdowania trasy pomiędzy dwoma miastami.

Listing 3. Kod Typescript tworzący trasę pomiędzy Warszawą a Katowicami. Źródło: Opracowanie własne.

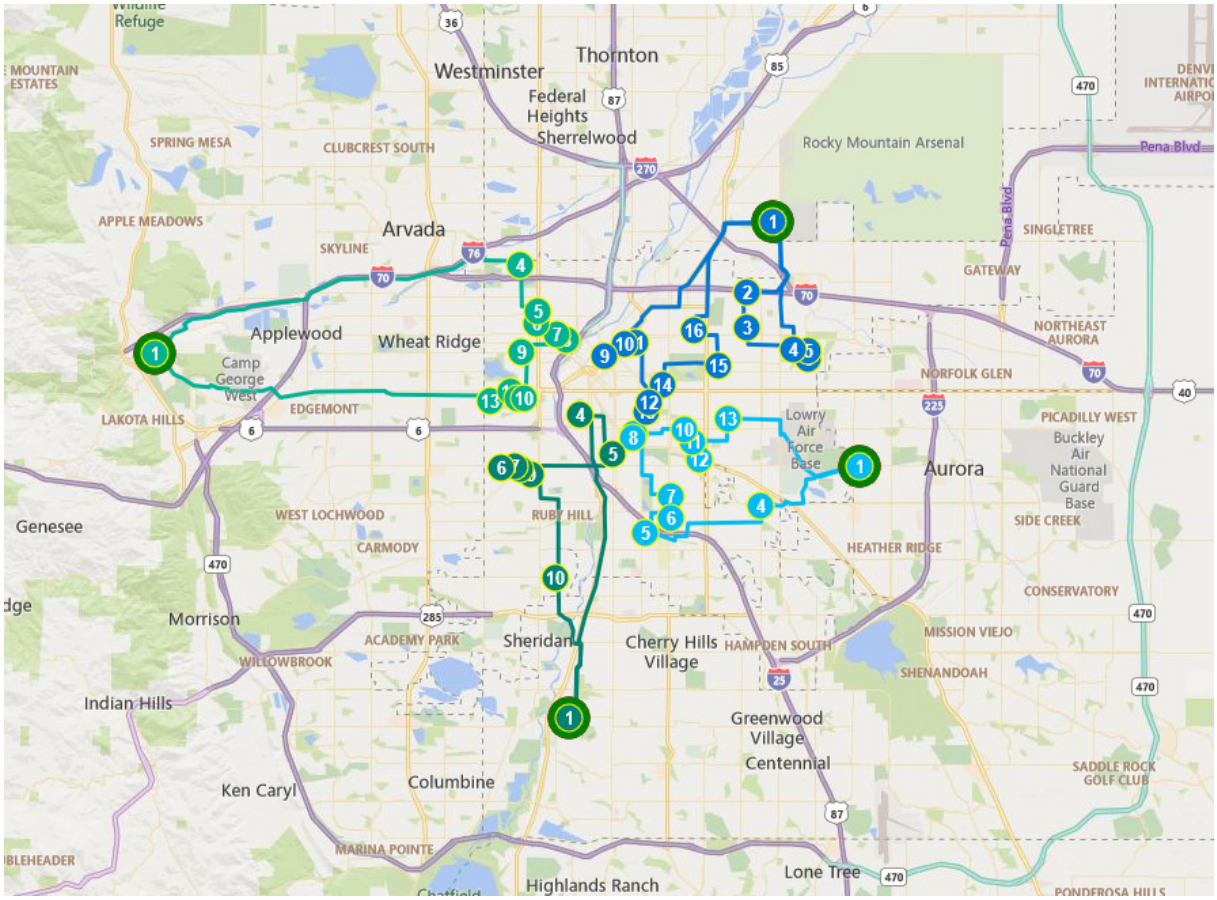
```
var map = new Microsoft.Maps.Map(
document.getElementById('myMap'),
    {
```

```
        /* No need to set credentials if already passed in URL */
        center:new Microsoft.Maps.Location(52.22977, 21.01178),
        zoom:12
    }
);

Microsoft.Maps.loadModule('Microsoft.Maps.Directions', () => {
var directionsManager =new
Microsoft.Maps.Directions.DirectionsManager(map);
    // Set Route Mode to driving
    directionsManager.setRequestOptions({ routeMode:
Microsoft.Maps.Directions.RouteMode.driving });
var waypoint1 =new Microsoft.Maps.Directions.Waypoint({ address:'Warsaw',
location:new Microsoft.Maps.Location(52.22977, 21.01178) });
var waypoint2 =new Microsoft.Maps.Directions.Waypoint({ address:'Katowice',
location:new Microsoft.Maps.Location(50.270908, 19.039993) });
    directionsManager.addWaypoint(waypoint1);
    directionsManager.addWaypoint(waypoint2);
    // Set the element in which the itinerary will be rendered
    directionsManager.setRenderOptions({
itineraryContainer:document.getElementById('printoutPanel') });
    directionsManager.calculateDirections();
});
```



Rysunek 28. Bing Maps SDK – uruchomienie kodu z listingu 3. Źródło: [25]

Microsoft umożliwia wykorzystanie map w celach komercyjnych udostępniając zróżnicowane, dopasowane do klienta API. Niniejsza praca wymagała zastosowania rozwiązania z oferty produktów do zarządzania flotą pojazdów. Stworzenie prototypu aplikacji umożliwiło wykorzystanie *Multi-Itinerary Optimization API*. Rozwiązanie stworzone jako serwis REST automatyzuje proces planowania zoptymalizowanych tras dla wielu pojazdów. Użytkownik może uwzględniać aktualną sytuację na drogach oraz otrzymywać w oparciu o nią najkrótszą, bądź najszybszą trasę. API umożliwia określenia priorytetu obsługi zadanych punktów na mapie. Funkcjonalność jest niezwykle pomocna w sprecyzowaniu konkretnej godziny odbioru bądź dostarczenia towaru w dane miejsce. Kolejnym ważnym parametrem wejściowym jest czas obsługi danej lokalizacji. Rola użytkownika sprowadza się do określenia liczby kierowców, ich lokalizacji oraz czasu pracy. W darmowej wersji użytkownik dysponuje trzema pojazdami oraz dwudziestoma punktami do obsługi na jedno wysyłane zapytanie. System automatycznie przydziela im odpowiednie punkty do obsłużenia w oparciu o wszystkie parametry wejściowe. Komunikacja z API możliwa jest przy użyciu zapytań HTTP - GET oraz POST. API akceptuje zapytania synchroniczne oraz asynchroniczne. Otrzymane odpowiedzi są zależnie od żądania w formacie JSON bądź XML. Podstawowo odpowiedź zawiera listę kierowców bądź pojazdów oraz instrukcje realizacji trasy. Rysunek 29 przedstawia przykład wykorzystania MIO API dla czterech kierowców oraz czterdziestu punktów do obsługi.



Rysunek 29. Bing Maps MIO API. Źródło: [26]

4.6. Angular Material

Popularność frameworka Angular poskutkowała rozwojem wielu bibliotek interfejsu użytkownika. Dostępność oraz łatwość zastosowania gotowych komponentów zniechęca zespoły

deweloperskie do wdrażania własnych. Do stworzenia prototypu aplikacji autor pracy zastosował popularną bibliotekę Angular Material. Wydaną przez firmę Google na licencji *open source* bibliotekę bardzo łatwo zintegrować z samym *framework'iem*. Pozwala na importowanie określonych elementów biblioteki zamiast całego pakietu [22]. Producent deklaruje kompatybilność z wszystkimi nowoczesnymi przeglądarkami internetowymi.

4.7. Moment.js

Podczas pracy z datą i czasem częstym problemem jest odpowiedni ich format. Podstawowe biblioteki programistyczne mogą okazać się niewystarczające aby sprostać specyficznym wymaganiom formatowania. Podczas tworzenia prototypu pracy problem ten rozwiązał Moment.js. Biblioteka oferuje bogaty zestaw funkcji parsowania i formatowania danych ułatwiających pracę z kodem. Produkt wydany na licencji MIT jest *open source'owy*.

4.8. Git

System kontroli wersji opracowany w 2005 roku przez Linusa Torvaldsa [23]. Obecnie stał się najpopularniejszym narzędziem wersjonowania kodu. Cechują go prosta składnia, szybkość, rozproszona architektura, wysoka wydajność zarówno w dużych jak i w małych projektach. Obiekty identyfikuje przy pomocy funkcji kryptograficznej SHA1. Historię zmian śledzi w oparciu o migawki wykonywane po każdym poleceniu `commit`. Jeżeli dany plik był modyfikowany przed wykonaniem migawki Git tworzy migawkę kolejnej wersji pliku. Oprócz zdalnego system udostępnia lokalne repozytorium dzięki czemu nie wymaga od użytkownika połączenia z siecią Internet. Narzędzie cechuje się prostotą wdrożenia nawet dla osoby niemającej doświadczenia w użytkowaniu systemu. Wpisanie w terminalu polecenia `git` wyprowadzi możliwe do wydania za jego pomocą najpopularniejsze komendy.

4.9. IntelliJ IDEA

Środowisko programistyczne stworzone przez firmę JetBrains. Podstawowo dedykowane dla Javy jednak wspiera również wiele innych technologii. Poprzez zbiór wbudowanych narzędzi jak refaktoryzacja kodu znacznie podnosi wydajność programisty. Umożliwia integrację z systemami kontroli wersji oraz zapewnia graficzną reprezentację historii zmian.

4.10. GitLab

Webowy menedżer repozytoriów współpracujący z systemem kontroli wersji Git. Udostępnia intuicyjny interfejs użytkownika znacznie ułatwiający wspólną pracę programistów oraz śledzenie dokonywanych zmian w projekcie.

4.11. pgAdmin

Platforma do wykonywania zapytań SQL oraz zarządzania bazą danych PostgreSQL. Udostępnia intuicyjny interfejs użytkownika dzięki czemu zwiększa efektywność pracy nad bazą danych. Stanowi zamiennik konsolowego programu Psql.

4.12. Apache Tomcat

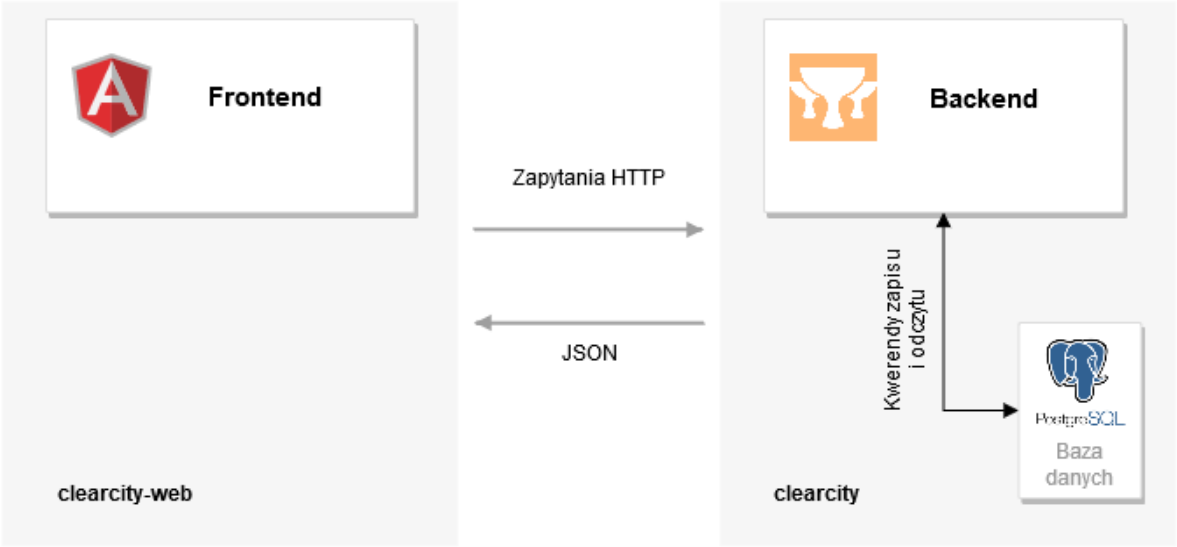
Wydany w 1998 roku przez firmę Apache serwer dedykowany aplikacjom webowym opartym na technologiach Java. Rozwijany przez lata, *open source'owy* projekt stał się jednym z najpopularniejszych rozwiązań dostępnych na rynku. Wdrożenie aplikacji z użyciem Tomcat'a sprowadza się do umieszczenia skompilowanego projektu w postaci pliku `war` w katalogu `webapps` serwera. Uruchomiony serwer samodzielnie rozpakowuje i udostępnia aplikację.

5. Projekt i implementacja

Rozdział przedstawia opis prototypu aplikacji proponowanego przez autora pracy. Po przejściu przez szczegóły implementacyjne zostaje przedstawiony projekt wdrożenia systemu na przykładzie wybranego miasta.

5.1. Aplikacja web wyznaczająca trasy pojazdom zbierającym odpady

Prototyp aplikacji opiera się na architekturze REST. Wymiana danych odbywa się przy pomocy formatu JSON. Autor stworzył dwie oddzielne aplikacje komunikujące się ze sobą w architekturze klient – serwer. Ogólna architektura systemu przedstawiona została na rysunku 30.



Rysunek 30. Architektura prototypu aplikacji. Źródło: Opracowanie własne.

5.1.1. Clearcity

Oprogramowanie działające po stronie serwera napisane zostało przy użyciu Frameworka Grails. Aplikacja łączy się z bazą danych PostgreSql i zapisuje w niej dane przychodzące z aplikacji klienckiej. Komunikację zapewniają zapytania HTTP zdefiniowane w klasie *UrlMappings*. Listing 4 przedstawia mapowania dla dodawanych w aplikacji pojazdów. Metody odwołują się do konkretnego kontrolera oraz akcji. W poniższym przykładzie przedstawione metody `get` oraz `post` służą do pobrania oraz zapisu listy pojazdów do bazy danych. Zastosowane `delete` oraz `put` odpowiednio usuwają oraz aktualizują wybrany poprzez identyfikator `id` pojazd.

Listing 4. Zapytania HTTP w pliku UrlMappings.groovy. Źródło: Opracowanie własne.

```
get "/trucks"(controller: "truck", action: "search")
post "/trucks"(controller: "truck", action: "save")
delete "/trucks/${id}"(controller: "truck", action: "delete")
put "/trucks/${id}"(controller: "truck", action: "update")
```

Obsługą zapytań w aplikacji zajmuje się kontroler. Akcje deklarowane są w postaci metod posiadających unikalne URI. Widoczna na listingu 4 akcja `search` odwołuje się do metody z kontrolera `Truck` (patrz listing 5). Mechanizm zapewnia pobranie listy pojazdów uwzględniając maksymalnie 100 obiektów na pojedyncze zapytanie.

Listing 5. Deklaracja akcji search w kontrolerze Truck. Źródło: Opracowanie własne.

```
def search(Integer max ) {
        respond Truck.list(max: Math.min( max ?: 10, 100))
}
```

Rolę modelu z wzorca MVC w Grails zapewnia klasa domenowa (*Domain Class*). Obiekty obecne w klasie są mapowane na tabele w bazie danych. Przedstawiona w listingu 6 klasa `Truck` odwołuje się do tabeli `truck` w bazie danych. Tworzenie nazw tabel na podstawie nazw klas wymaga od programisty stosowania unikalnych określeń dla klas domenowych bądź zastosowanie mapowania nazw tabel.

Listing 6. Reprezentacja tabeli truck w klasie domenowej. Źródło: Opracowanie własne.

```
class Truck {
    String name
    Double latitude
    Double longitude
    Integer dailyWorkLimit
    TruckType truckSizeType
    Integer workingTime
    Integer maxCapacity
    Boolean isActive
}
```

Pomocne przy tworzeniu aplikacji okazało się zastosowanie obiektów typu `enum`. Umożliwiły one mapowanie statycznych wartości pól w bazie danych w oparciu o wprowadzone przez użytkownika dane. Listing 7 prezentuje implementację `enum` w celu definiowania pojemności określonego typu pojazdu. Wywołanie w kontrolerze metody `getMaxCapacity()` dla instancji obiektu wiąże wartość `maxCapacity` w tabeli `truck` z typem śmieciarki. Wybranie przez użytkownika pojazdu typu `MINI` automatycznie zapisuje w bazie danych również jego maksymalną pojemność $10m^3$.

Listing 7. Typ enum. Źródło: Opracowanie własne.

```
package enums

enum TruckType {
    MINI(10), MEDIUM(16), XXL(28)

    TruckType(Integer maxCapacity) {
        this.maxCapacity = maxCapacity
    }
```

```
    private final Integer maxCapacity
    Integer getMaxCapacity() {
        maxCapacity
    }
}
```

5.1.2. Clearcity-web

Aplikacja interfejsu użytkownika powstała przy użyciu *frameworka* Angular. Stworzone zostały odpowiednie modele reprezentujące obiekty w systemie. Z pomocą platformy Angular nieskomplikowane okazało się wdrożenie wzorca MVC. Stworzenie prototypu wymagało zastosowania serwisu mapowego. Autor pracy zdecydował się wykorzystać rozwiązanie firmy Microsoft – Bing Maps. Wymiana danych z aplikacją Grails odbywa się asynchronicznie przy wykorzystaniu zapytań HTTP. Widoczny na listingu 8 kod używa biblioteki RxJS do pobrania listy pojazdów z REST API. Zapytanie odwołuje się do modelu `Truck` oraz adresu URL aplikacji clearcity. Po otrzymaniu odpowiedzi z serwera dane zostają zapisane w przytoczonym modelu.

Listing 8. Zapytanie HTTP skierowane do REST API. Źródło: Opracowanie własne.

```
public getAllTrucks(): Observable<Truck[]> {
    return this.http.get<Truck[]>(this.urlAddress).pipe(
      map(data => data.map( dataVal => new Truck().deserialize(dataVal)))
    );
}
```

Prototyp aplikacji udostępnia funkcjonalność wymiany danych pomiędzy komponentami mapy oraz formularzami menu. Dwukrotne kliknięcie lewym przyciskiem myszy na dodany pojemnik bądź pojazd umożliwia zmianę jego parametrów w dedykowanym formularzu. W celu dodania nowego obiektu na mapie użytkownik może skorzystać z pobrania lokalizacji poprzez kliknięcie prawym przyciskiem myszy w docelowym punkcie mapy. Komunikacja została zaimplementowana z wykorzystaniem zdarzeń klasy `Events` z Bing Maps. Wystąpienie określonego zdarzenia powoduje wysłanie poprzez serwis informacji do komponentu formularza.

Optymalizacją tras pojazdów zajmuje się dostarczone przez producenta serwisu mapowego MIO API. Funkcjonalność skupia się na wysyłaniu zapytań asynchronicznych do API Microsoftu i pobieraniu odpowiedzi po zakończeniu obliczeń przez API. Listing 9 przedstawia treść zapytania POST w formacie JSON. Wysyłane dane opisują kierowcę oraz punkty odbioru. API pobiera listę kierowców bądź pojazdów z informacją o ich nazwach, pojemnościach oraz listach zmian w formie daty i miejsca rozpoczęcia oraz zakończenia pracy. Na liście punktów obsługi (`itineraryItems`) znajdują się:

- nazwy obiektów,
- czasy otwarcia oraz zamknięcia punktu,
- czasy trwania obsługi `(dwellTime),`
- parametry pierwszeństwa obsługi poszczególnych punktów (`priority`),
- ilości ładunków do odebrania (`quantity`),
- parametry umożliwiające odwiedzenie lokalizacji więcej niż raz (`depot`),
- lokalizację obiektów w postaci współrzędnych GPS (`location`),
- listy obiektów wymagających odwiedzenia przed obsługą danego punktu (`dropOffFrom`).

Poza parametrami przedstawionymi w listingu 9 API umożliwia określenie formy realizacji tras. Przekazując parametr `type` użytkownik decyduje o uwzględnianiu sytuacji drogowej podczas trasowania, zaś modyfikując atrybut `costvalue` nakazuje wyszukiwanie najkrótszych relacji bez względu na czas ich trwania.

Listing 9. Treść zapytania POST wysyłanego do MIO API. Źródło: [27].

```
{
    "agents": [
        {
            "name": "agentName",
            "shifts": [
                {
                    "startTime": "...",
                    "startLocation": {
                        "latitude": ...,
                        "longitude": ...
                    },
                    "endTime": "...",
                    "endLocation": {
                        "latitude": ...,
                        "longitude": ...
                    }
                }
            ],
            "capacity": [...]
        }
    ],
    "itineraryItems": [
        {
            "name": "locationName",
            "openingTime": "...",
            "closingTime": "...",
            "dwellTime": "...",
            "priority": ...,
            "quantity" :[...],
            "depot": ...,
            "location": {
                "latitude": ...,
                "longitude": ...
            },
            "dropOffFrom": [
                "..."
            ]
        }
    ]
}
```

API odpowiada zoptymalizowanym zestawieniem tras dla pojazdów w formacie JSON. Przykład zapytania wraz z wizualizacją odpowiedzi został zamieszczony w Dodatku B. Widoczne w nim pole `agentItineraries` zawiera listę tras z uwzględnieniem czasu i miejsca rozpoczęcia oraz zakończenia podróży wraz z listą instrukcji dla kierowców. Zoptymalizowana trasa przedstawia listę kolejnych obiektów wraz z pokonaną odległością oraz czasem przyjazdu oraz zakończenia obsługi. Ze względu na obszerność listingi zostały zredukowane do postaci zawierającej najważniejsze elementy.

Przedstawieniem uzyskanych tras na dołączonej do aplikacji mapie zajmuje się klasa `DirectionsManager` będąca częścią pakietu Bing Maps. Implementacja metody `setRenderOptions()` umożliwia manipulację prezencją rysowanych tras. Przydzielenie każdej trasie unikatowego koloru pozwoliło na stworzenie intuicyjnego interfejsu użytkownika. Poprawie czytelności sprzyja również pogrubienie szerokości linii aktualnie oglądanej trasy.

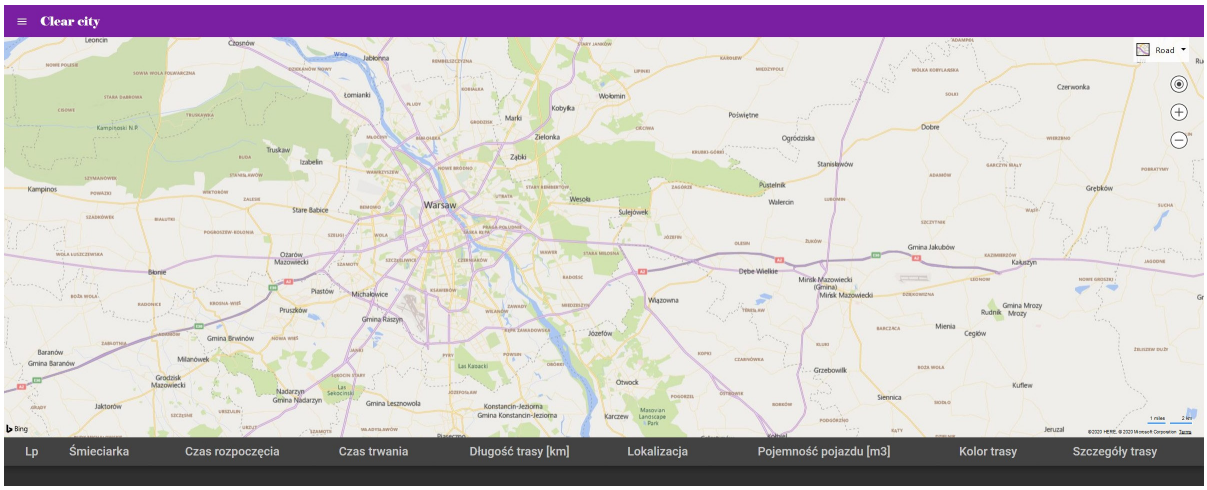
Ze względu na nałożone przez dostawcę API ograniczenia prototyp umożliwia wykonywanie trasowania dla trzech pojazdów oraz dwudziestu czujników zapełnienia. Wykupienie licencji MIO API umożliwi rozszerzenie funkcjonalności systemu do 50-ciu śmieciarek oraz 500 punktów obsługi.

5.2. Implementacja systemu na przykładzie wybranego miasta

Prezentację funkcjonowania prototypu aplikacji autor pracy zdecydował się przeprowadzić na przykładzie wybranych dzielnic miasta Warszawa. Symulacja zakłada zbiórkę odpadów segregowanych w trzech frakcjach jednocześnie. Koncepcja ma za zadanie przybliżyć działanie prototypu do warunków świata rzeczywistego.

5.2.1. Podstawowy widok aplikacji

Uruchomienie aplikacji w przeglądarce skutkuje pojawieniem się widoku przedstawionego na rysunku 31. Pierwotnie użytkownikowi prezentowana jest czysta mapa, bez dodanych obiektów. Uruchomienie aplikacji po raz kolejny gdy dodano już pojemniki wyświetli na mapie te których zapełnienie przekracza 60% pojemności.

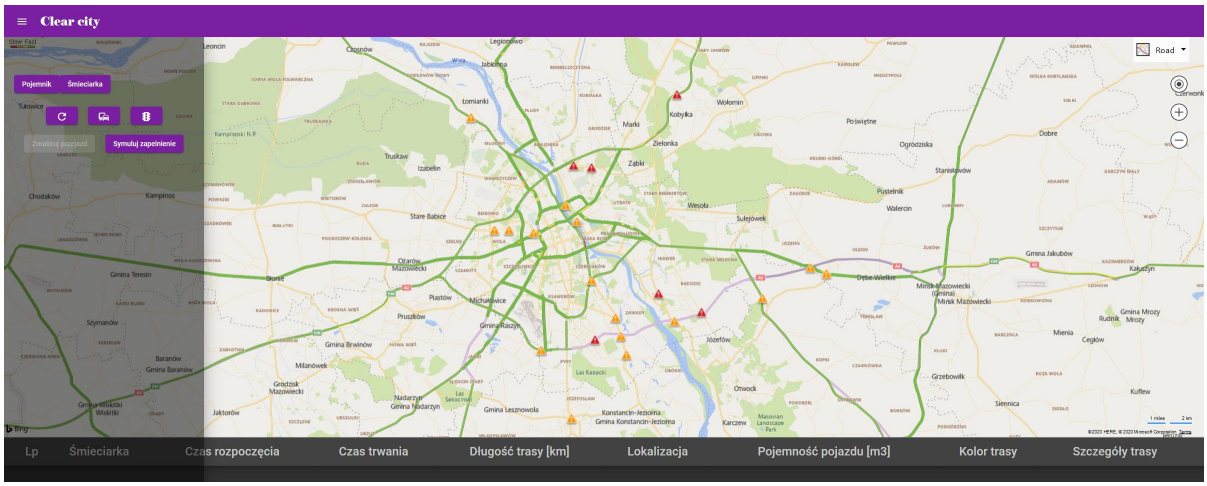


Rysunek 31. Widok aplikacji. Źródło: Opracowanie własne.

Na rysunku 31 widoczny jest nagłówek tabeli dla wyznaczanych relacji przejazdów. Będzie on wykorzystany przy interpretacji szczegółów zaplanowanych tras. Kliknięcie ikony po lewej stronie od nazwy aplikacji otwiera panel zarządzania aplikacją (patrz rysunek 32). Na rysunku przedstawiona

została również warstwa mapy z aktualną sytuacją drogową. Wraz z panelem zarządzania użytkownik otrzymuje przyciski służące kolejno do:

- wyświetlenia formularza dla pojemnika,
- wyświetlenia formularza dla pojazdu,
- odświeżenia mapy,
- wyszukania tras zbiórki odpadów,
- wyświetlenia aktualnej sytuacji drogowej,
- realizacji przejazdu dla wybranej trasy,
- symulacji zapełnienia dostępnych pojemników.



Rysunek 32. Panel zarządzania aplikacją. Źródło: Opracowanie własne.

Dodanie pojemnika wymaga wypełnienia formularza widocznego na rysunku 33. Użytkownik dostarcza:

- nazwę obiektu,
- szerokość geograficzną,
- długość geograficzną,
- typ pojemnika,
- zapełnienie,
- czas opóźnienia obsługi.

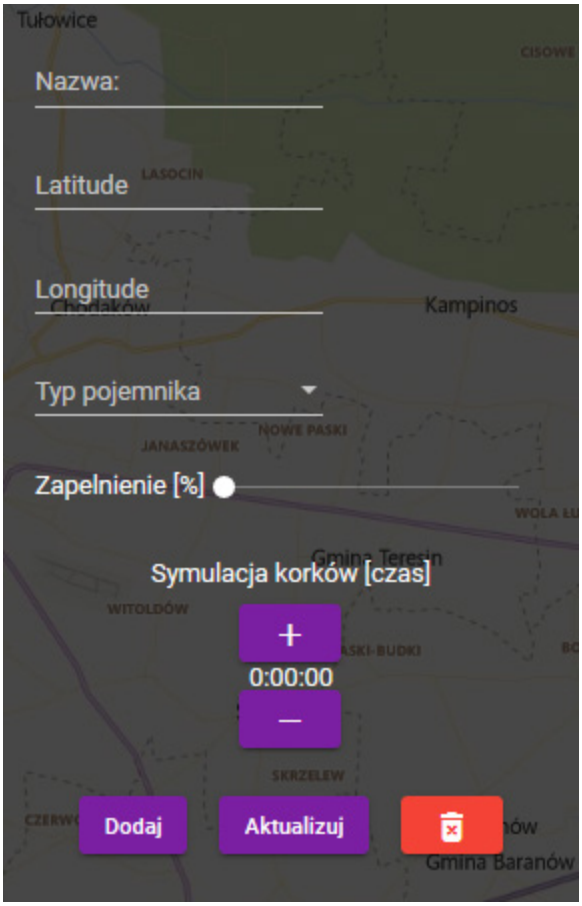
Współrzędne obiektu użytkownik może wprowadzić poprzez kliknięcie prawym przyciskiem myszy w wybranym miejscu mapy. Opóźnienie obsługi pojemnika zostało zaimplementowane w celu symulacji korków w danym punkcie. Opcjonalne zwiększenie parametru spowoduje zwiększenie czasu pobytu załogi w danym obiekcie. Na mapie pojemniki przybiorą odpowiednio kolory:

- zielony – brak opóźnień,
- żółty – opóźnienie do 20 minut,
- czerwony – powyżej 30 minut opóźnienia.

Wprowadzenie zmian do systemu umożliwiają dostępne przyciski. Modyfikacje są automatycznie wizualizowane przez komponent mapy. Użytkownik ma możliwość kolejno:

- dodania nowego pojemnika,

- aktualizacji istniejącego obiektu,
- usunięcia wybranego pojemnika.



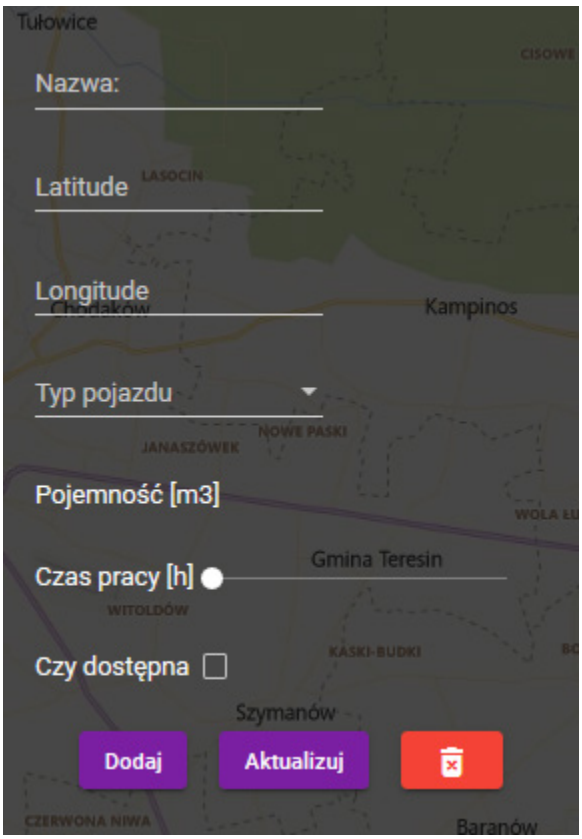
Rysunek 33. Formularz dodawania pojemnika. Źródło: Opracowanie własne.

Aplikacja wymaga również zdefiniowania pojazdów dla których wykonywane będzie trasowanie. W tym celu użytkownik wypełnia formularz przedstawiony na rysunku 34. W celu poprawnego dodania śmieciarki należy dostarczyć:

- nazwę pojazdu,
- szerokość geograficzną,
- długość geograficzną,
- typ pojazdu,
- pojemność skrzyni załadunkowej,
- maksymalny czas pracy,
- aktualną dostępność.

Pole do określenia pojemności pojazdu zostało powiązane z typem pojazdu. Wybranie wartości typu pojazdu skutkuje pojawieniem się suwaka do określenia pojemności. Udostępnione pole `Czy dostępna` typu *checkbox* definiuje aktywację danej śmieciarki w procesie zbiórki odpadów. Niezaznaczone pole nie uwzględni danego pojazdu podczas wyszukiwania tras. Dostępność pojazdów jest odpowiednio wizualizowana na mapie poprzez zaciemnienie śmieciarek aktualnie nieaktywnych.

Zatwierdzenie zmian następuje przy pomocy przycisków działających analogicznie jak dla formularza pojemnika.

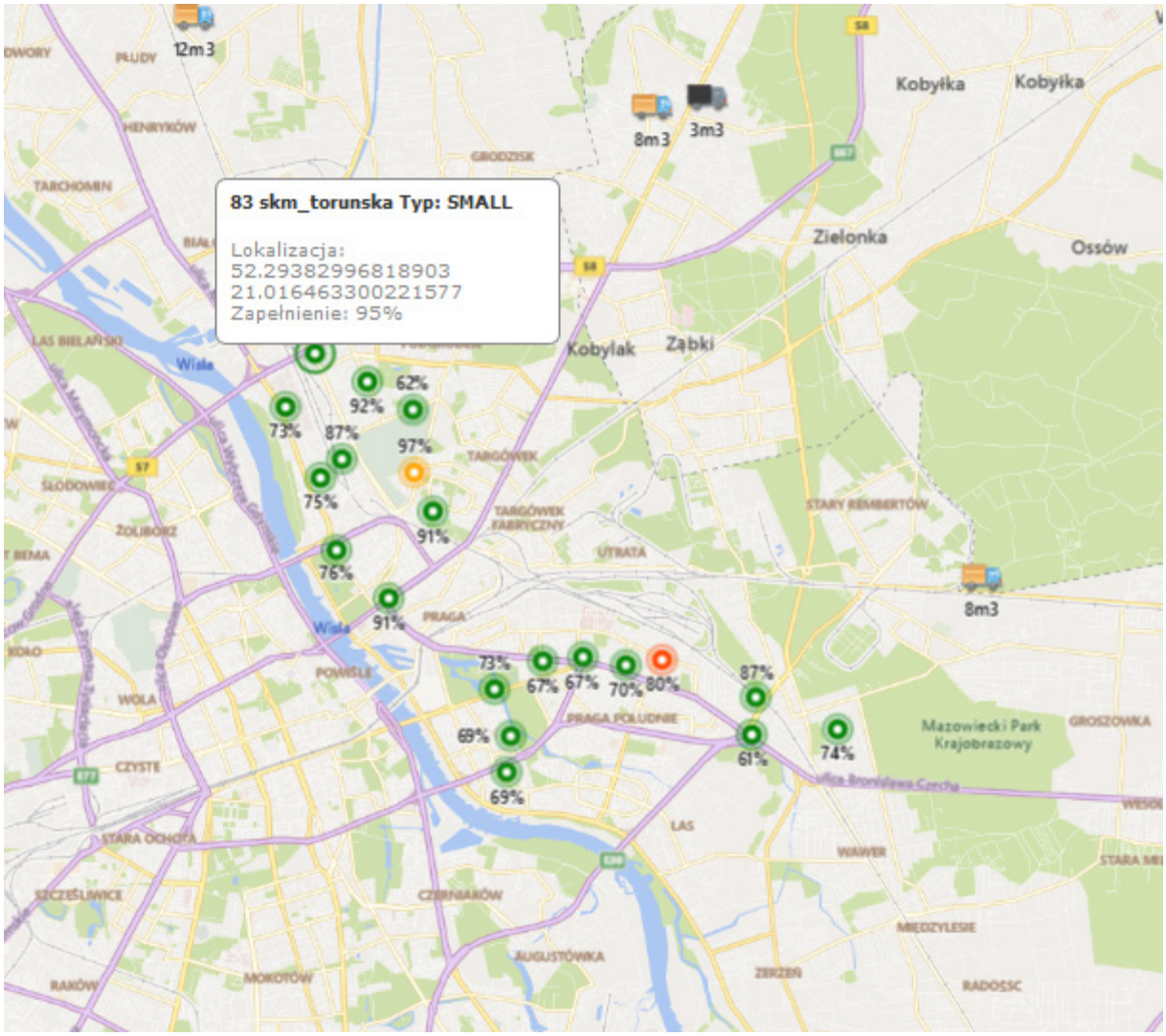


Rysunek 34. Formularz dodawania pojazdu. Źródło: Opracowanie własne.

5.2.2. Wyszukiwanie tras

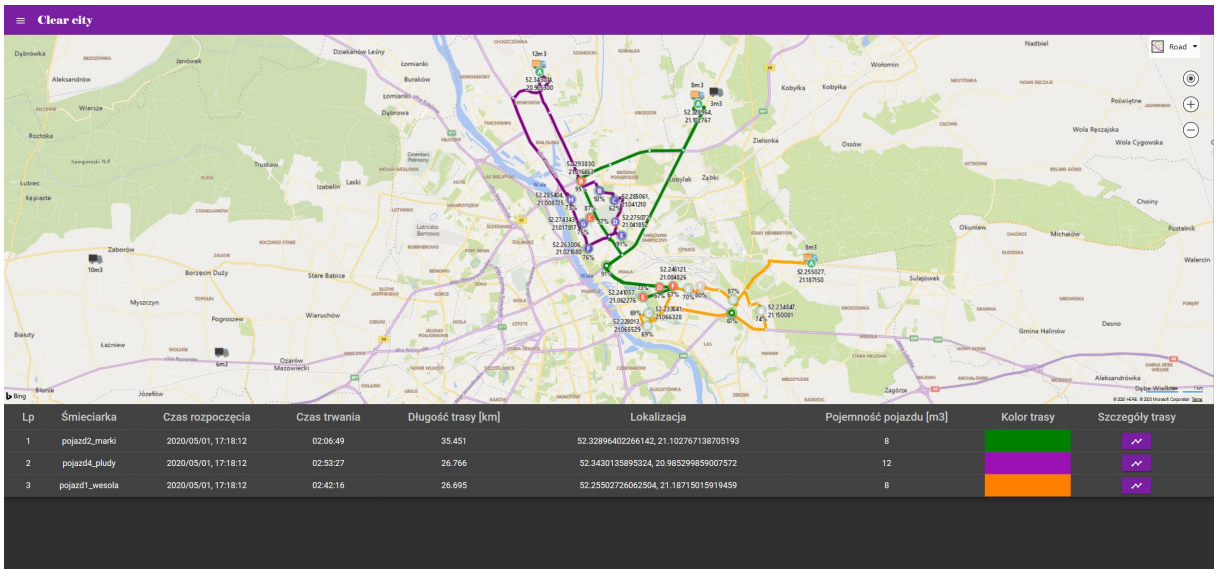
Autor pracy przygotował przykładowe rozmieszczenie pojemników oraz pojazdów w celu prezentacji funkcjonowania prototypu aplikacji. Proces zbiórki odpadów zasymulowany został dla dwóch dzielnic Warszawy. Ograniczenia systemu uniemożliwiły dokładnie odtworzyć warunki świata rzeczywistego jednak otrzymane wyniki pozwalają na optymalizację parametrów wejściowych planowanych w przyszłości procesów.

Widok mapy przedstawiający rozmieszczenie pojemników oraz pojazdów dla których zostały wykonane symulacje obrazuje rysunek 35. Po najechaniu wskaźnikiem myszy na dodany obiekt pojawia się etykieta z podstawowymi informacjami o pojemniku bądź pojeździe. Na ilustracji można zauważyć wyróżnione kolorystycznie pojemniki z ustawionymi opóźnieniami obsługi oraz nieaktywny pojazd. Opisy widoczne przy obiektach na mapie informują o aktualnym zapełnieniu procentowym dla pojemników oraz pojemności w m^3 dla pojazdów.

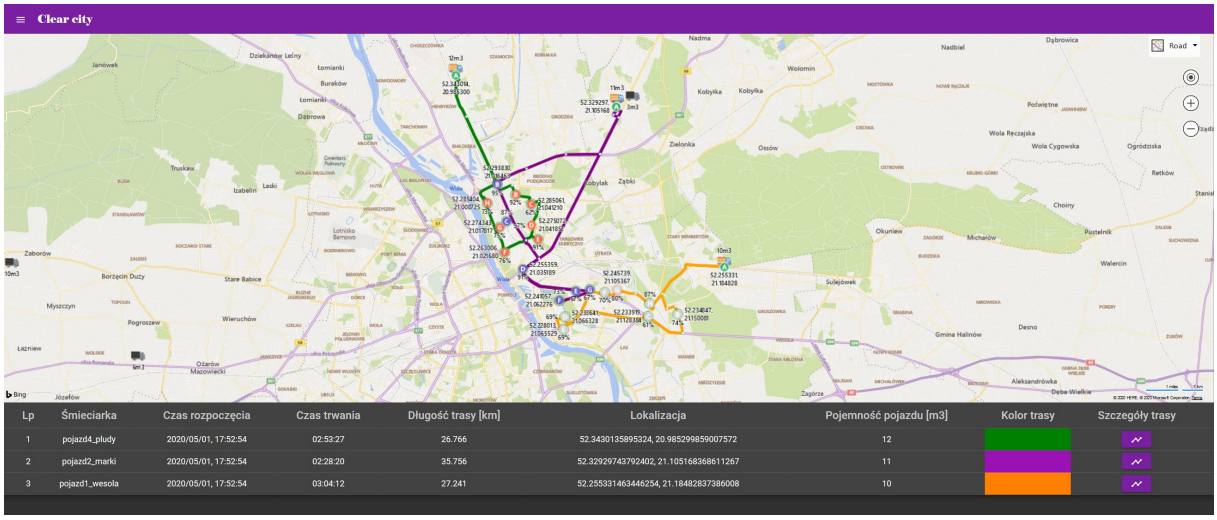


Rysunek 35. Mapa aplikacji z wstawionymi obiektami. Źródło: Opracowanie własne.

Dysponując zapełnionymi pojemnikami oraz aktywnymi pojazdami można zasymulować proces wyszukiwania tras zbiórki odpadów. Rysunek 36 przedstawia zoptymalizowane trasy dla dostępnych śmieciarek. Widoczne są ciągle nieobsłużone pojemniki. Taki stan ma miejsce ze względu na ograniczone pojemności pojazdów. W celu uniknięcia kolejnej iteracji zbiórki można ustalić większe rozmiary śmieciarek. Manipulacja parametrami aplikacji może pomóc zobrazować źródła wąskich gardeł procesu zbiórki odpadów. Rysunek 37 obrazuje działanie aplikacji po zwiększeniu pojemności pojazdów. Tym razem wszystkie pojemniki zostały uwzględnione w wyjściowych trasach. Wysłanie pojazdów z większymi skrzyniami załadunkowymi wyeliminowało konieczność ponownego ich delegowania.

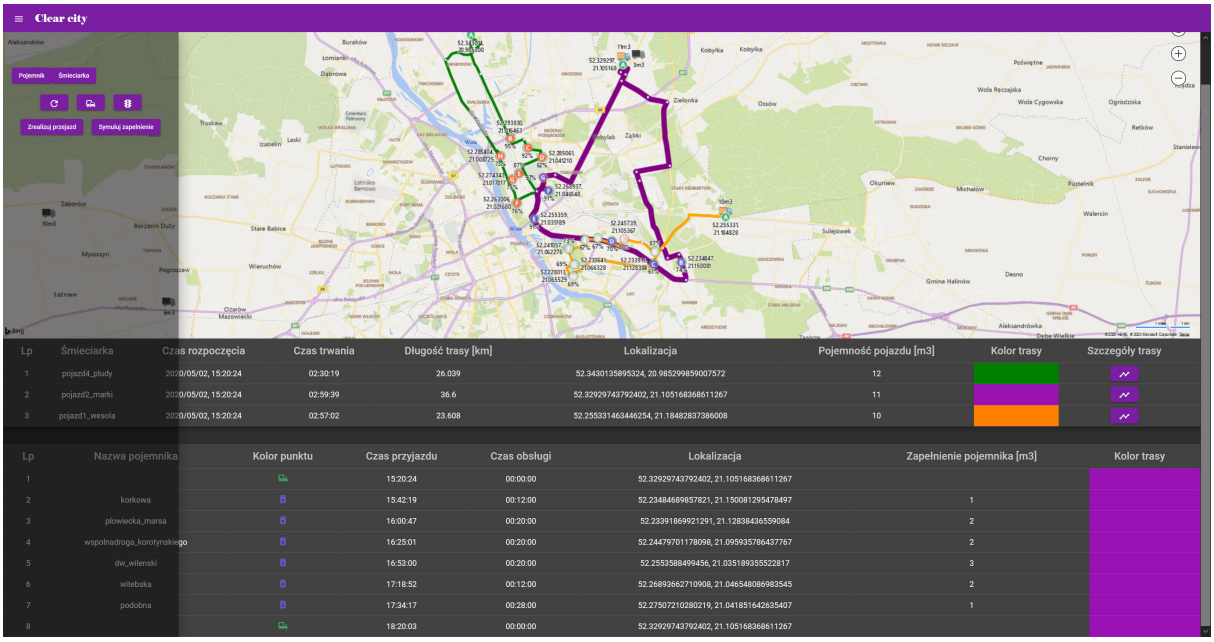


Rysunek 36. Aplikacja po wyznaczeniu tras zbiórki odpadów zawierająca nieopróżnione pojemniki. Źródło: Opracowanie własne.

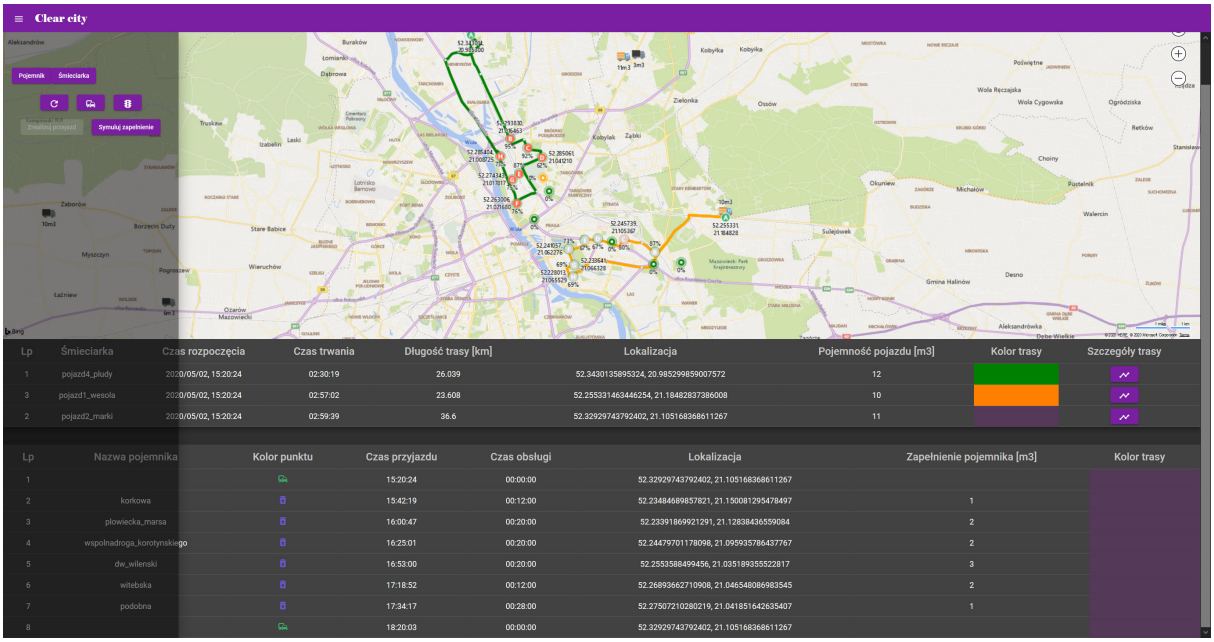


Rysunek 37. Aplikacja po wyznaczeniu tras zbiórki odpadów, zwiększona pojemność pojazdów. Źródło: Opracowanie własne.

Po kliknięciu przycisku w kolumnie `Szczegóły trasy` użytkownik otrzymuje tabelę zawierającą zestawienie kolejno odwiedzanych pojemników dla zadanej trasy (patrz rysunek 38). Punkty obsługi mają zdefiniowany czas przyjazdu brygady wykonującej zbiórkę oraz długość trwania obsługi. Użytkownikowi prezentowane jest również zapełnienie pojemników przeliczone na m^3. Przeglądana trasa zostaje pogrubiona na widoku mapy a w tabelach opisana odpowiednim kolorem. Aktywowany zostaje przycisk `Zrealizuj przejazd`, po kliknięciu którego pojemniki przypisane danej trasie zostają opróżnione. Trasa przestaje być widoczna na mapie, w tabelach jej kolor zostaje zaciemniony a przycisk `Zrealizuj przejazd` przestaje być dla danej trasy aktywny (patrz rysunek 39).



Rysunek 38. Szczegóły wybranej trasy. Źródło: Opracowanie własne.



Rysunek 39. Zrealizowany przejazd. Źródło: Opracowanie własne.

Zapełnienie pojemników użytkownik może definiować dla każdego obiektu z osobna bądź posłużyć się dostarczoną funkcjonalnością losowej zmiany parametru dla wszystkich dostępnych jednostek. Druga z opcji zaimplementowana została poprzez przycisk `Symuluj zapełnienie`.

5.2.3. Wpływ parametrów wejściowych na proces zbiórki odpadów

Ograniczone pojemności śmieciarek uniemożliwiły realizację procesu zbiórki dla wszystkich pojemników w jednej iteracji. Użytkownik może wybrać pomiędzy dwoma dostępnymi opcjami – wysłanie pojazdów ponownie do pozostałych obiektów lub zmiana wielkości stosowanych pojazdów.

Druga opcja w rzeczywistości wymaga posiadania bogatej floty śmieciarek. Aplikacja umożliwia analizę zasadności nabycia większych pojazdów już na etapie planowania procesów. Dane przedstawione w tabelach 3 i 4 odnoszą się do ponowienia iteracji zbiórki dla nieopróżnionych pojemników. Wykonanie usługi dla wszystkich obiektów daje finalnie:

- 09:00:44 [hh:mm:ss]
- 110,938 [km]

Alternatywna opcja – zastosowanie większych pojazdów (patrz tabela 5) wymaga:

- 08:21:17 [hh:mm:ss]
- 85,199 [km]

Firma wywozowa dysponująca określoną flotą pojazdów jest w stanie na podstawie przedstawionych danych określić opłacalność poszczególnych opcji. Autor pracy uważa że zmiana śmieciarek na większe będzie optymalnym rozwiązaniem dla przedstawionego problemu. Użytkownik zyskuje zarówno czas pracy jak również zmniejsza pokonany dystans przez pojazdy.

Tabela 3. Niewystarczająca pojemność pojazdów do opróżnienia wszystkich pojemników. Źródło: Opracowanie własne.

Nazwa pojazdu	Czas trwania trasy [hh:mm:ss]	Długość trasy [km]	Pojemność pojazdu [m3]
pojazd1_wesola	02:48:49	23.314	10
pojazd2_marki	02:17:56	42.470	8
pojazd4_pludy	02:20:04	22.379	9

Tabela 4. Ponowne wysłanie śmieciarek do pozostałych pojemników. Źródło: Opracowanie własne.

Nazwa pojazdu	Czas trwania trasy [hh:mm:ss]	Długość trasy [km]	Pojemność pojazdu [m3]
pojazd1_wesola	01:33:55	22.775	10

Tabela 5. Zwiększenie pojemności pojazdów. Źródło: Opracowanie własne.

Nazwa pojazdu	Czas trwania trasy [hh:mm:ss]	Długość trasy [km]	Pojemność pojazdu [m3]
pojazd1_wesola	03:39:48	28.335	16
pojazd2_marki	02:48:50	33.409	11
pojazd4_pludy	01:52:39	23.455	9

Współczesne miasta borykają się z problemem zatłoczonych dróg. Aplikacja pozwala zasymulować ten stan poprzez ustawienie opóźnień w obsłudze pojemników. Mając dostępne 3 pojazdy pracujące po 4 godziny każdy zrealizowanie tras w zasymulowanych przez autora pracy godzinach szczytu okazuje się niemożliwe. Sytuacja wymaga zwiększenia czasu pracy jednej z załóg. Wcześniejsze przewidywanie podobnych sytuacji może uchronić przedsiębiorstwo od nadmiernych kosztów oraz wspomóc planistów procesu. Widoczne w tabeli 6 dane obrazują źle zaplanowane godziny pracy załóg. Nie wszystkie pojemniki zostają opróżnione w związku z czym autor pracy

postanowił zwiększyć czas pracy jednego z pojazdów. Zmiana parametru spowodowała zmianę tras przejazdu i zapewniła obsługę wszystkich obiektów (patrz tabela 7).

Tabela 6. Symulacja korków, niewystarczający czas pracy załóg. Źródło: Opracowanie własne.

Nazwa pojazdu	Czas trwania trasy [hh:mm:ss]	Długość trasy [km]	Pojemność pojazdu [m3]
pojazd1_wesola	03:47:29	22.308	16
pojazd2_marki	03:51:43	31.140	11
pojazd4_pludy	03:48:58	37.404	9

Tabela 7. Symulacja korków, zwiększony czas pracy załogi pojazdu pojazd2_marki. Źródło: Opracowanie własne.

Nazwa pojazdu	Czas trwania trasy [hh:mm:ss]	Długość trasy [km]	Pojemność pojazdu [m3]
pojazd1_wesola	03:55:13	24.570	16
pojazd2_marki	05:00:08	41.712	11
pojazd4_pludy	03:56:10	27.881	9

Zmiana parametrów systemu przekłada się na realizację założonych tras. Symulacja wybranych wartości może pomóc w doborze odpowiedniej ilości oraz rozmiaru pojazdów.

6. Zalety, wady oraz plany rozwojowe

Realizacja tematu pracy skupiała się na przedstawieniu koncepcji usprawnienia procesu zbiórki odpadów. W konsekwencji autor stworzył aplikację webową usprawniającą czynność wyznaczania tras ruchu śmieciarek. Zastosowano ogólnie dostępne narzędzia oraz technologie umożliwiające sprawną implementację oprogramowania. Prototyp aplikacji spełnia podstawowe założenia autora pracy jednak można byłoby go usprawnić o dodatkowe funkcjonalności stosując przedstawione w poniższym rozdziale rozwiązania.

6.1. Zalety oraz wady przyjętych rozwiązań

Potencjalnymi odbiorcami aplikacji będącej częścią niniejszej pracy mogą być firmy wywożące odpady oraz zarządy gmin. Dostarczone rozwiązanie implementacyjne umożliwia prezentację funkcjonowania Internetu Rzeczy nie posiadając rzeczywistych elementów. Cecha ta wyróżnia prezentowany system od dostępnych na rynku używających fizycznych czujników pomiarowych. Ułatwia to przedstawienie działania oraz sprzedaż systemu potencjalnym nabywcom. Aplikację można stosować dla dowolnych obszarów, z wyjątkiem tych których sytuacja drogowa nie jest dostępna na mapach Bing. Rozwiązanie jest przenośne oraz intuicyjne. Użytkownik zmieniając parametry wejściowe systemu symuluje warunki świata rzeczywistego. Firma wywozowa przy wykorzystaniu aplikacji określi:

- czas pracy załogi,
- optymalne trasy zbiórki odpadów,
- zapotrzebowanie na pojazdy w swojej flocie,
- najbardziej dogodne godziny zbiórki odpadów.

Zarządy gmin za pomocą dostarczonego rozwiązania skorzystają podczas:

- planowania rozmieszczenia pojemników,
- oceniania kosztów zbiórki odpadów.

Wadą prototypu aplikacji jest ograniczenie liczby obiektów na mapie. Producent narzędzia optymalizującego trasy przejazdu wprowadził restrykcje co do ilości obsługiwanych pojazdów i punktów pośrednich. Autor pracy w odpowiedzi na ograniczenie uwzględnił pewne założenia przy obliczaniu zapełnienia pojazdów. Zupełne wyeliminowanie ograniczenia nastąpi wraz z integracją do płatnej wersji API Microsoftu.

6.2. Plany rozwojowe

Komercjalizacja aplikacji wymaga wykupienia płatnej wersji MIO API Microsoftu. Rozwiązanie umożliwi wykonywanie obliczeń dla większej ilości obiektów jednocześnie lepiej odwzorowując warunki świata rzeczywistego. Integracja wymagać może drobnych poprawek w wizualizacji zoptymalizowanych tras na mapie.

Listę potencjalnych odbiorców aplikacji powiększyć można dodając kolejne typy pojemników wraz z ich specyficznymi parametrami. Wdrożenie będzie wiązało się z dołączeniem nowych typów pojazdów realizujących zbiórkę. Podczas tworzenia prototypu nie zostały uwzględnione między innymi:

- pojemniki na używaną odzież,
- zbiorniki na odpady płynne.

Istotnym usprawnieniem w docelowej aplikacji może być uwzględnienie wymiarów pojazdów podczas wyznaczania tras przejazdu. Funkcjonalność umożliwi ocenienie oraz ominięcie przeszkód występujących na drogach. Wdrożenie polegało będzie na zastosowaniu Truck Routing API od Microsoftu. Jednocześnie w aplikacji przygotować należy odpowiednie formularze do wprowadzania dla pojazdu:

- wysokości,
- długości,
- szerokości,
- wagi,
- promienia skrętu.

7. Podsumowanie

Rosnąca pozycja Internetu Rzeczy była inspiracją realizacji tematu niniejszej pracy. Autor pracy przeanalizował istniejące problemy zarządzania zbiórką odpadów. Po przeglądzie zasad gospodarki odpadami przedstawione zostały istniejące na rynku rozwiązania wspomagające proces.

Stworzony przez autora prototyp aplikacji pozwala zasymulować działanie systemu zarządzania zbiórką odpadów. Aplikacja nie wymaga od użytkownika posiadania fizycznych urządzeń pomiarowych. Proponowane przez autora pracy rozwiązanie może wspomóc zarządy miast oraz firmy zajmujące się zbiórką odpadów. Rozwiązanie może znaleźć zastosowanie przy prezentacji działania systemu docelowo wyposażonego w fizyczne urządzenia.

Klienci wdrażający rozwiązania oparte finalnie na fizycznych czujnikach oszczędzą na procesie zbiórki odpadów jednocześnie zmniejszając zanieczyszczenie środowiska. Śmieciarki kursować będą rzadziej i wyłącznie do miejsc gdzie jest na to zapotrzebowanie. Wdrażanie przedstawionych w niniejszej pracy rozwiązań może pomóc w zmniejszeniu zakorkowania współczesnych miast.

Bibliografia

[1]. Ustawa o odpadach [Dz. U. 2016 poz. 1987]

[2]. J.Baran, M. Karlewska: Teoretyczne aspekty logistycznego system gospodarki odpadami, Szkoła Główna Gospodarstwa Wiejskiego w Warszawie, Warszawa 2016

[3]. A. Merkisz-Guranowska: Logistyka recyklingu odpadów, jako jeden z elementów system logistycznego Polski, Politechnika Warszawska, Warszawa 2010

[4]. Eco portal [Online] http://ecoportal.com.pl/urzadzenia-do-transportu-usuwania-odpadow/ [data dostępu: 16.11.2019].

[5]. K. Lelicińska-Serafin, P. Manczarski: Rozwój techniki komunalnej - zbieranie i transport odpadów komunalnych. Gaz, Woda i Technika Sanitarna 11, 2012, Warszawa

[6]. EWISEL [Online]http://gem.katowice.pl/ [data dostępu: 14.12.2019]

[7]. Elte Gps [Online]https://www.eltegps.pl/produkty/branza-oczyszczania/identyfikacja-pojemnikow-na-odpady.html [data dostępu: 23.11.2019]

[8]. Eco Bins [Online]http://ecobins.pl/ [data dostępu: 06.12.2019]

[9]. Logistyka [Online] https://www.logistyka.net.pl/bank-wiedzy/logistyka/item/84334-logistyczny-system-gospodarki-odpadami [data dostępu: 20.11.2019]

[10]. Navicar [Online]http://navicar.pl/ [data dostępu: 16.12.2019]

[11]. Abys-Technika [Online]https://abrys-technika.pl/produkty/pojemniki-do-segregacji-dzwon-at-s002/ [data dostępu: 25.11.2019]

[12]. JK [Online]www.jk.com.pl [data dostępu: 25.11.2019]

[13]. ELTE GPS [Online] https://www.eltegps.pl/pdf/Elte-GPS-branza-komunalna-katalog.pdf [data dostępu: 14.03.2020]

[14]. ELTE GPS [Online] https://www.eltegps.pl/produkty/6_fpsph/prod_main.html [data dostępu: 14.03.2020]

[15]. NaviSoft https://navisoft.pl/navicar [data dostępu: 14.03.2020]

[16]. M. Masse: REST API Design Rulebook, O'Reilly, ISBN: 978-1-449-31050-9, Sebastopol 2012

[17]. Itnext [Online] https://itnext.io/javascript-fundamentals-an-introduction-to-rest-apis-7cbe8a809d3b [data dostępu: 16.03.2020]

[18]. T. Marrs JSON at Work, O'Reilly, ISBN: 978-1-449-35832-7, Sebastopol 2017

[19]. RxJS [Online] https://rxjs-dev.firebaseapp.com/ [data dostępu: 18.03.2020]

[20]. G. Smith, P. Ledbrook: Grails in Action, Manning, ISBN: 978-1-933988-93-1, Greenwich 2009

[21]. Groovy [Online] http://groovy-lang.org/ [data dostępu: 18.03.2020]

[22]. J. Wilken: Angular w akcji. Helion, ISBN: 978-83-283-4799-1, 2019

[23]. L. Loeliger: Kontrola wersji z systemem Git, Helion, ISBN: 978-83-246-8179-2, 2014

[24]. Postgesql [Online] https://www.postgresql.org/about/ [data dostępu: 20.03.2020]

[25]. Bing Maps [Online] https://www.bing.com/maps [data dostępu: 01.04.2020]

[26]. Bing Maps MIO API [Online] https://www.microsoft.com/en-us/maps/multi-itinerary-optimization [data dostępu: 01.04.2020]

[27]. Bing Maps Optimize Multiple Itineraries [Online] https://docs.microsoft.com/en-us/bingmaps/rest-services/routes/optimized-itinerary [data dostępu: 26.04.2020]

[28]. GUS [Online] https://stat.gov.pl/ [data dostępu: 17.11.2019]

[29]. M. Kalisiak-Mędelska: Logistyka a gospodarka odpadami w miastach, Uniwersytet Łódzki, 2017

[30]. D. Guinard, V. Trifa: Internet rzeczy. Budowa sieci z wykorzystaniem technologii webowych i Raspberry Pi, Helion, ISBN: 978-83-283-2969-0, 2017

[31]. PRIME Faraday Partnership: An Introduction to MEMS (Micro-electromechanical Systems) ISBN 1-84402-020-7, Loughborough University 2002

[32]. TechWasti [Online] https://www.techwasti.com/architecture-for-iot-applications-d50ece031d38/ [data dostępu: 20.05.2020]

Spis rysunków

Spis listingów

Spis tabel

Dodatki

Dodatek A: Słownik użytej terminologii i skrótów

ACID –Atomicity, Consistency, Isolation, Durability

API – Application Programming Interface

CLI – Command Line Interface

DSL – Domain Specific Language

GORM –Grails Object Relational Mapping

GPS – Global Positioning System

GSM – Global System for Mobile Communications

GUS – Główny Urząd Statystyczny

HTTP – Hypertext Transfer Protocol

Internet - globalna sieć komputerowa

IoT – Internet of Things

JSON – Javascript Object Notation

MEMS – Micro-Electro-Mechanical Systems

MIT - Massachusetts Institute of Technology

MS SQL Server – Microsoft SQL Server

MVC – Model View Controller

Npm – Node package manager

REST – Representational State Transfer

RF – Radio Frequency

RFID – Radio-Frequency Identification

SQL - Structured Query Language

URI – Uniform Resource Identifier

UTF-8 – 8-bit Unicode Transformation Format

XML - Extendible Markup Language

Dodatek B: Optymalizacja MIO API

Listing 10. Zapytanie wysyłane do MIO API. Źródło: Opracowanie własne.


```
{
  "agents": [
    {
      "shifts": [
        {
          "startTime": "2020-05-30T20:02:39",
          "startLocation": {
            "latitude": 52.34274517397725,
            "longitude": 20.988076088692075
          },
          "endTime": "2020-05-31T00:02:39",
          "endLocation": {
            "latitude": 52.34274517397725,
            "longitude": 20.988076088692075
          }
        }
      ],
      "capacity": [
        9
      ],
      "name": "pojazd4_pludy"
    },

    .
    .
    .

  ],
  "itineraryItems": [
    {
      "quantity": [
        2
      ],
      "dropOffFrom": [],
      "name": "plowiecka_marsa",
      "openingTime": "2020-05-30T18:02:39",
      "closingTime": "2020-05-31T02:02:39",
      "dwellTime": "00:30:00",
      "priority": 5,
      "location": {
        "Latitude": 52.23370142619327,
        "Longitude": 21.129038883271612
      }
```

```
    },

    .
    .
    .

  ],
  "type": "TrafficRequest",
  "roadnetwork": true
}
```

Listing 11. Odpowiedź MIO API przedstawiająca zoptymalizowane trasy. Źródło: Opracowanie własne.

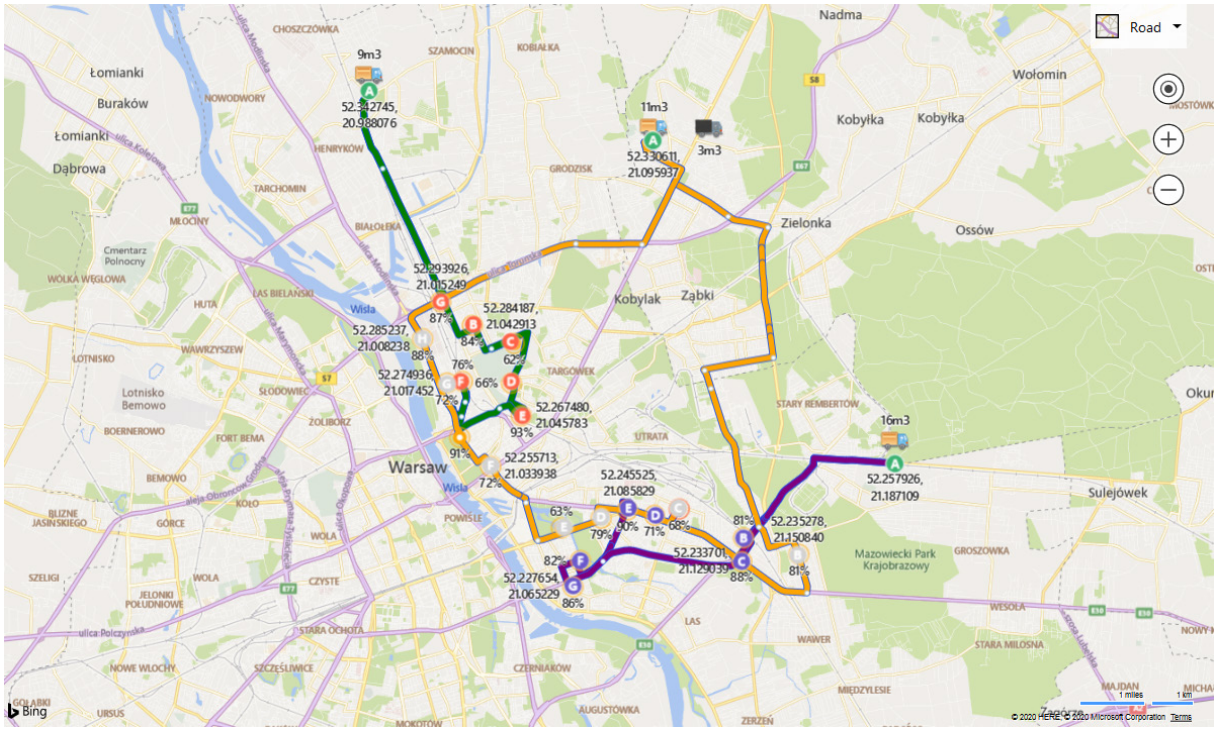
```
"agentItineraries": [
  {
    "agent": {
      "capacity": [
         9
      ],
      "name": "pojazd4_pludy",
      "shifts": [
        {
          "endLocation": {
            "latitude": 52.34274517397725,
            "longitude": 20.988076088692075
          },
          "endTime": "2020-05-31T00:02:39+00:00",
          "startLocation": {
            "latitude": 52.34274517397725,
            "longitude": 20.988076088692075
          },
          "startTime": "2020-05-30T20:02:39+00:00"
        }
      ]
    },
    "instructions": [
      {
        "duration": "00:00:00",
        "endTime": "2020-05-30T20:02:39+00:00",
        "instructionType": "LeaveFromStartPoint",
        "itineraryItem": {
          "closingTime": "2020-05-31T00:02:39+00:00",
          "dropOffFrom": [],
          "dwellTime": "00:00:00",
          "location": {
            "latitude": 52.34274517397725,
```

```
          "longitude": 20.988076088692075
        },
        "name": "",
        "openingTime": "2020-05-30T20:02:39+00:00",
        "priority": 1,
        "quantity": []
      },
      "startTime": "2020-05-30T20:02:39+00:00"
    },
    {
      "distance": 7555,
      "duration": "00:15:25",
      "endTime": "2020-05-30T20:18:04+00:00",
      "instructionType": "TravelBetweenLocations",
      "startTime": "2020-05-30T20:02:39+00:00"
    },
    {
      "duration": "00:32:00",
      "endTime": "2020-05-30T20:50:04+00:00",
      "instructionType": "VisitLocation",
      "itineraryItem": {
        "closingTime": "2020-05-31T02:02:39+00:00",
        "dropOffFrom": [],
        "dwellTime": "00:32:00",
        "location": {
          "latitude": 52.28838695191588,
          "longitude": 21.02712059020996
        },
        "name": "galeria_renova",
        "openingTime": "2020-05-30T18:02:39+00:00",
        "priority": 5,
        "quantity": [
          2
        ]
      },
      "startTime": "2020-05-30T20:18:04+00:00"
    },
    {
      "distance": 1480,
      "duration": "00:04:20",
      "endTime": "2020-05-30T20:54:24+00:00",
      "instructionType": "TravelBetweenLocations",
      "startTime": "2020-05-30T20:50:04+00:00"
    },
    {
      "duration": "00:40:00",
      "endTime": "2020-05-30T21:34:24+00:00",
```

```
    "instructionType": "VisitLocation",
    "itineraryItem": {
      "closingTime": "2020-05-31T02:02:39+00:00",
      "dropOffFrom": [],
      "dwellTime": "00:40:00",
      "location": {
        "latitude": 52.284186628775686,
        "longitude": 21.04291343688965
      },
      "name": "matkiteresy",
      "openingTime": "2020-05-30T18:02:39+00:00",
      "priority": 5,
      "quantity": [
        2
      ]
    },
    "startTime": "2020-05-30T20:54:24+00:00"
  },

  .
  .
  .

  {
    "distance": 6165,
    "duration": "00:12:39",
    "endTime": "2020-05-30T23:58:49+00:00",
    "instructionType": "TravelBetweenLocations",
    "startTime": "2020-05-30T23:46:10+00:00"
  },
  {
    "duration": "00:00:00",
    "endTime": "2020-05-30T23:58:49+00:00",
    "instructionType": "ArriveToEndPoint",
    "itineraryItem": {
      "closingTime": "2020-05-31T00:02:39+00:00",
      "dropOffFrom": [],
      "dwellTime": "00:00:00",
      "location": {
        "latitude": 52.34274517397725,
        "longitude": 20.988076088692075
      },
      "name": "",
      "openingTime": "2020-05-30T20:02:39+00:00",
      "priority": 1,
      "quantity": []
    },
```

```
        "startTime": "2020-05-30T23:58:49+00:00"
      }
    ],
    "route": {
      "endLocation": {
        "latitude": 52.34274517397725,
        "longitude": 20.988076088692075
      },
      "endTime": "2020-05-30T23:58:49+00:00",
      "startLocation": {
        "latitude": 52.34274517397725,
        "longitude": 20.988076088692075
      },
      "startTime": "2020-05-30T20:02:39+00:00",
      "totalTravelDistance": 27881,
      "totalTravelTime": "01:04:10",
      "wayPoints": [
        {
          "latitude": 52.28838695191588,
          "longitude": 21.02712059020996
        },
        {
          "latitude": 52.284186628775686,
          "longitude": 21.04291343688965
        },
        {
          "latitude": 52.27507210280219,
          "longitude": 21.041851642635407
        },
        {
          "latitude": 52.267480223303885,
          "longitude": 21.045783115449346
        },
        {
          "latitude": 52.27536465396692,
          "longitude": 21.023344039916992
        },
        {
          "latitude": 52.2939263360164,
          "longitude": 21.015248609817437
        }
      ]
    }
  },

  .
  .
```

```
    .


    }
]
```



Rysunek 40. Wizualizacja zoptymalizowanych tras przez MIO API. Źródło: Opracowanie własne.