

# PAP - „PA jak Pojadę”

## Autorzy:

Bartosz Han 318658

Mykhailo Marfenko 323558

Igor Matynia 318693

**Prowadzący zespołu:** Michał Chwesiuk

## 1. Wstęp

Celem naszego projektu było stworzenie aplikacji desktopowej która ma pomóc użytkownikowi w planowaniu swoich podróży komunikacją miejską w aglomeracji warszawskiej. Użytkownik może wyświetlać dane o poszczególnych liniach, przystankach, odjazdach oraz może zaplanować sobie podróż używając komunikacji miejskiej.

Nasza aplikacja została napisana w języku Python, a UI do niej zostało stworzone za pomocą Qt.

## 2. Obsługa programu

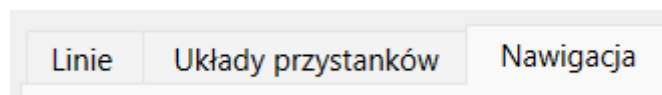
W celu uruchomienia programu lub utworzenia pliku wykonywalnego, potrzebny będzie **Python w wersji 3.10 lub wyższej**, oraz następujące biblioteki do Pythona:

- folium
- PySide6
- python\_oracledb
- pyinstalller

Aby uruchomić program, należy w folderze z projektem wpisać komendę w terminalu „make”, która uruchomi nasz program. Możemy też stworzyć plik wykonywalny, za pomocą komendy ‘make executable’. Po użyciu tej komendy, nasz plik wykonywalny będzie się znajdował w folderze ‘dist/app’, jako ‘Pa jak podajde’ (‘.exe’ jeśli uruchamiamy to na Windowsie).

Po uruchomieniu program potrzebuje kilka sekund, aby pobrać najpotrzebniejsze dane o transporcie, które będą później wielokrotnie wykorzystywane np. informacje o przystankach.

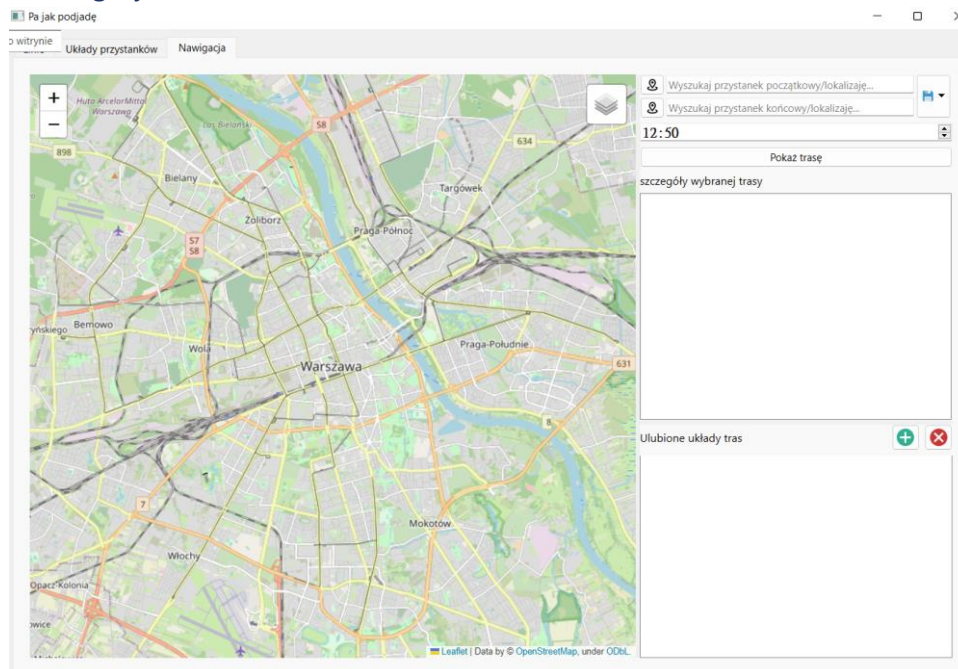
Aplikacja składa się z trzech okienek: ‘Linie’, ‘Układy przystanków’ i ‘Nawigacja’:



Aby zmienić okienko, należy kliknąć interesującą nas sekcję.

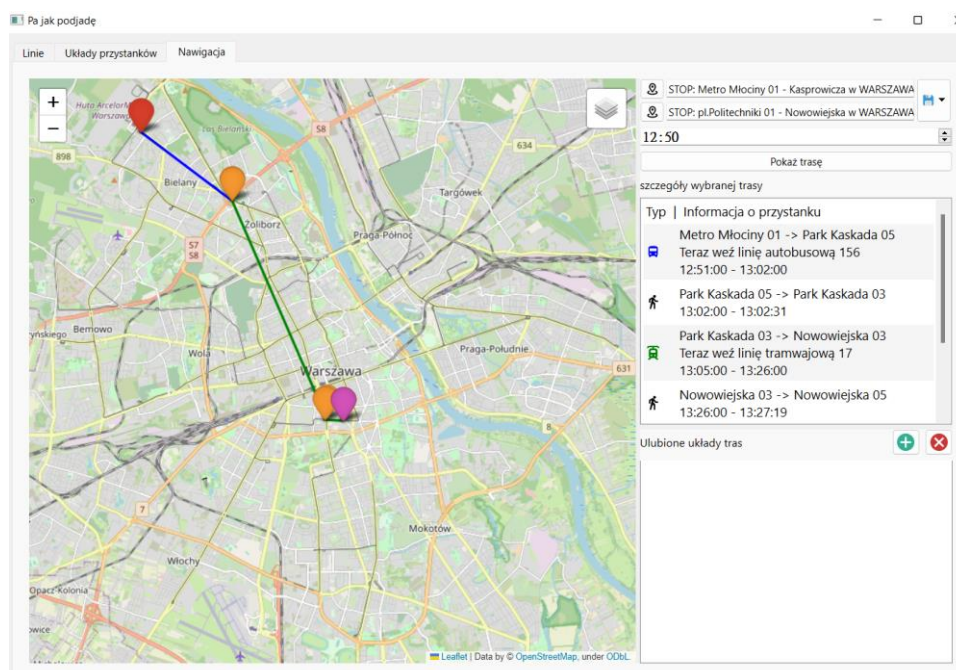
Poniżej przedstawione są poszczególne sekcje aplikacji:

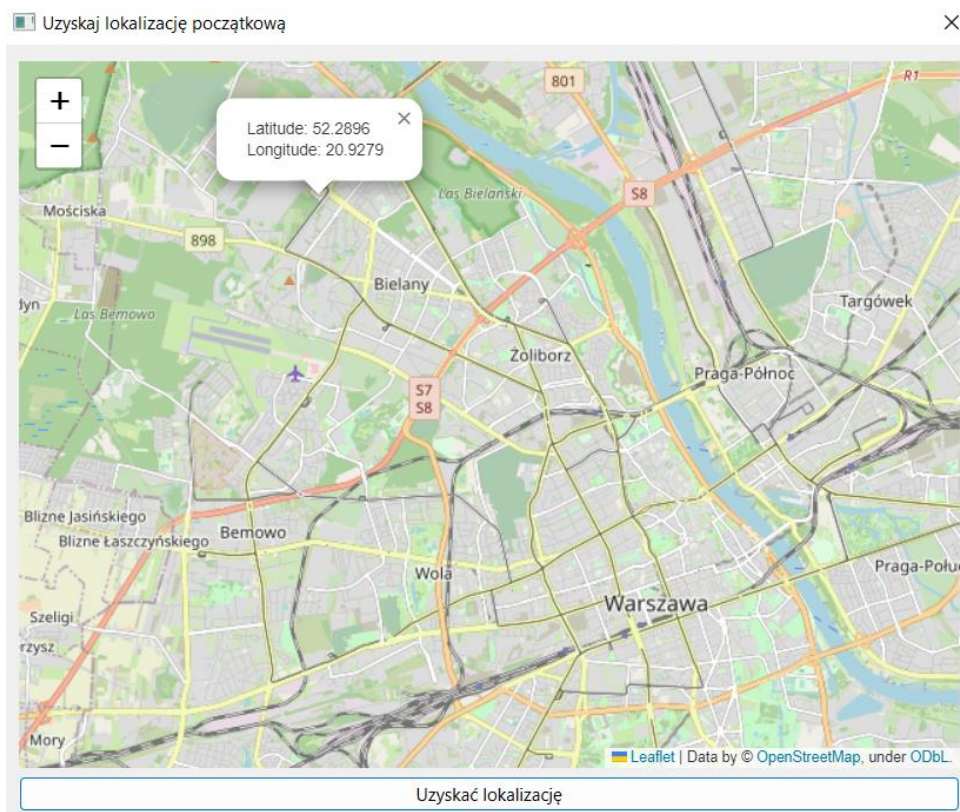
## 2.1 Nawigacja



Jest to okienko, w którym to można wyznaczyć trasę, którą chcemy pokonać transportem publicznym. Po lewej stronie mamy mapę Warszawy, a po prawej – informacje o trasie.

Aby zacząć wyszukiwać trasę, należy wpisać interesujące nas przystanki: początkowy i końcowy, oraz odpowiednią godzinę. Możemy także wybrać dowolną lokalizację, poprzez przycisk po lewej stronie względem pola do wpisywania przystanków. Po wpisaniu trasy aplikacja zacznie szukać optymalnego przejazdu komunikacją miejską:

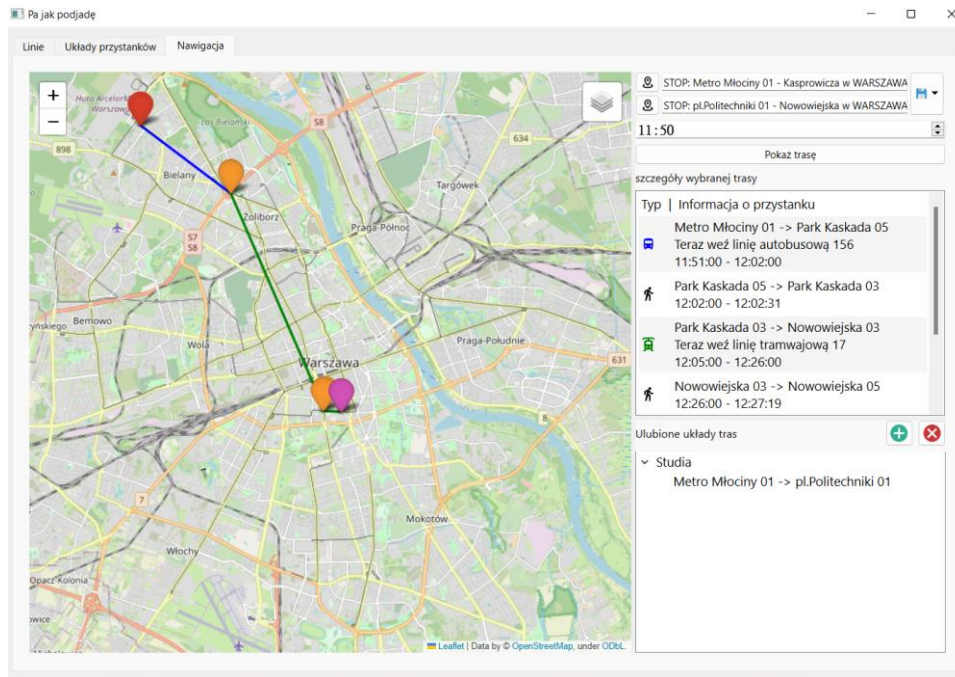




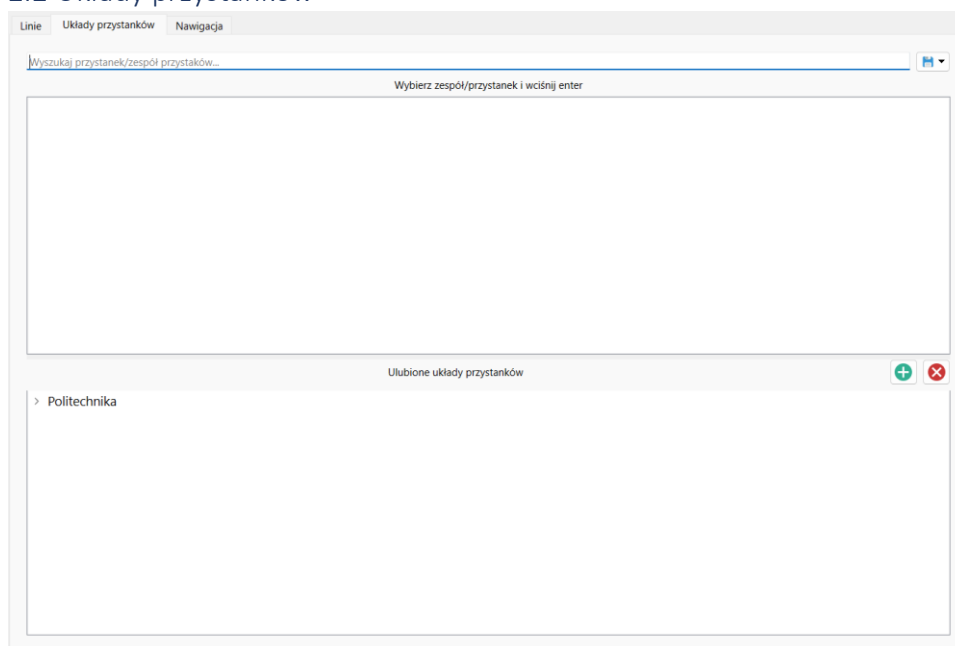
Ze względu na dość dużą liczbę danych do przetworzenia i pobrania, wyszukiwanie trasy może chwilę zająć: od kilkunastu sekund do, w najgorszym przypadku, paru minut. Im więcej się będzie wyszukiwało tras w ramach pojedynczego użycia programu, tym więcej danych będzie już pobranych i wyszukiwanie będzie nieco szybsze.

Na mapie pojawiła się wyznaczona przez algorytm trasa, natomiast po prawej stronie znajdują się istotne informacje o trasie: na jaki przystanek trzeba się udać, kiedy i jakim środkiem transportu publicznego się udać, czy godziny odjazdów.

Jeśli często się wybieramy jakąś trasą, możemy ją sobie zapisać. Należy w tym celu stworzyć grupę tras (zielony przycisk z plusem), a następnie zapisać, klikając przycisk po prawej stronie od pól do wpisywania przystanków. Trasa zostanie zapisana, a użytkownik będzie mógł szybciej wyszukać swoją trasę.



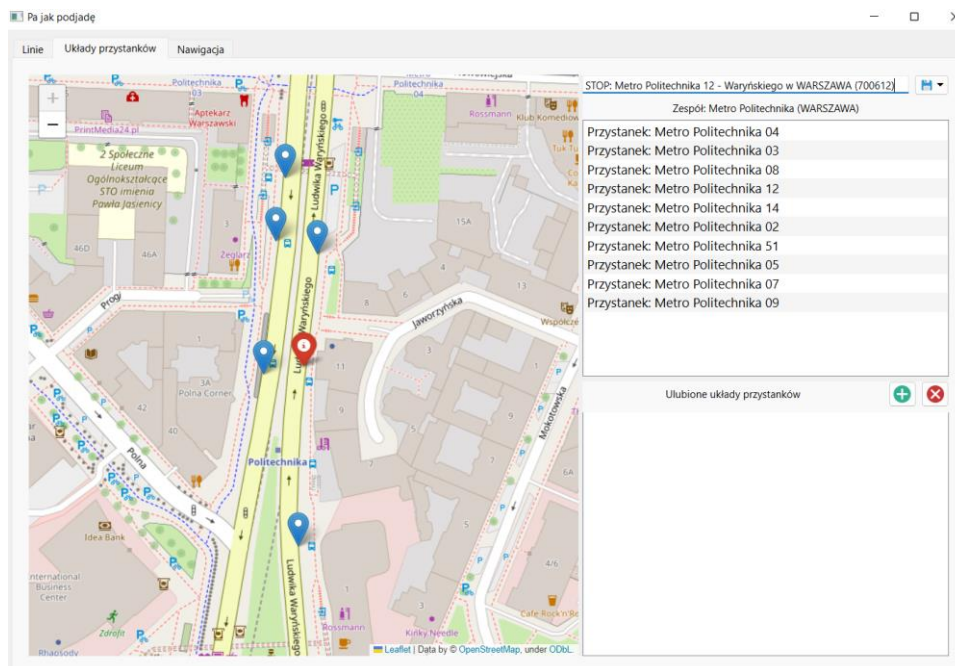
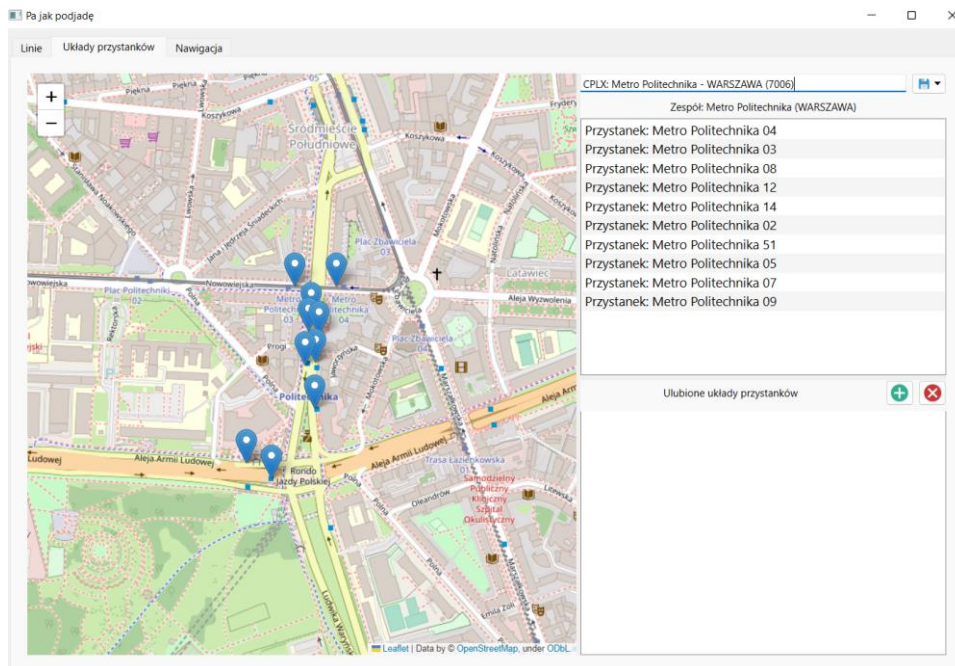
## 2.2 Układy przystanków



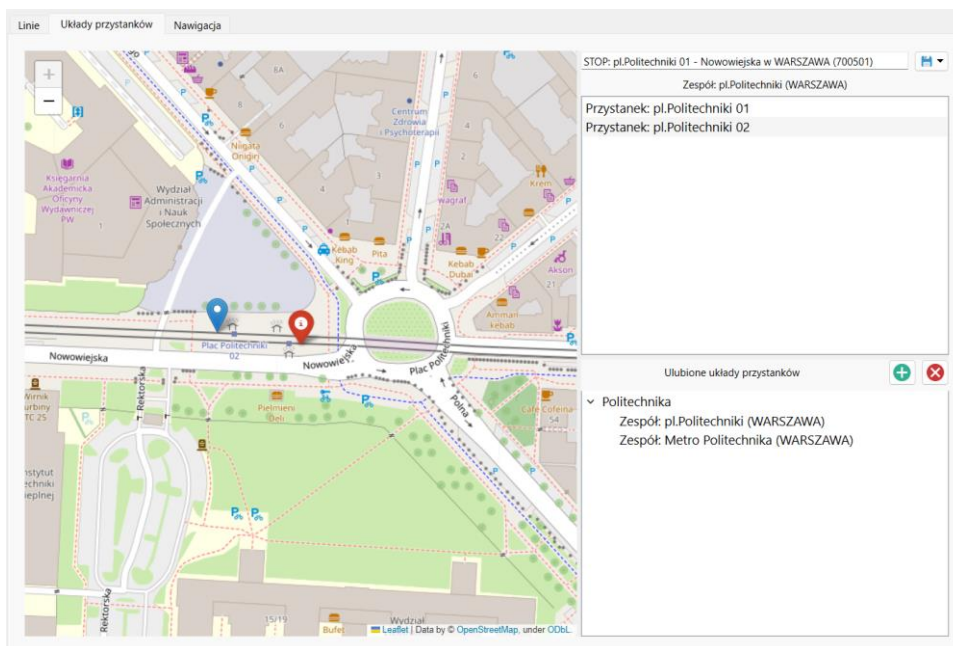
Okienko te służy do wyświetlania informacji o przystankach i ich położeniach.

Aby wyszukać jakiś przystanek, wystarczy wpisać jego nazwę do wyszukiwarki, wybrać interesujący nas przystanek i wcisnąć enter. Można wybrać zarówno jeden, pojedynczy przystanek, jak i całe zespoły przystankowe. Po wyszukaniu przystanku pojawi się mapa z przystankami składającymi się na zespół przystankowy, oraz lista tych przystanków.





Swoje ulubione przystanki można też zapisywać. Trzeba w tym celu utworzyć nową grupę i dodać do niej interesujące nas przystanki. Po ponownym uruchomieniu aplikacji będzie można szybciej wyszukać informacje o interesujących nas przystankach.

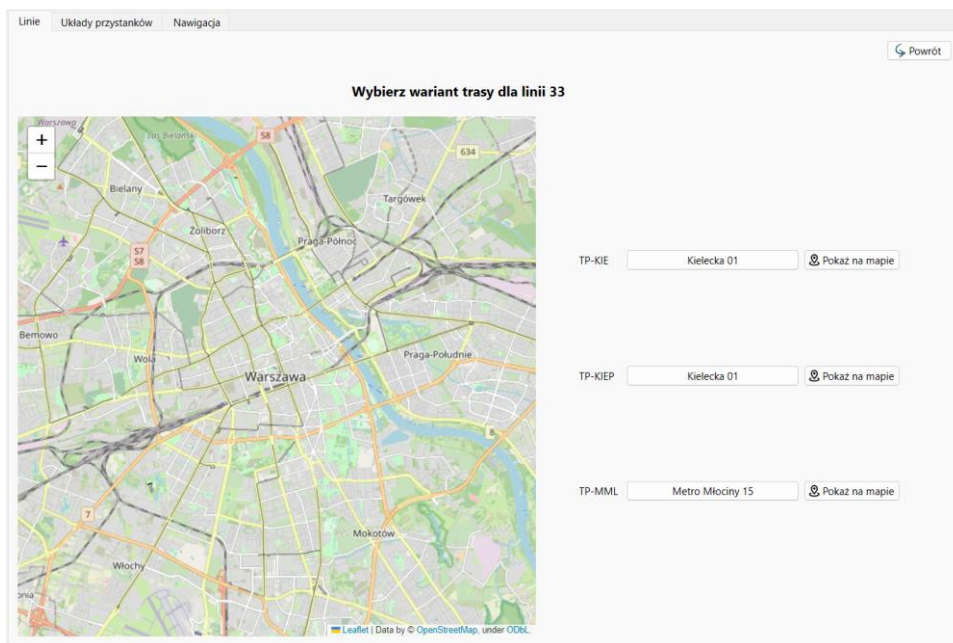


## 2.3 Linie

Linie	Układy przystanków	Nawigacja
1	2	3
4	6	7
9	11	13
15	17	20
22	23	
24	25	26
27	28	31
33	35	75
78	102	103
104	105	
106	107	108
109	110	111
112	114	115
116	117	118
119	120	
121	122	123
124	125	126
127	128	129
131	132	133
134	135	
136	138	139
140	141	142
143	145	146
147	148	149
150	151	
152	153	154
156	157	158
159	160	161
162	163	164
165	166	
167	168	169
170	171	172
173	174	175
176	177	178
179	180	
181	182	183
184	185	186
187	188	189
190	191	192
193	194	
196	197	198
199	200	201
202	203	204
207	208	209
210	211	
212	213	217
218	220	221
222	225	226
228	233	234
239	240	
245	249	250
251	255	256
262	263	264
303	305	308
311	314	
317	319	320
326	328	331
332	338	339
340	356	379
401	402	
409	411	414
500	501	502
503	504	507
509	511	512
514	516	
517	518	519
520	521	522
523	525	527
702	703	704
705	706	
707	709	710
711	712	713
714	715	716
717	719	720
721	722	
723	724	727
728	729	730
731	733	735
736	737	738
739	742	
743	750	809
815	817	850
900	E-1	E-2
L-1	L-2	L-3
L-4	L-5	
L-6	L-7	L-8
L-9	L10	L11
L12	L13	L14
L15	L16	L17
L18	L19	
L20	L21	L22
L23	L24	L25
L26	L27	L28
L29	L30	L31
L32	L34	
L35	L36	L37
L38	L39	L40
L41	L43	L45
L46	L47	L48
L49	L50	
L51	L52	N01
N02	N03	N11
N12	N13	N14
N16	N21	N22
N24	N25	
N31	N32	N33
N34	N35	N36
N37	N38	N41
N42	N43	N44
N45	N46	
N50	N56	N58
N61	N62	N63
N64	N71	N72
N81	N83	N85
N86	N88	
N91	N95	S1
S10	S2	S3
Z-4	Z42	Z76

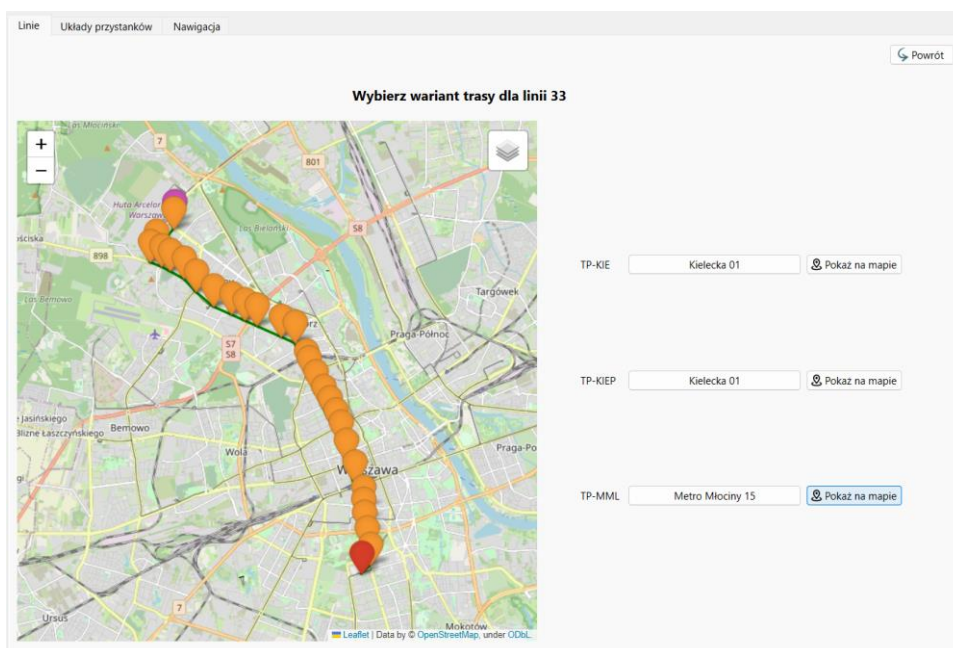
Okienko te służy do wyświetlania informacji o dostępnych liniach autobusowych, tramwajowych i kolejowych.

Aby wybrać interesującą nas linię, należy wcisnąć przycisk o odpowiedniej linii. Po krótkim pobraniu danych, zostaniemy przekierowani do nowego okienka:



Po prawej stronie otrzymujemy listę dostępnych wariantów wybranej przez nas linii.

Gdy zdecydujemy się na dany wariant, możemy wyświetlić przebieg trasy tego wariantu, klikając na przycisk 'Pokaż na mapie' przy tymże wariantcie:




W okienku tym możemy uzyskać informację o odjazdach z poszczególnych przystanków w ramach danego wariantu linii. Należy wówczas wcisnąć przycisk, z odpowiednim kierunkiem wariantu, znajdujący się po lewej stronie od przycisku 'Pokaż na mapie'. Po wciśnięciu tego przycisku pojawi się lista przystanków tego wariantu:

Linie

Układy przystanków

Nawigacja



Nr1	Kielecka 02	Rakowiecka
Nr2	Metro Pole Mokotowskie 04	al.Niepodległości
Nr3	Biblioteka Narodowa 04	al.Niepodległości
Nr4	GUS 04	al.Niepodległości
Nr5	Nowowiejska 04	al.Niepodległości
Nr6	Koszykowa 04	al.Niepodległości
Nr7	Dw.Centralny 10	al.Jana Pawła II
Nr8	rondo ONZ 04	al.Jana Pawła II
Nr9	Hala Mirowska 04	al.Jana Pawła II
Nr10	Kino Femina 10	al.Jana Pawła II
Nr11	Nowolipki 06	al.Jana Pawła II
Nr12	Anielewicz 06	al.Jana Pawła II
Nr13	Stawki 06	al.Jana Pawła II
Nr14	rondo "Radosława" 06	al.Jana Pawła II
Nr15	rondo "Radosława" 10	al.Jana Pawła II
Nr16	pl.Grunwaldzki 08	Broniewskiego
Nr17	Sady Żoliborskie 04	Broniewskiego
Nr18	Włocławska 04	Broniewskiego
Nr19	Park Olszyna 04	Broniewskiego
Nr20	Romaszewskiego 04	Broniewskiego
Nr21	Piaski 04	Broniewskiego

Możemy wówczas wybrać przystanek i sprawdzić tablicę odjazdów:

Linie	Układy przystanków	Nawigacja		
<div><div></div><div>Powrót</div></div>				
5:18:00	5:38:00	5:48:00	6:04:00	6:10:00
6:22:00	6:28:00	6:34:00	6:41:00	6:47:00
6:54:00	6:58:00	7:06:00	7:10:00	7:18:00
7:22:00	7:30:00	7:34:00	7:42:00	7:46:00
7:54:00	7:58:00	8:06:00	8:10:00	8:18:00
8:22:00	8:30:00	8:34:00	8:42:00	8:47:00
8:53:00	8:59:00	9:05:00	9:10:00	9:16:00
9:22:00	9:28:00	9:34:00	9:40:00	9:46:00
9:52:00	9:58:00	10:04:00	10:10:00	10:16:00
10:22:00	10:28:00	10:34:00	10:40:00	10:46:00
10:52:00	10:58:00	11:04:00	11:10:00	11:16:00
11:22:00	11:28:00	11:34:00	11:40:00	11:46:00
11:52:00	11:58:00	12:04:00	12:10:00	12:16:00
12:22:00	12:28:00	12:34:00	12:40:00	12:46:00
12:52:00	12:58:00	13:04:00	13:10:00	13:16:00
13:22:00	13:28:00	13:34:00	13:40:00	13:46:00
13:52:00	13:58:00	14:04:00	14:11:00	14:17:00
14:22:00	14:30:00	14:34:00	14:42:00	14:46:00
14:54:00	14:58:00	15:10:00	15:18:00	15:22:00
15:30:00	15:34:00	15:42:00	15:46:00	15:54:00
15:58:00	16:06:00	16:10:00	16:18:00	16:22:00
16:30:00	16:34:00	16:42:00	16:46:00	16:54:00
16:58:00	17:06:00	17:10:00	17:18:00	17:22:00
17:30:00	17:34:00	17:42:00	17:46:00	17:54:00
17:59:00	18:05:00	18:10:00	18:16:00	18:22:00
18:28:00	18:34:00	18:40:00	18:46:00	18:52:00
18:58:00	19:10:00	19:22:00	19:28:00	19:40:00
19:52:00	19:58:00	20:08:00	20:18:00	20:28:00
20:38:00	20:48:00	20:58:00	21:08:00	21:18:00

## 3. Szczegóły projektu

### 3.1 Pierwotnie założenia

Naszymi pierwotnymi założeniami, zapisanymi we dokumentacji wstępnej, były:



Linie autobusowe/tramwajowe/metra/skm

- > Po wyborze linii, można wybrać przystanek
- > Wyświetla się wtedy rozkład jazdy
- > Dla każdej godziny w rozkładzie można podejrzeć czasy przyjazdu na inne przystanki dla danego autobusu
- > W tym miejscu można też wyświetlić zespół przystankowy
- > Wyświetlanie, gdzie można dojechać z tego miejsca

Szukanie połączenia

Podejrzenie spóźnień dla danego autobusu na danym przystanku o danej porze

Ze względu na obszerność podjętych przez nas prac nad projektem, nie udało nam się zrealizować podejrzenia czasu odjazdów z danego przystanku o danej godzinie, oraz wyświetlania miejsc skąd można dojechać z danego przystanku. Nie udało nam się też zrobić podejrzenia spóźnień dla danego autobusu, jednak tutaj problemem był brak dostatecznie dobrego API do wyszukiwania spóźnień i lokalizacji pojazdów.

### 3.2 Szczegóły techniczne backendu

#### 3.2.1 Modele danych

W ramach projektu zamodelowaliśmy transport publiczny, skupiając się na czterech najważniejszych, naszym zdaniem, elementach komunikacji publicznej: przystanku (Stop), zespole przystankowego (Stop Complex), wariantowi (Variant) oraz kursowi (Course).

**Stop** – jest to pojedynczy przystanek.

**Stop Complex** – jest to zespół przystankowy. Na jeden Stop Complex składa się wiele pojedynczych Stopów.

Przykład – Dw. Centralny, jako całość, jest pojedynczym Stop Complex. Jednak ten zespół przystankowy składa się z wielu Stopów, np. takiego, skąd startuje jakiś określona linia autobusowa, albo innego, który jest przystankiem tylko dla określonych tramwajów.

**Variant** – jest to określony wariant danej linii, określający trasę, po której ma się poruszać pojazd. W najprostszym przypadku linia ma dwa warianty – po jednym dla każdego kierunku. Jednak często zdarza się, że linia dla jednego kierunku ma więcej niż jeden wariant. Przykładowo – gdy tramwaj zjeżdża do zajezdni, zazwyczaj musi on pojechać inną trasą niż zazwyczaj. Trasa podstawowa tramwaju i trasa, która prowadzi do zajezdni są dwiema różnymi wariantami trasy.

**Course** – oznacza pojedynczy kurs pojazdu komunikacji miejskiej.

Przykład: pojedynczym kursem będzie kurs pojedynczego tramwaju linii 33, który jedzie w kierunku Kielecka, a który rozpoczął swój bieg o godzinie 5:26.

Każdy z tych czterech elementów posiada w naszej aplikacji model oraz odpowiednią klasę. Klasa służy do przechowywania odpowiednich wartości każdego elementu, natomiast Model, będący singletonem, służy do tworzenia instancji tych klas i pobierania danych do nich.

Do modeli dołożyliśmy catche, które mają za zadanie zredukować pobieranie danych, które już były w pewnym momencie pobrane i przetworzone.

### 3.2.2 Baza danych

Dane do naszej aplikacji są tworzone na bazie danych rozkładowych udostępnianych przez Zarząd Transportu Publicznego w Warszawie (więcej informacji: <https://www.ztm.waw.pl/pliki-do-pobrania/dane-rozkładowe/>).

Dane rozkładowe to obszerny plik tekstowy, w którym są zawarte wszystkie informacje o transporcie publicznym w Warszawie.

Dane rozkładowe są przez nas pobierane, przetwarzane i wysyłane do zdalnej bazy danych Oracle. Nasz projekt zawiera w sobie część poświęconą tworzeniu tych danych oraz ich wysyłaniu, jednak kod ten nie jest wykonywany przy tworzeniu aplikacji.

Przetwarzanie danych polega na znajdowaniu odpowiednich sekcji w danych rozkładowych (sekcje są odpowiednio oznakowane w tym pliku). W czasie wykonywania programu. Każda tabela w naszej bazie danych ma oddzielną funkcję do wyszukiwania danych i wysłania ich. W czasie parsowania danych, wszędzie tam, gdzie się to da, dane pobierane z pliku są odpowiednio walidowane.

### 3.2.3 Algorytm wyszukiwania trasy

W celu wyszukiwania trasy został stworzony odpowiedni algorytm z użyciem algorytmu A\*.

Algorytm stara się znaleźć najlepszą trasę kilka razy, za każdym razem nieco poprawiając swoją heurystykę.

Poszczególne ponowne wyszukania trasy są podzielone na 2 etapy: pobierający i doskonalący. W etapie pobierającym graf wciąż jest rozszerzany o dodatkowe kursy z bazy danych – wykonywane jest maksymalnie 3 razy. W doskonalącym, algorytm operuje tylko na pobranych dotychczas danych i stara się na ich podstawie ulepszyć trasę – wykonywane jest maksymalnie 15 razy. Ponadto, każda próba ponownego wyszukania optimum jest ograniczona ilością iteracji.

Ilość kursów w bazie danych jest bardzo duża, więc pobranie ich wszystkich na początku programu zajęło by zbyt sporo czasu. Z drugiej strony, gdyby algorytm za każdym razem pobierał po jednym opisie 1 przystanku, ilość pojedynczych małych zapytań byłaby liczona w tysiącach. W naszym rozwiązaniu staraliśmy się znaleźć optymalny złoty środek pomiędzy czasem a ilością pobieranych zbędnych danych. Doszliśmy do wniosku, że najlepszym rozwiązaniem będzie podzielenie Warszawy na "chunki czaso-przestrzenne".

Każdy chunk oznacza przedział długości, wysokości geograficznych oraz czasu, tworząc swego rodzaju kostkę w przestrzeni 3-wymiarowej. Gdy algorytm napotyka na przystanek, który znajduje się poza chunkami uwzględnionymi w grafie, wykonuje polecenie SQL pobierające wszystkie potrzebne kursy dla każdego przystanku w tym obszarze i czasie. W ostatecznym rozwiązaniu podzieliliśmy Warszawę na 32x32 chunki przestrzenne i 32 chunki czasowe. Dzięki temu każdy z chunków jest reprezentowany przez 5 bitów identyfikatora.

Jednym z problemów tego rozwiązanie jest fakt, że algorytm nie patrzy daleko w przyszłość. Przyjeliśmy, że na jednym przystanku użytkownik będzie skłonny czekać maksymalnie około 1-2 godzin. Rezultatem tego jest to, że algorytm w pewnym sensie staje się "niecierpliwy". Woli wtedy iść na piechotę do następnego przystanku i liczy na to, że natrafi na jakiś autobus w przeciągu następnych 1-2 godzin i tak dalej. Znalaziona wtedy trasa może się w dużej mierze składać z przechodzenia z jednego przystanku na drugi.

### 3.3 Ograniczenia aplikacji

Nasza aplikacja posiada pewne ograniczenia, które pojawiły się w toku jej pisania:

- Nasza baza danych nie posiada informacji o kursowaniu metra; jest to spowodowane tym, iż dane rozkładowe udostępniane przez ZTM nie zawierają informacji o jego kursowaniu
- Wyszukiwanie trasy nie działa w czasie rzeczywistym – czasem nawet stosunkowo proste trasy mogą zająć parędziesiąt sekund do wyliczenia. Jest to spowodowane czasochłonnym pobieraniem potrzebnych danych z bazy. W szczególności widać to w przypadku pobierania kursów w okresie szczytu w centrum czy na śródmieściu.
- Wyszukiwanie trasy w nocy (w rejonach godzin 23 – 4) może być mało skuteczne. Jest to spowodowane ograniczoną ilością kursujących środków komunikacji miejskiej, co może powodować problemy spowodowane poruszoną wcześniej "niecierpliwością" algorytmu.
- W naszym projekcie zakładamy, iż każdy dzień jest dniem powszednim; nie da się wyszukać odjazdów np. w weekendy.

### 3.4 Podział pracy

Bartosz Han – stworzenie bazy danych, oraz stworzenie parsera do pobierania danych.

Mykhailo Marfenko – Stworzenie UI (linie oraz nawigacja).

Igor Matynia – Stworzenie algorytmu do wyszukiwania najlepszej ścieżki i stworzenie UI do wyświetlania przystanków.

## 4. Podsumowanie

W ramach tego projektu stworzyliśmy aplikację desktopową, która ma za zadanie wyświetlanie informacji o transporcie publicznym. Na przestrzeni 3 miesięcy poznaliśmy sposoby planowania i tworzenia aplikacji, budowania interfejsu użytkownika, oraz tworzenia aplikacji z użyciem danych pobranych poprzez kursory na bazę danych. Poznaliśmy także nowe możliwości, jakie daje Qt oraz biblioteka PySide, w tym tworzenie okienek z mapami. Projekt ten poszerzył naszą dotychczasową wiedzę o tworzeniu i budowaniu aplikacji i dał nam cenne doświadczenie w ich tworzeniu.