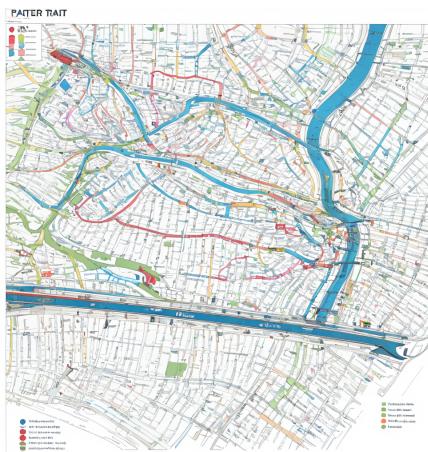


Symulacja Dyskretna Krakowskich lini tramwajowych

Dominik Breksa, Robert Barcik, Miłosz Góralczyk

22 grudnia 2023



1 Wstęp

Poniższa dokumentacja stanowi uzupełnienie / dopełnienie informacji przedstawionych w na laboratorium oraz dostępnych na prezentacjach w formacie *.pdf* wysyłanych zawsze na dzień przed zajęciami. Część informacji z prezentacji jest również zaktualizowana, aby oddać najświeższą wersję systemu.

1.1 Cel

Celem projektu było zebranie danych na temat kształtu i ruchu krakowskich linii tramwajowych, oraz symulacja ruchu oraz pasażerów, w celu ustalenia parametrów typu obłożenia tramwaju w zależności od czasu, linii, przystanku. Na podstawie symulacji powinna powstać wizualizacja przedstawionych danych.

Nasze założenia co do realizacji projektu obejmowały:

- Mapa:
 - Lokacje przystanków z Krakowa
 - Graf dróg z przejazdem dla tramwaju z ulic Krakowa
- Tramwaje:
 - Modele tramwajów:
 - Pojemność
 - Ilość członów / wagonów
 - Przynależność do linii
 - Przejездność / typ
- Trasa:
 - Ścieżka przez graf

- Czas między przystankami
- Czas oczekiwania na przystankach
- Przystanki:
 - Linie tramwajowe - punkty na trasie między którymi odbywa się transport
 - Obłożenie
 - Dokonanie metryki systemu

1.2 Plan

Aby móc zrealizować wcześniej wspomniany cel, postanowiliśmy go podzielić na mniejsze podzadania, celem ułatwienia implementacji oraz rozplanowania sobie czasu pracy podczas trwania semestru. Poniżej znajduje się szczegółowy plan działania:

Plan zawiera również daty poszczególnych spotkań na sali (laboratorium), podczas których przedstawialiśmy zrealizowane lub wtedy częściowo zrealizowane punkty. Można go też traktować jako kalendarz co kiedy było zrealizowane.

1. **(20 października 2023)** - Harmonogram i badanie źródeł.
 - Budowa wstępnego harmonogramu projektu z datami, wraz z zarysowaniem architektury.
 - Podział zadań na poszczególnych członków zespołu. Za co będą odpowiedzialni, ewaluacja umiejętności i znanych technologii.
 - Stworzenie prymitywnego stosu technologicznego w oparciu o umiejętności (szczegóły na końcu raportu).
 - Przeszukanie źródeł wiedzy, np. internetu, celem odnalezienia serwisów udostępniających dane przydatne podczas budowy modeli.
2. **(10 listopada 2023)** - Zebranie danych do plików masowych (*.pickle* / *.xlsx* / *.json*).
 - Jednoznaczne zdecydowanie się na metody obróbki danych i zaprojektowanie modelu (relacji między danymi) zapisu danych.
 - Zgromadzenie i bardzo prymitywne przetworzenie danych, aby umożliwić ich zapis.
 - Napisanie programów zbierających dane z zewnątrz.
 - Wizualizowanie relacji między danymi w postaci diagramów.
3. **(27 października 2023)** - Agregacja danych Alfa(α).
 - Wstępne wyodrębnienie i przystosowanie danych do zaprojektowanych relacji.
 - Przetworzenie formatu ze stron internetowych na bardziej użyteczny (np. konwersja jednostek, wyodrębnienie danych na części anomiczne).
4. **(10 listopada 2023)** - Agregacja danych Beta(β).
 - Przetworzenie danych, tak aby można je w pełni kompatybilne zapisać do plików masowych.
 - Pełne podzielenie danych na zadane relacje.
5. **(10 listopada 2023)** - Modelowanie grafu przystanków.
 - Przetworzenie danych w taki sposób aby móc jednoznacznie powiedzieć gdzie dany przystanek się znajduje.
 - Złączenie tych danych tak, aby móc ułożyć na pojedyncze przystanki (wierzchołki) linie (krawędzie).
 - Uwzględnienie faktu, iż nie zawsze ciąg przystanków ruch lini w kierunku zasadniczym jest taki sam jak w kierunku przeciwnym. Np. dla danej linii tramwaj w jednym kierunku jedzie przez *Filharmonia 02*, a wraca przez *Filharmonia 03* - przystanki różnią się znacznie lokalizacją o około 20 - 30 m, co musi być uwzględnione w modelu i nie pozwala na ich "zlepienie".

6. **(15 grudnia 2023)** - Modelowanie obiektu pojazdów - rodzaj / trasa / czas / pojemność.

- Wyodrębnienie z zgromadzonych danych informacji szczegółowych charakterystycznych dla danej jednostki tramwaju. Poniżej znajduje się szczegółowy opis:
 - **Rodzaj** - Charakterystyka modelu - marka, przeznaczenie, model w sensie np. "Krakowiak."
 - **Trasa** - Na jakiej linii dany tramwaj jeździ.
 - **Czas** - Czas w jakim operuje w trakcie dnia.
 - **Pojemność** - Ile dany tramwaj ma pojemności - ilość miejsc siedzących vs stojących.

7. **(17 listopada 2023)** - Modelowanie danych o położeniu.

- Stworzenie spójnego modelu rozkładu jazdy t.j. jaki konkretnie tramwaj przyjedzie na dany przystanek o której godzinie.
- Przedawnienie danych w formie dostępnej dla front-endu projektu.

8. **(1 grudnia 2023)** - Modelowanie obłożenia przystanków tramwajowych.

- Stworzenie modelu generowania populacji.
- Przeliczenie i zapisanie przetworzonych danych o populacji dla front-endu systemu.
- Zaprojektowanie i wyodrębnienie zmiennych wpływających na gęstość populacji.

9. **(15 grudnia 2023)** - Uwzględnienie zmiennych.

- Dostosowanie zmiennych i rozmiaru projektu, aby móc jak najlepiej oddać rzeczywistą sytuację na torach.

10. **(27 października 2023)** - Wizualizacja - szkielet mapy.

- Wizualizowanie pozycji przystanków względem siebie oraz mapy torów tramwajowych.
- Produkcja generowanego zdjęcia na bazie zebranych danych.

11. **(1 grudnia 2023)** - Wizualizacja - dodanie ruchu.

- Przedstawianie ruchu tramwajów na mapie wzduż rzeczywistych lokacji.
- Przyjęcie dyskretnego modelu ruchu - jedna tura, jedna minuta w rzeczywistości.
- Wizualizacji i przedstawienie modelu w *Godocie*.

12. **(1 grudnia 2023)** - Wizualizacja - uwzględnienie zmiennych i obłożenia.

- Uwzględnienie w wizualizacji danych o populacji.
- Przeliczenie czasu czekania na przystanków przez odpowiednich przechodniów.

13. **(15 grudnia 2023)** - Analiza danych, zebranie statystyk i oddanie projektu.

- Analiza danych w sensie weryfikacji poprawności modelu.
- Statystki np. ilość osób używających linii tramwajowych podczas symulacji.
- Przygotowanie dokumentacji aby móc finalizować oddanie projektu.

1.3 Efekty pracy

Aby móc w pełni przedstawić rozwój systemu poniżej przedstawimy efekty naszej pracy z okresu budowy projektu względem dat spotkań projektowych. Bardziej szczegółowy opis funkcjonalności znajduje się w następnej sekcji.

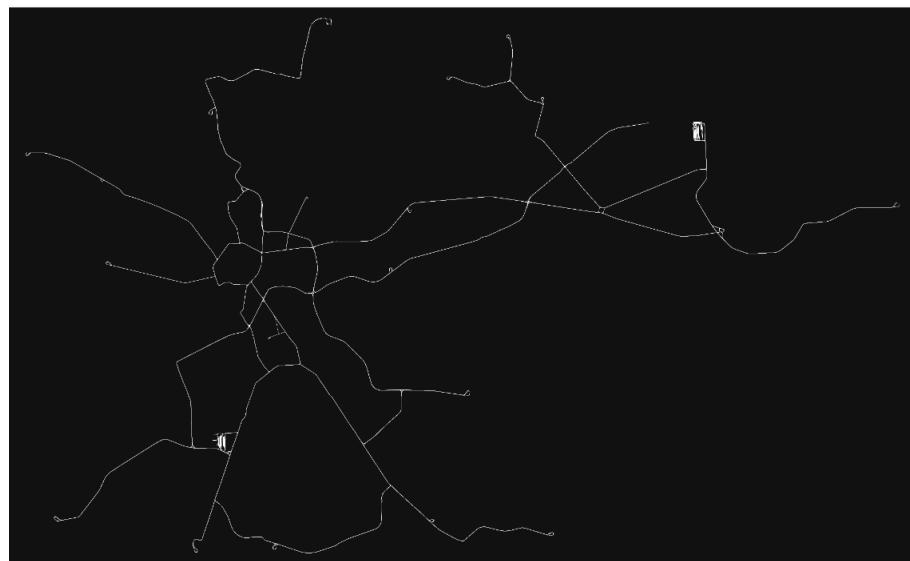
Sekcja będzie też zawierać wszelkie maści zdjęcia / filmy zawarte na wysyłanych wcześniej prezentacjach celem przedstawienia efektów.

1. (13 października 2023)

- Zebranie informacji
- Zebranie źródeł danych
- Utworzenie listy narzędzi
- Wybranie technologii projektu

2. (20 października 2023)

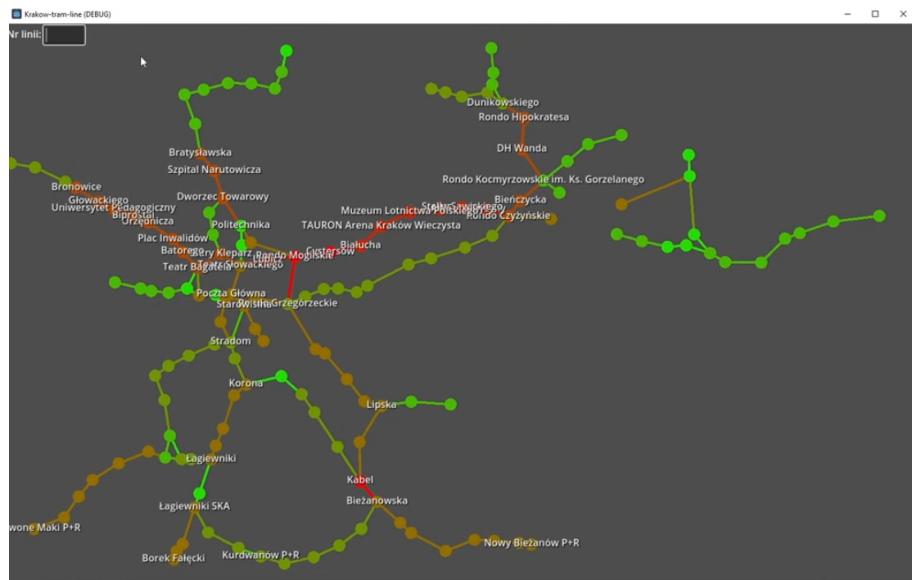
- Ustalenie harmonogramu działań na kolejne tygodnie
- Utworzenie programu web scrapera do uzyskania danych na temat przystanków
- Zebranie danych technicznych na temat modeli tramwajów linii krakowskich
- Zebranie danych na temat przystanków - nazwa, położenie, numer
- rozpoczęcie agregacji pozyskanych danych oraz eksport plików JSON z danymi na temat przystanków do wykorzystania w wizualizacji
- Utworzenie projektu wizualizacyjnego w GODOT i nałożenie przystanków oraz połączeń



Rysunek 1: Szkic mapy

3. (27 października 2023)

- Utworzenie programu web scrapera do uzyskania wszystkich danych na temat czasów przyjazdów tramwajów na wszystkie przystanki
- Agregacja uzyskanych danych w celu uzyskania pliku masowego zawierającego informacje o przemieszczaniu tramwajów
- Zakończenie procesu zbierania danych do plików masowych typu .pkl, .xlsx i .json
- Dodanie funkcjonalności do programu wizualizacyjnego: przełączanie widoczności względem linii, przystanki za modelowane jako reprezentacja obiektów, koloryzacja względem liczby połączeń



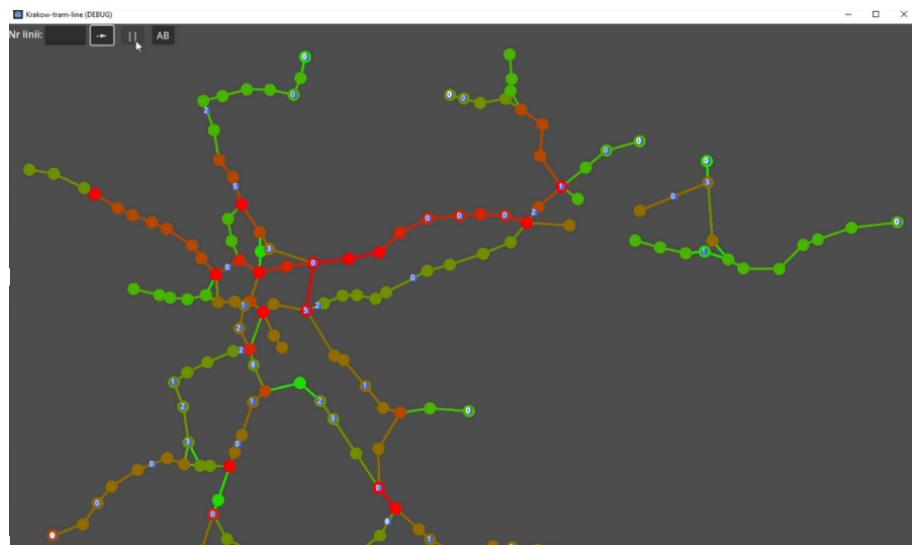
Rysunek 2: <https://drive.google.com/file/d/1RVnxwehyufaEP26DxnU-tRLupXcbPVbT/view?resourcekey>

4. (10 listopada 2023)

- dokończenie prac nad web scraperami oraz zebranie wszystkich danych na temat ruchu tramwajów
- dodanie modelu ruchu tramwajów po odpowiadającym ich liniach w programie wizualizacyjnym

5. (17 listopada 2023)

- Utworzenie klas reprezentujących przystanki i ludzi (Stop, Person), dodanie konstruktorów oraz potrzebnych metod
- Napisanie programu symulującego dyskretną generację pasażerów na przystankach linii tramwajowych, posiadających przypisane przystanki początkowy i końcowy, czas pojawienia, oczekiwana linia tramwajowa.
- Eksport listy wygenerowanych ludzi do pliku .json i przekazanie do wizualizacji.
- Zmiany w programie wizualizacyjnym poprzez dodanie ruchu pasażerów zaimplementowanymi liniami tramwajowymi.



Rysunek 3: Dodanie tramwajów

6. (1 grudnia 2023)

- Utworzenie funkcji modelującej prawdopodobieństwo generacji pasażerów, na podstawie rozkładu normalnego modelując prawdziwe zmiany przepływu w trakcie dnia
- Dobranie parametrów modelu w celu uzyskania prawdziwego modelu zapełnienia tramwajów.
- Implementacja zawijania tramwajów pod koniec linii w celu odzwierciedlenia prawdziwych warunków ruchu w modelu
- Dodanie opcji dostosowujących generację pasażerów

7. (15 grudnia 2023)

- Poprawa błędów oraz niedociągnięć wszystkich elementów
- optymalizacja czasu działania programu wizualizacyjnego
- dekoracja elementów wizualnych oraz dostosowanie wyświetlanych danych w celu optymalizacji UI

2 Architektura

Dokonując szczegółowej analizy planu doszliśmy, że nie możliwe będzie stworzenie jednego programu typu *"Monolit"*, w związku z czym przy projektowaniu systemu rozdzieliśmy go na trzy główne komponenty przetwarzania. Komponenty będą się komunikować poprzez system plików. Każda strona interfejsu będzie również zaznajomiona z ich schematem.

Logiczny schemat połączenia komponentów prezentuje się następująco:

$$\text{OutsideData} \longrightarrow_0 \text{Acquisition} \longrightarrow_1 \text{Aggregation} \longrightarrow_2 \text{Visualization} \quad (1)$$

1. **Outside Data** - Reprezentuje nasze źródło danych z zewnątrz systemu.
2. \longrightarrow_0 - HTTP / HTTPS
3. **Acquisition** - Komponent odpowiedzialny za zbieranie danych z poza systemu.
4. \longrightarrow_1 - Pliki *.xlsx*, *.pickle*
5. **Aggregation** - Komponent odpowiedzialny za przeliczenie i zwrócenie modelu na bazie zgromadzonych danych.
6. \longrightarrow_2 - Pliki *.json*
7. **Visualization** - Komponent odpowiedzialny za wizualizowanie modelu.

Model ten zawiera dużo plusów, ponieważ pozwala na indywidualizacje i odosobnienie rozwoju poszczególnych warstw systemu.

2.1 Zbieranie danych

Ta sekcja dedykowana jest komponentowi **Acquisition**. Łączy się on za pomocą protokołu HTTP z stronami i dzięki wykorzystaniu różnych API / Bibliotek, dokonuje zbiór danych, a następnie zapis ich na dysk.

2.1.1 Dane z API OpenStreetMap

Jedną z wykorzystanych przez system API jest **OpenStreetMap**. W naszym przypadku odpowiedni skrypt w *Pythonie* wysyła zapytania do bazy za pomocą biblioteki *OSMNX*, celem uzyskania dokładnych danych o rozkładzie torów i danych o położeniu przystanków.

Dane o rozkładzie torów są zapisywane w postaci grafu do pliku *.pickle*. Jest to niestety plik binarny, więc nie możemy go tutaj przedstawić, lecz jest on obecny w naszym repozytorium na platformie *Github* - link na dole sprawozdania.

Dane o położeniu uzyskiwane za pomocą zapytania, które zwraca obiekt *GeoPandasDataFrame*, co pozwala je zapisać w mnogiej ilości formatów. My wybierzemy *.pickle* - natywny dla Pythona sposób przechowywania danych w pamięci (prędkość i łatwość w przechowywaniu) oraz *.excel*, aby móc łatwo podglądać schemat danych.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	train	Y	Z	AA	AB	AC	AD	AE	AF	AG	ATwoj
1	element_tyd	osmid	name	network	workwid	operator	istor:wikidblic	transi	railway	ref	tram	geometry	url	wikidblic	layer	name:en	Q	wheelchair	source	informati	bench	shelter	note	X	train	Y	isused:rail	bus	ispublic	trai:old_name	discuss				
2	node	2,144+08	Wągrowia K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	125-1t	yes	POINT (20.0552148 50.0049232)																							
3	node	2,144+08	Kombinat K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	107-1t	yes	POINT (20.05500112 50.0091355)																							
4	node	2,144+08	Plac Solidarnosci i Wolnosci	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	108-1t	yes	POINT (20.0529435 50.0073321)																							
5	node	2,144+08	Teatr Lutej K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	119-1t	yes	POINT (20.0529435 50.0079056)																							
6	node	2,144+08	Rondo Lutej K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	118-2t	yes	POINT (20.0529435 50.0079056)																							
7	node	2,144+08	Bronisławska K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	101-2t	yes	POINT (20.0504583 50.0743017)																							
8	node	2,144+08	Os. Pasterska K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	101-2t	yes	POINT (19.989789 50.0740094)																							
9	node	2,144+08	AWI 02 K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	810-2t	yes	POINT (19.9623745 50.0588268)																							
10	node	2,144+08	Teatr Dzieci i Młodzieży K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	810-2t	yes	POINT (19.9623745 50.0588268)																							
11	node	2,144+08	Os. Tadeusza Kościuszki K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	811-2t	yes	POINT (19.9623821 50.0501733)																							
12	node	2,144+08	Francesco D'Adda K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	811-2t	yes	POINT (19.9623821 50.0501733)																							
13	node	2,144+08	Rondo 30-lecia Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	115-2t	yes	POINT (20.0505755 50.0073987)																							
14	node	2,144+08	Os. Kolejowa K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	104-2t	yes	POINT (20.0535576 50.0073351)																							
15	node	2,144+08	Os. 3 Maja K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	104-2t	yes	POINT (20.0535576 50.0073351)																							
16	node	2,144+08	Dł Wykona K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	135-1t	yes	POINT (20.0206008 50.0645139)																							
17	node	2,144+08	Rondo Kor Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	118-1t	yes	POINT (20.0254242 50.0600221)																							
18	node	2,144+08	Rondo Hip Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	162-1t	yes	POINT (20.0212424 50.11648050)																							
19	node	2,144+08	Plac Solidarnosci i Wolnosci K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	103-2t	yes	POINT (20.0319333 50.0745211)																							
20	node	2,144+08	Pasta K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	147-1t	yes	POINT (20.0319333 50.0798326)																							
21	node	2,144+08	Os. Pastwiskowa K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	148-1t	yes	POINT (20.0319386 50.1019885)																							
22	node	2,144+08	Podgórze K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	149-1t	yes	POINT (19.9899467 50.0645024)																							
23	node	2,144+08	Mistrzow K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	149-1t	yes	POINT (19.9899467 50.0645024)																							
24	node	2,144+08	Mistrzow Główny Tunel 03 K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	139-1t	yes	POINT (19.993344 50.116503513)																							
25	node	2,771+08	Dworzec Główny Tunel 03 K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	103-1t	yes	POINT (19.9477449 50.0682455) /2 Main Station Tunnel																							
26	node	2,888+08	Nowy Bieg K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	702-1t	yes	POINT (20.0206008 50.070702)																							
27	node	2,888+08	Protekcja K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	702-1t	yes	POINT (19.9477449 50.0682455) /2 Main Station Tunnel																							
28	node	2,888+08	Podgórze K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	642-1t	yes	POINT (19.9477449 50.0682455) /2 Main Station Tunnel																							
29	node	2,906+08	Plac Solidarnosci i Wolnosci K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	584-1t	yes	POINT (19.9477449 50.0682455) /2 Main Station Tunnel																							
30	node	2,906+08	Witosa D1 K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	583-1t	yes	POINT (19.9477449 50.0682455) /2 Main Station Tunnel																							
31	node	2,906+08	Borek Fałk K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	583-1t	yes	POINT (19.9477449 50.0682455) /2 Main Station Tunnel																							
32	node	3,216+08	Borek Fałk K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	562-1t	yes	POINT (19.9477449 50.0682455) /2 Main Station Tunnel																							
33	node	3,216+08	Borek Fałk K Komunikie	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	562-1t	yes	POINT (19.9477449 50.0682455) /2 Main Station Tunnel																							
34	node	3,216+08	Brama 4 Skomunike	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	213-1t	yes	POINT (20.0173284 50.0712386)																							
35	node	3,216+08	Brama 4 Skomunike	Q1784404	MPK Krak	G1178029	stop_point	tram	stop	215-1t	yes	POINT (20.0171241 50.0691427)																							

Rysunek 4: OSMNX dane o przystankach. W faktycznym modelu będziemy używać w zasadzie tylko kolumny *name* oraz *geometry*.

2.1.2 Dane z prostego web-scraperra

Dane z tej strony: [\[Link\]](#) możemy w łatwy sposób pozyskać używając biblioteki *pandas*, ponieważ na stronie są przedstawione jako statyczne tabele HTML.

Pierwszą tabelę pominiemy całkowicie, bo dane w niej zawarte można agregować z dwóch następnych tabel na stronie (jaki model ma jaką linie?).

Druga powie nam, jaki tramwaj ma model konstrukcji, a także jaką pełni funkcje i z jakiej zajezdni pochodzi. Ponieważ kod tramwaju ma następujący schemat: "XY000", gdzie pierwsza litera to kod zajezdni, druga to typ / przeznaczenie tramwaju np. czy to tramwaj techniczny i trzy ostatnie cyfry to unikalny identyfikator maszyny.

A	B	C	D	E
	id	m_depo_cctram_code		name
2	0	101 H	W	E1
3	1	104 H	W	
4	2	118 H	W	E1
5	3	121 H	W	E1
6	4	126 H	W	E1
7	5	129 H	W	
8	6	130 H	W	
9	7	132 R	W	E1
10	8	135 H	W	
11	9	140 H	W	E1
12	10	142 H	W	E1
13	11	143 H	W	E1
14	12	144 H	W	
15	13	145 H	W	
16	14	146 H	W	E1
17	15	149 R	W	E1
18	16	151 H	W	
19	17	155 R	W	
20	18	164 H	W	E1
21	19	213 R	Z	
22	20	216 R	Z	
23	21	219 R	Z	
24	22	226 R	Z	105N
25	23	231 R	Z	105N
26	24	234 R	Z	
27	25	237 R	Z	
	--	--	--	--

Rysunek 5: Zapisane dane w programie Excel. Od lewej: index, id tramwaju - trzy cyfrowy kod, kod zajezdni, kod przeznaczenia, i nazwa modelu. Część danych brakuje, więc taki id tramwajów nie braliśmy pod uwagę podczas tworzenia modelu.

Trzecia tabela z tej strony posłużyła nam jako wyznacznik jak id tramwaju ma się do jego linii, na której pracuje. Dane typu *Last seen*, czy *Position* nie mają dla nas specjalnego znaczenia ponieważ nie są za specjalnie aktualne dla każdego tramwaju.

A	B	C	D	E	F	G
	line	last_seen	latitude	longitude	direction	id
2	0	21 now	50,07281	20,11681	Os.Piastów	101
3	1	4 68 days ago	50,08547	20,06299	Zaj. Nowa I	104
4	2	21 now	50,07691	20,03094	Pleszów	118
5	3	44 now	50,06527	19,95219	Dworzec T	121
6	4	44 now	50,07356	20,01576	Dworzec T	126
7	5	21 16 hours ago	50,07745	20,05722	Kombinat	129
8	6	44 3 days ago	50,07713	20,05611	Kombinat	130
9	7	44 now	50,07563	19,93969	Dworzec T	132
10	9	21 31 days ago	50,07322	20,11757	Pleszów	135
11	10	16 now	50,0949	19,99542	Mistrzejow	140
12	11	16 now	50,0688	20,06993	Bardosa	142
13	12	49 now	50,01618	20,0124	Nowy Bież	143
14	13	44 2 days ago	50,07832	20,06056	Kombinat	144
15	14	44 10 days ago	50,07889	20,0625	Kombinat	145
16	15	44 now	50,07433	20,005	Kopiec Wa	146
17	16	44 now	50,06982	20,06691	Kopiec Wa	149
18	17	21 14 days ago	50,07269	20,03917	Kombinat	151
19	18	16 3 hours ago	50,06746	20,05962	Bardosa	155
20	19	44 now	50,06659	19,93896	Kopiec Wa	164
21	20	52 3 hours ago	50,05935	19,94226	Os.Piastów	213
22	21	13 72 days ago	50,01536	20,02385	Nowy Bież	216
23	22	13 63 days ago	50,07711	19,90146	Bronowice	219
24	23	52 now	50,10208	20,01177	Czerwone I	226
25	24	52 now	50,06504	19,95052	Os.Piastów	231
26	25	13 4 days ago	50,03026	19,93403	Nowy Bież	234
27	26	52 3 days ago	50,01869	19,88896	Czerwone I	237
28	27	13 now	50,01775	19,90073	Nowy Bież	242

Rysunek 6: Zapisane dane w programie Excel. Wszystkie kolumny są zatomizowane - np. jest wycięty id z nazwy tramwaju, ponieważ posłuży man jako klucz obcy do połączenia z poprzednią tabelką.

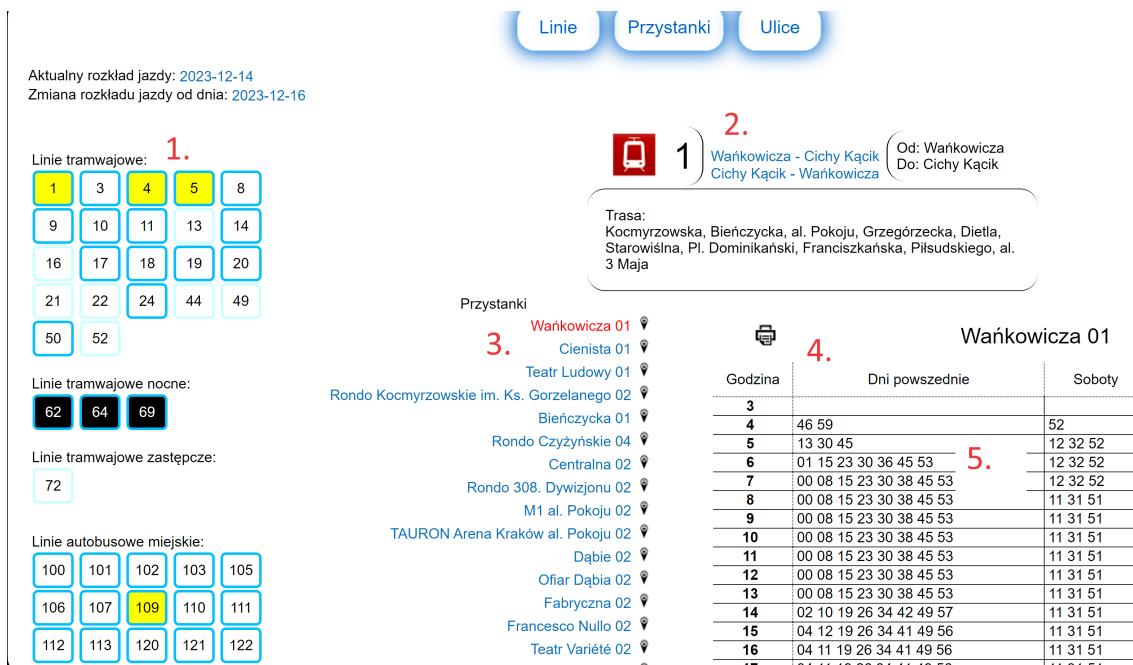
2.1.3 Dane z wyrafinowanego web-scraperra

Aby móc zebrać dane o odjazdach i przyjazdach tramwajów na danej linii musielibyśmy skorzystać z serwisu MPK z tego linku [\[Link\]](#), nie jest możliwe zbieranie danych starą metodą, ponieważ za każdym użyciem linku użytkownik jest przekierowywany na stronę główną.

Z pomocą przychodzi użycie biblioteki do testowania user interface (UI) stron internetowych *Selenium* wraz z *chrome drierverami* do testowania chroma. Ta biblioteka w połączeniu z *pandas* pozwoli nam odczytać dane dotyczące rozkładu jazdy Krakowskich tramwai.

Finalnie program zbierze około 100 000 rekordów godzin i minut, i całość zajmie około 10 - 15 min czasu liczenia w zależności od internetu.

Schemat działania jest następujący:



Rysunek 7: Zdjęcie ekranu z narysowanymi znacznikami liczbowymi przedstawiające kroki zbierania danych przez webscrapera.

1. Prze-iteruj przez wszystkie linie dzienne (kliknij w nie).
2. Dla każdej linii przejdź przez wszystkie kierunki, na jakich jeżdżą tramwaje (kliknij w kierunek).
3. Następnie kliknij w każdy przystanek dla konkretnego kierunku i lini.
4. Zbierz dane z tabelki za pomocą *pandas* - *pandas.read_html(...)*.
5. Wyłuskaj konkretne godziny ze stringa tabelki dla dni powszednich.

A	B	C	D	E	F
line	name	direction	hour	minute	
0	1	Wańkowic:START_Zapisano w: ten komputer	46		
1	1	Wańkowic:START_ENI	4	59	
2	1	Wańkowic:START_ENI	5	13	
3	1	Wańkowic:START_ENI	5	30	
4	1	Wańkowic:START_ENI	5	45	
5	1	Wańkowic:START_ENI	6	1	
6	1	Wańkowic:START_ENI	6	15	
7	1	Wańkowic:START_ENI	6	23	
8	1	Wańkowic:START_ENI	6	30	
9	1	Wańkowic:START_ENI	6	36	
10	1	Wańkowic:START_ENI	6	45	
11	1	Wańkowic:START_ENI	6	53	
12	1	Wańkowic:START_ENI	7	0	
13	1	Wańkowic:START_ENI	7	8	
14	1	Wańkowic:START_ENI	7	15	
15	1	Wańkowic:START_ENI	7	23	
16	1	Wańkowic:START_ENI	7	30	
17	1	Wańkowic:START_ENI	7	38	
18	1	Wańkowic:START_ENI	7	45	
19	1	Wańkowic:START_ENI	7	53	
20	1	Wańkowic:START_ENI	8	0	
21	1	Wańkowic:START_ENI	8	8	
22	1	Wańkowic:START_ENI	8	15	
23	1	Wańkowic:START_ENI	8	23	
24	1	Wańkowic:START_ENI	8	30	
25	1	Wańkowic:START_ENI	8	38	
26	1	Wańkowic:START_ENI	8	45	
27	1	Wańkowic:START_ENI	8	53	
28	1	Wańkowic:START_ENI	9	00	

Rysunek 8: Zdjęcie danych. "START-END" oznacza w kierunku zasadniczym, a "END-START" w przeciwnym.

Podczas trwania semestru był rozwijany jeszcze jeden webscrapper zbierający dane z tej strony [Link]. Zbierał dane dotyczące aktualnego opóźnienia na linach. Nie zdecydowaliśmy się go rozwijać ponieważ za bardzo komplikował by model oraz aby móc zebrać dostatecznie dużo danych musiał by ciągle działać przez kilka godzin, odbierając tym samy możliwość korzystania z serwisu postronnym użytkownikom.

2.1.4 Dane zgromadzone ręcznie

Dane na temat modeli tramwajów, jak ich gabaryty, masa, ilość miejsc pasażerskich, zostały zebrane do pliku .xlsx na podstawie udostępnionych danych na stronie [Link] uzupełnionych w miejscach wybrakowania informacji danymi z Wikipedii oraz forów tramwajowych.

A	T	B	T	C	T	D	T	E	T	F	T	G	T	H	T	I	T	J	T	K	T	L	T	M	T	N	T	O	T	P	T
name	Wielkość	Rodzaj	Długość	Szerokość	Wysokość	Rozstaw	Rozsta	Masa w Masa	Miejsca ogółem	Miejsca siedzące	Wysokość	Liczba silników	Moc silnik	Średnica kot																	
105 N	Wagon silni wysokopodł	13.50	2.40	3.06	6	1.90	16,500	25,500	125	23	0.89	4	40	0.65																	
N8S-NF	Wagon silni wysokopodł	26.08	2.30	3.65	6.20	1.80	35,300	50,000	207	46	0.28	2	125	0.67																	
NGT6 (1)	Wagon silnik niskopodłog	26	2.40	3.46		1.80	30,000	46,650	182	76	0.36	4	125																		
NGT6 (2)	Wagon silnik niskopodłog	26	2.40	3.46		1.80	30,000	46,650	182	76	0.36	4	125																		
NGT6 (3)	Wagon silnik niskopodłog	26	2.40	3.46		1.80	30,000	46,650	182	76	0.36	4	125																		
E1	Wagon silni wysokopodł	19.71	2.20	3.20	6	1.80	24,000	35,000	151	33	0.90	2	150	0.65																	
C3	Wagon poj wysokopodł	14.10	2.20	3.20	6	1.90	11,580	21,500	138	28	0.89	0	0	0.65																	
GT8N	Wagon silni wysokopodł	26.20	2.42	3.46	6.20	1.80	35,000	49,000	200	51	0.88	2	150	0.67																	
405N	Wagon silni wysokopodł	40.57	2.35	3.75	5.60	1.90	63,500		297	57	0.37	12	50																		
EU8N	Wagon silni wysokopodł	26.62	2.31	3.24	6	1.80	33,410		158	48	0.29	2	190	0.69																	
NGT8	Wagon silni niskopodłog	32.83	2.40	3.60		1.80	42,000	62,400	225	77	3.60	4	105																		
2014N	wagon silni niskopodłog	42.83	2.40	3.60	8.60	1.80	64,036		322	93	0.30	6	105																		
Stadler	Wagon silni niskopodłog	33.40	2.40	3.60		1.80	59,000		222	83	0.30	4	105																		
Stadler Tango	Wagon silni niskopodłog	33.40	2.40	3.60		1.80	59,000		222	83	0.30	4	105																		
Stadler Tango	Wagon silni niskopodłog	33.40	2.40	3.60		1.80	59,000		222	83	0.30	4	105																		

Rysunek 9: Dane modeli

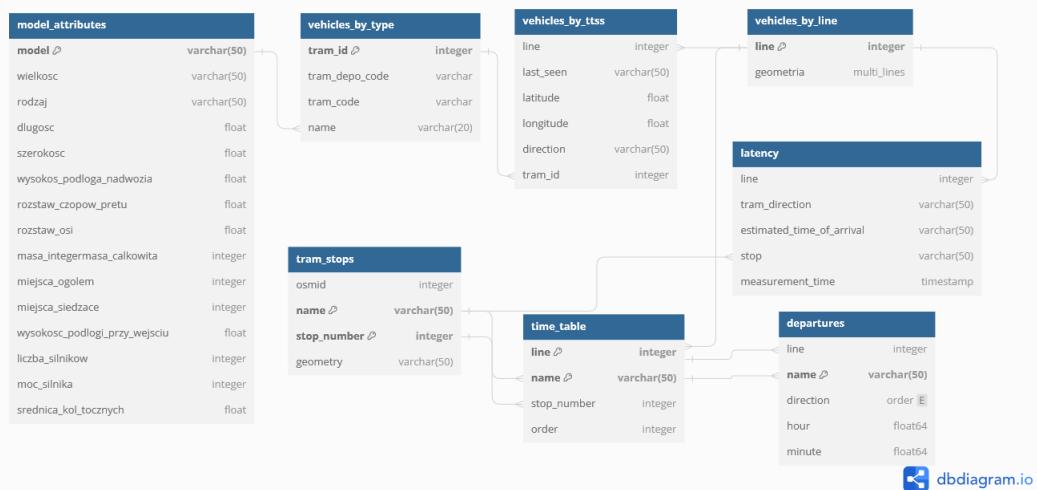
Dodatkowo ręcznie zostały zebrane dane na temat modeli przemieszczającymi się konkretnymi liniami tramwajowymi, jako że te dane nie były ogólnie dostępne w żadnym gotowym formacie, zostały one zebrane poprzez asocjację linii z modelami tramwajowymi na nich kursujących, wpisując je do pliku **.xlsx**.

	A	B
1	merge	name
2	HW151	E1
3	RW155	E1
4	RZ231	105N
5	RZ255	105N
6	RF302	GT8N
7	RF304	GT8N
8	RF307	GT8N
9	RF320	GT8N
10	RF322	GT8N
11	RF323	GT8N
12	RF326	GT8N
13	HL402	EU8N
14	HL405	EU8N
15	HL412	EU8N
16	HL416	EU8N
17	HL427	EU8N
18	HL433	EU8N
19	HL436	EU8N
20	HL438	EU8N
21	HL439	EU8N
22	HK451	N8C-NF

Rysunek 10: Id - model

2.1.5 Schemat danych

Całość możemy przedstawić za pomocą następującego diagramu:



Rysunek 11: Diagram przedstawiający całkowity schemat danych zebranych przez ten komponent.

Tabele na schemacie odpowiadają następującym zbiorom danych:

1. **model_attributes** - Dane zgromadzone ręcznie.
2. **vehicles_by_type** - Dane z prostego webscrappera - tabela dwa.
3. **vehicles_by_ttss** - Dane z prostego webscrappera - tabela trzy.
4. **vehicles_by_line** - Dane z prostego webscrappera - tabela dwa, ale ograniczone tylko do linii.

5. **latency** - Dane z nie kontynuowanego webscrappera (nie zebrane).
6. **departures** - Dane z wyrafinowanego webscrappera z użyciem Selenium.
7. **time_table** - Dane z wyrafinowanego webscrappera z użyciem Selenium.
8. **tram_stops** - Dane z API OSMNX.

2.2 Modelowanie

Zdecydowaliśmy się na format *.JSON*, ponieważ silnik GODOT, ma wbudowane biblioteki do przetwarzania tego formatu, a jednocześnie można w nim przedstawić fragmenty relacji przedstawionych powyżej.

Skrypty w pythonie generują dane przetworzone z plików *.pickle*, następnie zwracają pliki *.json*. Kluczowe słowo to skrypty, bo nie zdecydowaliśmy się na centralny plik tworzący model, a na wiele mniejszych implementujących daną gałąź modelu. Poniżej opiszemy dokładnie jak to działa.

2.2.1 Tworzenie populacji

Z poziomu *Pythona* została utworzona funkcja generująca na podstawie agregowanych danych na temat przystanków, pasażerów czekających na wybrane dyskretnie linie tramwajowe. W każdym kroku, której długość wynosi sekundę, program iteruje po wszystkich przystankach, w celu próby wygenerowania czekającego pasażera. Próba polega na porównaniu losowej liczby z przedziału $(0,1)$ z prawdopodobieństwem uzyskanym poprzez przekazanie czasu symulacji do funkcji *get_probability(time)* zwracającą wynik złożenia funkcji rozkładu normalnego, z dobranymi parametrami aby jak najlepiej symulować rzeczywisty rozkład zagęszczenia ruchu w trakcie dnia, ze szczytami w godzinach 8 i 17. Szansa generacji człowieka na przystanku wachodzi wraz z czasem między **5%** a **30%**

```
def two_peak_normal_distribution(x, x1, x2, amplitude1, amplitude2, sigma1=1, sigma2=1):
    peak1 = amplitude1 * np.exp(-(x - x1)**2) / (2 * sigma1**2)
    peak2 = amplitude2 * np.exp(-(x - x2)**2) / (2 * sigma2**2)
    return peak1 + peak2

def tram_probability (time):
    amplitude1 = 0.3
    amplitude2 = 0.3
    amplitude_mean = (amplitude1 + amplitude2) / 2
    rush_1 = 8 * 60
    rush_2 = 16 * 60
    baseline = two_peak_normal_distribution(time, rush_1, rush_2, 0.3, 0.3, 100, 100)
    result = np.power(baseline / amplitude_mean, 0.8) * amplitude_mean

    return result
```

Rysunek 12: uzyskiwanie prawdopodobieństwa

W momencie sukcesu generacji wywołany zostaje instancja obiektu człowieka, któremu dyskretnie przypisany zostaje przystanek z listy przystanków osiągalnych z przystanku początkowego. Zostaje jednocześnie zapisana instancja czasu, w której dany pasażer pojawił się na przystanku, kierunek w którym się porusza, jak i unikatowe wygenerowane imię i nazwisko.

Pod koniec przeprowadzenia w ten sposób symulacji, uzyskana lista przechowująca tak wygenerowanych ludzi zostaje następnie odpowiednio zakodowana, oraz przekazana, wraz ze swoimi parametrami **person_id**, **name**, **stop_start**, **stop_end**, **line**, **time**, **direction**, do funkcji pakującej wszystkie dane do pliku *JSON* w celu wykorzystania przez komponent wizualizacji w programie *GODOT*.

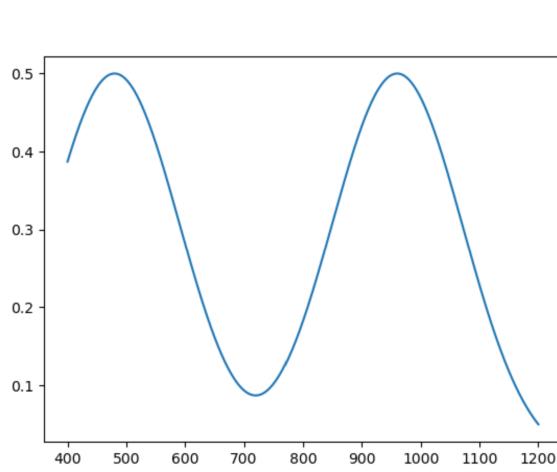
```

410537      {
410538        "id": 53520,
410539        "name": "Jenelle Milano",
410540        "start_stop": "Dworzec Towarowy 03",
410541        "end_stop": "Nowy Kleparz 02",
410542        "line": "44",
410543        "time": 1199,
410544        "direction": "END_START"
410545      },
410546      {
410547        "id": 53521,
410548        "name": "Ingunna Hertberg",
410549        "start_stop": "Rondo Hipokratesa 01",
410550        "end_stop": "Teligi 02",
410551        "line": "9",
410552        "time": 1200,
410553        "direction": "END_START"
410554      },
410555      {
410556        "id": 53522,
410557        "name": "Shane Weston",
410558        "start_stop": "Kurdwanów P+R 08",
410559        "end_stop": "Mały Płaszów P+R 01",
410560        "line": "11",
410561        "time": 1200,
410562        "direction": "START_END"
410563      },
410564      {
410565        "id": 53523,
410566        "name": "Sonny Hester",
410567        "start_stop": "Szpital Narutowicza 02",
410568        "end_stop": "Muzeum Lotnictwa Polskiego 01",
410569        "line": "5",
410570        "time": 1200,
410571        "direction": "END_START"
410572      },

```

Rysunek 13: Wygenerowani ludzie

Przy standardowych parametrach, cykl dnia generuje statystycznie około **90000** unikalnych reprezentantów pasażerów (agentów) wszystkich linii tramwajowych Miejskiego Przedsiębiorstwa Komunikacyjnego w Krakowie.



Rysunek 14: funkcja prawdopodobieństwa

2.2.2 Umiejscawianie przystanków

Aby móc wizualizować graf przystanków musieliśmy dostarczyć następujące informacje warstwie wyższej:

- Nazwa przystanku - aby móc się potem do niego odwoływać np. w modelu ludzi.

- Pozycja w przestrzeni 2D (współrzędne geograficzne).
- Number lini, która jeździ na danym przystanku.
- Numer przystanku w kolejności względem danej linii.
- Kierunek w którym dany przystanek jest zwrócony.

```

": [
  [
    4,
    26,
    "Kombinat 01",
    [
      20.0650412,
      50.0791355
    ],
    "END_START"
  ],
  [
    ...
  ]
]

```

Rysunek 15: Przykładowe wyjście z modelu obrazujące schemat. Od góry kolejność przystanku na linii, numer linii, nazw przystanku, pozycje geograficzne, kierunek w którym przystanek stoi.

Możemy w ten sposób skonstruować graf przystanków, ponieważ za wierzchołki grafu można przyjąć pozycje przystanków w przestrzenni. Następnie za poszczególne połączania z wierzchołka, należy wziąć wszystkie linie wychodzące z danego przystanku w kolejności $n + 1$ względem jego kolejności (n). Tym sposobem front-end tworzy graf skierowany wszystkich przystanków.

Metodologia generacji jest następująca. Jest to znany z SQL INNER JOIN pomiędzy tabelami tram_stops, a time_table przy użyciu name jako klucz główny operacji w obu tabelach. Reszta to posortowanie wyjścia w tak, aby były według rosnącego order oraz przygotowanie rezultatu do przedstawienie w formacie .json.

2.2.3 Tworzenie rozkładu jazdy dla konkretnych tramwai

Przy rozdzielaniu rozkładu jazdy na poszczególne tramwaje posłużyliśmy się następującym autorskim algorytmem:

1. Na początku tworzy pusty słownik out, który będzie przechowywał wynikowe dane. Ustalamy również godziny startu (**7:00**) i zakończenia (**19:59**). Godziny są dobrane tak, aby nie nadwyręzać wizualizacji, a jednocześnie aby móc mieć dostatecznie duże pole do manewru do przetworzenia transportu populacji w scybach godzinowych.
2. Na początku filtrujemy dane o odjazdach (departures), aby uwzględnić tylko te, które mieszczą się w określonym przedziale czasowym. Dodatkowo, dla każdego odjazdu oblicza wartość absolutną czasu w minutach od północy według proporcji $(g * 60 + m)$.
3. Algorytm przechodzi przez każdy rekord z tabeli vehicles_by_type. Dla każdego wiersza tworzy ekstraktu-je identyfikator (identifier) i linie (line). Przechodzi dla każdego tramwaju.
4. Następnie, algorytm wyszukuje przystanki w ramce danych time_table, które odpowiadają danej linii i kierunkowi jazdy. Jeżeli nie ma takich przystanków, algorytm przechodzi do następnego tramwaju (np. jeżeli tramwaj porusza się tylko na liniach nocnych).
5. Algorytm sortuje przystanki według kolumny order i wybiera pierwszy przystanek (first_stop). Następnie wyszukuje w ramce danych departures czas odjazdu z tego przystanku.

6. Algorytm wchodzi w pętlę, w której dla każdego przystanku na linii wyszukuje godziny odjazdu (suitable_hours), które są późniejsze niż poprzedni czas odjazdu i je usuwa, ponieważ tylko jeden tramwaj może o danej porze nadjechać na dany przystanek. Jeżeli nie ma takich godzin, algorytm kończy pętlę. W przeciwnym razie, algorytm aktualizuje prev_absolute_time i dodaje informacje o odjeździe do słownika wyjściowego.
7. Po przejściu przez wszystkie przystanki w kolejności zasadniczej, algorytm zmienia przejeżdża przez przeciwny kierunek ruchu linii i powtarza kroki 6-7.
8. Algorytm kontynuuje te kroki, aż przejdzie przez wszystkie linie tramwajów. Na końcu zwraca słownik out, który zawiera informacje o odjazdach dla każdej linii tramwaju.

Wyjściem algorytmu jest następujący obiekt `.json`. Kluczem w tym słowniku obiektu jest id tramwaju i lista kolejnych przystanków na które ma zajechać. Jest podana nazwa, godzina, która jest zawsze większa na kolejnych przystankach oraz kierunek ruchu. Lista ta się zawiąza - tramwaj jeździ w kółko od 7:00 do 19:59, bez przerwy, chyba, że skończy się dostępny przedział czasowy.

```

1   "101": [
2     [
3       [
4         "Pleszów 01",
5         7.0,
6         19.0,
7         "START-END"
8       ],
9       [
10        "Koksochemia 02",
11        7.0,
12        20.0,
13        "START-END"
14      ],
15      [
16        "Meksyk 02",
17        7.0,
18        21.0,
19        "START-END"
20      ],
21      [
22        "Brama nr 5 02",
23        7.0,
24        22.0,
25        "START-END"
26      ],
27      [
28        "Giedroycia 02",
29        7.0,
    ]
  ]
]
  
```

Rysunek 16: Przykład wyjścia z modelu.

2.2.4 Wzbogacanie danych o konkretnych tramwajach

Aby front end wiedział na jakiej linii jadą jakie tramwaje oraz jakie mają charakterystyki (ilość miejsc siedzących - aby móc sprawdzić, czy tramwaj nie jest przepełniony), musimy mu dostarczyć dane z tabel: model_attributes, vehicles_by_type i vehicles_by_ttss.

Zrobimy to wykorzystując dwa Inner Joiny, oraz obcinając niepotrzebne kolumny, celem przyspieszenia przetwarzania symulacji. Najpierw pomiędzy tabelami: model_attributes i vehicles_by_type używając z lewej strony kolumny model, a z prawej kolumny name. Następnie pomiędzy rezultatem i vehicles_by_ttss, używając tram_id.

```
{  
    "id":101,  
    "tram_depo_code":"H",  
    "tram_code":"W",  
    "name":"E1",  
    "merge":"HW101",  
    "Wielkość":"Wagon silnikowy dwuczłonowy",  
    "Rodzaj":"wysokopodłogowy",  
    "Długość":19.71,  
    "Szerokość":2.2,  
    "Wysokość nadwozia":3.2,  
    "Rozstaw czopów skrętu":6.0,  
    "Rozstaw osi":1.8,  
    "Masa własna":24000,  
    "Masa całkowita":35000.0,  
    "Miejsca ogółem":151,  
    "Miejsca siedzące":33,  
    "Wysokość podłogi przy wejściu":0.897,  
    "Liczba silników":2,  
    "Moc silnika":150,  
    "Średnica kół tocznych":0.654,  
    "line":21  
},  
{  
    "id":118,
```

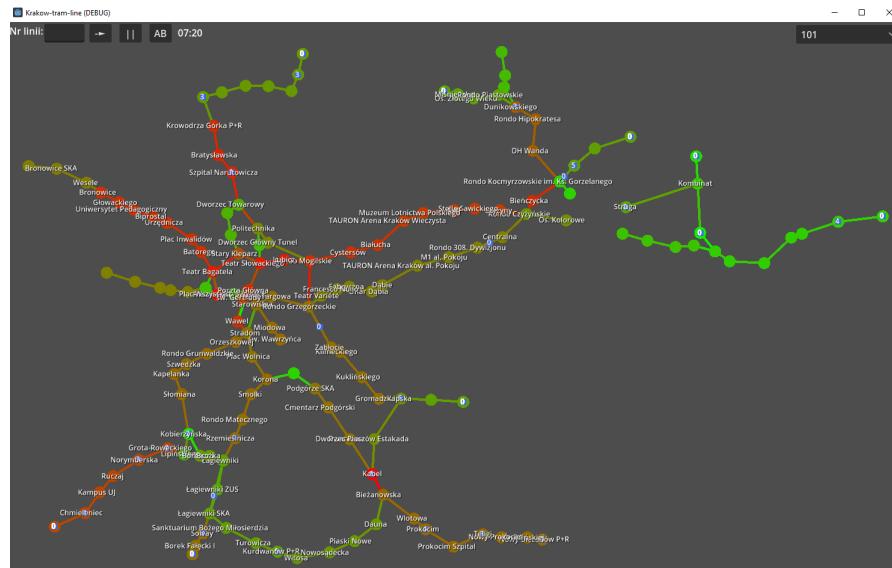
Rysunek 17: Przykładowy rekord ilustrujący jak wygląda rezultat operacji.

2.3 Wizualizacja

Do wizualizacji użyliśmy silnika Godot 4, do którego przenieśliśmy istotne dane za pomocą plików json. Dane dotyczyły przystanków, tramwajów, rozkładów jazdy oraz wygenerowanych ludzi.

2.3.1 Podstawowa wizualizacja

Za pomocą wymienionych wcześniej danych nakładane są przystanki, połączenia między nimi (w postaci prostych odcinków), jak i same tramwaje. Linie przyjmują kolory, które pokazują natężenie różnych linii w danym obszarze, gdzie czerwony to największe zatężenie. Nad tramwajami wyświetla się bezpośrednio liczba mówiąca o ilości obecnych pasażerów. Tramwaje poruszają się po mapie zgodnie ze swoim rozkładem jazdy. Po drodze liczba pasażerów oczywiście się zmienia.



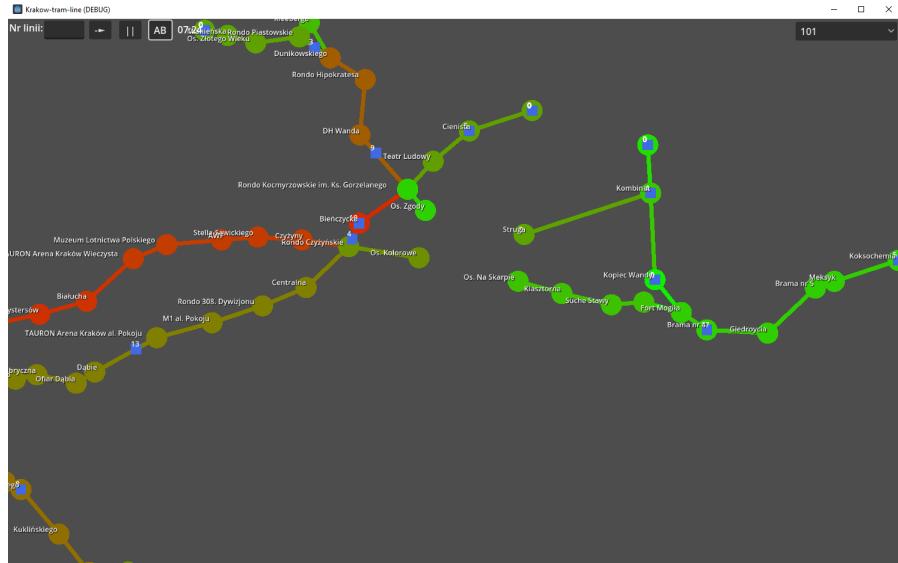
Rysunek 18: Wygląd symulatora zaraz po włączeniu

2.3.2 Podpisy przystanków

Każdy przystanek przypisany ma swoją nazwę bezpośrednio przy sobie. W celu utrzymania czytelności projektu, niektóre z nazw wyświetla się dopiero po przybliżeniu widoku.

2.3.3 Przesuwanie i zmiana rozmiaru mapy

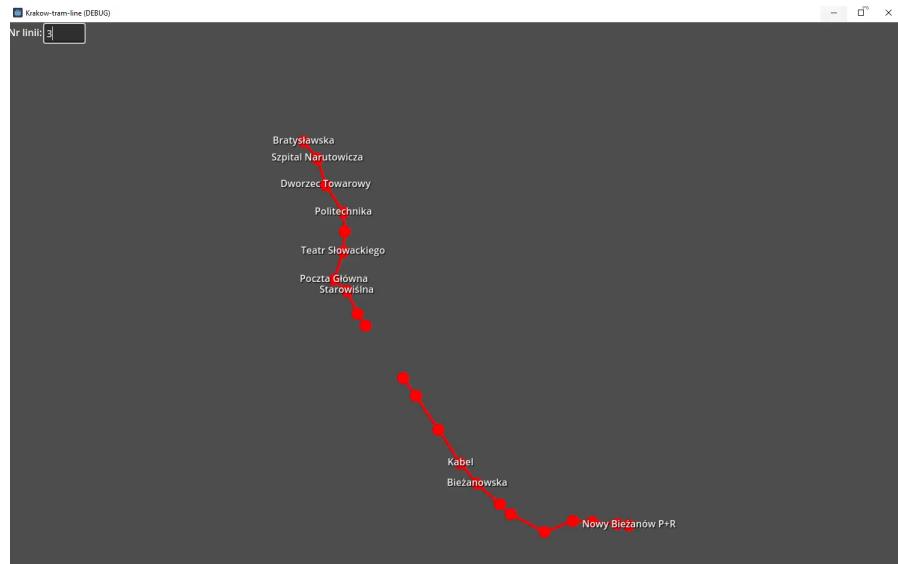
Po przytrzymaniu przycisku myszy mapę można przesuwać. Ograniczeniami są tu oczywiście miejsca, w których żaden jej element nie byłby już widoczny. Za pomocą środkowego przycisku myszy mapę można również przyblizać i oddalać. Przy odpowiednich rozmiarach zmieniają się również np. rozmiary czcionki w projekcie.



Rysunek 19: Przybliżenie. Wyświetlane są nazwy przystanków, które wcześniej wyświetlane nie były, wszystkie mają odpowiednio dopasowany rozmiar czcionki

2.3.4 Wyświetlenie konkretnej linii

W lewym górnym rogu ekranu istnieje pole, w który możemy wpisać konkretny numer linii. Jeśli to zrobimy, wyświetli nam się mapa tylko tej linii. Po wpisaniu 0 otrzymamy wszystkie linie

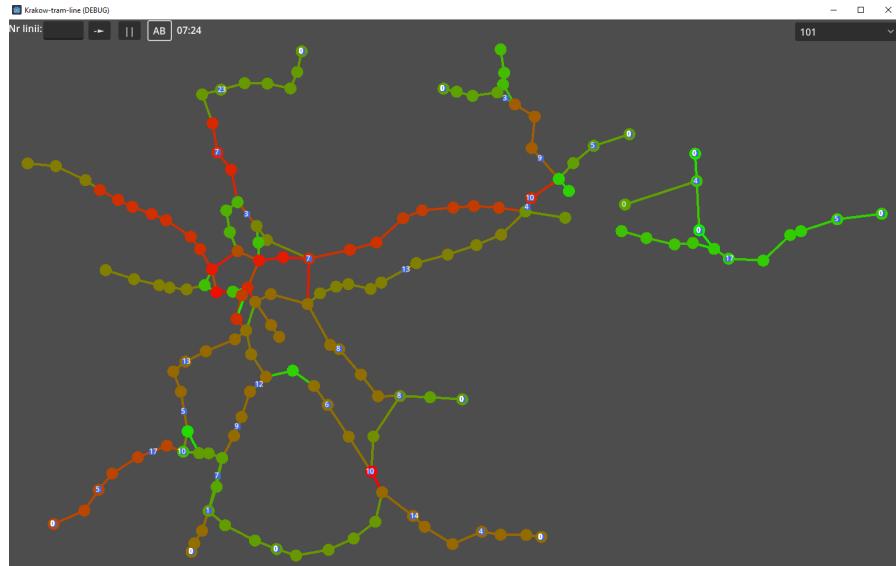


Rysunek 20: Wyświetlenie tylko jednej linii

2.3.5 Dodatkowe przyciski

Zaimplementowano kilka dodatkowych przycisków pomocniczych do specyficzniejszych symulacji. Mamy przycisk następnego kroku (strzałki), który wczytuje następną minutę symulacji, przycisk pauzy wstrzymujący ją (nie neguje działania pierwszego przycisku, ale automatyczna animacja zostaje wstrzymana) i przycisk napisu, wyłączający lub włączający wszystkie etykiety przystanków

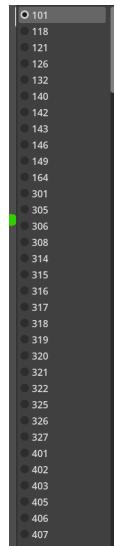
na mapie. Ponadto obok tych przycisków wyświetla się czas, który obecnie jest symulowany przez program.



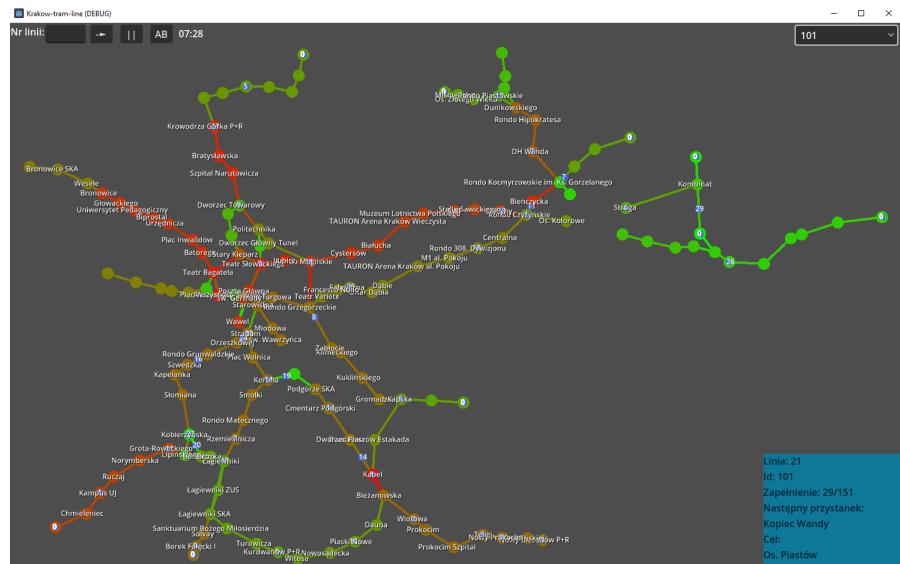
Rysunek 21: Prezentacja przycisków: Wciśnięte są przyciski pauzy i napisów

2.3.6 Dane tramwaju

W prawym górnym rogu ekranu wyświetla się menu, w którym możemy wybrać sobie tramwaj na podstawie jego id. Po wybraniu w prawym dolnym rogu wyświetli się pole z informacjami o danym tramwaju: jego linia, id, zajęte miejsca, następny przystanek oraz obecny cel.



Rysunek 22: Menu wyboru ID



Rysunek 23: Cały wygląd z wyświetlonym w rogu polem z danymi wybranego tramwaju

3 Walidacja i Statystyki

Walidacja danych projektu została znacząco utrudniona z powodu negatywnej odpowiedzi o udzielenie danych i statystyk przez Politechnikę. Walidację jak i samo modelowanie mogliśmy oprzeć jedynie o empiryczne obserwacje oraz garść statystyk zebranych samodzielnie. Najnowsze oficjalne statystki znalezione w internecie pochodzą z czasów przed pandemią, i opisują jedynie nieaktualną liczbę kursów bądź pokrycie kosztów MPK. Aby pozyskać dane poniższej tabelki, każdy z nas na przestrzeni kilku dni zanotował i sfotografował liczbę osób na przystanku tuż przed przyjazdem tramwaju, zdjęć nie umieszczały oczywiście. **art. 81 UstPrAut**

Przystanek	Kierunek	Godzina	Liczba oczekujących
Wesele	do centrum	06:48	7
Wesele	do centrum	12:02	3
Wesele	do centrum	17:56	10
Wesele	do centrum	19:59	1
Rzemieślnicza	od centrum	13:31	3
Rzemieślnicza	od centrum	15:53	5
Rzemieślnicza	od centrum	18:27	8
Teligi	od centrum	07:50	4
Teligi	od centrum	09:15	6
Teligi	od centrum	11:35	5
Teligi	od centrum	13:00	3

Rysunek 24: Manualnie zebranie danych

Na podstawie danych przybliżyliśmy wartości funkcji modelującej zagęszczenie pasażerów, jak i dostosowaliśmy parametry aby ustawić statystyczną liczbę pasażerów w trakcie dnia na 90000 osób. Ilość kursów tramwajów w trakcie dnia, z pozyskanych danych została wyliczona na 2516 kursów w okresie od 7:00 do 19:59. Na tych danych oparliśmy nasz model.

4 Technologie

4.1 Silnik i obliczenia

- Python 3.12 oraz biblioteki
 - numpy

- pandas
- geopandas
- matplotlib
- json
- osmnx
- random
- pyrosm
- selenium

- GODOF

4.2 Zapis i odczyt danych

- Excel
- JSON
- .pkl

5 Kontrybucje

- Dominik Wojciech Breksa
 - **Fokus: Zebranie i przetwarzanie danych.**
 - Utworzenie programów web-scraperów.
 - Agregacja danych do plików masowych (Z webscrapperów i z modelu).
 - Napisanie algorytmu przydzielania tramwajom rozkładu jazdy.
 - Uzyskanie większości danych na których projekt bazował.
- Robert Andrzej Barcik
 - **Fokus: wizualizacja projektu.**
 - Całokształt wizualizacji danych.
 - Utworzenie UI projektu .exe.
 - Optymalizacja czasu symulacji.
- Miłosz Bogdan Góralczyk
 - **Fokus: przetwarzanie danych i modelowanie symulacji.**
 - Zebranie ręczne danych.
 - Agregacja danych do plików masowych.
 - Utworzenie klas reprezentujących elementy projektu.
 - Napisanie symulacji pasażerów.
 - Modelowanie funkcji probabilistycznej pasażerów.

6 Archiwum danych oraz projektu

Cały kod, zebrane dane, oraz programy znaleźć można na publicznym repozytorium projektu <https://github.com/ForNeus57/krakow-tram-lines>

folder **src/ktl** jest podzielony na sekcje **acquisition**, **aggregation**, **model**, **visualisation**, które przechowują kod związany z odpowiadającymi kolumnami projektu. folder **data** zawiera wszystkie zebrane, oraz zagregowane pliki wykorzystane przy modelowaniu projektu.

Pełen opis instalacji wszystkich programów wchodzących w skład projektu znajduje się w naszym README.md na głównej gałęzi projektu.