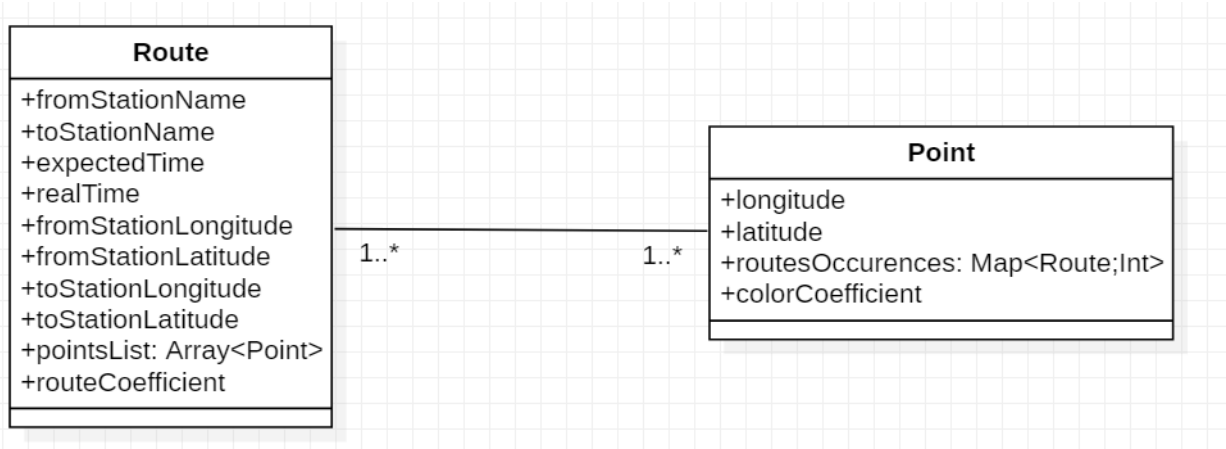
1 Model danych



- **Route**
  - *fromStationName, toStationName* - nazwy stacji wrocławskiego roweru miejskiego, z API wrocławskiego roweru miejskiego
  - *fromStationLongitude, fromStationLatitude, toStationLongitude, toStationLatitude* - współrzędne stacji początkowej i końcowej, z API wrocławskiego roweru miejskiego
  - *expectedTime* - oczekiwany czas trasy, z API ORS
  - *realTime* - czas rzeczywisty trasy, jako średnia ze wszystkich przejazdów na tej trasie, z API wrocławskiego roweru miejskiego
  - *pointsList* - lista punktów trasy
  - *routeCoefficient* - współczynnik niepewności trasy
- **Point**
  - *longitude, latitude* - współrzędne geograficzne
  - *routesOccurrences* - mapa, która mapuje trasę na liczbę jej wystąpień
  - *colorCoefficient* - współczynnik koloru

2 Algorytm

Algorithm 1: Algorytm wyznaczania GeoJSON

Result: GeoJSON z pokolorowanymi punktami według intensywności wykorzystania tras przez rowerzystów

Input: Lista stacji rowerów ze współrzędnymi i nazwami

$routes$: Array<Route>;
$points$: Array<Point>;

initializeRoutes($routes$);
/* Każda trasa jest reprezentowana przez osobny obiekt, inicjalizujemy nazwy stacji oraz współrzędne, czyli atrybuty fromStationName, toStationName, fromStationLongitude, fromStationLatitude, toStationLongitude, toStationLatitude */

foreach *r in the routes* **do**

getRealTime(r);
/* zapytanie do API Wrocławskiego roweru miejskiego o średni czas na tej trasie (funkcja agregująca avg), ustawia pole Route (realTime) */

getRealTimeAndRoutePoints(r);
/* Zapytanie do ORS o czas na tej trasie i współrzędne punktów na tej trasie, ustawia pola Route (expectedTime i pointsList) */

routeOccurrences = getRoutesOccurrences(r);
/* Zapytanie do API Wrocławskiego roweru miejskiego o częstość występowania tej trasy (funkcja agregująca count) */

countCoefficient(r);
/* Oblicza współczynnik niepewności dla trasy ze wzoru podanego poniżej (patrz pod algorytmem) i ustawia jego wartość (routeCoefficient) */

updatePointsRoutes(r, routeOccurrences);
/* Iteruje po punktach trasy r i jeśli punktu nie ma w tablicy to go do niej dodaje i wpisuje trasę do mapy tras wraz z jej liczbą wystąpień, a jeśli jest w tablicy to dodaje tylko trasę do mapy tras tego punktu wraz z liczbą wystąpień */

updatePointsColorCoef(r, routeOccurrences);
/* Iteruje po punktach trasy r, do każdego z punktów do współczynnika koloru dodaje wartość routeOccurrences * routeCoefficient */

end

foreach *p in the points* **do**

assignColorToPoints();
/* Każdemu z punktów przypisuje kolor na podstawie wartości pola colorCoefficient oraz tablicę tras i liczby ich wystąpień na podstawie mapy, która jest atrybutem */

end

$$W_n = \frac{T_R - |T_R - T_O|}{T_R}$$

$T_R -$ czas rzeczywisty

$T_O -$ czas oczekiwany

Wspolczynnik ma wartosci od 0 do 1, 1 gdy czas rzeczywisty jest rowny czasowi oczekiwanemu