**Coś a la wstęp**

Świat się zmienia. I jak jeszcze do niedawna ogólnie rozumiany postęp był postrzegany przez pryzmat nowinek technicznych ułatwiających – w mniejszym bądź większym stopniu – codzienne czynności, tak od pewnego czasu niezwykle ważną kwestią stała się ekologia. Wkrada się ona do niemal każdego aspektu naszego życia: od naklejek na sprzęcie AGD informującym o efektywności energetycznej danego urządzenia, przez powszechne i coraz bardziej rygorystyczne normy emisji silników spalinowych, po ograniczenia w spalaniu opału w prywatnych piecach. Fala „pro eko” nabiera mocy z każdym rokiem, co jest jak najbardziej zrozumiałe w kontekście zmian do jakich dochodzi na Ziemi z powodu działalności energetycznej człowieka. Jednym z ogromnej liczby postulatów ruchu proekologicznego jest ograniczenie ruchu samochodowego, zwłaszcza tego na krótki dystans. Co więc w zamian? Odpowiedź jest prosta: niemal w stu procentach ekologiczny rower. Ten wymaga jednak odpowiedniej infrastruktury oraz nieco zmiany w podejściu do kwestii transportu.

Jednak pewien trend jest widoczny już do dłuższego czasu: rower staje się coraz popularniejszym środkiem transportu. Wpływ na owe zjawisko ma wiele zróżnicowanych czynników i nie wszystkie można wytłumaczyć powszechną modą na ekologię. Z pewnością dokładają się również takie aspekty jak oszczędność pieniędzy (nie licząc kupna roweru, jazda na nim w zasadzie nic nie kosztuje), a w nierzadkich przypadkach i czasu, łatwość w dotarciu bezpośrednio pod cel podróży (samochód, bądź komunikacja miejska nie zawsze to umożliwiają), czy niewątpliwe walory zdrowotne. Nic więc dziwnego, że rowerzystów-amatorów przybywa w niezwykle szybkim tempie i to nie tylko tych jeżdżących dla przyjemności w swoim czasie wolnym. Tendencję tę można zauważyć zwłaszcza w miastach o bardzo dużym natężeniu ruchu samochodowego. Wynika to z bardzo prostego faktu: w miastach odległości są względnie niewielkie, a powstające na głównych ciągach komunikacyjnych zatory w ruchu pojazdów mechanicznych, powodują znaczne wydłużenie czasu podróży przy użyciu tychże. Stąd rower całkiem często okazuje się wybawieniem – środkiem transportu, który zaoszczędza niemało czasu i niejako przy okazji daję okazji do regularnej aktywności fizycznej. Dlatego też widok dużej ilości stojaków rowerowych ustawionych przy siedzibach dużych firm nikogo nie dziwi. Wręcz przeciwnie – nie raz pracownicy jakiegoś przedsiębiorstwa potrafią wprost wypomnieć brak takiego udogodnienia.

Miastem, w którym coraz wyraźniej widać zainteresowanie transportem rowerowym, z pewnością jest Kraków, co zdają się zauważać także lokalne władze, sukcesywnie rozwijając infrastrukturę rowerową. W wyniku tych prac, zaczyna ona coraz dokładniej oplatać niemal wszystkie dzielnice małopolskiej metropolii. Dzięki temu, obecnie możliwa jest podróż rowerem z jednego krańca Krakowa na drugi, poruszając się w większości po trasach stworzonych specjalnie dla jednośladów. Oczywiście istnieją od tej reguły drobne wyjątki, jednakże większość dużych skupisk ludzkich (gdzie na co dzień mieszka i sypia sporo mieszkańców) oraz dzielnic usługowo-handlowych czy też biznesowych (gdzie owi mieszkańcy pracują lub udają się na zakupy bądź relaks) jest ze sobą nad wyraz poprawnie skomunikowana pod względem rowerowym.

Jednakże wielu krakowskich rowerzystów – często nieświadomie – nie wykorzystuje potencjału drzemiącego w infrastrukturze rowerowej swego miasta. Nierzadko poruszają się zwykłymi drogami w towarzystwie samochodów, co stanowi zagrożenie dla bezpieczeństwa zarówno samego rowerzysty, jak i kierowcy. Istnieje również niemałe grono ludzi zastanawiając się nad ową formą transportu, które jednak rezygnuje z tego pomysłu właśnie z powodu niechęci do jazdy głównymi arteriami miasta, lawirowaniu pomiędzy pojazdami mechanicznymi i wdychaniu generowanych przez nich spalin. Sytuacja ta wynika z dynamicznie rozwijających się sieci dróg rowerowych w mieście, a w konsekwencji trudności w nadążeniu za owymi zmianami. Przeciętny mieszkaniec Krakowa nie jest po prostu świadom istnienia wielu udogodnień przeznaczonych dla bicykli.

Właśnie z powyższych powodów podjęto się w niniejszej pracy stworzenia aplikacji do wyznaczania optymalnej trasy rowerowej oraz nawigowaniu po niej. Ma ona na celu nie tylko pomóc przeciętnemu mieszkańcowi Krakowa w przemieszczaniu się po mieście przy pomocy roweru, ale także uzmysłowić jak duże możliwości drzemią w infrastrukturze rowerowej tego miasta. Wystarczy rzut oka na wyznaczoną trasę, by przekonać się, iż z punktu A do punktu B można dostać się bez konieczności jazdy po ruchliwych ulicach, czy zatłoczonych chodnikach, co zresztą jest prawnie zabronione. Działanie owej aplikacji opiera się na oficjalnej bazie danych krakowskiego ZIKiT-u, co – przynajmniej w teorii – gwarantuje ich poprawność i aktualność.

**Rozdział 1 – coś ogólnie o temacie**

**Nawigacje rowerowe**

Stworzenie sprawnie działającej nawigacji dla rowerzystów nie jest rzeczą prostą – podstawowym wyzwaniem jest tutaj stworzenie odpowiedniego silnika, na którym owa nawigacje będzie się opierać. Ów silnik musi w sposób szybki i powtarzalny wyznaczać najbardziej optymalne trasy w oparciu o ścieżki rowerowe lub odpowiednie drogi. Już na wstępie należy uzmysłowić sobie fakt, iż zadanie to wymaga przede wszystkim oparcia się na sprawdzonych, poprawnych i aktualnych danych opisujących infrastrukturę. I właśnie problem ich dostępności jest największym z jakim trzeba się zmierzyć.

Infrastruktura rowerowa jest specyficzną dziedziną. Nie da się w jej przypadku – tak jak odbywa się to przy okazji dróg samochodowych – łatwo i jednoznacznie określić wszystkich odcinków, które są możliwe do przemierzenia. Ścisłe wyznaczenie każdych dostępnych dla rowerzystów ciągów komunikacyjnych jest w zasadzie możliwe jedynie w niezwykle przejrzyście i mądrze zaprojektowanym mieście, a – nie ma się co oszukiwać – takie ideały w zasadzie nie istnieją. W normalnych warunkach urbanistycznych nie sposób pomyśleć o wszystkich możliwych przejazdach, skrótach i odcinkach. Równie trudną kwestią jest nieunikniona konieczność decyzji, które drogi można uznać za przyjazne rowerzystom, a które nie. Te pierwsze teoretycznie powinny odznaczać się małą ruchliwością, drugie zazwyczaj są głównymi arteriami miasta. I tutaj pojawia się konflikt interesów – lokalne, drobne ulice są wprawdzie wręcz stworzone do przemieszczania się nimi za pomocą bicykli, jednak ich celem jest jedynie doprowadzać do ulic większych.

W Krakowie urzędnicy wpadli na całkiem udany pomysł – stworzenie bazy informacji o infrastrukturze rowerowej powierzono instytucji, która jest odpowiedzialna za ogólną infrastrukturę miasta, czyli Zakładowi Infrastruktury Komunalnej i Transportu. Dzięki tej idei powstała obszerna, aktualna baza danych, która z powodzeniem może posłużyć do tworzenia nawigacji dla rowerzystów. Niestety, jak dotąd zainteresowanie bazą danych autorstwa ZIKiT-u było dość nikłe, co – ze względu na pewną jej unikalność – może tylko dziwić.

W bazie tej zostały wyznaczone nie tylko wszystkie ścieżki rowerowe, ale także drogi przyjazne rowerzystom, które ZIKiT zdecydował się zdefiniować jako te z ograniczeniem prędkości dla samochodów do 40 km/h. Bardzo trudno o rzetelny osąd takiego, a nie innego wyboru. Prawdopodobnie wynikało to z chęci stworzenia jak najbardziej kompletnej sieci możliwych dróg i być może przy wyborze np. tylko dróg z ograniczeniem prędkości do 30 km/h, na mapie miasta znalazłoby się zbyt dużo „białych plam”.

Gdy pominie się problem precyzyjnych danych o infrastrukturze rowerowej, zrobienie nawigacji dla amatorów kolarstwa nie wydaje się już zadaniem szczególnie nietypowym. Nie jest to do końca prawdą. Wprawdzie praca ta faktycznie sprowadza się do stworzenia prostych algorytmów odnajdujących najkrótszą trasę z możliwych, a później nawigowaniu po niej, jednak dochodzi do tego konieczność ustalenia wag i korelacji pomiędzy infrastrukturą stricte rowerową, a drogami przyjaznymi kolarzom. Wartości te muszą zostać dobrane tak, by wyznaczona trasa odznaczała się jednocześnie optymalną długością i satysfakcjonującym udziałem ścieżek rowerowych. Wymaga to, by autorzy takiej aplikacji byli zaznajomieni ze specyfiką danego miasta, w przeciwnym wypadku wyznaczenie równowagi pomiędzy powyższymi typami tras może nastręczać bardzo dużych problemów.

**Istniejące na rynku rozwiązania**

Aktualnie rynek nawigacji rowerowych w Polsce jest dość ubogi. Istnieją oczywiście profesjonalne urządzenia nazywane „nawigacją rowerową”, jednak są to produkty dość drogie i skierowane raczej do co najmniej średnio-zaawansowanych rowerzystów. Inną sprawą jest, ze tego typu nawigacje starają się za wszelką cenę unikać zatłoczonych tras i większych miast, siłą rzeczy więc nie nadają się do nawigowania po metropoliach. Istnieją także serwisy internetowe pomocne dla rowerzystów – jak chociażby naviki, czy Mapy Google – jednakże żadna z nich nie potrafi rowerzysty nawigować, a do tego wpadają w proste pułapki projektów o zbyt dużej skali jak niedokładność danych i czasami absurdalne dobory tras. Rozwiązania te często nie uwzględniają pasów jezdni wydzielonych specjalnie dla rowerzystów, potrafią też prowadzić użytkownika wzdłuż wielkich arterii miasta, bądź przez schody, czy inne rozwiązania urbanistyczne trudne do pokonania rowerem. Czasem też zdarza im się nadgorliwość – przejechanie dosłownie paru metrów wzdłuż bardziej ruchliwej ulicy jest zamieniane na odcinek prowadzący niezwykle okrężną drogą. Do tego dochodzi jeszcze kwestia aktualności - wiele ze ścieżek powstałych w ciągu ostatniego roku, czy dwóch, w ogóle nie została odnotowana w tego typu rozwiązaniach.

Za powyższy stan rzeczy z pewnością odpowiedzialne są wymienione wcześniej problemy przy tworzeniu nawigacji rowerowej, czyli trudna dostępność precyzyjnych danych. Jednakże nie jest to z pewnością jedyna przyczyna, gdyż w ślad za Krakowem starały się pójść takie miasta jak Warszawa, Wrocław, czy Gdańsk, które także stworzyły własne bazy danych o infrastrukturze rowerowej. Nie wiedzieć jednak czemu w żadnym z tych miast nie poskutkowało to powstaniem aplikacji wyszukującej trasy.

**Rozwiązania na platformy mobilne.**

Zdrowy rozsądek podpowiada, że programy wspomagające rowerzystów powinny występować na urządzeniach mobilnych w bardzo dużej ilości – w końcu zarówno rowery, jak i wszelkiego rodzaju elektroniczne gadżety, mają tę samą grupę docelową, którą są głównie ludzie młodzi. Dodatkowo wystarczy spojrzeć na ilość dostępnych aplikacji nawigujących kierowców, by dojść do wniosku, że jeśli się poruszać, to tylko ze smartfonem w ręku. Jednakże tylko samochodem, gdyż rzeczywistość nie ma w tym przypadku nic wspólnego z logiką – znalezienie dedykowanej dla rowerzystów aplikacji nie jest łatwe, a te na które można się natknąć, albo są znane ze swych pełnoprawnych i wielce niedoskonałych w danym temacie wersji (Mapy Google), albo są produktem świeżym, jednak bazującym na niesprawdzającym się w rowerowym świecie API OpenStreetMaps (Locus Mapa).

**Powiedzmy, że rozdział 2 czyli opisy środowisk**

Do stworzenia kompletnego oprogramowania potrzebna jest zarówno znajomość konkretnych języków programowania, jak i architektury urządzeń, na które ów produkt jest tworzony. Bez tej wiedzy wynik prac może być wielce niesatysfakcjonujący, a w najgorszym wypadku nawet niekompatybilny z docelową platformą. Dlatego też zdecydowano się przedstawić niżej ogólną specyfikę narzędzi, czy środowisk, które okazały się niezbędne to wytworzenia omawianej aplikacji, a także użytych algorytmów.

**Urządzenia mobilne**

Poprzez termin „urządzania mobilne” w niniejszej pracy rozumie się telefony klasy smartfon – czyli posiadające autonomiczny, pełnoprawny system operacyjny – oraz tablety. Wśród takich produktów w obecnym czasie dominują te oparte o systemy Android oraz iOS.

W porównaniu do owych dwóch systemów udział pozostałych jest marginalny, wręcz ocierający się o granicę błędu statystycznego. Z tego też powodu opisywana w niniejszej pracy aplikacja powstała właśnie na wspomniane wyżej dwa systemy – na Androida i iOS-a. Ich popularność jest bezapelacyjna, co ułatwiłoby rozpropagowanie ewentualnego produktu.

**- System Android**

System operacyjny Android to najbardziej rozpowszechniony system operacyjny na urządzenia mobilne. Jest on częściowo otwartym systemem, opartym na jądrze Linux, którego początki sięgają końcówki roku 2005. Wtedy to też, kupiona niespełna pół roku wcześniej przez Google, niewielka firma Android rozpoczęła pracę nad stworzeniem systemu operacyjnego przeznaczonego dla urządzeń mobilnych. Jednak na pierwsze wydanie systemu przyszło czekać aż do 21 października 2008 roku. Niewiele później zostało wprowadzone do sprzedaży pierwsze urządzenie napędzane przez ów system. Był to, wyprodukowany przez tajwańską firmę HTC, telefon o nazwie *Dream*. Pracował on pod kontrolą Androida w wersji 1.0, posiadał jednordzeniowy procesor firmy Qualcomm oraz 192 MB pamięci RAM. Jak na czasy, w których powstał, był sprzętem całkiem mocnym i wartym uwagi.

Pomimo niepozornych początków popularność systemu Android rosła w bardzo szybkim tempie – tylko do końca 2009 roku wprowadzono do sprzedaży ponad 30 urządzeń z tym systemem. Poskutkowało to faktem, iż w owym czasie do użytkowników trafiło około 8 milionów egzemplarzy urządzeń z systemem firmy Google. Przyrost był więc zauważalny gołym okiem. Zdobycie większości udziałów w rynku systemów mobilnych nie zajęło Androidowi wiele czasu. Jego silnej ekspansji nie wytrzymały konkurencyjne systemy jak Symbian, czy Windows Phone. Jedynym nieniszowym oponentem dla Androida pozostał system iOS, tworzony przez firmę Apple.

Amerykańska korporacja upodobała sobie, by nazwy co bardziej ważnych wersji systemu zapożyczać od określeń różnorakich słodyczy, w porządku alfabetycznym. I tak wersja 1.0 nosiła nazwę *Applepie* (szarlotka), 1.1 *Banana Bread* (chleb bananowy), 1.5 *Cupcake* (babeczka), a 1.6 *Doughnut* (pączek). Na chwilę obecną najnowszą wersją systemy Android jest ta oznaczona numerem 9.0 i nazwą Pie(ciasto).

Wspólny system operacyjny to jednak jedyna cecha, którą dziedziczą smartfony z Androidem – różnorodność dostępnego na rynku sprzętu, opartego o wyżej wymieniony system, jest naprawdę ogromna. Można tutaj spotkać przedstawicieli wielu firm, które implementują w swych produktach różne podzespoły odmiennych producentów (od Qualcomma, przez Mediateka i Broadcomma, aż po autorskie układy Huaweia, Samsunga, czy Xiaomi). W wyniku takiego stanu rzeczy, urządzenia z Androidem występuję we wszystkich segmentach rynku, począwszy od propozycji budżetowych, aż po najbardziej wydajne i prestiżowe pozycje premium.

Jednakże takiego rozdrobnienie skutkuje trudnościami w odpowiednim zoptymalizowaniu systemu, jak i aplikacji pod niego pisanych. Z tego też powodu do Androida przyległa łatka systemu powolnego i niezwykle zasobożernego, co w wielu przypadkach okazuje się prawdą. Urządzenia z owym systemem niezwykle prosto doprowadzić do wyraźnego spadku wydajności, a nawet nieszczególnie wymagające czynności na mocnej konfiguracji sprzętowej potrafią mieć niezwykle niską responsywność, rzędu sekundy, czy dwóch.

Środowiskiem programistycznym systemu Android jest lekko zmodyfikowany język Java, wzbogacony o dodatkowe biblioteki. Z tego też powodu tzw. „próg wejścia” dla osób zaznajomionych z Javą był bardzo niski, co skutkowało bardzo szybkim spopularyzowaniem platformy wśród Twórców oprogramowania. Miało (i nadal ma) to jednak i swoje gorsze strony – jest to chociażby jedna z przyczyn spadków wydajności, gdyż konieczna jest obecność wirtualnej maszyny interpretującej kod programu na dyrektywy procesora. Efektywność takiej maszyny (do Androida w wersji 4.4 był to tzw. Dalvik, w nowszych wersjach coraz bardziej ustępujący środowisku uruchomieniowemu ART) nie należy do wysokich.

Jedną z głównych zalet systemu Android jest, wspomniana wcześniej, otwartość oraz podatność na modyfikacje. Dzięki temu, oraz pracy społeczności zgromadzonej wokół systemu, powstały setki alternatywnych tzw custom ROMów. Istnieje również ogromna bazą wszelkiego rodzaju aplikacji, tworzonych zarówno przez samego Google’a jak i podmioty trzecie. Zapewne jest to efektem tego, że zdecydowano się na bazę w postaci popularnego języka programowania jakim jest Java. Z pewnością zadania nie ułatwiała sama amerykańska korporacja, gdyż dość długo nie było jednego, spójnego IDE stworzonego dla deweloperów chcących budować aplikacje pod system Android. Jedną z możliwości była praca w środowisku Eclipse wspomaganym przez odpowiednie wtyczki, dostarczane przez firmę Google. Na szczęście wraz z majem 2014 roku amerykańska firma zaczęła udostępniać zutylizowane narzędzie o nazwie Android Studio, które bazowało na platformie IntelliJ. Początkowo były to wersje alpha, później beta, aż w końcu w grudniu owego roku wyszła pierwsza stabilna wersja powyższego narzędzia, które jest stale poprawiane.

**- System iOS**

Zupełnym przeciwieństwem, opisanego w poprzednim podrozdziale systemu, jest system operacyjny iOS. Jest to system autorstwa firmy Apple, który cechuje się całkowicie zamkniętą architekturą. Bazuje on na systemie Mac OS X stworzonym wcześniej na potrzeby urządzeń typu Macintosh. Oba te produkty opierają się na jądrze Darwin i mają silne uniksowe korzenie.

Początkowo iOS – gdy zaprezentowano światu pierwszego iPhone’a napędzanego przez ów system – nie posiadał oficjalnej nazwy. Dopiero w marcu 2008 roku, 9 miesięcy po wydaniu pierwszej wersji systemu, Apple wprowadzając na rynek pierwsze narzędzia deweloperskie dla tego środowiska, nadała mu nazwę iPhone OS. W czerwcu 2010 została ona przemianowana na iOS, która to funkcjonuje do dnia dzisiejszego.

W odróżnieniu od swych odpowiedników z Androidem, urządzenia pracujące pod kontrolą systemu iOS są zdecydowanie bardziej spójne sprzętowo, jak i cenowo. Wszystkie oficjalnie dostępne produkty oferujące ów system należą do półki premium, zarówno pod względem jakości wykonania, jak i wydajności. Brak tu więc takiej różnorodności i możliwości wyboru, jaką cechuje się rynek sprzętu mobilnego z systemem Android.

Firma Apple, która jest odpowiedzialna za wszystkie dostępne na rynku urządzenia z systemem iOS, sama projektuje nie tylko ich wygląd, ale i większość komponentów koniecznych do ich budowy, jak chociażby jednostki centralne. Dzięki takiemu podejściu oraz faktowi, iż system iOS jest systemem całkowicie zamkniętym, dużo łatwiej osiągnąć odpowiednio dobrą optymalizację i stabilność systemu oraz aplikacji dla niego dedykowanych. Wynika to z bardzo prostego faktu, iż podczas tworzenia, czy rozwijania oprogramowania znana jest dokładna architektura i specyfikacja sprzętu, na którym owe oprogramowanie ma działać. Dzięki temu urządzenia kontrolowane przez system iOS charakteryzują się bardzo dobrą kulturą pracy, stabilnością działania i odpornością na spadki wydajności.

**Dane geograficzne**

By stworzyć aplikację, czy to nawigującą użytkownika, czy też tylko wyznaczającą trasę, niezbędnym jest skorzystanie z baz danych infrastruktury miejsca, w którym owa aplikacja ma działać. Bazy te zazwyczaj zawierają mnóstwo użytecznych dla twórców informacji, jednak w przypadku wyznaczania tras absolutnie kluczowymi są dane geograficzne. Ich struktura jest bardzo rozmaita i zależy od konkretnych rozwiązań.

**- Dane ZIKiT-u pobierane poprzez API CartoDB**

Aplikacja, której zrobienia podjęto się na potrzeby niniejszej pracy, korzysta z danych na temat infrastruktury rowerowej Krakowa zgromadzonych przez ZIKiT w formie zbioru danych CartoDB. Jest to baza danych pracująca w chmurze, umożliwiająca przechowywanie oraz szybką wizualizację danych z poziomu przeglądarki internetowej. Sporym udogodnieniem, które charakteryzuje CartoDB, jest obsługa bardzo dużej ilości formatów danych wejściowych. W przypadku bazy ZIKiT-owej mamy do czynienia z danymi zapisanymi w formacie JSON, w których – w przypadku danych geograficznych, jak np. koordynaty – zagnieżdżony jest format GeoJSON. Specyfika tworzenia map w CartoDB – która umożliwia tworzenie ich wizualnie – nastręcza pewne problemy podczas przetwarzania danych. Jednym z głównych jest rozbieżność punktów początkowych konkretnych segmentów w miejscu skrzyżowania tychże. W bazie stworzonej przez ZIKiT każdy segment w miejscu skrzyżowania z innymi, ma punkt początkowy bardzo minimalnie przesunięty względem punktów początkowych innych segmentów. W terenie rzeczywistym są to odległości rzędu kilku centymetrów, jednak patrząc pod kątem tworzenia aplikacji, te różnice mogą być bardzo kłopotliwe.

Inną charakterystyczną rzeczą dla używanego w stworzonej aplikacji API, jest zaznaczanie punktów dla danych segmentów wyłącznie na początku, końcu oraz miejscu załamania się segmentu. W wyniku takiego podejścia skrzyżowanie jednego segmentu z drugim, który jest na danym odcinku idealnie prosty (a więc nie wymaga dodatkowych punktów), nie powoduje obecności bliskiego sąsiedztwa punktów segmentów przecinających się. Z tego też powodu niejednokrotnie ciężko byłoby ustalić miejsca skrzyżowań co poniektórych segmentów i w związku z tym wymagało to dodatkowych rozwiązań algorytmicznych.

**- Google Maps API**

Google Maps API jest potężnym narzędziem tworzonym przez amerykański koncern Google. Umożliwia zaadaptowanie własnych map opartych o to rozwiązania na praktycznie każdą platformę, od Androida, przez iOS-a, po zwykłe strony internetowe czy web serwisy. Powyższe API charakteryzuje się gigantyczną bazą danych obejmującą w zasadzie cały świat, sporą dokładnością oraz pewnością danych – w końcu firma Google sygnuje je swoim logiem, dba więc o ich wiarygodność.

**Użyte algorytmy**

W niniejszym podrozdziale zaprezentowane będą znane algorytmy, które posłużyły do stworzenia aplikacji. Zostanie wyjaśnione ich ogólne działanie oraz celowość użycia w tym projekcie inżynierskim.

**- Algorytm Dijkstry**

Jako, że w temacie pracy jest mowa o wyszukiwaniu najkrótszej trasy, oczywistym jest, iż niezbędny był algorytm do rozwiązania owego problemu. Chyba najbardziej znanym i wysoce efektywnym w działaniu jest ten opracowany przez holenderskiego informatyka Edsgera Dijkstre. Jak można przeczytać w książce *Wprowadzenie do algorytmów* „algorytm Dijkstry służy do rozwiązywania problemu najkrótszych ścieżek z jednym źródłem w ważonym grafie skierowanym w przypadku, gdy wagi wszystkich krawędzi są nieujemne”. Jak wygląda to w praktyce? Otóż algorytm Dijkstry polega na zapamiętywaniu kosztów ścieżek oraz wielokrotnym powielaniu pewnego łańcuch czynności. Pierwszym etapem jest stworzenie tablicy kosztów dotarcia do każdego wierzchołka z wierzchołka początkowego (koszt dotarcia do niego samego wynosi 0, a do wierzchołków niedostępnych bezpośrednią drogą nieskończoność). Następnie w pętli wykonywane są niniejsze czynności: najpierw w tablicy wyszukiwany jest wierzchołek o najniższym, sumarycznym koszcie dotarcia z wierzchołka początkowego, po czym zostaje on usunięty ze zbioru wierzchołków do rozważenia. Kolejnym krokiem jest sprawdzenie dla wszystkich sąsiadów właśnie usuniętego wierzchołka sumarycznych kosztów dotarcia z wierzchołka początkowego. Te z kosztów dla danych wierzchołków, które są niższe niż obecnie istniejące w tablicy, zostają zapisane w owej tablicy poprzez nadpisanie poprzednich danych. Pętla kończy swe działanie w momencie, gdy zbiór wierzchołków do przeszukania zostanie opróżniony (liczenie najkrótszej drogi z wierzchołka początkowego do wszystkich możliwych wierzchołków), bądź też gdy wierzchołkiem wyrzuconym właśnie ze zbioru jest wierzchołek końcowy (liczenie najkrótszej drogi pomiędzy wierzchołkiem początkowym a końcowym).

Dzięki zapamiętywaniu kosztów poszczególnych wierzchołków algorytm ten jest niezwykle efektywny ze względu na fakt, iż dokonuje on redukcji potencjalnych tras już w trakcie wykonywania obliczeń. Pozwala to zaoszczędzić czas, gdyż nie wszystkie potencjalne ścieżki zostają sprawdzane, lecz tylko te o najniższych kosztach względem wierzchołka początkowego.

**- Metoda haversine → dodałem, bo chyba przydałoby się tego użyć – implementacja powinna być banalna, a wyniki dzięki temu będą dokładniejsze no i zawsze coś się walnie do teorii**

Wszystkie dane na jakich bazują wszelakie API dotyczące map występują w formie współrzędnych geograficznych. Z tego też powodu wyliczanie odległości nie było sprawą oczywistą, ani tym bardziej prostą. W tej sytuacji z pomocą przychodzi tak zwana *metoda haversine*. Jest to sposób wyliczania odległości pomiędzy dwoma punktami znajdującymi się na olbrzymiej kuli (jaką przecież jest Ziemia) na podstawie długości i szerokości geograficznej oraz promienia tejże kuli (w tym przypadku Ziemi, czyli około 6371 kilometrów). Dokładny sposób obliczania odległości *metodą haversine* przedstawiają wzory:

**tu będą wzory (za nic nie mogłem zmusić mojego open office'a do poprawnego ich wyświetlenia)**

We wzorze (1) znak Δφ oznacza różnicę szerokości geograficznych punktów ( Δφ = φ2 – φ1), a Δλ to różnica ich długości geograficznych (Δλ = λ2 – λ1). Trzeba zaznaczyć, że wszystkie dane współrzędnych geograficznych muszą zostać przeliczone ze stopni, minut i sekund kątowych na radiany, by otrzymane wyniki były prawidłowe.

Warto również wyjaśnić czym jest funkcja atan2, która pojawia się we wzorze (2). Otóż jest to dwuargumentowa funkcja arcus tangens i w wyniku jej działania uzyskuje się wartość kąta zawartego pomiędzy osią odciętych (OX), a półprostą wychodzącą z początku układu współrzędnych i przechodzącą przez punkt o współrzędnych (X1, Y1). Współrzędna Y1 jest pierwszym argumentem owej funkcji, a współrzędna X1 drugim.

W wyniku powyższych obliczeń otrzymuje się odległość pomiędzy dwoma punktami wyrażoną w metrach.

**Języki programowania i użyte narzędzia**

W przypadku projektu, który ma działać na kilku odmiennych platformach oczywistym jest, iż przy jego tworzeniu wykorzystanych zostanie więcej języków programowania. W niniejszym projekcie koniecznym było stworzenie aplikacji zarówno na system Android jak i iOS. Jedna na etapie projektowania zdecydowano, że zamiast pisać osobno dwie podobne do siebie aplikacje na dwa różne systemy operacyjne w dwóch różnych językach programowania, lepiej będzie skorzystać z biblioteki języka programowania JavaScript o nazwie ReactJS. Umożliwia ona tworzenie interfejsów użytkownika, a w naszym przypadku pozwala na stworzenie aplikacji zarówno na system Android jak i na iOS, co zaoszczędza zasoby.

Proces tworzenia grafu relacji wszystkich ścieżek, a także późniejsze wyliczanie optymalnej ścieżki, samo w sobie wymaga całkiem sporej wydajności obliczeniowej, a dodatkowo powinno przebiegać szybko (nawigacja musi bardzo często odświeżać pozycję użytkownika, by monitorować jego pozycję i informowac o następnym manewrze lub wyznaczyć nową trasę w przypadku zboczenia z obecnej), dlatego też zdecydowano by proces ten odbywał się na zewnętrznym serwerze. Ma to oczywiście swoje wady i zalety, jednakże w mniemaniu autorów projektu było to rozwiązaniem najlepszym w danej sytuacji. Z tego też powodu zdecydowano się na realizację owego skryptu w języku JavaScript, który jest wprost stworzony do budowania aplikacji webowych działających w czasie rzeczywistym.

**- Skryptowy język programowania JavaScript oraz biblioteka ReactJS**

Język JavaScript powstał dzięki firmie Netscape i zadebiutował na rynku 4 grudnia 1995 roku. Jest to język programowania zorientowany obiektowo (choć nie tylko – w jego paradygmatach widnieją także funkcyjność i imperatywność) w głównej mierze wykorzystywany do tworzenia stron internetowych. Skrypty napisane przy pomocy JavaScriptu służą zazwyczaj do zapewnienia interakcji poprzez reagowanie na pewne zdarzenia, a także do walidacji danych wprowadzanych w formularzach. Czasami wykorzystuje się je również do tworzenia złożonych efektów wizualnych, stąd nie raz idę one w parze z CSS i HTML. Język ten opracowano właśnie w taki sposób, aby jego kod mógł być przemieszany z kodem HTML oraz wszystkie jego interakcje z ekranem i użytkowaniem mogły korzystać właśnie z tego kodu. Same skrypty, które uruchamiane są przez strony internetowe, mają mocno limitowany dostęp do urządzenia, na którym wyświetlana jest strona.

Strony internetowe to jednak nie jedyne zastosowanie języka JavaScript. Może on być również wykorzystany w celu stworzenia pełnoprawnej, działającej autonomicznie aplikacji. Wiele firm lub organizacji udostępnia odpowiednie środowiska i biblioteki umożliwiające zbudowanie takich aplikacji – jak chociażby Microsoft, który dołącza takie biblioteki jako część środowiska Windows Scripting Host, czy też Mozilla.

Biblioteka ReactJS – jak już zostało to wcześniej wspomniane – służy do tworzenia interfejsów graficznych aplikacji internetowych. Jej pierwsze wydanie miało miejsce 1 marca 2013 roku, a jej twórcą jest Jordan Walke, jeden z programistów firmy Facebook. Jak twierdził sam Walke inspiracją do stworzenia Reacta była bilbioteka do języka programowania PHP o nazwie XHP.

ReactJS wyróżnia się na tle podobnych bibliotek przede wszystkim obecnością wirtualnego Obiektowego Modelu Dokumentu (ang. Document Object Model, w skrócie DOM). React cały DOM przechowuje w pamięci, a po jakiejś zmianie wyszukuje jedynie ewentualne różnice pomiędzy prawdziwym DOM, a tym wirtualnym i w przypadku wykrycia istnienia tychże aktualizuje zmiany. Kolejną unikatową cechą jest język JSX, czyli nakładka na język programowania JavaScript dodająca możliwość wstawiania kodu HTML (lub też komponentów samego Reacta) bezpośrednio w kodzie, zamiast ciągu znaków.

Mimo krótkiego stażu na rynku biblioteka ReactJS szybko zdobyła niezwykle dużą popularność. Już w styczniu 2017 roku według statystyk gromadzonych przez popularny wśród programistów serwis GitHub (wykorzystujący system kontroli wersji Git) projekty z użyciem biblioteki React były na piątym miejscu pod względem popularności (liczebności). Obecnie na jej bazie działają strony takich gigantów branży usługowej jak chociażby PayPal (płatności), Imgur (hosting grafiki) czy Netflix (streaming wideo).