AKADEMIA GÓRNICZO-HUTNICZA
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I ELEKTRONIKI
KIERUNEK INFORMATYKA

STUDIO PROJEKTOWE I

Trajektorie dzienne w technologii blockchain

Przechowywanie trajektorie przemieszczeń przy użyciu technologii blockchain

Dmytro Kuzmin Vyacheslav Trushkov

Kraków, 21 kwietnia 2021

Spis treści

1 Wprowadzenie

Przedstawiamy dokumentację dotyczącą projektu realizowanego w ramach przedmiotu Studio Projektowe I. Projekt polega na zrobieniu systemu do przechowywania trajektorii przy użyciu technologii blockchain.

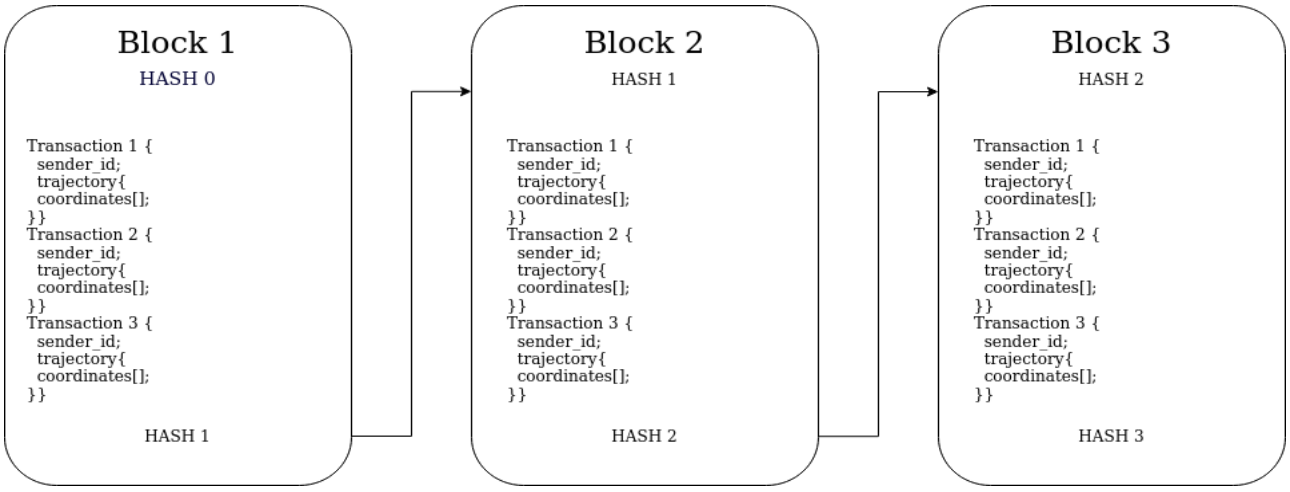
2 Motywacja

Załóżmy że mamy flotę samochodów zajmującą się dostawą towarów. Chcemy mieć podgląd do przebytej drogi każdego samochodu. Dlatego możemy stworzyć rozproszoną sieć która przechowuję historię przebytych tras każdego z samochodów należących do naszej floty. Dane będą przechowywane bez możliwości edycji co zapewni nam pewność że otrzymujemy dane wiarygodne. Wyżej opisane własności możemy otrzymać używając technologii blockchain.

3 Wprowadzenie teoretyczne

3.1 Blockchain

Do tworzenia rozproszonej bazy danych używamy blockchain.
Blockchain trajektorii przemieszczeń to jest zdecentralizowany rejestr łączonych do bloków transakcji, zawierających w sobie trasę i jej autora, funkcjonujący jako rosnąca lista jednokierunkowa. To jest kluczowy sposób strukturyzacji danych, odróżniający Blockchain od zwykłej bazy danych. Blockchain gromadzi informacje w grupach, zwanych również blokami, które przechowują zestawy informacji. Jeden blok w Blockchain trajektorii przemieszczeń jest tworzony średnio raz na 8 godzin. Bloki nie mają określonej pojemności, więc wszystkie otrzymane transakcję są łączone blokiem z poprzednim raz na 8 godzin, tworząc łańcuch danych znany jako „łańcuch bloków".Wszystkie nowe informacje wynikające ze świeżo dodanego bloku są kompilowane do nowo utworzonego bloku, który po wypełnieniu zostanie również dodany do łańcucha. System ten z natury tworzy również nieodwracalny harmonogram danych, gdy jest wdrażany w sposób zdecentralizowany.



Rysunek 1: Rysunek pokazuje strukturę blockchain'u trajektorii przemieszczeń

3.2 Budowa blockchain'u

Blockchain składa się z bloków które z kolei składają się z transakcji.

3.2.1 Transakcje

Transakcje są podstawową jednostką informacji w naszym systemie.
Transakcja zawiera w sobie takie pola:

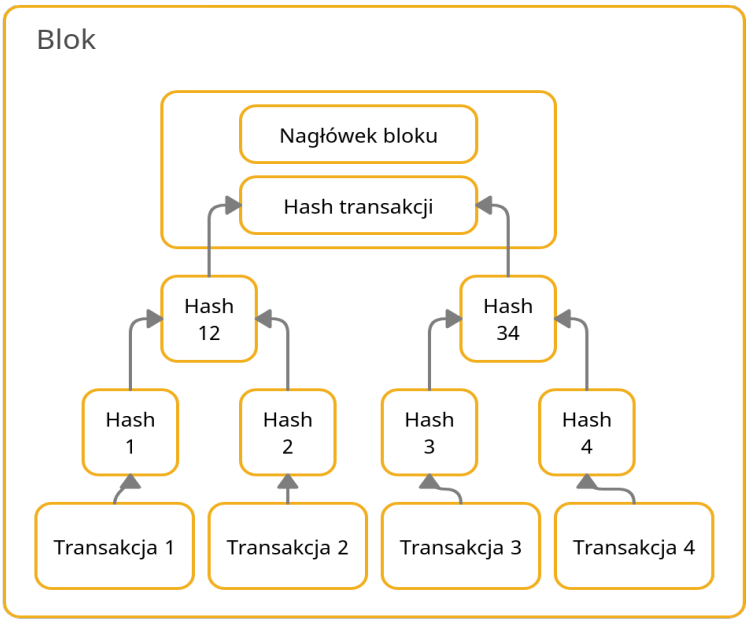
- ID samochodu
- Długość trasy
- Lista współrzędnych trasy

Kiedy samochód rozpoczyna trasę to transakcja zostaję stworzona lokalnie i co jakiś czas zostaję aktualizowana. Transakcja zostaję zakończona wraz z zakończeniem trasy. Co 8 godzin transakcji ze wszystkich samochodów zostają przesłane do głównego komputera na którym tworzymy Block.

3.2.2 Block

Block będzie tworzony każde 8 godzin (zakładamy że zmiana kierowcy będzie 8 godzin). Blok zawiera w sobie transakcji i hash'e poprzednich bloków.
Block zawiera w sobie takie pola:

- Hash poprzedniego bloku
- Data utworzenia
- Hash transakcji za ostatnią zmianę

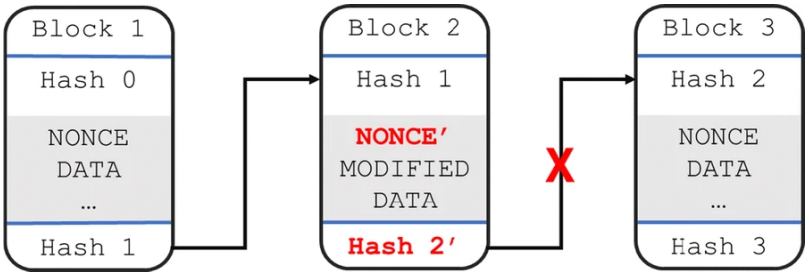


Rysunek 2: Rysunek pokazuje strukturę blocku

3.3 Bezpieczeństwo

Jedną z głównych zalet blockchainu jest niezmienność danych. Jest to powodowane tym że gdy chcemy zmienić jakiś blok to musimy wprowadzić zmianę do wszystkich kopij naszego blockchainu co w praktyce staję się praktycznie nierealnie.

Z racji na to że każdy następny blok powstaję z hash'ów poprzednich bloków to nie możemy zmieniać bloków. Gdyż zostanie wprowadzona zmiana do już istniejącego bloku to hash zmienionego bloku już się nie będzie zgadzał z tym hashem który jest zapisany w następnym bloku, i możemy powiedzieć że obcinamy relację pomiędzy blokami.



Rysunek 3: Rysunek pokazuje blockchain z modyfikowanymi danymi w block 2

3.4 Uzasadnienie wyboru

3.4.1 Jeden blok zawiera wszystkie transakcji

Koncepcja polega na tym że nasz blockchain tworzy swojego rodzaju dziennik. W danym dzienniku jedna strona reprezentuje jedną zmianie, a strona z kolei reprezentuje jeden blok.
Przechowując wszystkie trajektorie dzienne, a nie każdą trajektorie w bloku. Unikamy sytuacji kiedy blockchain będzie się rozrastał. Na przykład może się zdarzyć że jeden kierowca będzie wskazywał początek i koniec trasy co 10m i otrzymamy duży łańcuch bloków który będzie bezsensownym.

3.4.2 Nowy blok co 8 godzin

Jak jest opisane powyżej, chcemy przechowywać jedną zmianę w blockchainu, z tego wynika że potrzebujemy stałego czasu zapisywania do bloku, żeby ujednolicić wszystkie zapisy do blockchainu.

3.4.3 Algorytm Konsensusu: Proof Of Work

Najważniejszą celą metod osiągania konsensusu jest zapewnianie bezpieczeństwa sieci za rachunek tworzenia 'kosztu' każdego utworzonego bloku. Wiedząc że hash każdego z bloków bazuje się na hashu bloku poprzedniego, widzimy, że jeżeli jakaś informacja w historii bloków została zmieniona to wtedy trzeba byłoby wykonać pracę mainerów (w naszym przypadku komputera głównego) od zmodyfikowanego bloku do ostatniego. Czyli tworzymy sytuację, kiedy straty czasu i kosztów na zmianę historii blockchain'u są za duże, czym zapewniamy niezmienność historii.
Do wyboru mieliśmy zasadniczo dwa algorytmy konsensusu:

- **Proof of Space** - Jest formą kryptograficznego potwierdzenia, w którym jedna strona udowadnia innym, że pewna ilość przestrzeni została poświęcona na jakiś cel.
- **Proof of Work** - Jest formą kryptograficznego potwierdzenia, w którym jedna strona udowadnia innym, że pewien wysiłek obliczeniowy został poświęcony na jakiś cel.

Podstawową różnicą wad i zalet pomiędzy pierwszym a drugim jest wysokie zużycie prądu algorytmem Proof od Work, wtedy kiedy Proof of Space jest bardziej ekologiczny i ma dużo niższe zużycie. Z drugiej strony Proof of Work jest prostszy w implementacji i przy niskiej mocy sieci zwiększa koszt bloku z upływem czasu, gdyż Proof of Space potrzebuje dodatkowej kontroli w postaci Proof of Elapsed Time. Licząć że będziemy mieć sieć zamkniętą o niskiej mocy, stosunkowe zużycie prądu będzie nieistotne, więc w naszym przypadku decydujemy, że Proof of Time będzie lepszy.

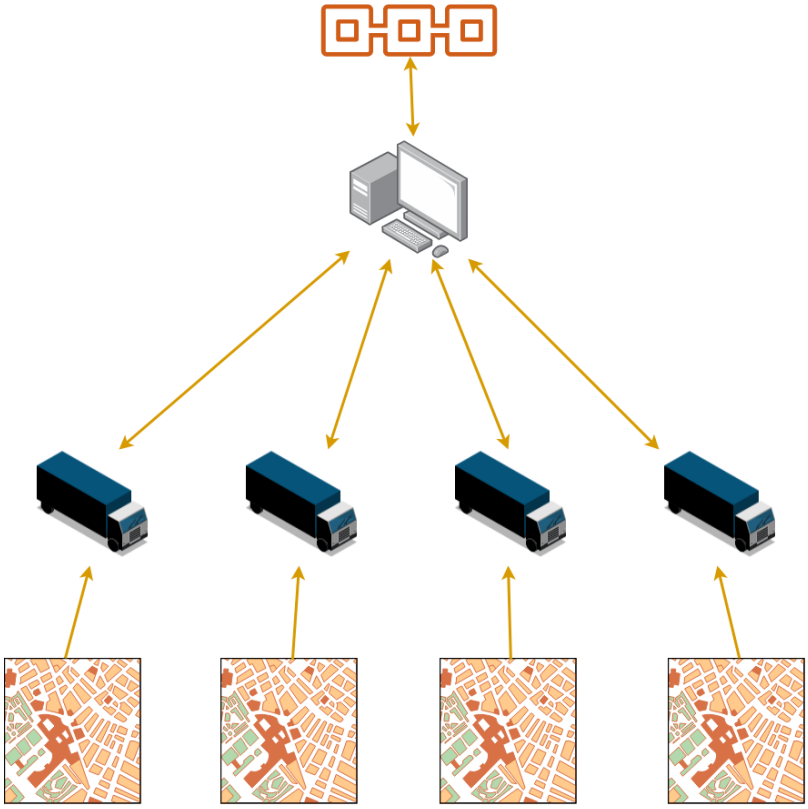
3.5 Nonce o stałej złożoności

Przewidujemy, że sieć będziemy zamknięta i będzie posiadała stałą moc obliczeniową, więc ustalając jeden raz złożoność nie będziemy mieli potrzeby ją zmieniać.

4 Opis systemu

4.1 Ogólny opis

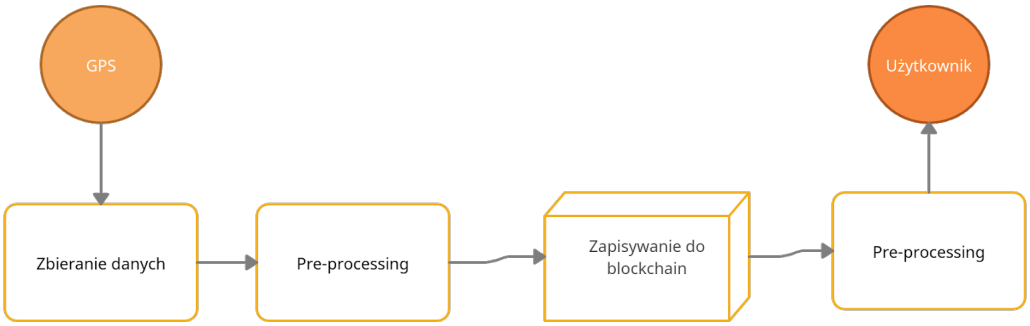
System będzie się składał z **komputera głównego**, który będzie tworzył **blockchain**, z **floty samochodów** które przesyłają swoje trasy do głównego komputera. Każdy samochód po prze-



Rysunek 4: Rysunek pokazuję ogólny schemat systemu

byciu trasy będzie zapisywała to lokalnie na urządzeniach wbudowanych do samochodu. Każdy samochód będzie miał swoje ID co pozwoli na szybką identyfikację który samochód jaką trasę przebył. Po rozpoczęciu trasy urządzenie które jest podłączone do GPS będzie próbkować współrzędne z ustawionym okresem. Lista współrzędnych która jest sama w sobie strukturą losową formuję transakcję.
Jedną transakcja jest to jedna trasa. Zapisujemy do transakcji ID samochodu dlatego żeby było możliwie przydzielić samochód do trasy. Trasę zapisujemy dlatego że interesuję nas historia przejazdów danego samochodu. Długość trasy zapisujemy dlatego żeby nie tracić dokładności przy wyliczaniu długości z współrzędnych.

4.2 Algorytm



Rysunek 5: Rysunek pokazuję ogólny schemat algorytmu

Algorytm składa się z czterech punktów.

- **Zbieranie danych** polega na próbkowaniu danych z GPS.
- **Pre-processing** polega na tworzeniu transakcji która zostanie zapisana do blockchain.
- **Zapis do blockchain**, na tym etapie zostaną zebrane wszystkie transakcję z ostatniej zmiany i z tych transakcji zostanie utworzony nowy block.
- **Pre-processing** dany pre-processing służy do odczytania danych z blockchain i przedstawieniu ich użytkowniku.

5 Implementacja

5.1 Opis technologii

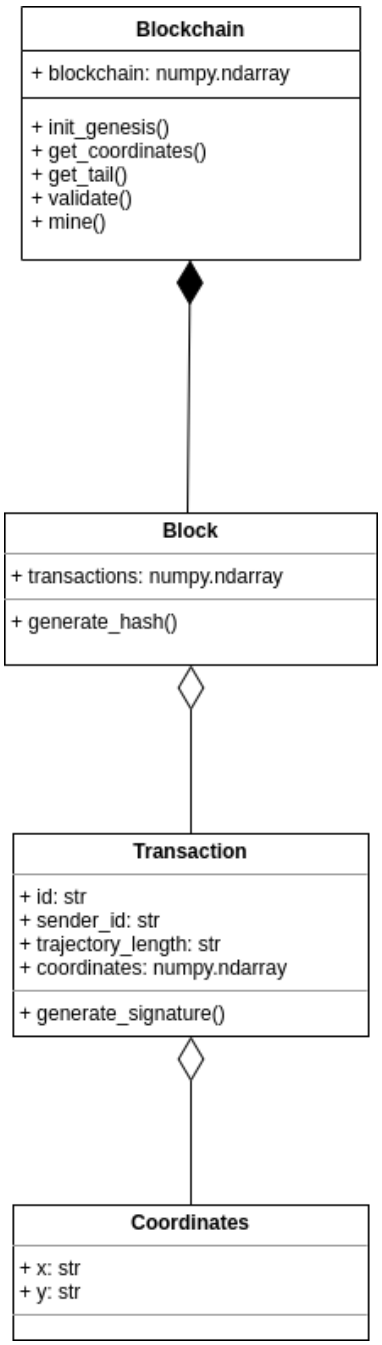
- Język programowania: **Python**
  - Python wspiera różne paradygmaty programowania - pozwala zastosować programowanie obiektowe żeby implementować każdą składową łańcuchu bloków osobno oraz programowanie funkcyjne żeby w łatwy i czytelny sposób zarządzać dużą ilością uporządkowanych danych.
  - Automatyczne zarządzanie pamięcią w Pythonie za pomocą Garbage Collector wezmę na siebie odpowiedzialność za de-alokację niepotrzebnej pamięci, za rachunek czego zwiększa się poziom bezpieczeństwa systemu.
  - Wielka ilość wspieranych platform.
- Biblioteki: **Numpy, json**
  - Wiadomo że Python ma stosunkowo niską wydajność. Ten problem rozwiązujemy za pomocą biblioteki Numpy, która jest zaimplementowana w języku C. Takie rozwiązanie pozwoli pozwoli zapewnić wysoką szybkość obliczeń na na tablicach danych.

  - JSON będzie wykorzystany żeby przedstawić dane w wygodnej postaci dla funkcji hashującej SHA256.

- API: **Mapbox**
  - Mapbox zostanie wykorzystany do wizualizacji otrzymanych tras

- Algorytm Kosensusu: **Proof Of Work**
  - w technologii blockchain to jest technika osiągania porozumienia w kwestii dodawania nowych bloków. Pozwala komputeru udowodnić że na dodanie nowego bloku został poświęcony pewny wysiłek obliczeniowy. W naszym przypadku to będzie znajdowanie liczby Nonce o **stałej** złożoności.

5.2 Diagramy klas



Rysunek 6: Rysunek pokazuję klasy programu oraz najważniejsze pola i funkcję

5.3 Endpoints

- \- genesis
- \nodes\register - dodać nową nodę
- \nodes\resolve - znaleźć konsensus
- \transactions\new - dodać nową transakcję
- \mine - zapisać blok
- \chain - otrzymać cały blockchain