Egy olyan projekt ötlet jutott eszembe, hogy **"'AI'-alapú autó elosztó alkalmazást"** (webes) készítsek. Az lenne a célja, hogy a program GPS kordinátákat / Pontos címeket kapna, azokról az autókról, amiket át lehetne / kellene mozgatni egyik helyről a másikra (vagy egyik kerületből másikba).

Ezek alapján pedig:

* Összegyűjtené az autók adatait,
* Utána eltárolna az adatbázisban,

Majd kiszámolná / megtervezné az útvonalat. Kezdésnek 3 autó esetében mely három autó van egymáshoz a legközelebb. Azt is figyelembe véve, hogy 1 autóval (amivel kimenne a 4 ember a 3 autóért) mennének.

**Amit a program kapna meg adat** (elsőnek én generálnám le faker alapú Laravel Backend, phpmyadminnal-, hogy legyen mivel dolgoznia):

* Autó **GPS** **koordinátája** / **címe**,
* Mikor **foglalták** **utoljára** (dátum, idő)
* Az akkumulátor **töltöttségi** szintje / **Üzemanyagszintjét**.
* **5 db** fix **helyszín** koordinátája / címe – ahová le kell „szállítani” őket.

**Amit a program adna vissza:**

1. Teljes útvonal hossza (1-2-3-4. autó közötti különbség (távolság) km-ben).
2. Az autók adatait töltöttségi / üzemanyagszintjét, rendszám, szín.
3. 5db fix helyszín közül hova kell leszállítani – Budapesten belül:
   1. Keleti-pályaudvar parkolójának 500m sugarú körben (fix zóna)
   2. Erzsébet híd pesti hídfőjének 500m sugarú körében. (fix zóna)
   3. Nyugati-pályaudvar parkolójának 500m sugarú körben (fix zóna)
   4. Citadella 500m sugarú körében. (fix zóna)
   5. Liszt Ferenc Nemzetközi repülőtér P+R parkolója.

Az 5db fix **helyszín** **amire** szállítani kell az autókat, ahhoz képest kell **meghatározni** azt **az 5 autót**, ami:

1. Legközelebb van(nak) a célállomáshoz (szállítási helyszínhez)
2. Az autók egymáshoz viszonyított távolsága a legkisebb.
3. Az autókat „Mikor **foglalták** **utoljára**” időpont alapján a lehető legtöbb idő eltelt a ’most’-hoz képest.

**Reprezentatív példa számítás:**

* 10 db autó van, amit mozgatni kellene ’A’ pontból (különböző helyszínek Budapesten) a ’B’ pontba 🡪 Nyugati pályaudvarra.
  + Ebből 3 autó van az egyik a:
    - Balzac u. 1-7 szám alatt,
    - Másik az Eötvös u. 44 szám alatt,
    - A harmadik pedig a Király u. 35-37 szám alatt.
* A tervező programnak ki kell számolnia, hogy:

1. Az első autó a Király utcai lesz,
2. Utána az Eötvös utcai,
3. Majd a Balzac utcai autó, amit fel kell venniük és lerakniuk (leparkolniuk) a megkapott Nyugati-Pályaudvar parkolójának 500m-es sugarú körében.
4. Ehhez 3 „mozgató ember” és 1 sofőr kell, aki kocsival elviszi őket az adott autókhoz. Ezalapján kellene az utat megterveznie a programnak.

**Kiegészítő fejlesztések ezekhez:**

* Be lehessen állítani a csapatlétszámot, amivel a „gyűjtés” kezdődne (3-4-5-6) fő maximum.
* Térségre, kerületekre szűrés.
* Töltöttségi szintek alapján szűrés.

**Útvonaltervező logika**:

* **Adatok lekérése**: Kérje le az autók adatait az adatbázisból a megadott szűrési feltételek alapján.
* **Távolságok számítása**:
  + Használjon külső API-t (pl**. Google Maps Distance Matrix** **API**, **OpenStreetMap**) a távolságok és időtartamok kiszámításához.
  + Implementálja a **Haversine** **formulát**, ha **egyszerűbb** **megoldásra** van szükség **a földrajzi távolságok kiszámításához**.

**Algoritmusok implementálása**:

* **Kiválasztás az autók közül**:
  + Rendelje súlyokat a kritériumokhoz (távolság a célhoz, egymáshoz való közelség, utolsó foglalás ideje).
  + Használjon rendezési algoritmust vagy optimalizálási technikákat (pl. súlyozott pontszámítás) a legmegfelelőbb autók kiválasztásához.

**Eredmények előkészítése**:

* Készítse elő az adatokat a frontend számára (JSON formátumban), beleértve az útvonalak koordinátáit és a szükséges autóadatokat.