Gépkocsi kölcsönzés nyilvántartás

Készítsünk alkalmazást, amely gépkocsik nyilvántartását végzi.

1. Hozzunk létre egy adatbázist, amely két táblát tartalmaz:

Cars

id:	szám, automatikus számozás, elsődleges kulcs
producer:	szöveg(20), az autó gyártója, pl. Opel, Suzuki
type:	szöveg(30), az autó típusa, pl. Corsa, Swift
vintage:	szam, az autó forgalomba helyezésének éve
fuel:	szöveg(10), az üzemanyag típusa, pl. benzin, dízel

A táblákat töltsük fel 4-5 teszt rekorddal értelemszerűen! A Database mappa tartalmazza a szükséges sql fájlt!

2. Hozzunk létre egy header állományt, mely az alábbi menüpontokat tartalmazza:

- Cars - összes autó listája
- New Car - új autú hozzáadása

A mappastruktúra az alábbi szerint alakuljon:

- A gyökérmappában foglal helyet az index.php és a config.json és az *Application* mappa
- Az *Application* mappában a *Core*, *Style, Templates és Database* mappa található, valamint a functions.php.
- A *Core* mappába kerül a programlogika → **deleteCar**, **modifyCar**, **newCar,** cars.php
- A *Style* mappába kerül a stíluslap → style.css
- A *Templates* mappába kerül a _layout.php és az egyes megjelenítésért felelő view-k → headerView, carsView.php, newCarView.php

A header állomány minden esetben jelenjen meg az oldalon!

3. Hozzuk létre a _layout.php-t és a menüpontokhoz tartozó feldolgozó és megjelenítő oldalakat. Csak a **_layout.php** tartalmazza a teljes *HTML* fájlt! Az alábbi view-k nem tartalmazhatnak teljes HTML struktúrát!

A) **cars.php**: Adjon egy listát az összes autó összes adatáról!

B) **carsView.php**: Táblázatos formában jeleníti meg a kapott adatokat! Legyen lehetőség a módosításra, törlésre, amit az egyes soroknál megadott *Modify* és *Delete gombok-ra kattintva lehessen megtenni!* Ha a felhasználó a *Modify* gombra kattint, az oldal dobjon át a *modifyView.php*-ra, a kiválasztott autó adataival együtt.

C) **modifyView.php**: A *Modify* gomb megnyomása után megkapja a carsView.php által elküldött adatokat és azokat jeleníti meg egy űrlapban, majd, ha a felhasználó submiteli, elküldi a modifyCar.php logikának.

D) **modifyCar.php**: A modifyView.php hívja meg. A modifyView.php ürlap által elküldött adatokat ez a fájl update-eli az adatbázist, majd átirányít → header("Location: index.php").

E) **deleteCar.php**: Ha a felhasználó a *Delete* gombra kattint a táblázatban, az átadja az azonosítót és ez a fálj törli az adott sort az adatbázisból, majd átirányít.→ header("Location: index.php").

F) **newCarView.php**: Ha a felhasználó a fejléc "New Car" *ancore* tag-re kattint, ez a fájl egy üres űrlapot jelenít meg, melyben az új autó adatait lehet megadni.

G) **newCar.php**: A newCarView.php által elküldött adatok alapján insertál az adatbázisba, majd átirányít a főoldalra. → header("Location: index.php").

A kékkel jelölt oldalakhoz nem tartozik megjelenítés!

A feldolgozó fájlok csak adatokat kezelnek, írnak-olvasnak az adatbázisba, a megjelenítő oldalak (View) csak megjelenítik az átadott adatokat html-be ágyazva. Itt már nincsen logika/feldolgozás.

A view-k esetében kötelező az az alapelv követése, hogy csak azt írjuk ki php-val, ami dinamikus!

Pl.: Ez hibás:

```
<?php
if($akarmi)
{
         echo '<table>
                  <tr>
                     <td>column</td>
                     <td>{$data}</td>
                  </tr></table>';
}
?>
```

Ez helyes:

```
<?php if($akarmi): ?>
  <table>
    <tr>
      <td>column</td>
      <td><?=$data?></td>
    </tr>
  </table>
<?php endif ?>
```

Az elkészült program legyen esztétikus, használjon CSS stíluslapot! Használjon routing-ot, PDO adatbázis kapcsolatot és SQL paraméterezést, az egyes adatbázis műveleteket függvényben valósítsa meg, adatbázis

módosítás után használjon átirányítást, az egyes formok POST metódussal küldjék az adatokat! Törekedjen a megadott mappastruktúra követésére!

A kész feladat github repository linkjét küldje el email-ben.