SzalkaCar

Fejlesztői Dokumentáció

Kádár Marcell, Paragh Tibor
2025.03.31.

Fejlesztői Dokumentáció

Projekt Leírása

A **SzalkaCar** célja, az autó szerviz digitalizálása, a folyamat megkönnyebbítése és megbízható információ nyújtása az ügyfelek számára. A gyors, hatékony munka a cég jellemzője, törekednek arra, hogy ügyfeleik járműve minél kevesebb időt töltsenek a szervízbe, ez köszönhető a csapat szorgalmának.

Használt Eszközök

A Fejlesztés során több eszközt használtunk. Ezek listája a következő:

- Code-Server
  - Egy centralizált fejlesztő szerver, amivel elérhetjük projektünket akárhonnan.
- Docker
  - A Code-Serverek konténerben futtatása
- Nginx
  - A web elérési utak kezelése, mint például a Code-Server1-nek vagy Code-Server2-nek.
- Visual Studio Code
  - A weboldal, és szerver kódjának fejlesztés.
- phpMyAdmin
  - Az adatbázis kezelése, kiépítése

A hardver eszközök, amin a projekt fejlesztése történt:

- Acer Veriton M430
  - Szerver gép, amely Port Forwardolással lehetővé teszi a szerverek, és konténerek elérését.
  - 4 magos AMD Phenom II X4 850 processzor
  - 12GB DDR3 memória

    System information

Model	Acer Veriton M430
Machine ID	a85c10f68fa145a180c31d40568396ba
Uptime	about 18 hours

  -
- Lenovo Legion 5 15ACH6H 2020
  - Kádár Marcell fejlesztő által használt laptop
  - 6 mag, 12 szálas AMD Ryzen 5 5600H notebook processzor
  - 16GB DDR4 memória
  - 15 colos, 1920x1080 kijelző felbontás
- Saját építésű számítógép
  - Paragh Tibor fejlesztő által használt számítógép
  - 6 mag, 12 szálas AMD Ryzen 5 1600 processzor
  - 16GB DDR4 memória
  - 24 col, 1920x1080 kijelző felbontás

Célközösség

A weboldal célközössége, az olyan ember aki járműve iránti törődésre vágyik, de nincs ideje, vagy csak szimplán nem ért annak karbantartásához és szervízeléséhez. Életkortól függetlenül, ha van szervízelendő vagy karbantartandó járműve, szívesen várjuk.

Fő koncepciók

Az oldalnak több koncepciója van ennek megvalósításához, mint például az online árajánlás, közösségi piac és interaktív időpont foglalás.

- Árajánlat
    - Ha a felhasználó be van jelentkezve
        - Ahogy a felhasználó megnyitja az oldalt, a szerverből lekérdezzük a járműveit, majd a legördülő listába betöltjük. Majd muszáj megadnia a címet, és leírást a problémának. Amint rányom az „Árajánlat kérése” gombra, a kérés feltöltődik a APIn keresztül adatbázisunkba, amit admin panelen keresztül lehet kezelni, és válaszolni rá. Ahogy válaszolunk rá, a felhasználó egyből fogja látni a választ az oldalon, és emailen keresztül megkapja szintén, ha éppen nem nézné a weboldalt.
    - Ha a felhasználó nincs bejelentkezve
        - A kérésnek meg kell adnia a címét meg leírását, majd email címét meg kell adnia, hogy ezen keresztül tudjuk értesíteni. A backend ugyan úgy működik, mint a bejelentkezett felhasználónál.
- Időpont foglalás
    - Ezt csak bejelentkezett felhasználók érik el, amit a Cookiekon keresztül nézünk meg.
    - A bejelentkezett felhasználó kap egy görgethető naptárat, és ha ezen belül valamelyik elérhető napra rányom, egy modalon keresztül beviheti panaszát, hogy kell az adott problémát replikálni és a járművet, amelyen ezt a problémát észleli.
- Közösségi Pac
    - Bejelentkezett felhasználó érheti el csak, amint betölt az oldal, a szerverről lekérjük a hírdetéseket. Bal oldalt a szűrőket meg lehet adni, majd ha Szűrés gombra rányom, egy Post requestbe, a szűrési feltételeket bodyba betéve átküldi a szervernek, majd a szerver visszaadja a szűrt adatot.

Levelező rendszer

Az Email szervert a nethely.hu üzemeltetni. Széleskörű szolgáltatásokkal, például távoli elérés, saját levelező platform, stb.

Ez nálunk PHP-ban valósul meg. Egyedi sablonnal rendelkezünk, amelyet az adott levél tartalmához igazítunk.

Mikor kap a felhasználó levelet tőlünk?

- Fiók megerősítése regisztráció után
- Autójának státusz változásakor. (Amikor szervízben van)
- Ajánlatkérésére érkezett válaszát ott kapja meg
- Hirdetés, Közösségi bejegyzés jóváhagyásakor / elutasításakor
- Jelszó visszaállításkor megkapja a visszaállításhoz szükséges kódot.

Fájl struktúra

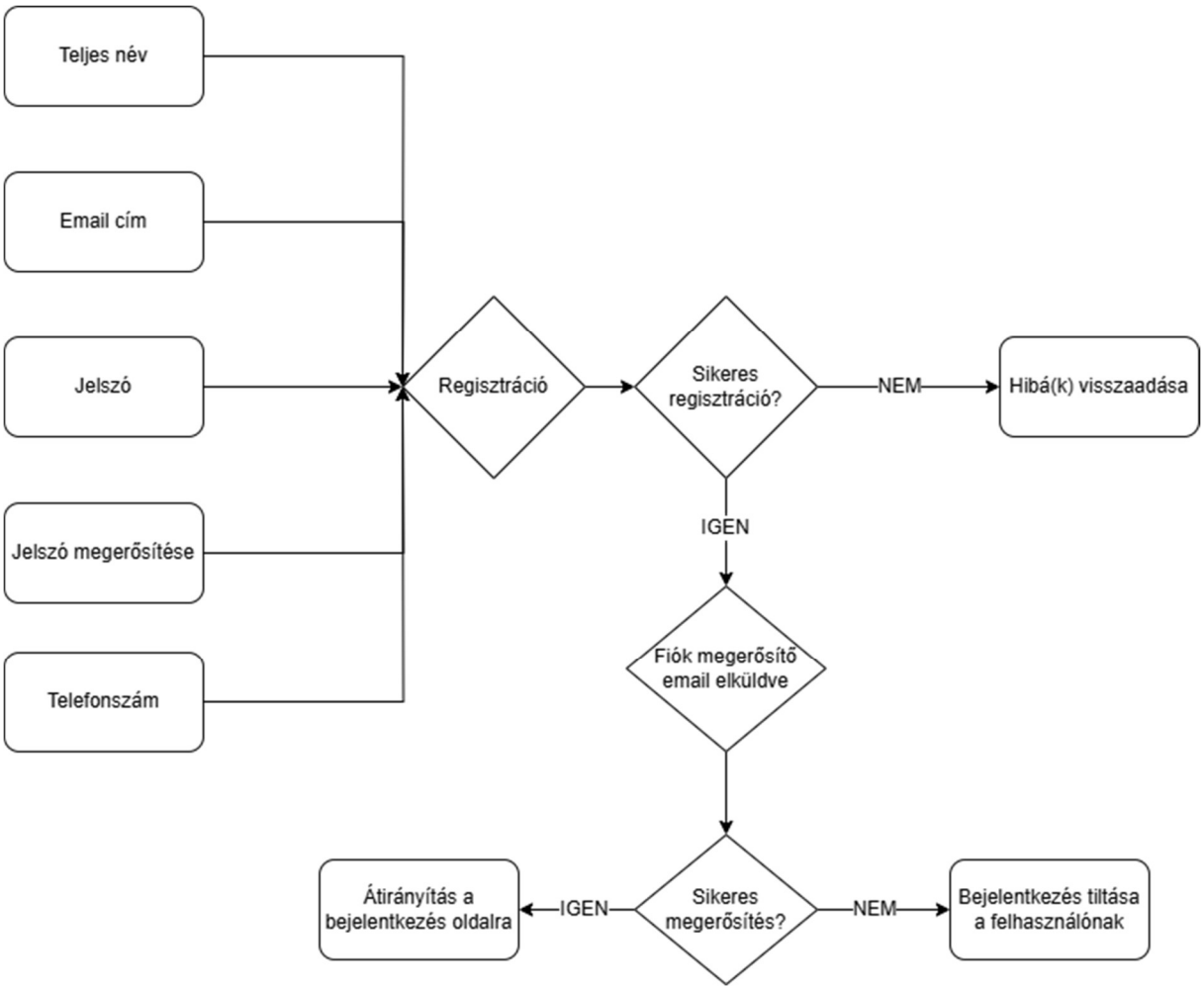
A Kliens/Frontend oldalon a következő a fájl struktúra:

- node_modules
  - A nodejs által használt modulok mappája
- public
  - A React által használt fájlok, mint például az index.html, ami visszaadja, hogy a Javascriptnek engedélyezve kell lennie ha nincs.
  - png
    - Itt található a weboldal által használt png fájlok egy része
- src
  - Components
    - E mappa tárolja a komponensek mappáit, amibe találhatóak a kódok.
    - Authentication
      - A bejelentkezés és regisztráció által használt részkódok.
    - Community
      - A közösségi oldalhoz kapcsolódó kódok, mint például a LikeCommunityPost, ami visszaad egy szív ikont és amellett a likeok száma, az ikon színe attól függ, hogy az adott bejegyzést a felhasználó likeolta e.
    - CommunityAdminPanel
      - A közösségi oldal admin panel komponensei, amellyel lehet a bejegyzéseket kezelni.
    - Marketplace
      - A közösségi piac szükségei kódjai, hogy betöltse, és készítsen új hirdetéseket a felhasználó.
    - MarketplaceAdminPanel
      - A közösségi piac adminisztrátor oldala, ahol a beküldött hirdetéseket lehet elfogadni, vagy elutasítani.
    - MyAccount
      - A felhasználó saját profilja, és ahhoz használt részek, mint például a saját bejegyzések betöltése.
    - Navigator
      - A Navbar és Footer mappája.
    - OwnCar
      - A felhasználó járműveinek komponense, azok megjelenítése és új jármű felvéteke.
    - PriceQuote
      - Árajánlás kérés küldéséhez szükséges kódok, fájlok.
    - Rating
      - A főoldalon található értékelések.
    - RequestsAdminPanel

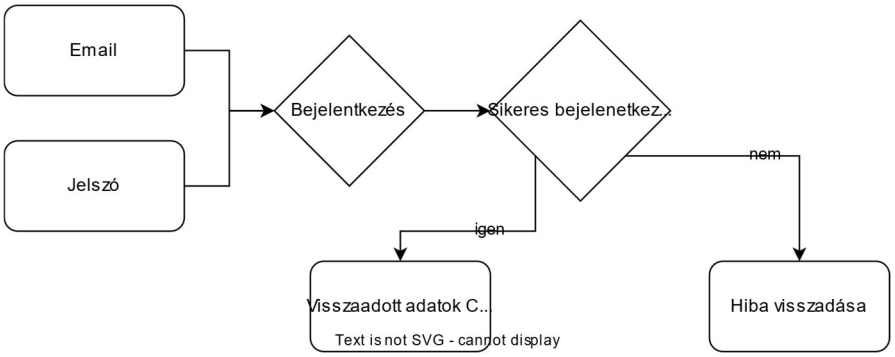
        - Árajánlat kérések válaszolásához és kezeléséhez szükséges komponensek.
- Media
    - A weboldal által használt JPG és PNG fájlok mappája
- Pages
    - Az elérési utakhoz rendelt fájlok itt találhatóak
- Providers
    - A ReactJS providerek mappája, mint például a ModalProvider, aminek importálása, és 2 új sor megadásával lehet egy modalt készíteni ami a felhasználó képernyőjén megjelenik.

Folyamatábrák

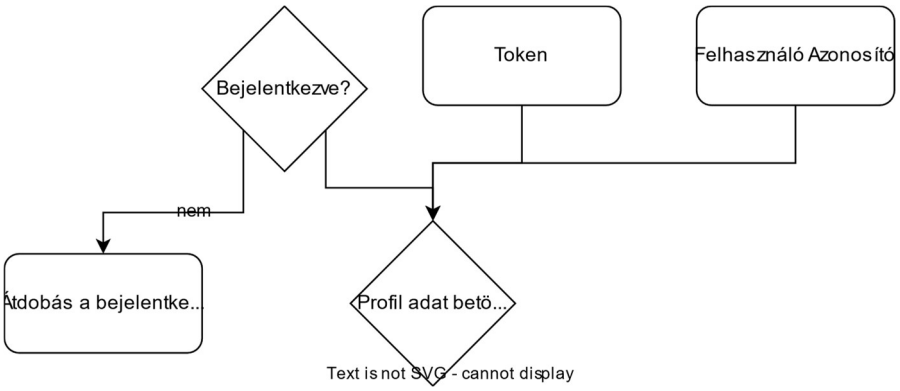
Regisztráció



Bejelentkezés

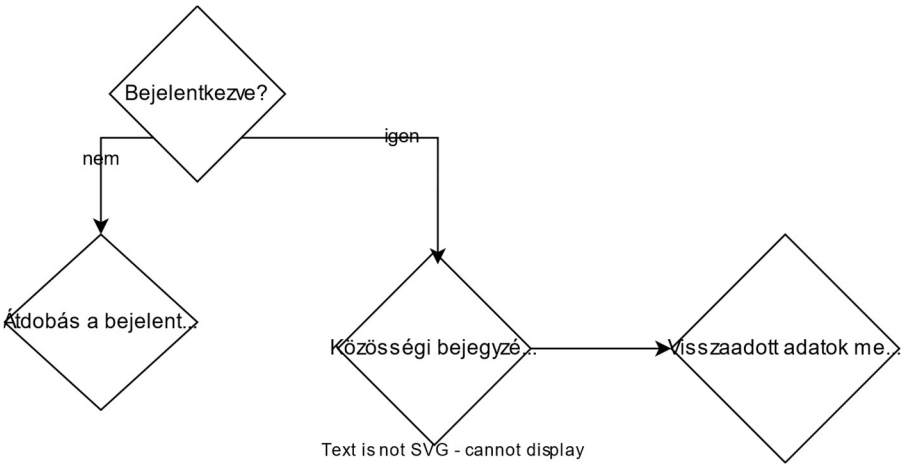


Profil

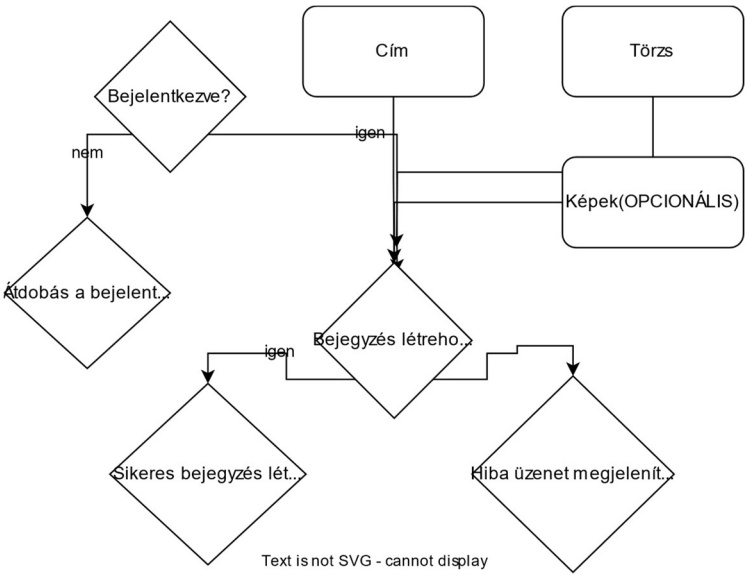

Bejelentkezve?
Token
Felhasználó Azonosító
nem
Átdobás a bejelentke...
Profil adat betö...
Text is not SVG - cannot display

Közösségi oldal

Bejegyzések betöltése

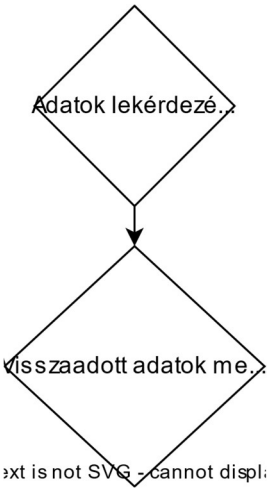


Bejegyzés létrehozása

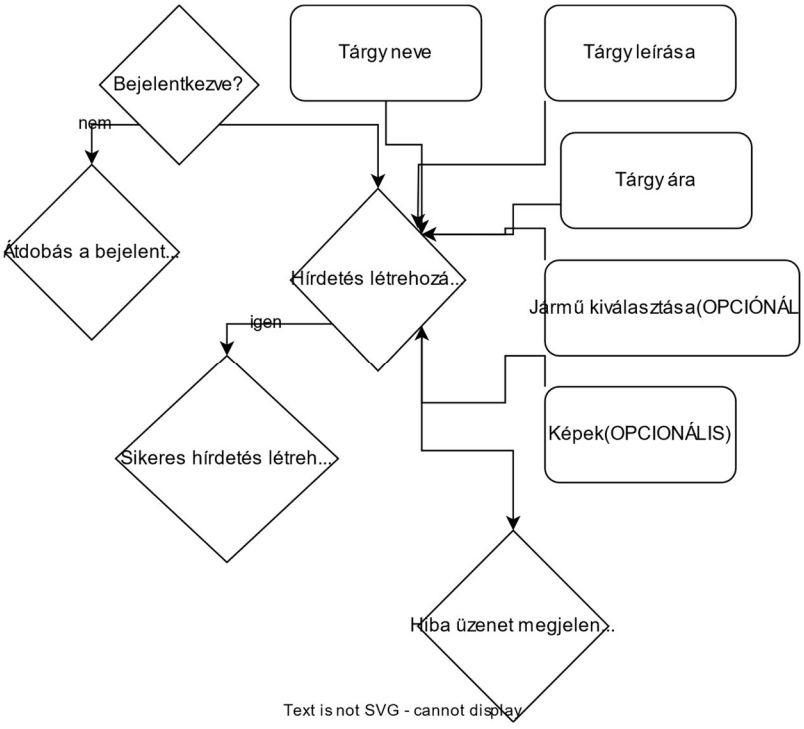


Közösségi Piac

Közösségi Piac betöltése

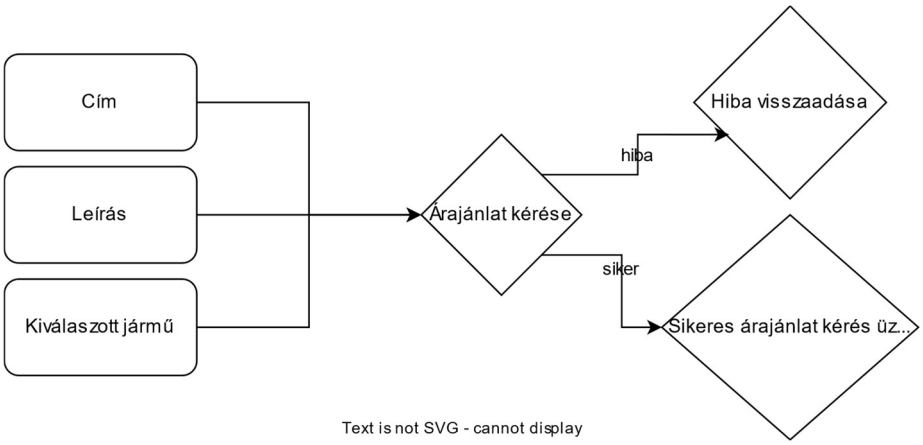


Közösségi Piac hírdetés létrehozása



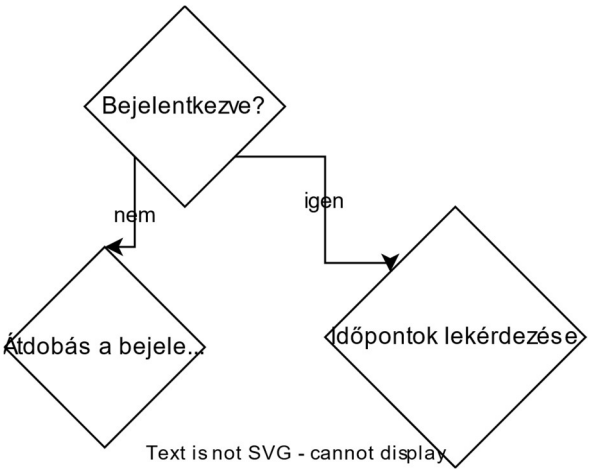
Árajánlat kérés

Árajánlat kérése

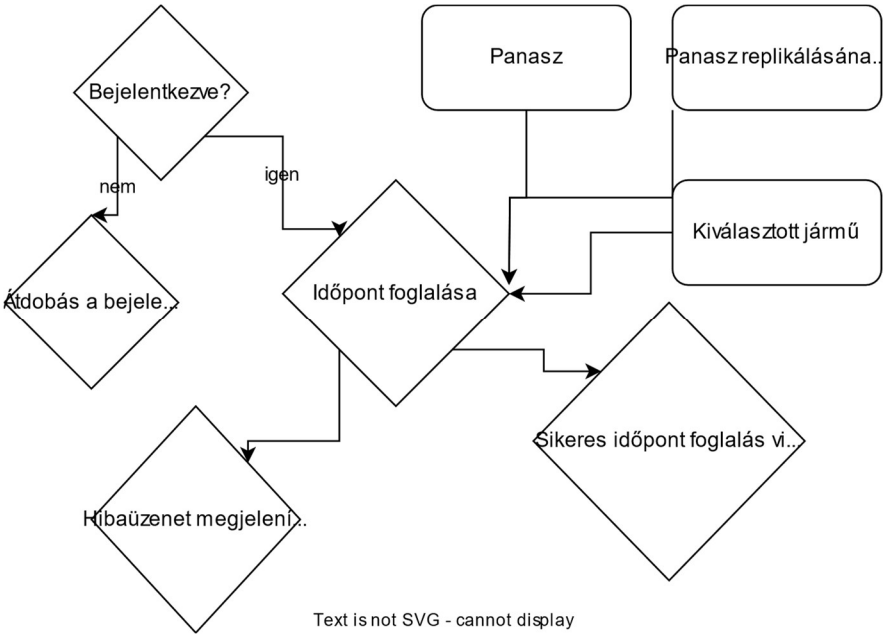


Időpont foglalás

Időpontok betöltése



Időpont foglalása



API Végpontok

/user

/register

Metódusa: POST

A szükséges paraméterek a következők:

- fullName
- email
- phone
- password

Az API ezen az elérési úton a következőt csinálja:

> Leellenőrzi, hogy az adott email szerep-e az adatbázisba. Ha nem szerepel, vissza ad egy „Az email cím már foglalt!” hiba kódot. szerepel, a következő dolog amit elvégez, az adatok adatbázisba bevitele. Ha sikeres az adatbevitel, vissza ad egy „Sikeres regisztráció” üzenetet. Ha a folyamat közbe hiba történik, akkor egy „Failed to register user!” hibát ad vissza, ami után a szerver beilleszti a hiba kódot ami okozta a kód futásának meghíusulását.

/login

Metódusa: POST

A szükséges paraméterek a következők:

- email
- password

Az API ezen az elérési úton a következőt csinálja:

> Elsőként, kiválasztja azt a sort az adatbázisból, ahol az email cím egyenlő a megadott email címmel. Ha nem létezik ilyen sor, akkor vissza ad egy „Hibás email cím vagy jelszó!” hibát. Ha létezik ilyen sor, a „password_verify($arg1, $arg2);” beépített PHP függvénnyel ellenőrzi, hogy a megadott jelszó megegyezik az adatbázisban megadott jelszóval. Ha nem, ugyan úgy vissza adja a „Hibás email cím vagy jelszó!” hibát. Ha megegyzik, vissza adja a tokent amely szükséges adott végpontokhoz, és a felhasználó adatait amit a weboldal használ.

/profile-data/{userid}

Metódusa: POST

A szükséges paraméterek a következők:

- userid

Az API ezen az elérési úton a következőt csinálja:

> Az adatbázisból lekérdezi azt a sort a user táblából, ahol az id egyenlo a megadott userid értékkel. Vissza adja a queryben megadott értékeket ha létezik a felhasználó.

/car

/load/all/{userid}

Metódusa: POST

A szükséges paraméterek a következők:

- userid

Az API ezen az elérési úton a következőt csinálja:

A userid paraméterben megadott értékkel kiválasztja az API az adatbázisból a felhasználóhoz kötött járműveket és visszaadja őket.

/delete

Metódusa: POST

A szükséges paraméterek a következők:

- userid
- carid

Az API ezen az elérési úton a következőt csinálja:

A user_car táblából, azt az értéket törli ki, ahol a táblának userid és a carid értéke egyenlő a

paraméterekben megadott userid és carid-vel. Majd a következő dolog amit csinál, a car táblában ahol az id egyenlő a paraméterben megadott carid-vel, azt törli.

/add

Metódusa: POST

A szükséges paraméterek a következők:

- userid
- brand
- model
- year
- licenseplate
- vin
- registration
- images

Az API ezen az elérési úton a következőt csinálja:

A megadott adatokból csinál egy JSON objektet, a vin paraméteren kívül, majd beviszi a car

adatbázisba, és az utolsólag beszúrt elem ID-jét lekéri, ami ebben az esetben az újonnan bevitt car. Ezt megjegyezve, az API bevisz egy sort a user_car adatbázisba, ahol a tábla userid és a carid értéke egyenlő a paraméterben megadottal.

/change/status

Metódusa: POST

A szükséges paraméterek a következők:

- carid
- status

Az API ezen az elérési úton a következőt cisnálja:

Az adott carid paraméterben megadot azonosítójú sort a car táblában módosítja, a status oszlopot a paraméterekben megadott status értékével teszi egyenlővé.

/community

/load-posts/{page}

Metódusa: POST

A szükséges paraméterek a következők:

- page

Opcionális paraméterek a következők:

- profilepage

Az API ezen az elérési úton a következőt csinálja:

> Ha a felhasználó a saját profil oldalán van, a profilepage paraméter használt, és csak a saját bejegyzéseit adja vissza az API. Ha viszont nem saját profil oldalán van, hanem a normális Közösségi oldalon, akkor minden bejegyzést visszaad.

/comments

Metódusa: POST

A szükséges paraméterek a következők:

- postid
- userid

Az API ezen az elérési úton a következőt csinálja:

> Az adatbázisból kiszűri azokat az adatokat, amelyeknél a community_posts_comments.postid egyenlő a paramétereknél megadott postidvel, és a community_posts_comments.userid egyenlő a paraméterekben megadott useridvel, hogy a felhasználó nevét és többi szükséges adatait vissza tudja adni.

/create-post

Metódusa: POST

A szükséges paraméterek a következők:

- userid
- title
- description
- images

Az API ezen az elérési úton a következőt csinálja:

> A maximum 3 kép elérési útját egybe teszi egy String típusú objektbe, vesszővel elválasztva, majd az adatbázisba beszúrja a paramétereket, az előbb említett string típusú objekt értékét.

/like

Metódusa: POST

A szükséges paraméterek a következők:

- postid
- userid

Az API ezen az elérési úton a következőt csinálja:

> Egy q nevű változóba eltárolja a következőt az adatbázisból: A sorok száma, ahol a community_post_likes táblában a postid és a userid egyenlő a paraméterekben megadottakkal. Mad ha ez a változónak a liked nevű értéke egyenlő nullával, beszúrja a community_post_likes táblába a postid és userid értékeket. Ha nem nulla az értéke, akkor kitörli a sorokat ahol a postid és a userid egyenlő a paraméterekben megadottakkal.

/comment

Metódusa: POST

A szükséges paraméterek a következők:

- postid
- userid
- comment

Az opcionális paraméterek a következők:

- images

Az API ezen az elérési úton a következőt csinálja:

> Az adatbázisba, a community_posts_comments nevű táblába beszűrja a szükséges paraméterek adatait, és az opcionális paraméterek adatait, ha meg vannak adva.

/posts/delete

Metódusa: POST

A szükséges paraméterek a következők:

- userid
- postid

Az API ezen az elérési úton a következőt csinálja:

> Megnézi, hogy a megadott userid és postid érték pár létezik-e a community_posts táblában. Ha nem, hibát ad vissza. Ha létezik, az adatbázisból az API törli a következő sorrendben:
>
> 1. community_post_likes
> 2. community_comments_likes
> 3. community_posts_comments
> 4. community_posts

/comment/like/{commentid}

Metódusa: POST

A szükséges paraméterek a következők:

- commentid
- userid

Az API ezen az elérési úton a következőt csinálja:

> Hasonlóan mint a bejegyszés likeolásnál, ha a commentid és userid érték pár létezik a community_comments_likes táblában, akkor kitörli az adott érték párt, ha viszont nem létezik, akkor beviszi azokat.

/ratings

/ratings/get
Metódusa: GET

Az API ezen az elérési úton a következőt csinálja:

> Egy avg nevű változóba lekéri az átlag értékét a count oszlopnak. Egy all nevű változóba, az összes értékelés számát menti el. Egy ratings változóba kiválasztja a felhasználó nevét, és az értékelés értékét, és kommentjét, a user táblát inner joinnal köti össze, user.id = ratings.userid. Sp nevű változóba az értékelések értékét, csillagok számaként menti el, majd ezt visszaadja.

/ratings/new
Metódusa: POST

A szükséges paraméterek a következők:

- userid
- star
- comment

Az API ezen az elérési úton a következőt csinálja:

> Adatbázis lekérdezések segítségével lekérdezi, hogy a felhasználó valóban igénybe vette e, már egyik szolgáltatásunkat. Ezt követően beszúrjuk az értékelést az adatbázisba.

/cdn
/cdn/get/{path}
Metódusa: GET

A szükséges paraméterek a következők:

- path

Az API ezen az elérési úton a következőt csinálja:

> Visszafejti a base64 titkosítással kódolt kép elérési útját, majd megnyissa a fájlt.
>
> Ezt követően beállításra kerül a fejléc típusa: image/png, így a kép megjelenik.

/marketplace
/marketplace/listings/load/{page}
Metódusa: POST

A szükséges paraméterek a következők:

- page

- userid

Az API ezen az elérési úton a következőt csinálja:

Elsősorban megtörténik a változók helyes típusúvá alakítása, majd ezt követően kiszámolásra kerül, hogy egy oldalon mennyi bejegyzés jelenjen meg. Adatbázis lekérdezés elkészítése metódusokkal, további lekérdezések lefuttatása INNER JOIN segítségével (táblák összekapcsolása), majd rendezésre kerülnek a lekérdezett adatok, majd visszaadja őket.

/marketplace/listings/load/admin/{page}

Metódus: POST

A szükséges paraméterek a következők:

- page
- userid

Az API ezen az elérési úton a következőt csinálja:

Megvizsgálásra kerül. hogy a felhasználó jogosult e a tartalom elérésez,

ammenyiben elérhet számára lekérdezésre kerülnek a jóváhagyásra váró hirdetéseket.

/marketplace/item/{id}

Metódus: GET

A szükséges paraméterek a következők:

- id

Az API ezen az elérési úton a következőt csinálja:

Az id alapján lekérdezésre kerül egy adott hirdetés.

/marketplace/listings/approve

Metódus: POST

A szükséges paraméterek a következők:

- userid
- rankid
- listingid

Az API ezen az elérési úton a következőt csinálja:

A sikeres azonosítás után a hirdetés elfogadásra kerül.

/marketplace/listings/decline

Metódus: POST

A szükséges paraméterek a következők:

- userid
- rankid
- listingid

Az API ezen az elérési úton a következőt csinálja:

A sikeres azonosítás után a hirdetés elutasításra kerül.

/marketplace/listings/delete

Metódus: POST

A szükséges paraméterek a következők:

- userid
- listingid

Az API ezen az elérési úton a következőt csinálja:

A sikeres azonosítás után a hirdetés törlésre kerül.

/marketplace/images/{postid}/{index}

Metódus: GET

A szükséges paraméterek a következők:

- postid
- index

API ezen az elérési úton a következőt csinálja:

Megjeleníti egy adott hirdetés fényképet, az index határozza meg, hogy melyik jelenik meg.

/requests

/requests/all

A szükséges paraméterek a következők:

- userid

API ezen az elérési úton a következőt csinálja:

Megvizsgálja a felhasználót, hogy jogosult-e az adatok lekérdezéséhez. Ammenyiben ez sikeres, akkor az adatok lekérdezése, majd visszaadása történik meg.

/requests/load

Metódus: POST

A szükséges paraméterek a következők:

- userid

API ezen az elérési úton a következőt csinálja:

> Ez a végpont visszaadja a felhasználóhoz tartozó összes kérést, azok státuszát és válaszait. A lekérdezés a user_car táblához is csatlakozik, hogy a megfelelő kocsikhoz rendelt kérések jelenjenek meg.

/requests/answer

Metódus: POST

Szükséges paraméterek:

- userid
- requestid
- response

Működés:

> Ez a végpont lehetővé teszi a kérésre való válaszadást. Ha a felhasználó jogosult, a válasz elmentésre kerül a request_replies táblába, és a kérés állapota frissül. Ezen kívül e-mail értesítés is küldésre kerül a felhasználó részére.

/requests/send/anon

Metódus: POST

Szükséges paraméterek:

- title
- description
- email

API ezen az elérési úton a következőt csinálja:

Ez a végpont lehetővé teszi egy anoním felhasználó számára, hogy kérdést küldjön. Az anoním felhasználóhoz kapcsolódó e-mail cím alapján a kérés mentésre kerül.

/requests/admin/delete
Szükséges paraméterek:

- userid
- requestid

Működés:

Ez a végpont lehetővé teszi az adminisztrátorok számára, hogy töröljenek egy kérést. Az admin jogosultságot a felhasználó rangjának ellenőrzésével biztosítja (rankid = 1). Ha a felhasználó nem admin, akkor 405-ös hibát kap.

/appointments

/appointments/get
Metódus: POST

Szükséges paraméterek:

- date

Működés:

- A kérést POST metódussal kell küldeni.

- A válaszban a két lista érkezik:

  - Elérhető időpontok (times)

  - Foglalt időpontok (taken_times)

/appointments/create
Metódus: POST

Szükséges paraméterek:

- userid (int): A felhasználó ID-ja.

- carid (int): Az autó ID-ja.

- complaint (string): A panaszos észrevétele.

- stepstorep (string): A szükséges lépések, amelyeket el kell végezni.

- date (string): A kívánt időpont a találkozóhoz.

- timeid

Működés:

- A kérést POST metódussal kell küldeni.

- A rendszer rögzíti a találkozót és a kapcsolódó adatokat az adatbázisba.

- A válaszban egy sikeres státuszkód és üzenet érkezik.

/appointments/get-all

Metódus: GET

Szükséges paraméterek: Nincs szükség paraméterekre.

Működés:

- A kérést GET metódussal kell küldeni.

- A válasz tartalmazza az összes találkozó részleteit, beleértve a felhasználó nevét, email címét, telefonját, autó adatait, a találkozó időpontját, a panaszokat, valamint a szükséges lépéseket.

/appointments/set-car-state/{carid}

Metódus: POST

Szükséges paraméterek:

- carid

- status

Működés:

- A kérést POST metódussal kell küldeni.

- A kért autó ID-ját és az új állapotot frissíti az adatbázisban.