Fejlesztői dokumentáció

A program 3 backend fájllal dolgozik, 2-2 endpointtal:

- users.php
- parcels.php
- connect.php

connect.php:

A connect.php kódrész egy osztályt tartalmaz, amely felelős a MariaDB/MySQL adatbázissal való kapcsolat létrehozásáért. A program más szerveren való használatához szükséges a kódban a csatlakozásnál a 'localhost' értéket átírni.

users.php

A users.php kódrész felelős a user táblával való kommunikálásért. 2 végpont található benne:

- getUsers:
    - ez a végpont egy olyan function, ami a users táblából kiolvassa az összes adatot az 'id' oszlop szerint rendezve, majd ezzel json formátumban visszatér. A végpont úgy elérhető egyszerűen, ha az oldalunkon, ahol el akarjuk érni, írunk egy JavaScript kódrészt, amely így néz ki:

```
<script>
    fetch('users.php', {
    method: 'GET',
    })
    .then(response => response.json())
    .then(data => {
        console.log(data);
    })
    .catch(error => console.error('Hiba történt:', error));
</script>
```

    Ezt a kódrészletet módosíthatjuk a felhasználás módja szerint. A script ezen formája egy GET hívással lekéri a users/getUsers végpontot, aminek eredményét a console-ra kiírja.
- insertUser:
    - ez a végpont egy olyan function, amely POST metódussal megkap egy felhasználóról minden adatot, majd ad neki egy egyedi ID-t és beilleszti a users táblába.
    - ahhoz, hogy elérjük ezt a végpontot, szükségünk van egy hívásra ami POST metódussal történik. Ezt legkönnyebben egy form html elemen keresztül tudjuk megtenni, ahol a form elemei a megfelelő id-kkal vannak ellátva, és a küldő gombnak a metódusa POST.

parcels.php

A parcels.php kódrész felelős a parcels táblával való kommunikációért. 2 végpont található benne:

- getParcel:
    - ez a végpont egy olyan function, amely GET metódussal érhető el, és vár egy payload-ot, ami a csomag száma. Ennek hiányában a function le sem fut. Ezek után a táblából lekérdezi azokat a sorokat, amelyeknek a parcel_number mezője megegyezik a payloadban megkapottal, ami mivel az esetünkben egy egyedi hexadecimális szám így mindenképpen egy rekordot kapunk vissza. A rekord userid mezője alapján még lekérdezzük a users táblából is a csomaghoz rendelt felhasználót, majd ezeket json formátumban visszaadja.
    - a végpont eléréséhez szintén JavaScript kóddal elérhető, ahol a fetch-nek GET metódusnak kell lennie és rendelkeznie kell egy payloaddal, ami a csomagnak a száma
- insertParcel:
    - ez a végpont egy olyan function, ami POST metódussal érhető el és a payloadja 2 adat, a csomag mérete (size) és a felhasználó id-ja (userid). A function ezen kívül még generál egy hexadecimális számot a fájlban megtalálható generateRandomHex function használatával, és addig készít újat, ameddig található egyező csomagszám az adatbázisban, azért, hogy a csomagszám egyedi legyen. Ez lesz a csomagnak a száma (parcel_number). Ezek után beszúrja a parcels táblába a kapott értékeket.
    - ezt a végpontot elérhetjük szintén egy POST metódussal ellátott form html elemmel, ahol az input mezőket megfelelő id-kkal látunk el, és a metódusnak 2 payloadot adunk, amelyek a size és a felhasználó id-ja.

Ezeken kívül még megtalálható mindkét fájlban egy metódus, ami a végpont hívásakor a hívás metódusa alapján eldönti melyik functiont hívja, és ellenőrzni, hogy léteznek-e a payloadok, valamint hibát dob hibás hívás esetén.

Futtatás

A futtatáshoz szükséges elsősorban egy MySQL vagy MariaDB kapcsolat, ahova be vannak importálva a parcels és a users táblák. Ezen túl a futtatáshoz szükséges egy webszerver, ahol ezek a végpontok elhelyezkednek, optimális esetben ez a szerver ne egyezzen azzal a szerverrel, ahol a frontendje helyezkedik el a weboldalunknak.