Feladatok

1. Telepíts egy json szervert, az alábbi lépéseket követve:

- Ha nincs a gépeden nodeJS akkor töltsd le, és telepítsd innen az LTS verziót: NodeJS letöltése
- Ezután add ki a konzolon az alábbi parancsot: `npm install -g json-server`
- A json-server dokumentációját itt találod: LINK
- A Mockaroo oldalán generálj le egy json file-t, ami 100 data id, firstName, lastName objektumot tartalmaz, a gyökérelem neve users
- a JSON filet helyezd a projekt mappádba
- indítsd el a json szervert konzolból az alábbi paranccsal: `json-server --watch aJsonFileUtvolana`
  Ezzel van is egy szervered, ami a kiszolgája majd az alkalmazásod. A szerver a 3000-es porton fut, ha a JSON-be a root elem users volt, és
  le akarod kérdezni az 1-es indexű elemet akkor csak ennyit kell beírnod a böngészőbe:
  (http://localhost:3000/users/1)[http://localhost:3000/users/1]
  és megjelenik az 1-es id-jú user összes adata.

Készíts egy `User` nevű class-t, ami rendelkezik az alábbi statikus privát adattaggal rendelkezi:

- `url` : a teljes URL a `users` erőforrásra

és statikus metódusokkal:

- `post(user)`: fetch-el küldj kérést a szervernek, ami létrehoz egy új usert a paraméterként megadott adatokkal
- `get(id)`: fetch-el küldj kérést a szervernek, ami visszaadja a paraméterként megadott id-jú user adatait
- `put(user, id)`: fetch-el küldj kérést a szervernek, ami lecserélni a paraméterként megadott id-jú user adatait a szintén paraméterként megadott adatokra
- `delete(id)`: fetch-el küldj kérést a szervernek, ami törli a paraméterként megadott id-jú user adatait

A paraméterben lévő `user` mindig egy objektum `firstName`, és `lastName` tulajdonságokkal
Mindenhol legyen kivételkezelés is!
Amennyiben a metódus futása sikeres a szerverről visszakapott JSON adatot objektummá alakítva írd ki a konzolra.
Hiba esetén a hibaüzenet szerepeljen a konzolon, tehát csak a `message`.

2. Az előző példát alakítsd át úgy, hogy a kód újra felhasználható(bb) legyen!
   Az alábbiak szerint:

- a `User` class-t nevezd át `Api`-ra
- a metódusok ne legyenek statikusak
- példányosításkor kapja megy a constructor az útvonalat (ne a teljes URL-t, cak az utolsó részt, pl: `users`)

- ha a kérés sikeresen lefutott, akkor ne a konzolra írd ki a objektummá alakított választ, hanem ez legyen a metódus visszatérési értéke
- hibakezelést itt nem kell megvalósítani
- hozz létre egy új class-t, ez lesz most a `User`
- a `User` az alábbi metódusokkal rendelkezzen:
  - `create(user)`: meghívja az `Api` példányának `post()` metódusát, a konzolra írja a visszatérési értékét az alábbi formában: 'created: Hans Zimmer'
  - `read(id)`: meghívja az `Api` példányának `get()` metódusát, a konzolra írja a visszatérési értékét az alábbi formában: 'read: Hans Zimmer'
  - `update(user, id)`: meghívja az `Api` példányának `put()` metódusát, a konzolra írja a visszatérési értékét az alábbi formábna: 'updated: Hans Zimmer'
  - `delete(id)`: meghívja az `Api` példányának `delete()` metódusát, a konzolra írja a visszatérési értékét az alábbi formában: 'deleted: Hans Zimmer'
- hibakezelést itt kell megvalósítani, hiba esetén maradhat a hibaüzenet konzolra írása
- készíts egy `main.js` állományt, amiben az alkalmazás összeáll (itt legyen importálva és példányosítva minden), ezt az állományt kell az `index.html`-be belinkelni

3. Az előző példát módosítsd, egészítsd ki a következőkkel:

- a html fájlba linkeld be a Bootstrap CSS fájlját
- hozz létre egy `Messenger` class-t, ami minden műveletnél annak eredményétől függően már a DOM-ba jelenít megy egy üzenet a művelet sikerességéről, vagy sikertelenségéről
- a `Messenger` két privát adattaggal rendelkezik, melyeknek példányosításkor lehet értéket adni:
  - `container`: milyen szelektorú elemen belül jelenjenek meg az üzenetek, alapértelmezett érték: '#messages'
  - `hideAfterMiliSec`: hány másodperc múlva kell az üzenetet elrejteni, alapértelmezett érték: 5000
- a `Messenger`nek két kötelező metódusa van (plusz amennyit írsz):
  - `error(message)`: megjelenít egy Bootstrap-es alert elemet a megfelelő színnel, és a paraméterként adott szöveg tartalommal
  - `success(message)`: megjelenít egy Bootstrap-es alert elemet a megfelelő színnel, és a paraméterként adott szöveg tartalommal
- az `User` példányosításkor kapjon meg egy `Messenger` példányt, és ha siker esetén a `success` metódust hívja meg a konzolra való logolás helyett
- hiba esetén a `Messenger` példány `error()` metódusát hívja meg, és írja a DOM-ba a hibaüzenetet
- a `main.js`-t egészítsd ki, a `Messenger` példányosításai is itt történjenek