Általános leírás

A DrinkIsland egy komplex webalkalmazás, amelyet Angular és Laravel keretrendszerek segítségével fejlesztettek ki. Célja, hogy széles körű ismereteket nyújtson a felhasználóknak a koktélozás világáról.

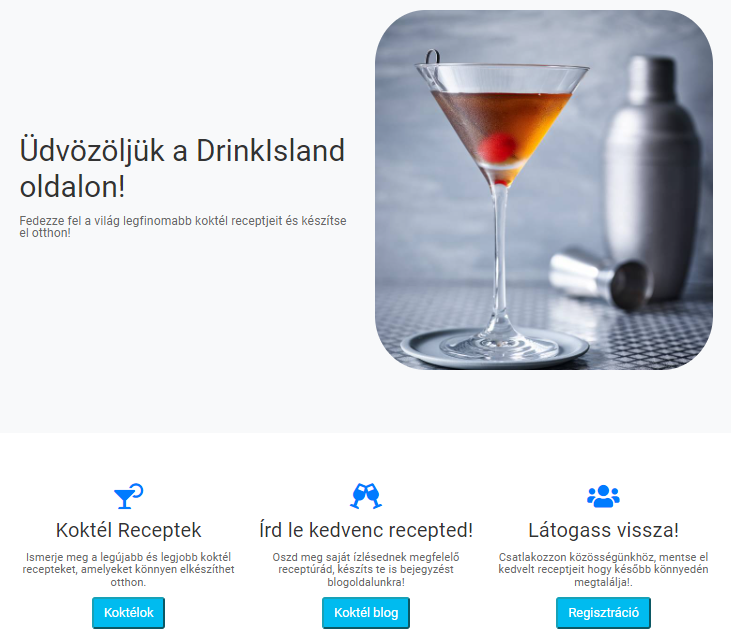
A weboldal fő célja, hogy segítsen a felhasználóknak felfedezni és megismerni a különféle koktélok világát. Legyen szó klasszikus receptekről, újító ötletekről vagy érdekességekről, a DrinkIsland mindenkit inspirál és segít abban, hogy új ízeket és italokat fedezzen fel.

Fontos megjegyezni, hogy mivel az alkoholos témákat is feldolgozza, a weboldal célközönsége 18 éven felüliek. Ennek megfelelően a felhasználóknak be kell mutatniuk az életkorukat, mielőtt teljes körű hozzáférést kapnának a tartalomhoz.

A DrinkIsland egy interaktív platform, ahol a felhasználók saját koktél recepteket is megoszthatnak

Home komponens

* Az első szakasz üdvözli a felhasználót és bemutatja az oldalt egy képpel.
* A második szakasz bemutatja az oldal főbb jellemzőit, mint például a koktél receptek, a blog és a regisztráció lehetőségét.



Navbar Komponens

A *NavComponent* felelős a navigációs sáv megjelenítéséért és működéséért az alkalmazásban. Az összecsukható menü és a navigációs elemek kezelése, valamint a felhasználói jogosultságok alapján történő elemek megjelenítése is ebben a komponensben történik.

**Tulajdonságok:**

* *isMenuOpen* - Logikai érték, amely azt jelzi, hogy az összecsukható menü éppen nyitva vagy zárva van.
* *isNavbarOpen* - Logikai érték, amely azt jelzi, hogy a navigációs sáv éppen nyitva vagy zárva van.
* *IsAdmin* – Logikai érték, amely adatbázis alapján eldönti, hogy azt a menüpontot az Admin vagy a felhasználó láthatja.
* isLoggedin – mely kiírja azokat a menüpontokat, melyeket bejelentkezés után lát a felhasználó/admin.
* Loggedout - mely kiírja azokat a menüpontokat, melyeket bejelentkezés nélkül lát a felhasználó/admin.

**Metódusok:**

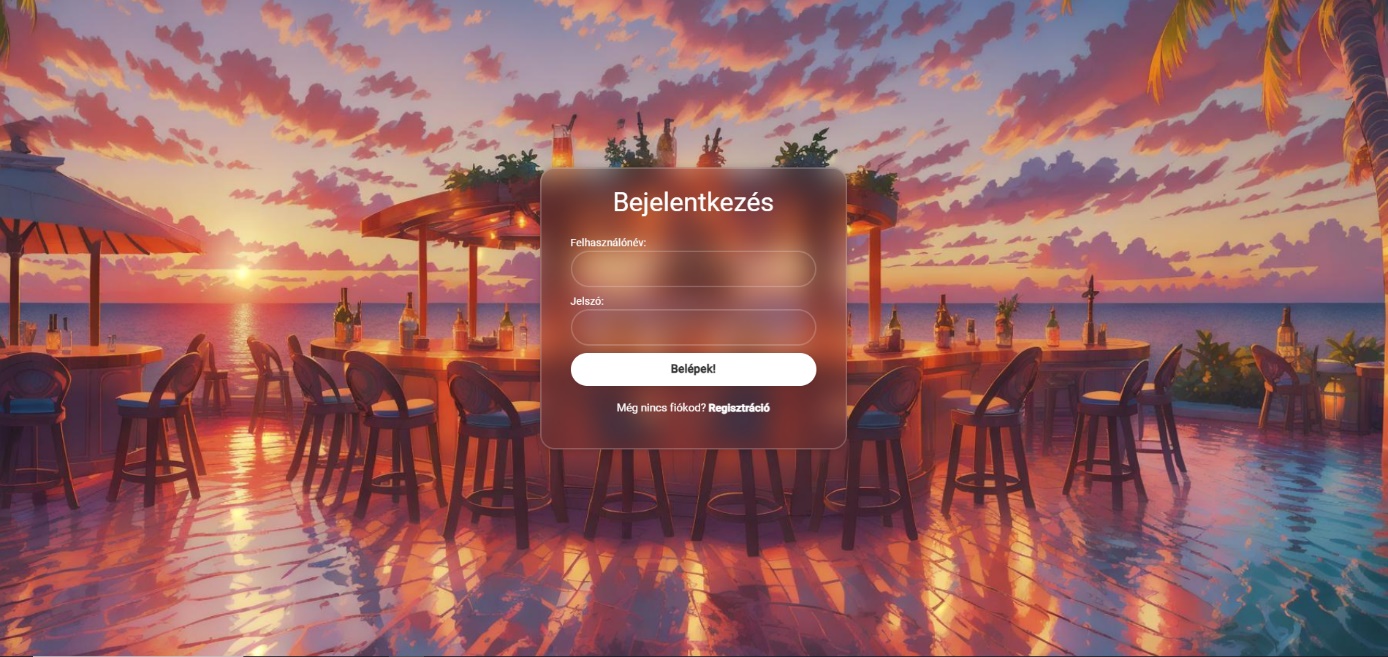
* *toggleMenu()* - Az összecsukható menü állapotának váltása (nyitva vagy zárva).
* *toggleNavbar()* - A navigációs sáv állapotának váltása (nyitva vagy zárva).
* *logout()* - Kijelentkezési funkció, ami a *Loginservice* segítségével végzi a kijelentkezést.

Login Komponens

A „Bejelentkezés” gombra kattintva továbbvisz minket az bejelentkezéshez, itt kell megadni a felhasználó nevet, illetve jelszót. Bejelentkezés után a Kezdőlapra irányít az oldal.

Működése:

Bejelentkezés során meg kell adni a következőket:

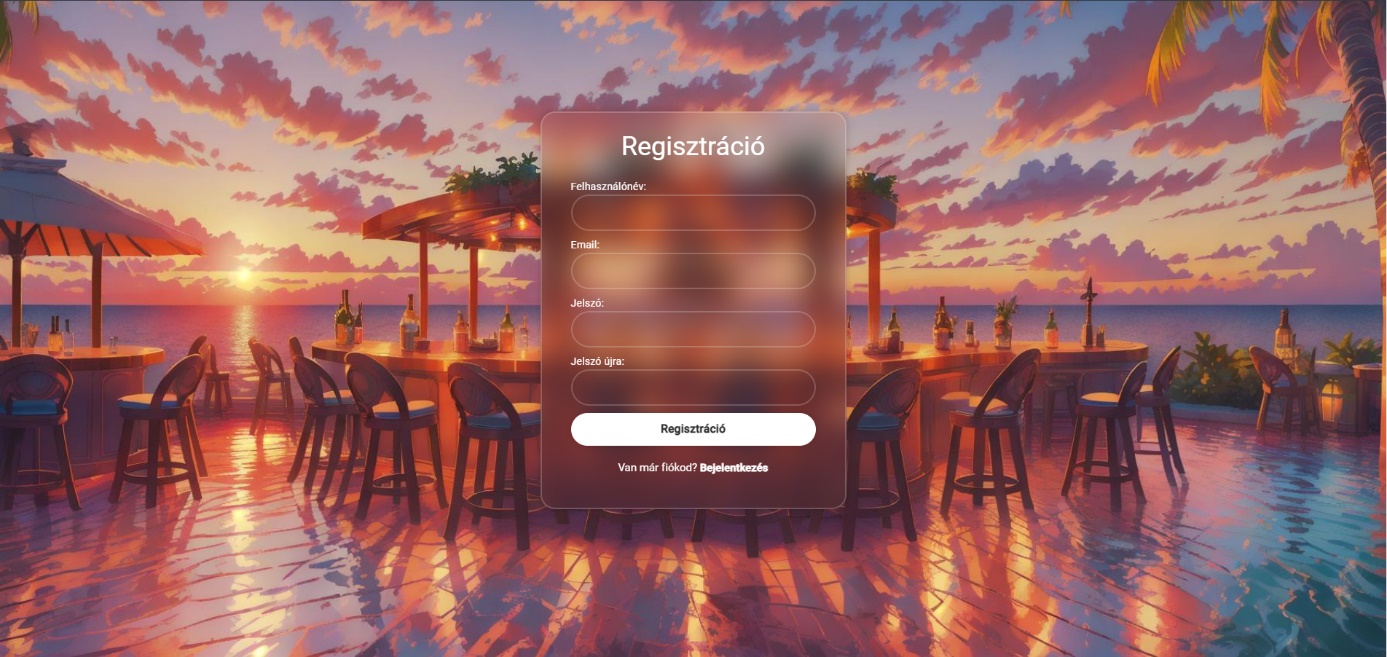
* Felhasználónév
* jelszó

JustSignup Komponens

Regisztrácóhoz szükséges komponens, ahol a szükséges adatok kitöltése után lehet regisztrálni vagy tovább lehet navigálni a bejelentkezésre, ha az illetőnek van már fiókja.

Működése:

Regisztráció során meg kell adni a következőket:

* Felhasználónév:
  + Ha egy másik felhasználó regisztrált már ugyanazzal a felhasználónévvel, nem lehet újból használni.
* Email:
* Jelszó:
  + A jelszót meg kell erősíteni és csak akkor fogadja el ha mind a két jelszó leírása azonos.

Blog Komponens

A Blog Komponens felelős a blogbejegyzések megjelenítéséért.

**Felhasználói felület (HTML)**

A komponens HTML kódja egy bejegyzések listáját tartalmazza, amelyet dinamikusan generálunk az **\*ngFor** direktívával. Minden bejegyzésnek van egy címe (**post.title**) és szöveges tartalma (**post.text**), amelyeket a komponens HTML kódban jelenítünk meg.

**Komponens logikája (TypeScript)**

A komponens TypeScript kódja felelős a blogbejegyzések lekérdezéséért a szolgáltatástól, és a megfelelő adatok megjelenítéséért a felhasználói felületen. Az **ngOnInit** metódus meghívja a **getPosts** metódust, amely lekéri az összes bejegyzést a szolgáltatástól, majd eltárolja azokat a **posts** változóban, amelyet a HTML kódban használunk az adatok megjelenítésére.

Calculator Komponens

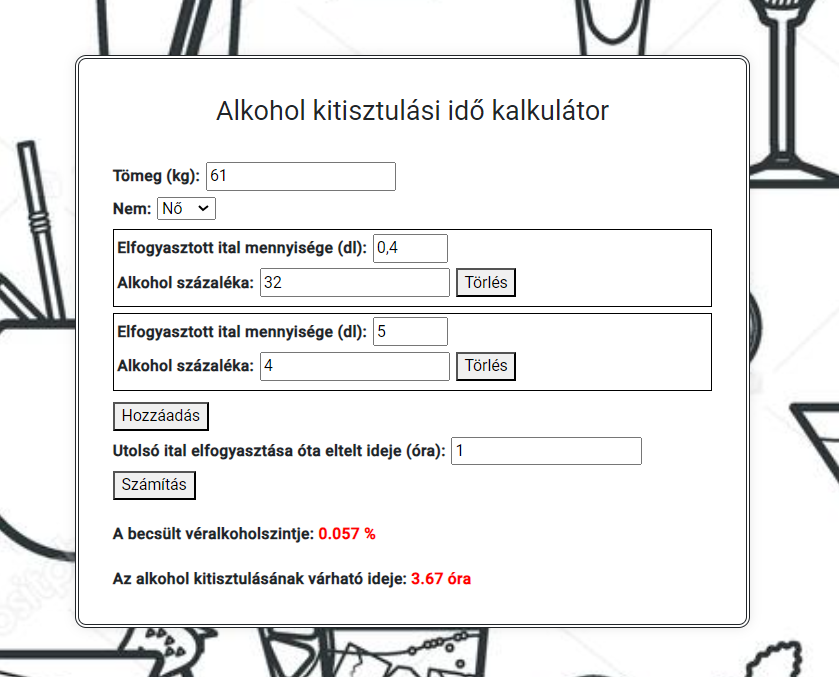
Az Alkohol kitisztulási idő kalkulátor komponens egy űrlapot biztosít a felhasználók számára, hogy kiszámítsák a becsült véralkoholszintjüket és az alkohol kitisztulásának várható idejét. A felhasználóknak lehetőségük van megadni a súlyukat, nemüket, az elfogyasztott italok mennyiségét és alkoholtartalmát, valamint az utolsó ital elfogyasztása óta eltelt időt.

**Felhasználói felület (HTML)**

Az űrlap lehetővé teszi a felhasználók számára, hogy megadják a súlyukat, nemüket, az elfogyasztott italok mennyiségét és alkoholtartalmát, valamint az utolsó ital elfogyasztása óta eltelt időt. Az "Hozzáadás" gomb segítségével új italokat adhatnak hozzá az űrlaphoz, és az "Eltávolítás" gomb segítségével törölhetik azokat. A "Számítás" gombra kattintva a rendszer kiszámítja a becsült véralkoholszintet és az alkohol kitisztulásának várható idejét, és ezeket megjeleníti a felhasználó számára.

**Komponens logikája (TypeScript)**

A komponens TypeScript kódja felelős az italok hozzáadásáért, eltávolításáért és a becsült véralkoholszint, valamint az alkohol kitisztulásának várható idejének kiszámításáért. A **CalculatorService** segítségével a komponens kiszámítja az alkohol kitisztulásának várható idejét és a becsült véralkoholszintet a megadott adatok alapján.



User-blog Komponens

Az új blogkomponens segítségével felhasználók tudnak posztokat hozzáadni, frissíteni és törölni az általuk létrehozott posztokat.

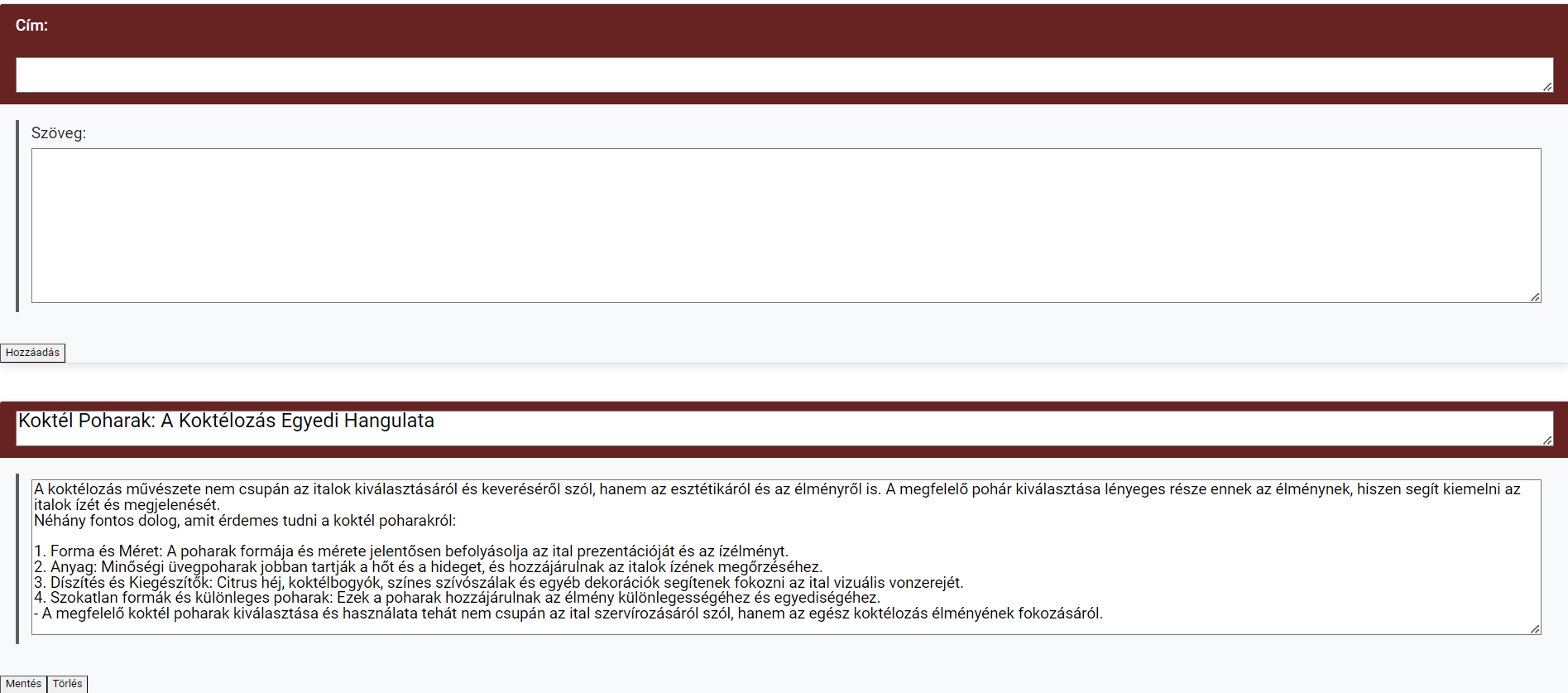
**Felhasználói felület (HTML)**

Az űrlap lehetővé teszi a felhasználók számára, hogy címet és szöveget adjanak meg az új poszthoz. Az "Mentés" gombra kattintva a rendszer hozzáadja az új posztot a listához.

A már meglévő posztok listája is megjelenik az oldalon. Minden poszthoz tartozik egy cím és egy szöveg mező, amelyek szerkeszthetők. Minden poszthoz tartozik egy "Mentés" gomb a módosítások mentéséhez, valamint egy "Törlés" gomb a poszt törléséhez.

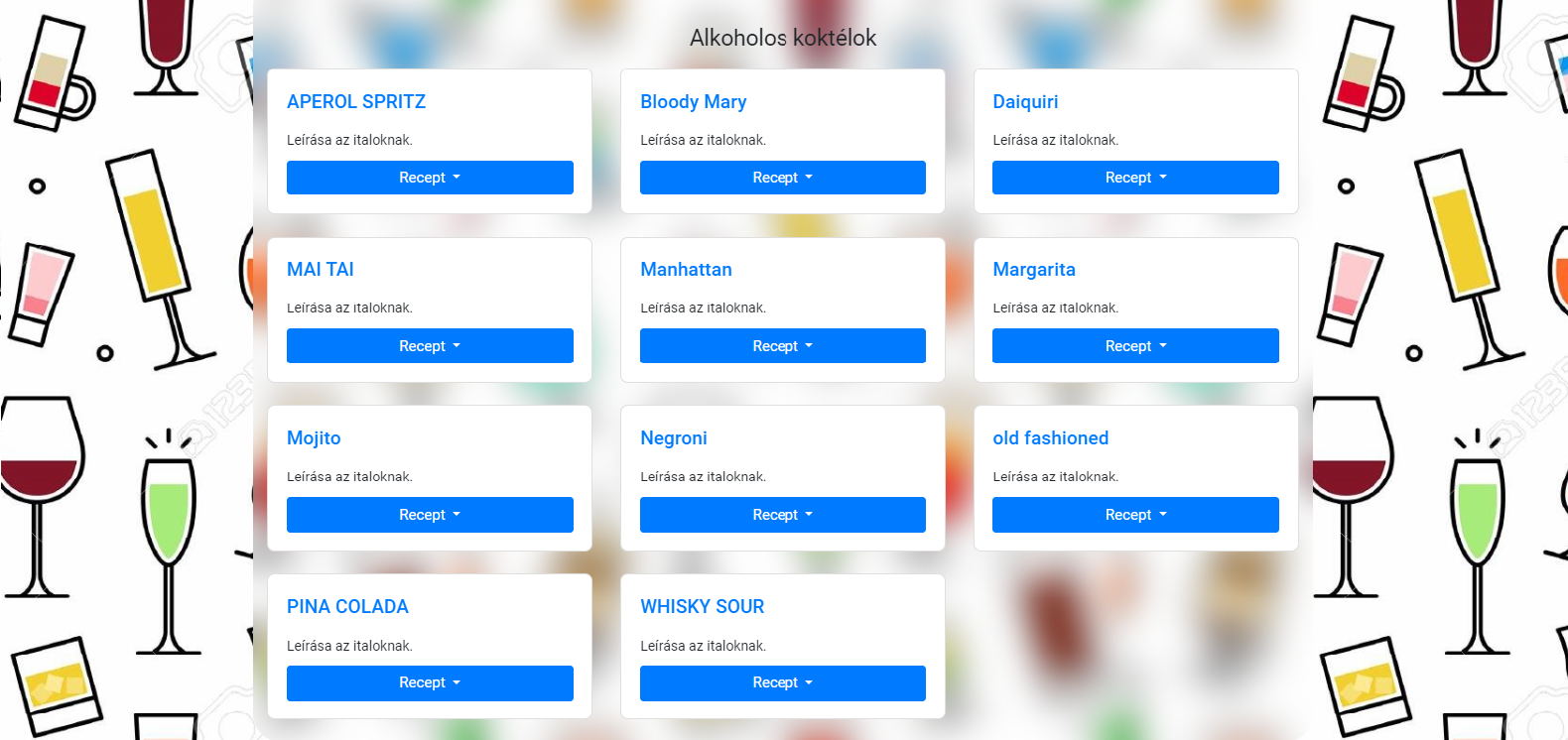
**Komponens logikája (TypeScript)**

A komponens TypeScript kódja felelős az új poszt hozzáadásáért, meglévő posztok frissítéséért és törléséért. Az adatok mentése, frissítése és törlése az **addPost**, **updatePost** és **deletePost** metódusok segítségével történik, amelyek hívják a megfelelő szolgáltatás metódusait.

A **loadPosts** metódus felelős a már meglévő posztok betöltéséért az oldal megnyitásakor.

Alcoholic-drinks Komponens

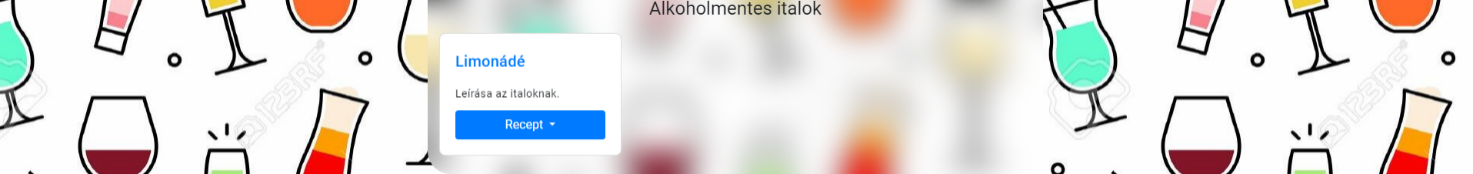
Ez a komponens kilistázza az összes olyan típusú koktélt, amely alkoholt tartalmaz.

Az adatbázist a dataservice-ből importáltuk be, és a Alcoholic-drinks component.ts-ben van rá metódus írva. \*NgFor-al hivatkozunk az adatbázisra html-ben, és íratjuk ki.

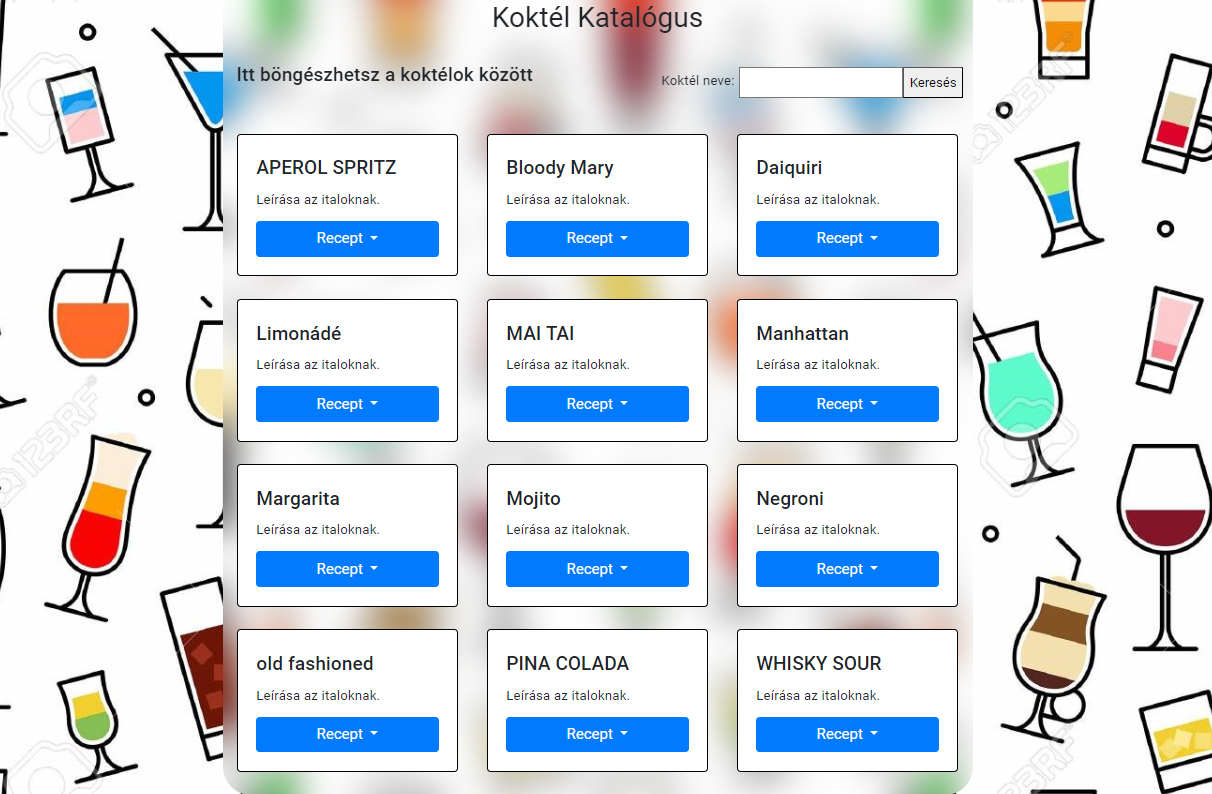
Non-Alcoholic-drinks komponens :

Ez a komponens kilistázza az összes olyan típusú koktélt, amely nem tartalmaz alkoholt.

Az adatbázist a dataservice-ből importáltuk be, és a Non-Alcoholic-drinks component.ts-ben van rá metódus írva. \*NgFor-al hivatkozunk az adatbázisra html-ben, és íratjuk ki.



Detailed-search Komponens :

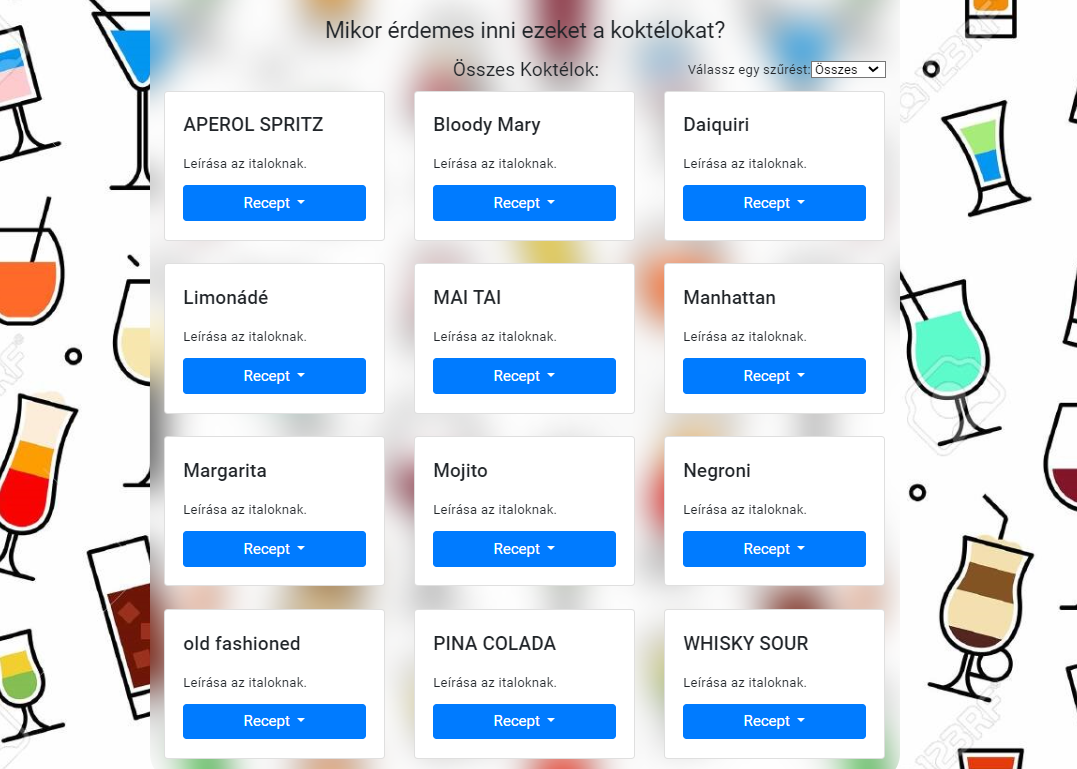
Ezen az oldalon a felhasználó megadhat egy szövegdobozba egy szöveget, amely alapján kilistázz azt a koktélt/koktélokat, amiknek a neve tartalmazza azt a szöveget. Az adatbázist a dataservice-ből importáltuk be, és a detailed-searchcomponent.ts-ben van rá metódus írva(getNonAlcoholicdata()). \*NgFor-al hivatkozunk az adatbázisra html-ben, és íratjuk ki.

Type-search Komponens

Ezen az oldalon a felhasználó megadhat egy szövegdobozba egy szöveget, amely alapján kilistázz azt a koktélt/koktélokat, amiknek a neve tartalmazza azt a szöveget. Az adatbázist a dataservice-ből importáltuk be, és a type-searchcomponent.ts-ben vannak rá metódus írva. A metódusok nevei:

* getData()
* getTeliData()
* getNyariData()
* getEdesData()
* getSavanyuData()
* getKeseruData()

\*NgFor-al hivatkozunk az adatbázisra html-ben, és íratjuk ki.



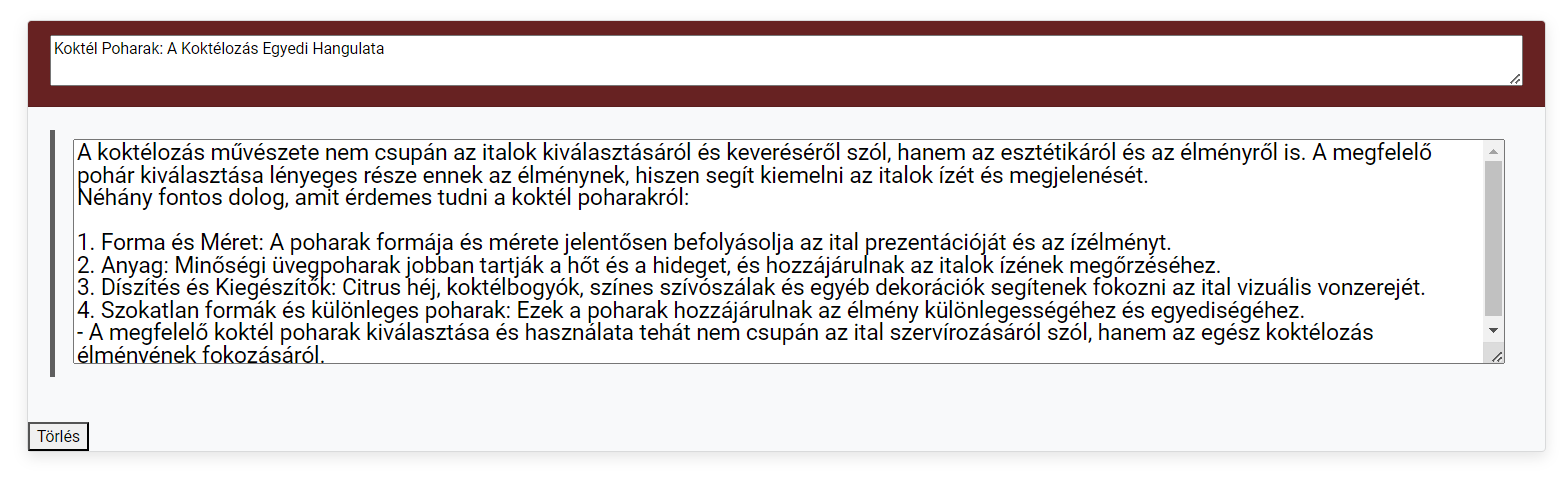
ADMIN | Admin Blog Komponens

Az Admin Blog komponens felelős a blogbejegyzések kezeléséért az adminisztrátor számára. Ennek a komponensnek a fő feladata az adminisztrátor számára biztosítani a blogbejegyzések listázását, szerkesztését és törlését.

Az admin kezelő felülete (HTML)

A komponens HTML kódja egy div elemet tartalmaz, amelynek osztályneve "blog -component". Ez a div egy tömb ciklusban megjeleníti a blogbejegyzéseket kártya formájában. Minden kártya tartalmazza a bejegyzés címét és szövegét egy textarea elemen belül, amelyek readonly módban vannak, így nem szerkeszthetők. Emellett minden kártyán van egy "Törlés" gomb is, amely lehetővé teszi a bejegyzés törlését.

Komponens logikája (TypeScript)

A komponens TypeScript kódja felelős a blogbejegyzések lekérdezéséért, megjelenítéséért, szerkesztéséért és törléséért. A komponens inicializálásakor meghívódik a **loadPosts** metódus, amely lekéri a blogbejegyzéseket a szolgáltatástól. Az **updatePost** és **deletePost** metódusok a blogbejegyzések szerkesztését és törlését végzik el, majd frissítik a megjelenített listát.

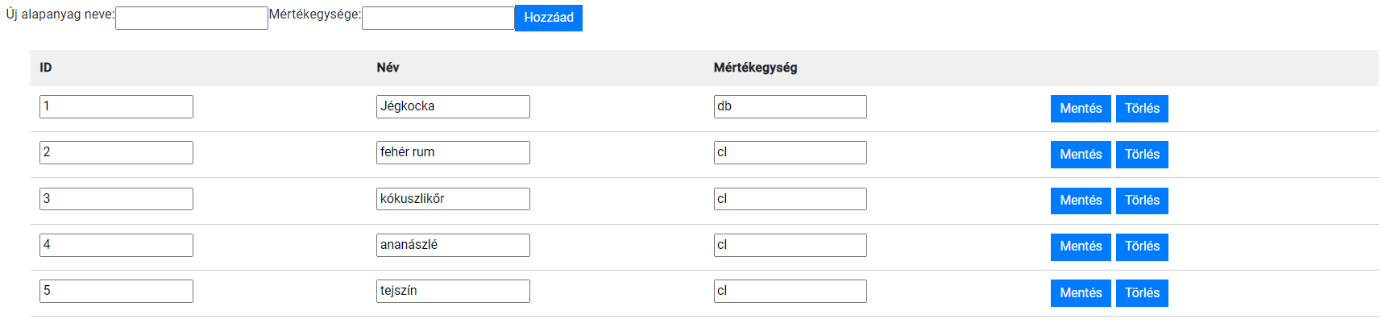
ADMIN | Alapanyag Komponens

Az Alapanyag Komponens felelős az alapanyagok kezeléséért, amelyeket a felhasználók hozzáadhatnak, szerkeszthetnek és törölhetnek az alkalmazásban.

**Felhasználói felület (HTML)**

A komponens HTML kódja egy űrlapot és egy listát tartalmaz. Az űrlap lehetővé teszi az új alapanyag hozzáadását, amelynek meg kell adni a nevét és mértékegységét. A lista megjeleníti az összes alapanyagot, minden alapanyaghoz tartozik egy ID, név és mértékegység mező. Minden alapanyag mellett találhatóak gombok a szerkesztéshez és törléshez.

**Komponens logikája (TypeScript)**

A komponens TypeScript kódja felelős az alapanyagok lekérdezéséért, megjelenítéséért, hozzáadásáért, szerkesztéséért és törléséért. Az **ngOnInit** metódus meghívja a **loadAlapanyagok** metódust, amely lekéri az összes alapanyagot a szolgáltatástól. Az **addAlapanyag**, **updateAlapanyag** és **deleteAlapanyag** metódusok felelnek az alapanyagok hozzáadásáért, szerkesztéséért és törléséért, majd frissítik a megjelenített listát.

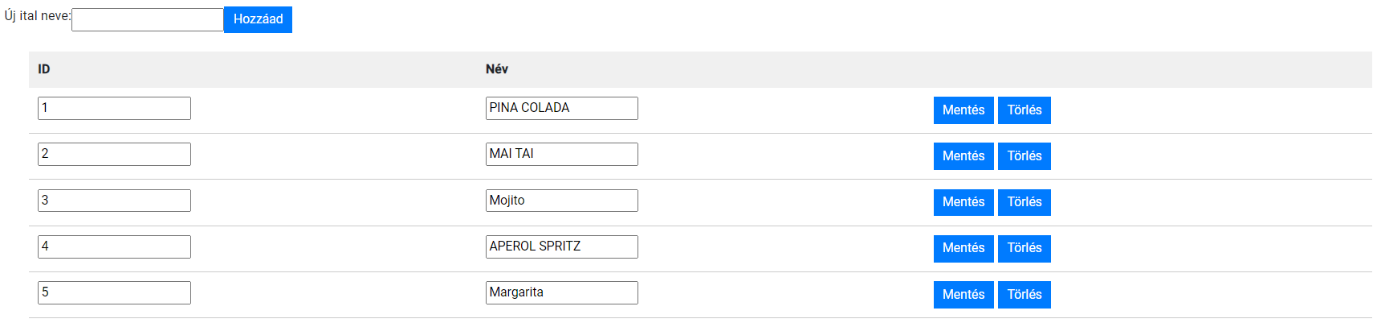
ADMIN | Ital Komponens

Az Ital Komponens felelős az italok kezeléséért, amelyeket a felhasználók hozzáadhatnak, szerkeszthetnek és törölhetnek az alkalmazásban.

**Felhasználói felület (HTML)**

A komponens HTML kódja egy űrlapot és egy listát tartalmaz. Az űrlap lehetővé teszi az új ital hozzáadását, amelynek meg kell adni a nevét. A lista megjeleníti az összes italt, minden italhoz tartozik egy ID és név mező. Minden ital mellett találhatóak gombok a szerkesztéshez és törléshez.

**Komponens logikája (TypeScript)**

A komponens TypeScript kódja felelős az italok lekérdezéséért, megjelenítéséért, hozzáadásáért, szerkesztéséért és törléséért. Az **ngOnInit** metódus meghívja a **loadItalok** metódust, amely lekéri az összes italt a szolgáltatástól. Az **addItal**, **updateItal** és **deleteItal** metódusok felelnek az italok hozzáadásáért, szerkesztéséért és törléséért, majd frissítik a megjelenített listát.

ADMIN | Recept Komponens

A Recept Komponens felelős az italokhoz és alapanyagokhoz tartozó receptek kezeléséért. A felhasználók hozzáadhatnak, szerkeszthetnek és törölhetnek recepteket az italokhoz és alapanyagokhoz.

**Felhasználói felület (HTML)**

A komponens HTML kódja egy űrlapot és egy listát tartalmaz. Az űrlap lehetővé teszi az új recept hozzáadását, amelyhez meg kell adni az ital ID-jét, az alapanyag ID-jét és a mennyiségét. A lista megjeleníti az összes receptet, minden recepthez tartozik egy ital ID, alapanyag ID és mennyiség mező. Minden recept mellett találhatóak gombok a szerkesztéshez és törléshez.

**Komponens logikája (TypeScript)**

A komponens TypeScript kódja felelős a receptek lekérdezéséért, megjelenítéséért, hozzáadásáért, szerkesztéséért és törléséért. Az **ngOnInit** metódus meghívja a **loadReceptek** metódust, amely lekéri az összes receptet a szolgáltatástól. Az **addRecept**, **updateRecept** és **deleteRecept** metódusok felelnek a receptek hozzáadásáért, szerkesztéséért és törléséért, majd frissítik a megjelenített listát.

ADMIN | Típus Komponens

A Típus Komponens felelős az italok típusainak kezeléséért. A felhasználók hozzáadhatnak, szerkeszthetnek és törölhetnek típusokat az italokhoz.

**Felhasználói felület (HTML)**

A komponens HTML kódja egy űrlapot és egy listát tartalmaz. Az űrlap lehetővé teszi az új típus hozzáadását, amelyhez meg kell adni az ital ID-jét és az egyes típusokhoz tartozó értékeket (téli, nyári, édes, stb.). A lista megjeleníti az összes típust, minden típus mellett találhatóak mezők az ital ID-vel és az egyes típusok értékeivel. Minden típus mellett gombok vannak a szerkesztéshez és törléshez.

**Komponens logikája (TypeScript)**

A komponens TypeScript kódja felelős a típusok lekérdezéséért, megjelenítéséért, hozzáadásáért, szerkesztéséért és törléséért. Az **ngOnInit** metódus meghívja a **loadTipusok** metódust, amely lekéri az összes típust a szolgáltatástól. Az **addTipus**, **updateTipus** és **deleteTipus** metódusok felelnek a típusok hozzáadásáért, szerkesztéséért és törléséért, majd frissítik a megjelenített listát.

ADMIN | User Komponens

A User Komponens felelős a felhasználók kezeléséért és az adminisztrátori jogosultságok kezeléséért.

**Felhasználói felület (HTML)**

A komponens HTML kódja egy listát tartalmaz, amely minden felhasználóhoz tartozik. Minden felhasználó mellett megjelenik a felhasználónév, és egy gomb, amely lehetővé teszi, hogy az adott felhasználót hozzáadjuk vagy eltávolítsuk az adminisztrátori jogosultságok listájáról. A felhasználó típusától függően a gombok szövege és funkcionalitása változik.

**Komponens logikája (TypeScript)**

A komponens TypeScript kódja felelős a felhasználók lekérdezéséért és az adminisztrátori jogosultságok kezeléséért. Az **ngOnInit** metódus meghívja a **loadUserek** metódust, amely lekéri az összes felhasználót a szolgáltatástól. A **setAdmin** metódus felelős az adminisztrátori jogosultságok beállításáért vagy eltávolításáért, majd frissíti a megjelenített listát.

Adatbázis: drinkisland

Táblák:

* Alapanyagok: Az alapanyagok nevét és mennyiségének mértékegységét tárolja.
* Italok: A koktélok nevét tárolja.
* Posts: A posztok írójának id-jét, a poszt címét és a szövegét tárolja.
* Receptek: Az italokhoz a recepteket tárolja.
* Tipus: Az italok típusát tárolja.
* Users: A felhasználók adatait tárolja.

alapanyagok:

* id integer
* name varchar(50)
* unit varchar(50)

italok:

* id integer
* name varchar(50)

posts:

* id integer
* user\_id integer
* title varchar(200)
* text varchar(2000)

receptek:

* id integer
* ital\_id integer
* alapanyag\_id integer
* amount integer

tipus:

* id integer
* ital\_id integer
* teli tinyint
* nyari tinyint
* edes tinyint
* savanyu tinyint
* keseru tinyint
* alkoholos tinyint
* alkoholmentes tinyint

users:

* id integer
* username varchar(255)
* email varchar(255)
* password varchar(255)
* remember\_token varchar(100)
* created\_at timestamp
* updated\_at timestamp
* login\_attempts integer
* banned\_time timestamp
* is\_admin tinyint

Kapcsolatok:

* A posts tábla user\_id mezője a users tábla id mezőjére hivatkozik.
* A receptek tábla ital\_id mezője az italok tábla id mezőjére hivatkozik.
* A receptek tábla alapanyag\_id mezője az alapanyagok tábla id mezőjére hivatkozik.
* A tipus tábla ital\_id mezője az italok tábla id mezőjére hivatkozik.

Kontrollerek:

* Auth/AuthController
* AlapanyagController
* ItalController
* KoktelController
* PostController
* ProfileController
* ReceptController
* TipusController

Eljárások:

**AuthController:**

register():

A ”/register” végponton érhető el, felhasználók regisztálását segítő eljárás. A felhasználó adatait a users tábába menti el. Paraméternek egy Request-et vár.

Request adatok:

* username varchar(255)
* email varchar(255)
* password varchar(255)
* password\_confirmation varchar(255)

Hogyha nem létezik ilyen felhasználónevű user, az e-mail cím megfelelő formátumú és a jelszók megegyeznek, akkor felveszi a felhasználót a users táblába.

login()

A ”/login” végponton érhető el, felhasználók bejelentkezését segítő eljárás. Paraméternek egy Request-et vár.

Request adatok:

* username varchar(255)
* password varchar(255)

Hogyha létezik ilyen felhasználónevű user, és a jelszó megfelelő, akkor belép a felhasználóval, és generál egy tokent a felhasználónak, amivel azonosíthatja magát, és visszaadja ezt.

logout()

A ”/ logout” végponton érhető el, hogyha a felhasználó be van jelentkezve. A felhasználók kijelentkezését segítő eljárás. Paraméternek egy Request-et vár, amiben a bejelentkezéskor kapott tokent várja.

Futtatásakor, ha a felhasználó megfelelő tokent ad vissza akkor kijelentkezik a felhasználó és törli ezt a tokent.

**ProfileController:**

getUserProfileData()

A /getUserProfileData végponton érhető el akkor, hogyha egy felhasználó be van jelentkezve. A felhasználó felhasználónevét adja vissza.

getUsers()

A /getUsers végponton érhető el, hogyha egy admin van bejelentkezve.

Ha meghívjuk, akkor visszatér az összes felhasználó összes adatával a users táblából.

setNewPassword()

A /setNewPassword végpontról érhető el, ha a felhasználó be van jelentkezve.

Request adatok:

* password varchar(255)
* password\_confirmation varchar(255)

Ha meghívjuk, és átadjuk a bejelentkezéskor kapott tokenünket, akkor megváltoztatja a felhasználó jelszavát.

setAdmin()

A /setAdmin végponton érhető el, ha adminként van a felhasználó bejelentkezve, és bemenetként egy Requestet vár.

Request adatok:

* username varchar(255)

Ha a username nevű felhasználó nem admin, akkor admin jogot ad neki, ha pedig admin, akkor elveszi azt.

deleteAccount()

A /deleteAccount végponton érthető el, ha a felhasználó be van jelentkezve.

Ha meghívja a felhasználó, és megfelelő tokent ad át, akkor törli a felhasználót és a felhasználóhoz tartozó posztokat.

**AlapanyagController:**

addAlapanyag()

A /addAlapanyag végponton érhető el, ha admin van bejelentkezve, és egy Requestet vár.

Request adatok:

* name varchar(50)
* unit varchar(50)

Ha olyan alapanyagnevet kapunk, ami még nem létezik, akkor az alapanyagok táblához hozzáad egy újabb alapanyagot és a mértékegységét.

getAlapanyagByName()

A /getAlapanyagByName végponton érjük el, és egy Requestet vár.

Request adatok:

* name varchar(50)

Ha létező nevet adunk meg, akkor visszaadja az ahhoz az névhez tartozó alapanyagot.

getAlapanyagById()

A /getAlapanyagById végponton érjük el, és egy Requestet vár.

Request adatok:

* id integer

Ha létező id-t adunk meg, akkor visszaadja az ahhoz az id-hez tartozó alapanyagot.

getAlapanyagok()

A /getAlapanyagok végponton érjük el.

Visszaadja az összes alapanyagot.

modifyAlapanyag()

A /modifyAlapanyag végponton érjük el, és egy Requestet vár, ahol id és egy name attribútumokat várja, ahol a name az új név, az unit pedig az új mértékegység.

Request adatok:

* id integer
* name varchar(50)
* unit varchar(50)

Ha létező id-t adunk meg, és nincs még olyan név, amire meg szeretnénk változtatni, akkor megváltoztatja az alapanyagok táblában az adatokat.

deleteAlapanyag()

A /deleteAlapanyag végponton érjük el, ha adminként vagyunk bejelentkezve, és egy Requestet vár.

Request adatok:

* id integer

Ha létezik ilyen id-jű alapanyag, és egyik receptben sincs összetevőként bent, akkor törli az alapanyagot. Azt, hogy benne van-e egy receptben a Receptkontroller getReceptByAlapanyagId() metódussal lehet lekérdezni, ami egy id-t kér Request-ként.

**ItalController**

addItal()

A /addItal végponton lehet elérni, ha admin van bejelentkezve a felhasználó és egy Requestet vár.

Request adatok:

* name varchar(50)

Ha olyan italnevet adunk, ami még nem létezik, akkor az italok táblához hozzáad egy újabb italt.

getItalByName()

A /getItalByName végponton érjük el, és egy Requestet vár.

Request adatok:

* name varchar(50)

Ha létező nevet adunk meg, akkor visszaadja az ahhoz az névhez tartozó italt.

getItalById()

A /getItalById végponton érjük el, és egy Requestet vár.

Request adatok:

* id integer

Ha létező id-t adunk meg, akkor visszaadja az ahhoz az id-hez tartozó italt.

getItalok()

A /getItalok végponton érjük el.

Visszaadja az összes italt.

modifyItal ()

A /modifyItal végponton érjük el, ha adminként vagyunk bejelentkezve, és egy Requestet vár, ahol id és egy name attribútumokat várja, ahol a name az új név.

Request adatok:

* id integer
* name varchar(50)

Ha létező id-t adunk meg, és nincs még olyan név, amire meg szeretnénk változtatni, akkor megváltoztatja az italok táblában az adatokat.

deleteItal ()

A /deleteItal végponton érjük el, ha adminként vagyunk bejelentkezve, és egy Requestet vár.

Request adatok:

* id integer

Ha létezik ilyen id-jű ital, akkor törli az italt. A receptek táblából kitörli az összes olyan receptet, ami ehhez az ital id-hez tartozik, és a tipusok táblából pedig a hozzá tartozó típust a TipusController deleteTipusByItalId() metódusával.

**KoktelController**

getKoktelok()

A /getKoktelok végponton érhető el, és a koktélok receptjeit adja vissza.

Egy JSON file-okat tartalmazó tömböt ad vissza, a következő példán látható módon:

[

{

"Ital": "Limonádé",

"Recept": {

"citromlé": "2 cl",

"szódavíz": "20 cl",

"Jégkocka": "6 db",

"mentalevél": "1 db",

"lime": "2 szelet",

"cukorszirup": "2 cl"

}

},

{

"Ital": "APEROL SPRITZ",

"Recept": {

"narancskarika": "1 szelet",

"szóda": "4 cl",

"prosecco": "12 cl",

"Jégkocka": "1 db"

}

}

]

searchKoktelokByName()

A /searchKoktelokByName végponton érjük el, és egy Requestet vár.

Request adatok:

* text varchar

Az előzőhöz hasonló JSON fájlokat tartalmazó tömböt ad vissza, csak azokat az italokat tartalmazva, amiknek a nevében benne van a Requestben kapott text paraméter.

getReceptByName()

A /getReceptByName végponton lehet elérni, és egy Requestre vár.

Request adatok:

* name varchar(50)

Egy JSON fájlt ad vissza, amiben annak a koktélnak a receptje van, amelyiknek a nevét a name attribútumban kapta.

getTeli()

A /getTeli végponton érhető el, és visszaadja az összes olyan koktél receptjét, melynek a típusa téli. Ezt az típusok táblából lehet lekérdezni.

Egy JSON fájlokat tartalmazó tömböt ad vissza.

getNyari()

A /getNyari végponton érhető el, és visszaadja az összes olyan koktél receptjét, melynek a típusa nyári. Ezt az típusok táblából lehet lekérdezni.

Egy JSON fájlokat tartalmazó tömböt ad vissza.

getEdes()

A /getEdes végponton érhető el, és visszaadja az összes olyan koktél receptjét, melynek a típusa édes. Ezt az típusok táblából lehet lekérdezni.

Egy JSON fájlokat tartalmazó tömböt ad vissza.

getSavanyu()

A /getSavanyu végponton érhető el, és visszaadja az összes olyan koktél receptjét, melynek a típusa savanyú. Ezt az típusok táblából lehet lekérdezni.

Egy JSON fájlokat tartalmazó tömböt ad vissza.

getKeseru()

A /getKeseru végponton érhető el, és visszaadja az összes olyan koktél receptjét, melynek a típusa keserű. Ezt az típusok táblából lehet lekérdezni.

Egy JSON fájlokat tartalmazó tömböt ad vissza.

getAlkoholos()

A /getAlkoholos végponton érhető el, és visszaadja az összes olyan koktél receptjét, melynek a típusa alkoholos. Ezt az típusok táblából lehet lekérdezni.

Egy JSON fájlokat tartalmazó tömböt ad vissza.

getAlkoholmentes()

A /getAlkoholmentes végponton érhető el, és visszaadja az összes olyan koktél receptjét, melynek a típusa alkoholmentesi. Ezt az típusok táblából lehet lekérdezni.

Egy JSON fájlokat tartalmazó tömböt ad vissza.

**PostController**

addPost()

A /addPost végponton lehet elérni, ha be van jelentkezve a felhasználó és egy Requestet vár.

Request adatok:

* user\_id integer
* title varchar(200)
* text varchar(2000)

A posts táblához hozzáadja az adatokat.

getPosts()

A /getPosts végponton érjük el.

Visszaadja az összes posztot.

getPostsByUserId()

A / getPostsByUserId végponton érjük el, és egy Requestet vár.

Request adatok:

* user\_id integer

Ha létező user\_id-t adunk meg, akkor visszaadja az ahhoz az id-hez tartozó posztokat.

modifyPost()

A /modifyPost végponton érjük el, ha be vagyunk jelentkezve, és egy Requestet vár.

Request adatok:

* user\_id integer
* title varchar(200)
* text varchar(2000)

Ekkor megváltoztatja a posts tábában a poszt címét és szövegét.

deletePost ()

A /deletePost végponton érjük el, ha be vagyunk jelentkezve, és egy Requestet vár.

Request adatok:

* id integer

Ha létezik ilyen id-jű poszt, akkor törli a posztot.

Ha adminként vagyunk bejelentkezve, akkor bármelyik posztot tudjuk törölni, különben csak a sajátunkat.

**ReceptController**

addRecept()

A /addRecept végponton lehet elérni, ha adminként van bejelentkezve a felhasználó és egy Requestet vár.

Request adatok:

* ital\_id integer
* alapanyag\_id integer
* amount integer

A receptek táblához hozzáadja az adatokat.

getReceptek()

A /getReceptek végponton érjük el.

Visszaadja az összes receptet.

getReceptByAlapanyagId()

A /getReceptByAlapanyagId végponton érjük el, és egy Requestet vár.

Request adatok:

* id integer

Visszaadja az első receptet, ahol a megadott id az alapanyag id.

modifyRecept()

A /modifyRecept végponton érjük el, ha adminként vagyunk bejelentkezve, és egy Requestet vár.

Request adatok:

* ital\_id integer
* alapanyag\_id integer
* amount integer

Ekkor megváltoztatja a receptek tábában a recept adatait.

deleteRecept()

A /deleteRecept végponton érjük el, ha adminként vagyunk bejelentkezve, és egy Requestet vár.

Request adatok:

* id integer

Ha létezik ilyen id-jű recept, akkor törli a receptet.

**TipusController**

addTipus()

A /addTipus végponton lehet elérni, ha adminként van bejelentkezve a felhasználó és egy Requestet vár.

Request adatok:

* ital\_id integer
* teli tinyint
* nyari tinyint
* edes tinyint
* savanyu tinyint
* keseru tinyint
* alkoholos tinyint
* alkoholmentes tinyint

A tipusok táblához hozzáadja az adatokat, ha még nem létezik ilyen ital id-hez rekord.

getTipusByItalId()

A /getTipusByItalId végponton érjük el, és egy Requestet vár.

Request adatok:

* ital\_id integer

Visszaadja az első típust, ahol a megadott ital\_id az ital\_id.

getTipusok()

A /getTipusok végponton érjük el.

Visszaadja az összes ital típusát.

modifyTipus()

A /modifyTipus végponton érjük el, ha adminként vagyunk bejelentkezve, és egy Requestet vár.

Request adatok:

* ital\_id integer
* teli tinyint
* nyari tinyint
* edes tinyint
* savanyu tinyint
* keseru tinyint
* alkoholos tinyint
* alkoholmentes tinyint

Ekkor megváltoztatja a tipus tábában a típus adatait.

deleteTipus()

A /deleteTipus végponton érjük el, ha adminként vagyunk bejelentkezve, és egy Requestet vár.

Request adatok:

* id integer

Törli a típust, ha annak az ital\_id-je az id.