Backend

**Konzulens:**

* Kaczur Sándor (it-tanfolyam.hu)

**Szükséges szoftverek:**

* xampp v3.3.0
* Composer v2.7.2
* Laravel v10

**Beüzemelés:**

A projekt clone-ozása után a xampp felületén importáljuk be a *ISABIKEDB1.0.sql* állományt

ez legenerálja nekünk az adatbázist a hozzá tartozó felhasználókat és a tárolt eljárásokat.

Ezt követően lépjünk be ebbe a könyvtárba *TMTK\_enjoyer\_vizsgaremek\Backend\api\isabikeAPI*,

és adjuk ki a következő parancsot *composer install*.

Miután ez a művelet le futót készítsünk másolatot a *.env.example* fájlból és nevezzük el *.env* -nek.

Ezután fútassuk le a projekt könyvtárában *TMTK\_enjoyer\_vizsgaremek\Backend\api\isabikeAPI* a

*php artisan key:generate* parancsot. Majd a server indításához adjuk ki a *php artisan serve* parancsot

ez localhost [*http://127.0.0.1:8000*](http://127.0.0.1:8000)fogja elindítani alapértelmezetten

**Táblák és kapcsolatok:**

*Táblák:*

* Felhasznalok: Felhasználók adatait tárolja.
* Viszaigazolo\_kod: A regisztrációhoz szükséges visszaigazoló kódokat tárolja.
* Tokenek: A bejelentkezéshez szükséges tokeneket tárolja.
* Ertekelesek: A termékek értékeléseit tárolja.
* Velemenyek: A termékekről írt véleményeket tárolja.
* Kedvezmenyek: A webáruházban elérhető kedvezményeket tárolja.
* Szalitok: A szállítókat tárolja.
* Fizetes\_opciok: A fizetési opciókat tárolja.
* Rendelesek: A rendelések adatait tárolja.
* Gyartok: A gyártókat tárolja.
* Kategoriak: A termékkategóriákat tárolja.
* Tomeg\_tulajdonsagai: A tömeg mértékegységeit tárolja.
* Termekek: A termékek adatait tárolja.
* Termek\_kepek: A termékekhez tartozó képeket tárolja.
* Ar\_valtozasok: A termékek árváltozásait tárolja.
* Kosarazot\_termekek: A kosárba tett termékeket tárolja.
* Rendelt\_termekek: A rendelt termékeket tárolja.

*Kapcsolatok:*

* A Velemenyek tábla felhasznalo\_id oszlopa a Felhasznalok tábla felhasznalo\_id oszlopára hivatkozik.
* A Velemenyek tábla ertekeles\_id oszlopa az Ertekelesek tábla ertekeles\_id oszlopára hivatkozik.
* A Rendelesek tábla kedvezmeny\_id oszlopa a Kedvezmenyek tábla kedvezmeny\_id oszlopára hivatkozik.
* A Rendelesek tábla szalito\_id oszlopa a Szalitok tábla szalito\_id oszlopára hivatkozik.
* A Rendelesek tábla fizetes\_opcio\_id oszlopa a Fizetes\_opciok tábla fizetes\_opcio\_id oszlopára hivatkozik.
* A Termekek tábla gyarto\_id oszlopa a Gyartok tábla gyarto\_id oszlopára hivatkozik.
* A Termekek tábla kategoria\_id oszlopa a Kategoriak tábla kategoria\_id oszlopára hivatkozik.
* A Termekek tábla tomeg\_tulajdonsaga\_id oszlopa a Tomeg\_tulajdonsagai tábla tomeg\_tulajdonsaga\_id oszlopára hivatkozik.
* A Termek\_kepek tábla termek\_id oszlopa a Termekek tábla termek\_id oszlopára hivatkozik.
* A Ar\_valtozasok tábla termek\_id oszlopa a Termekek tábla termek\_id oszlopára hivatkozik.
* A Kosarazot\_termekek tábla felhasznalo\_id oszlopa a Felhasznalok tábla felhasznalo\_id oszlopára hivatkozik.
* A Kosarazot\_termekek tábla termekek\_id oszlopa a Termekek tábla termek\_id oszlopára hivatkozik.
* A Rendelt\_termekek tábla rendeles\_id oszlopa a Rendelesek tábla rendeles\_id oszlopára hivatkozik.
* A Rendelt\_termekek tábla termek\_id oszlopa a Termekek tábla termek\_id oszlopára hivatkozik.

*Megjegyzések:*

* Az adatbázis UTF-8 karakterkód

**Tárolt eljárások:**

**get\_kategoriak\_procedure:**

*Leírás:*

A get\_kategoriak\_procedure tárolt eljárás lekéri az ÖSSZES rekordot a kategoriak táblából.

*Bemeneti paraméterek:*

A tárolt eljárásnak nincs bemeneti paramétere.

*Kimeneti paraméterek:*

A tárolt eljárás a kategoriak tábla minden rekordját visszaadja. A visszaadott rekordok a következő oszlopokat tartalmazzák:

* + kategoria\_id (integer): A kategória azonosítója
  + kategoria\_nev (string): A kategória neve

**check\_if\_felhasznalo\_exist\_procedure:**

*Leírás:*

A check\_if\_felhasznalo\_exist\_procedure tárolt eljárás ellenőrzi, hogy létezik-e a Felhasznalok táblában a megadott felhasznalo\_nev vagy email cím alapján.

*Bemeneti paraméterek:*

* felhasznalo\_nev\_p (varchar(20)): A felhasználónév
* email\_p (varchar(40)): Az email cím

*Kimeneti paraméterek:*

* result (integer): A Felhasznalok táblában található rekordok száma, amelyek megfelelnek a megadott feltételeknek.

**add\_felhasznalo\_procedure:**

*Leírás:*

Az add\_felhasznalo\_procedure tárolt eljárás új rekordot hoz létre a Felhasznalok táblában, és hozzárendel egy visszaigazoló kódot a Viszaigazolo\_kod táblában. A tárolt eljárás nem ad vissza kimeneti paramétert.

*Bemeneti paraméterek:*

* felhasznalo\_nev\_p (varchar(20)): A felhasználónév
* vezetek\_nev\_p (varchar(30)): A vezetéknév
* kereszt\_nev\_p (varchar(30)): A keresztnév
* email\_p (varchar(40)): Az email cím
* jelszo\_p (varchar(200)): A jelszó
* viszaigazolo\_kod\_p (varchar(5)): A visszaigazoló kód

Kimeneti paraméterek:

* felhasznalo\_id (integer): A Felhasznalok táblában újonnan létrehozott rekord azonosítója
* felhasznalo\_nev (string): A felhasználónév
* vezetek\_nev (string): A vezetéknév
* kereszt\_nev (string): A keresztnév
* vasarlo\_telefonszama (string): A vásárló telefonszáma (lehet null)
* email (string): Az email cím
* jelszo (string): A jelszó (titkosítva)
* szalitasi\_cime (string): A szállítási cím (lehet null)
* jogosultsag (integer): A jogosultsági szint
* regisztracio\_ideje (string): A regisztráció időpontja
* viszaigazolas\_statusz (integer): A visszaigazolási státusz
* aktiv (integer): Aktív-e a felhasználó
* viszaigazolo\_kod\_id (integer): A Viszaigazolo\_kod táblában azonosító
* viszaigazolo\_kod (string): A visszaigazoló kód

**verify\_felhasznalo\_procedure:**

*Leírás:*

A verify\_felhasznalo\_procedure tárolt eljárás ellenőrzi a viszaigazolo\_kod paraméterben megadott visszaigazoló kódot, és ha érvényes, akkor aktiválja a felhasználót a Felhasznalok táblában. A tárolt eljárás lekéri a Felhasznalok táblából az aktivált felhasználó adatait.

*Bemeneti paraméterek:*

* viszaigazolo\_kod\_p (varchar(5)): A visszaigazoló kód

*Kimeneti paraméterek:*

A tárolt eljárás a Felhasznalok táblából lekéri az aktivált felhasználó adatait.

* felhasznalo\_id (integer): A felhasználó azonosítója
* felhasznalo\_nev (string): A felhasználónév
* vezetek\_nev (string): A vezetéknév
* kereszt\_nev (string): A keresztnév
* vasarlo\_telefonszama (string): A vásárló telefonszáma (lehet null)
* email (string): Az email cím
* jelszo (string): A jelszó (titkosítva)
* szalitasi\_cime (string): A szállítási cím (lehet null)
* jogosultsag (integer): A jogosultsági szint
* regisztracio\_ideje (string): A regisztráció időpontja
* viszaigazolas\_statusz (integer): A visszaigazolási státusz
* aktiv (integer): Aktív-e a felhasználó
* viszaigazolo\_kod\_id (integer): A Viszaigazolo\_kod táblában azonosító
* viszaigazolo\_kod (string): A visszaigazoló kód

**logIn\_felhasznalo\_procedure:**

*Leírás:*

A logIn\_felhasznalo\_procedure tárolt eljárás ellenőrzi a beírt jelszót és email címet a Felhasznalok táblában. Ha a beírt adatok helyesek, és a felhasználó aktiválva van, a tárolt eljárás bejegyzést hoz létre a tokenek táblában, és aktiválja a felhasználót. A tárolt eljárás lekéri a Felhasznalok és a tokenek táblákból a bejelentkezett felhasználó adatait.

*Bemeneti paraméterek:*

* felhasznalo\_jelszo\_p (varchar(200)): A beírt jelszó
* email\_p (varchar(40)): A beírt email cím
* token\_p (varchar(120)): A token

*Kimeneti paraméterek:*

A tárolt eljárás vissza adja a Felhasznalok és a tokenek táblákból a bejelentkezett felhasználó adatait és egy „true” értéket result ként

* result (boolean): Az művelet sikeresége
* felhasznalo\_id (integer): A felhasználó azonosítója
* felhasznalo\_nev (string): A felhasználónév
* vezetek\_nev (string): A vezetéknév
* kereszt\_nev (string): A keresztnév
* vasarlo\_telefonszama (string): A vásárló telefonszáma (lehet null)
* email (string): Az email cím
* jelszo (string): A jelszó (titkosítva)
* szalitasi\_cime (string): A szállítási cím (lehet null)
* jogosultsag (integer): A jogosultsági szint
* regisztracio\_ideje (string): A regisztráció időpontja
* viszaigazolas\_statusz (integer): A visszaigazolási státusz
* aktiv (integer): Aktív-e a felhasználó
* token\_id (integer): A tokenek táblában azonosító
* token (string): A token

**set\_token\_procedure:**

*Leírás:*

A set\_token\_procedure tárolt eljárás bejegyzést hoz létre a tokenek táblában a megadott felhasznalo\_id\_p azonosítóhoz és token\_p tokenhez. Ezzel egyidejűleg aktiválja a felhasználót a Felhasznalok táblában.

*Bemeneti paraméterek:*

* felhasznalo\_id\_p (mediumint): A felhasználó azonosítója
* token\_p (varchar(120)): A token

*Kimeneti paraméterek:*

A tárolt eljárás nem ad vissza kimeneti paramétereket.

**logout\_procedure:**

*Leírás:*

A logout\_procedure tárolt eljárás inaktiválja a felhasználót a Felhasznalok táblában, és törli a bejegyzést a tokenek táblában a megadott token\_p token alapján.

*Bemeneti paraméterek:*

* token\_p (varchar(120)): A token

Kimeneti paraméterek:

A tárolt eljárás nem ad vissza kimeneti paramétereket.

**get\_felhasznalok\_procedure:**

*Leírás:*

Ez az eljárás lekéri a felhasználók adatait. A token alapján csak adminisztrátorok (jogosultság = 3) számára elérhető ez a procedure.

*Bemeneti paraméterek:*

* token\_p (varchar(120)): A felhasználó tokenje

*Kimeneti paraméterek:*

result (boolean): true és az összes felhasználó adata, ha a felhasználó adminisztrátor, false ha nem

* felhasznalo\_id (integer): A felhasználó azonosítója
* felhasznalo\_nev (string): A felhasználónév
* vezetek\_nev (string): A vezetéknév
* kereszt\_nev (string): A keresztnév
* vasarlo\_telefonszama (string): A vásárló telefonszáma (lehet null)
* email (string): Az email cím
* jelszo (string): A jelszó (titkosítva)
* szalitasi\_cime (string): A szállítási cím (lehet null)
* jogosultsag (integer): A jogosultsági szint
* regisztracio\_ideje (string): A regisztráció időpontja
* viszaigazolas\_statusz (integer): A visszaigazolási státusz
* aktiv (integer): Aktív-e a felhasználó

**get\_one\_felhasznalo\_procedure:**

*Leírás:*

Ez az eljárás lekéri egy felhasználó adatait a token alapján.

*Bemeneti paraméterek:*

token\_p (varchar(120)): A felhasználó tokenje

*Kimeneti paraméterek:*

result (boolean): true, ha a token érvényes, false ha nem

A felhasználó adatai (ha a token érvényes)

* felhasznalo\_id (integer): A felhasználó azonosítója
* felhasznalo\_nev (string): A felhasználónév
* vezetek\_nev (string): A vezetéknév
* kereszt\_nev (string): A keresztnév
* vasarlo\_telefonszama (string): A vásárló telefonszáma (lehet null)
* email (string): Az email cím
* jelszo (string): A jelszó (titkosítva)
* szalitasi\_cime (string): A szállítási cím (lehet null)
* jogosultsag (integer): A jogosultsági szint
* regisztracio\_ideje (string): A regisztráció időpontja
* viszaigazolas\_statusz (integer): A visszaigazolási státusz
* aktiv (integer): Aktív-e a felhasználó

**update\_felhasznalok\_procedure**

*Leírás:*

Ez az eljárás frissíti egy felhasználó adatait felhasználó ID alapján. Csak adminisztrátorok (jogosultság = 3) számára elérhető ezt a vizsgálatot token alapján végzi

*Bemeneti paraméterek:*

* token\_p (varchar(120)): A felhasználó tokenje
* felhasznalo\_id\_p (mediumint): A felhasználó ID-ja
* felhasznalo\_nev\_p (varchar(20)): A felhasználó neve
* vezetek\_nev\_p (varchar(30)): A vezetéknév
* kereszt\_nev\_p (varchar(30)): A keresztnév
* vasarlo\_telefonszama\_p (varchar(20)): A telefonszám
* email\_p (varchar(40)): Az email cím
* szalitasi\_cime\_p (varchar(40)): A szállítási cím
* jogosultsag\_p (tinyint): A jogosultsági szint (1 - felhasználó, 3 - adminisztrátor)

*Kimeneti paraméterek:*

result (boolean): true, ha a frissítés sikeres volt, false ha nem

A frissített felhasználó adatai (ha a frissítés sikeres volt)

* felhasznalo\_id (integer): A felhasználó azonosítója
* felhasznalo\_nev (string): A felhasználónév
* vezetek\_nev (string): A vezetéknév
* kereszt\_nev (string): A keresztnév
* vasarlo\_telefonszama (string): A vásárló telefonszáma (lehet null)
* email (string): Az email cím
* jelszo (string): A jelszó (titkosítva)
* szalitasi\_cime (string): A szállítási cím (lehet null)
* jogosultsag (integer): A jogosultsági szint
* regisztracio\_ideje (string): A regisztráció időpontja
* viszaigazolas\_statusz (integer): A visszaigazolási státusz
* aktiv (integer): Aktív-e a felhasználó

**update\_one\_felhasznalo\_procedure**

*Leírás:*

Ez az eljárás frissíti egy felhasználó adatait a token alapján.

*Bemeneti paraméterek:*

* token\_p (varchar(120)): A felhasználó tokenje
* felhasznalo\_nev\_p (varchar(20)): A felhasználó neve
* vezetek\_nev\_p (varchar(30)): A vezetéknév
* kereszt\_nev\_p (varchar(30)): A keresztnév
* vasarlo\_telefonszama\_p (varchar(20)): A telefonszám
* email\_p (varchar(40)): Az email cím
* szalitasi\_cime\_p (varchar(40)): A szállítási cím

*Kimeneti paraméterek:*

result (boolean): true, ha a frissítés sikeres volt, false ha nem

A frissített felhasználó adatai (ha a frissítés sikeres volt)

* felhasznalo\_id (integer): A felhasználó azonosítója
* felhasznalo\_nev (string): A felhasználónév
* vezetek\_nev (string): A vezetéknév
* kereszt\_nev (string): A keresztnév
* vasarlo\_telefonszama (string): A vásárló telefonszáma (lehet null)
* email (string): Az email cím
* jelszo (string): A jelszó (titkosítva)
* szalitasi\_cime (string): A szállítási cím (lehet null)
* jogosultsag (integer): A jogosultsági szint
* regisztracio\_ideje (string): A regisztráció időpontja
* viszaigazolas\_statusz (integer): A visszaigazolási státusz
* aktiv (integer): Aktív-e a felhasználó

**add\_termek\_procedure**

*Leírás:*

Ez az eljárás új terméket ad hozzá a Termekek táblához. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek:*

* token\_p (varchar(120)): A felhasználó tokenje
* kategoria\_id\_p (mediumint): A termék kategóriájának ID-ja
* termek\_nev\_p (varchar(50)): A termék neve
* gyarto\_id\_p (tinyint): A gyártó ID-ja
* raktarondb\_p (bigint): A termék raktárkészlete
* tomeg\_tulajdonsaga\_id\_p (tinyint): A tömegegység ID-ja
* tomeg\_erteke\_p (double): A termék tömege
* szine\_p (varchar(50)): A termék színe
* leiras\_p (varchar(500)): A termék leírása
* egyseg\_ar\_p (mediumint): A termék ára

*Kimeneti paraméterek:*

result (boolean): true, ha a termék hozzáadása sikeres volt, false ha nem

A hozzáadott termék adatai (ha a termék hozzáadása sikeres volt)

* termek\_id (mediumint)
* kategoria\_id (mediumint)
* termek\_nev (varchar(50) )
* gyarto\_id (tinyint )
* raktarondb (bigint )
* tomeg\_tulajdonsaga\_id (tinyint)
* tomeg\_erteke (double)
* szine (varchar(50) )
* leiras (varchar(500) )
* egyseg\_ar (mediumint)
* elerheto (boolean )

**update\_termek\_procedure**

*Leírás:*

Ez az eljárás frissíti egy termék adatait a Termekek táblában. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek:*

* token\_p (varchar(120)): A felhasználó tokenje
* termek\_id\_p (mediumint): A termék ID-ja
* kategoria\_id\_p (mediumint): A termék kategóriájának ID-ja
* termek\_nev\_p (varchar(50)): A termék neve
* gyarto\_id\_p (tinyint): A gyártó ID-ja
* raktarondb\_p (bigint): A termék raktárkészlete
* tomeg\_tulajdonsaga\_id\_p (tinyint): A tömegegység ID-ja
* tomeg\_erteke\_p (double): A termék tömege
* szine\_p (varchar(50)): A termék színe
* leiras\_p (varchar(500)): A termék leírása
* egyseg\_ar\_p (mediumint): A termék ára

*Kimeneti paraméterek:*

result (boolean): true, ha a frissítés sikeres volt, false ha nem

A frissített termék adatai (ha a frissítés sikeres volt)

* termek\_id (mediumint)
* kategoria\_id (mediumint)
* termek\_nev (varchar(50) )
* gyarto\_id (tinyint )
* raktarondb (bigint )
* tomeg\_tulajdonsaga\_id (tinyint)
* tomeg\_erteke (double)
* szine (varchar(50) )
* leiras (varchar(500) )
* egyseg\_ar (mediumint)
* elerheto (boolean )

**update\_termekek\_elerheto\_procedure**

*Leírás:*

Ez az eljárás frissíti egy termék elérhetőségét a Termekek táblában. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek*

token\_p (varchar(120)): A felhasználó tokenje

termek\_id\_p (MEDIUMINT): A termék ID-ja

*Kimeneti paraméterek:*

result (boolean): true, ha a frissítés sikeres volt, false ha nem

**get\_termekek\_procedure**

*Leírás:*

* Az eljárás lekéri a Termekek tábla adatait.
* Az eljárás becsatlakozik a Kategoria táblához a kategoria\_id alapján.
* Az eljárás becsatlakozik a Gyartok táblához a gyarto\_id alapján.
* Az eljárás becsatlakozik a Tomeg\_tulajdonsagai táblához a tomeg\_tulajdonsaga\_id alapján.
* Az eljárás becsatlakozik a Termek\_kepek táblához a termek\_id alapján.
* Az eljárás csoportosítja az eredményeket a termek\_id alapján.
* Az eljárás kiszámítja a legkisebb kep\_helye értéket minden termékhez.
* Az eljárás limit\_p számú eredményt ad vissza.

Bemeneti paraméterek:

* limit\_p (integer): A lekérdezendő termékek maximális száma

Kimeneti paraméterek:

A Termekek tábla adatai, kiegészítve a kategória nevével, a gyártó nevével, a mértékegységgel és a termékfotók URL-jével.

* termek\_id (mediumint)
* kategoria\_id (mediumint)
* kategoria\_neve (varchar(50))
* termek\_nev (varchar(50) )
* gyarto\_id (tinyint )
* gyarto\_neve (varchar(50))
* telefonszama (varchar(50))
* webhely (varchar(50))
* raktarondb (bigint )
* tomeg\_tulajdonsaga\_id (tinyint)
* mertek\_egysege (varchar(50))
* tomeg\_erteke (double)
* szine (varchar(50) )
* leiras (varchar(500) )
* egyseg\_ar (mediumint)
* elerheto (boolean )
* kep\_helye (text)

**add\_gyartok\_procedure**

*Leírás:*

Ez az eljárás új gyártót ad hozzá a Gyartok táblához. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek:*

* token\_p (varchar(120)): A felhasználó tokenje
* gyarto\_neve\_p (varchar(20)): A gyártó neve
* telefonszama\_p (varchar(25)): A gyártó telefonszáma
* webhely\_p (varchar(50)): A gyártó webhelye

*Kimeneti paraméterek:*

* result (boolean): true, ha a gyártó hozzáadása sikeres volt, false ha nem

**get\_gyartok\_procedure**

*Leírás:*

Ez az eljárás lekéri az összes gyártót a Gyartok táblából.

*Bemeneti paraméterek:*

Nincs

*Kimeneti paraméterek:*

* gyarto\_id (tinyint)
* gyarto\_neve (varchar(20))
* telefonszama (varchar(25))
* webhely (varchar(50))

**update\_gyartok\_procedure**

*Leírás:*

Ez az eljárás frissíti egy gyártó adatait a Gyartok táblában. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek:*

* token\_p (varchar(120)): A felhasználó tokenje
* gyarto\_id\_p (tinyint): A gyártó ID-ja
* gyarto\_neve\_p (varchar(20)): A gyártó neve
* telefonszama\_p (varchar(25)): A gyártó telefonszáma
* webhely\_p (varchar(50)): A gyártó webhelye

*Kimeneti paraméterek:*

* result (boolean): true, ha a frissítés sikeres volt, false ha nem

**get\_ertekelesek\_procedure**

*Leírás:*

Ez az eljárás lekéri az összes értékelést az ertekelesek táblából.

*Bemeneti paraméterek:*

Nincs

*Kimeneti paraméterek:*

* ertekeles\_id (tinyint )
* ertekeles (varchar(20))

**get\_tomeg\_tulajdonsagok\_procedure**

*Leírás:*

Ez az eljárás lekéri az összes tömegtulajdonságot a tomeg\_tulajdonsagai táblából.

*Bemeneti paraméterek:*

Nincs

*Kimeneti paraméterek:*

A tomeg\_tulajdonsagai tábla összes adata

* tomeg\_tulajdonsaga\_id (tinyint)
* mertek\_egysege (varchar(20))

**add\_tomeg\_tulajdonsagok\_procedure**

*Leírás:*

Ez az eljárás új tömegtulajdonságot ad hozzá a tomeg\_tulajdonsagai táblához. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek:*

Token\_p (varchar(100)): A felhasználó tokenje

mertek\_egysege (varchar(20)): A tömegtulajdonság mértékegysége

*Kimeneti paraméterek:*

result (boolean): true, ha a tömegtulajdonság hozzáadása sikeres volt, false ha nem

A hozzáadott tömegtulajdonság adatai (ha a hozzáadás sikeres volt)

* tomeg\_tulajdonsaga\_id (tinyint)
* mertek\_egysege (varchar(20))

**add\_velemenyek\_procedure**

*Leírás:*

Ez az eljárás új véleményt ad hozzá a velemenyek táblához.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* ertekeles\_id\_p (tinyint): Az értékelés ID-ja, amihez a vélemény kapcsolódik
* velemeny\_p (varchar(500)): A vélemény szövege

*Kimeneti paraméterek:*

result (boolean): true, ha a vélemény hozzáadása sikeres volt, false ha nem

**get\_velemenyek\_procedure**

*Leírás:*

Ez az eljárás lekéri az összes véleményt a velemenyek táblából.

*Bemeneti paraméterek:*

Nincs

*Kimeneti paraméterek:*

A velemenyek tábla összes adata

* velemeny\_id smallint
* felhasznalo\_id mediumint
* ertekeles\_id tinyint
* velemeny varchar(500)
* lathato boolean

**change\_velemenyek\_lathato\_procedure**

*Leírás:*

Ez az eljárás megváltoztatja egy vélemény láthatóságát a velemenyek táblában. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* velemeny\_id\_p (smallint): A vélemény ID-ja

*Kimeneti paraméterek:*

result (boolean): true, ha a láthatóság módosítása sikeres volt, false ha nem

**add\_termek\_kep\_procedure**

*Leírás:*

Ez az eljárás új termékfotót ad hozzá a termek\_kepek táblához. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* termek\_id\_p (mediumint): A termék ID-ja
* kep\_helye\_p (varchar(100)): A termékfotó fájlneve és elérési útja a szerveren

*Kimeneti paraméterek:*

result (boolean): true, ha a termékfotó hozzáadása sikeres volt, false ha nem

**get\_termek\_kepek\_procedure**

*Leírás:*

Ez az eljárás lekéri az összes termékfotót egy adott termékhez a termek\_kepek táblából.

*Bemeneti paraméterek:*

* termek\_id\_p (mediumint): A termék ID-ja

*Kimeneti paraméterek:*

A termek\_kepek tábla összes adata a megadott termékhez

* termek\_kep\_id mediumint
* termek\_id mediumint
* kep\_helye text

**delete\_termek\_kep\_procedure**

*Leírás:*

Ez az eljárás törli az összes termékfotót egy adott termékhez a termek\_kepek táblából. Csak adminisztrátorok (jogosultság = 3) számára elérhető.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* termek\_id\_p (mediumint): A termék ID-ja

*Kimeneti paraméterek:*

result (boolean): true, ha a termékfotók törlése sikeres volt, false ha nem

**add\_kosarazot\_termek\_procedure**

*Leírás:*

Ez az eljárás hozzáad egy terméket a kosárhoz a Kosarazot\_termekek táblában.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* termekek\_id\_p (mediumint): A termék ID-ja
* darabszam\_p (tinyint): A termék darabszáma

*Kimeneti paraméterek:*

* result (tinyint):
  + 0: Hibás token
  + 2: Nincs elegendő raktárkészlet
  + 3: Sikeres hozzáadás

**get\_one\_kosarazot\_termek\_procedure**

*Leírás:*

Ez az eljárás lekéri a kosárban lévő termékeket a Kosarazot\_termekek táblából egy adott felhasználó számára.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje

*Kimeneti paraméterek:*

result (boolean): true, ha a lekérés sikeres volt, false ha nem

A Kosarazot\_termekek tábla összes adata a bejelentkezett felhasználóhoz

* felhasznalo\_id mediumint
* termekek\_id mediumint
* darabszam tinyint
* kosarazas\_ideje DATETIME

**update\_kosarazot\_termeke\_darabszam\_procedure**

*Leírás:*

Ez az eljárás frissíti a kosárban lévő termék darabszámát a Kosarazot\_termekek táblában.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* termekek\_id\_p (mediumint): A termék ID-ja
* darabszam\_p (tinyint): A termék új darabszáma

*Kimeneti paraméterek:*

* result (tinyint):
  + 0: Hibás token
  + 1: Nincs elegendő raktárkészlet
  + 2: Sikeres frissítés

**delete\_kosarazot\_termekek\_procedure**

*Leírás:*

Ez az eljárás töröl egy terméket a kosárból a Kosarazot\_termekek táblában.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* termekek\_id\_p (mediumint): A termék ID-ja

*Kimeneti paraméterek:*

result (boolean): true, ha a termék törlése sikeres volt, false ha nem

**get\_one\_termek\_procedure**

*Leírás:*

eljárás lekér egy terméket a termekek táblából a termék ID-je alapján.

*Bemeneti paraméterek:*

* termek\_id\_p (mediumint): A termék ID-ja

*Kimeneti paraméterek:*

A termekek tábla összes adata a megadott termékhez, kiegészítve a kategória nevével, a gyártó nevével, a tömegegység mértékegységével, a termék elérhetőségével és a termékfotó URL-jével.

* termek\_id (mediumint)
* kategoria\_id (mediumint)
* kategoria\_neve (varchar(50))
* termek\_nev (varchar(50) )
* gyarto\_id (tinyint )
* gyarto\_neve (varchar(50))
* telefonszama (varchar(50))
* webhely (varchar(50))
* raktarondb (bigint )
* tomeg\_tulajdonsaga\_id (tinyint)
* mertek\_egysege (varchar(50))
* tomeg\_erteke (double)
* szine (varchar(50) )
* leiras (varchar(500) )
* egyseg\_ar (mediumint)
* elerheto (boolean )
* kep\_helye (text)

**get\_kategoriak\_procedure**

*Leírás:*

Ez az eljárás lekéri az összes kategóriát a kategoriak táblából.

*Bemeneti paraméterek:*

Nincs

*Kimeneti paraméterek:*

A kategoriak tábla összes adata

* kategoria\_id mediumint
* kategoria\_neve varchar(50)

**get\_one\_rendeles\_termekek\_procedure**

*Leírás:*

Ez az eljárás lekéri egy adott rendeléshez tartozó termékeket a rendelt\_termekek és a termekek táblákból.

*Bemeneti paraméterek:*

* rendeles\_id (mediumint): A rendelés ID-ja

*Kimeneti paraméterek:*

* termék\_nev (varchar(255)): A termék neve
* darabszam (tinyint): A termék darabszáma
* egyseg\_ar (decimal(10,2)): A termék egységára

**get\_rendelt\_termekek\_procedure**

*Leírás:*

Ez az eljárás lekéri az összes rendelt terméket a rendelt\_termekek táblából.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje

*Kimeneti paraméterek:*

result (boolean): true, ha a lekérés sikeres volt, false ha nem

A rendelt\_termekek tábla összes adata

* rendeles\_id mediumint
* termek\_id mediumint
* darabszam tinyint
* kezbesitve boolean
* kezbesites\_ideje DATETIME,

**get\_all\_rendeles\_procedure**

*Leírás:*

Ez az eljárás lekéri az összes rendelést a rendelesek táblából.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje

*Kimeneti paraméterek:*

result (boolean): true, ha a lekérés sikeres volt, false ha nem

A rendelesek tábla összes adata

* rendeles\_id mediumint
* felhasznalo\_id mediumint,
* rendeles\_ideje DATETIME default NOW(),
* megjegyzes varchar(200),
* szalito\_id tinyint,
* fizetes\_opcio\_id TINYINT,
* kedvezmeny\_id tinyint,

**get\_one\_rendeles\_procedure**

*Leírás:*

Ez az eljárás lekéri egy adott felhasználó egy rendelését a rendelesek táblából.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje

*Kimeneti paraméterek:*

result (boolean): true, ha a lekérés sikeres volt, false ha nem

A rendelesek tábla összes adata a megadott felhasználóhoz tartozó egyetlen rendeléshez

* rendeles\_id mediumint
* felhasznalo\_id mediumint,
* rendeles\_ideje DATETIME default NOW(),
* megjegyzes varchar(200),
* szalito\_id tinyint,
* fizetes\_opcio\_id TINYINT,
* kedvezmeny\_id tinyint,

**add\_rendeles\_procedure**

*Leírás:*

Ez az eljárás új rendelést hoz létre a rendelesek táblában, frissíti a felhasználó szállítási adatait a felhasznalok táblában, Át teszi a kosarazot\_termekek tábla felhasználóhoz kötőt tartalmát a rendelt termékek táblába és termekek táblában a termékek darabszámát csökkenti annyival amennyit rendeltek belőle

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* vasarlo\_telefonszama\_p (varchar(20)): A vásárló telefonszáma
* szalitasi\_cime\_p (varchar(40)): A szállítási cím
* megjegyzes\_p (varchar(200)): Megjegyzés a rendeléshez
* szalito\_id\_p (tinyint): A szállító ID-je
* fizetes\_opcio\_id\_p (tinyint): A fizetési opció ID-je
* kedvezmeny\_id\_p (tinyint): A kedvezmény ID-je
* felhasznalo\_id (integer): A Felhasznalok táblában újonnan létrehozott rekord azonosítója
* felhasznalo\_nev (string): A felhasználónév
* vezetek\_nev (string): A vezetéknév
* kereszt\_nev (string): A keresztnév
* vasarlo\_telefonszama (string): A vásárló telefonszáma (lehet null)
* email (string): Az email cím
* jelszo (string): A jelszó (titkosítva)
* szalitasi\_cime (string): A szállítási cím (lehet null)
* jogosultsag (integer): A jogosultsági szint
* regisztracio\_ideje (string): A regisztráció időpontja
* viszaigazolas\_statusz (integer): A visszaigazolási státusz
* aktiv (integer): Aktív-e a felhasználó
* viszaigazolo\_kod\_id (integer): A Viszaigazolo\_kod táblában azonosító
* viszaigazolo\_kod (string): A visszaigazoló kód

*Kimeneti paraméterek:*

result (boolean): true, ha a lekérés sikeres volt, false ha nem

A frissen létrehozót rendelés adatait és a hozzá tartozó felhasználó adatait

* rendeles\_id mediumint
* felhasznalo\_id mediumint,
* rendeles\_ideje DATETIME default NOW(),
* megjegyzes varchar(200),
* szalito\_id tinyint,
* fizetes\_opcio\_id TINYINT,
* kedvezmeny\_id tinyint,

*Működési leírás:*

* Az eljárás lekéri a felhasználó ID-t a token alapján a Tokenek táblából.
* Ha a felhasználó nem létezik, az eljárás false értéket ad vissza.
* Ha a felhasználó létezik, az eljárás a következőket teszi:
* Új rendelést hoz létre a rendelesek táblában a megadott adatokkal.
* Lekéri a frissen létrehozott rendelés ID-jét.
* Beilleszti a kosárban lévő termékeket a rendelt\_termekek táblába a frissen létrehozott rendelés ID-jével.
* Frissíti a termék raktárkészletét a rendelt termékek mennyiségével.
* Frissíti a felhasználó szállítási adatait a felhasznalok táblában.
* Törli a kosár tartalmát a kosarazot\_termekek táblából.
* Az eljárás true értéket, a frissen létrehozott rendelés adatait és a felhasználó e-mail címét adja vissza.

**update\_rendelt\_termek**

*Leírás:*

Ez az eljárás megjelöli egy adott rendelt terméket kézbesítettként a rendelt\_termekek táblában.

*Bemeneti paraméterek:*

* Token\_p (varchar(100)): A felhasználó tokenje
* rendelt\_termek\_id\_p (mediumint): A rendelt termék ID-je

*Kimeneti paraméterek:*

* result (boolean): true, ha a frissítés sikeres volt, false ha nem

**Controllerek:**

**TermekContoller**

**A Termék Controller osztály a vizsga projektben a termékekkel kapcsolatos műveleteket kezeli. Felelős a termékek adatainak lekérdezéséért, létrehozásáért, módosításáért és törléséért.**

*getTermekek:*

*A getTermekek metódus /termekek/{limit} végpont meghivásával indul el az adatbázis Termekek tábla rekordjait adja vissza paraméternek egy limitet(int) vár ami azt határozza meg hogy hány darab terméket kérjen le a táblából a get\_termekek\_procedure(int) eljárás meghívásával*

*visszatérési adatok:*

* *termek\_id (int)*
* *kategoria\_id (int)*
* *kategoria\_neve (str)*
* *termek\_nev (str)*
* *gyarto\_id (int)*
* *gyarto\_neve (str)*
* *telefonszama (str)*
* *webhely (str)*
* *raktarondb (int)*
* *tomeg\_tulajdonsaga\_id (int)*
* *mertek\_egysege (str)*
* *tomeg\_erteke (float)*
* *szine (str)*
* *leiras (str)*
* *egyseg\_ar (int)*
* *elerheto (boolean)*
* *kep\_helye(str)*

*getOneTermekek:*

*A getOneTermekek metódus /onetermekek/{id} végpont meghivásával indul el az adatbázis Termekek tábla egy rekordjait adja termek\_id alapján paraméternek egy id(int) vár ami azt határozza meg hogy melyik terméket kérje le a táblából a get\_one\_termek\_procedure eljárás meghívásával*

*visszatérési adatok:*

* *termek\_id (int)*
* *kategoria\_id (int)*
* *kategoria\_neve (str)*
* *termek\_nev (str)*
* *gyarto\_id (int)*
* *gyarto\_neve (str)*
* *telefonszama (str)*
* *webhely (str)*
* *raktarondb (int)*
* *tomeg\_tulajdonsaga\_id (int)*
* *mertek\_egysege (str)*
* *tomeg\_erteke (float)*
* *szine (str)*
* *leiras (str)*
* *egyseg\_ar (int)*
* *elerheto (boolean)*
* *kep\_helye(str)*

*addTermek:*

*A addTermek metódus /* *addtermek végpont meghivásával indul el az adatbázis Termekek tábla egy uj rekordot adhozzá paraméternek egy Request adatait várj ami alapján létrehozza a terméket add\_termek\_procedure eljárás meghívásával.*

*Request adatok:*

* *token (str)*
* *termek\_kateg (int)*
* *termek\_nev (str)*
* *gyarto\_id (int)*
* *raktarondb (int)*
* *tomeg\_tulajdonsaga\_id (int)*
* *tomeg\_erteke*
* *szine (str)*
* *leiras (str)*
* *egyseg\_ar (int)*

*visszatérési adatok:*

* *token (str)*
* *termek\_kateg (int)*
* *termek\_nev (str)*
* *gyarto\_id (int)*
* *raktarondb (int)*
* *tomeg\_tulajdonsaga\_id (int)*
* *tomeg\_erteke*
* *szine (str)*
* *leiras (str)*
* *egyseg\_ar (int)*

*updateTermek:*

*A updateTermek metódus /* *updatetermek* *végpont meghivásával indul el az adatbázis Termekek tábla egy meglévö terméket változtat paraméternek egy Request adatait várj ami alapján megváltoztassa a terméket update\_termek\_procedure eljárás meghívásával.*

*Request adatok:*

* *token (str)*
* *termek\_kateg (int)*
* *termek\_nev (str)*
* *gyarto\_id (int)*
* *raktarondb (int)*
* *tomeg\_tulajdonsaga\_id (int)*
* *tomeg\_erteke*
* *szine (str)*
* *leiras (str)*
* *egyseg\_ar (int)*

*visszatérési adatok:*

* *token (str)*
* *termek\_kateg (int)*
* *termek\_nev (str)*
* *gyarto\_id (int)*
* *raktarondb (int)*
* *tomeg\_tulajdonsaga\_id (int)*
* *tomeg\_erteke*
* *szine (str)*
* *leiras (str)*
* *egyseg\_ar (int)*

*deleteTermek:*

*A deleteTermekmetódus /* *deletetermek végpont meghívásával indul el az adatbázis Termekek tábla egy meglevő termék állapotát változtatja hogy elérhető vagy nem paraméternek egy Request adatait várj ami alapján megváltoztassa a terméket update\_termekek\_elerheto\_procedure eljárás meghívásával.*

*Request adatok:*

* *BearerToken (str)*
* *termek\_id (int)*

*visszatérési adatok:*

* *result*

*getKategoriak:*

*A getKategoriak /* *kategoriak* *végpont meghívásával indul el az adatbázis Kategoriak tábla rekordjait adja vissza get\_kategoriak\_procedure* *eljárás meghívásával.*

*visszatérési adatok:*

* *kategoria\_id (int)*
* *kategoria\_neve (str)*

**FelhasznaloController**

*registerUser:*

*A registerUser metodus a /* *registration* *végpont meghívásával indul el az adatbázis Felhasznalok tábla hoz létre egy új felhasználót paraméternek egy Request-et vár amiben a felhasználó létrehozásához szükséges adatok vannak check\_if\_felhasznalo\_exist\_procedure és a* *add\_felhasznalo\_procedure* *eljárás meghívásával.*

*Request adatok:*

* *felhasznalo\_nev (str)*
* *kereszt\_nev (str)*
* *vezetek\_nev (str)*
* *email (str)*
* *password (str)*

*visszatérési adatok:*

* *felhasznalo\_id (int)*
* *felhasznalo\_nev (str)*
* *vezetek\_nev (str)*
* *kereszt\_nev (str)*
* *vasarlo\_telefonszama (str)*
* *email (str)*
* *jelszo (str)*
* *szalitasi\_cime (str)*
* *jogosultsag (int)*
* *regisztracio\_ideje*
* *viszaigazolas\_statusz*
* *aktiv*
* *viszaigazolo\_kod\_id (int)*
* *viszaigazolo\_kod (str)*

*verifyUser:*

*A verifyUser metodus /* *verify végpont meghívásával indul el az adatbázis felhasznalok tábla rekordjait változtassa meg hogy a felhasználó vissza igazolt státusza true értéket vegyen fel egy Request-et vár el paraméterként verify\_felhasznalo\_procedure eljárás meghívásával.*

*Request adatok:*

* *code (str)*

*visszatérési adatok:*

* *felhasznalo\_id (int)*
* *felhasznalo\_nev (str)*
* *vezetek\_nev (str)*
* *kereszt\_nev (str)*
* *vasarlo\_telefonszama (str)*
* *email (str)*
* *jelszo (str)*
* *szalitasi\_cime (str)*
* *jogosultsag (int)*
* *regisztracio\_ideje*
* *viszaigazolas\_statusz*
* *aktiv*
* *viszaigazolo\_kod\_id (int)*
* *viszaigazolo\_kod (str)*

*logIn*

*A logIn metodus /* *login végpont meghívásával indul el az adatbázis felhasznalok táblában változtassa meg hogy a felhasználó akti státusza true értéket vegyen fel ha helyes email jelszó párossal próbál meg bejelentkezni, sikeres bejelentkezéskor egy generált tokent add vissza Request-et vár el paraméterként logIn\_felhasznalo\_procedure eljárás meghívásával.*

*Request adatok:*

* *password (str)*
* *email (str)*

*visszatérési adatok:*

* *data (str)*

*logOut*

*A logOut metodus /* *logout* *végpont meghívásával indul el Request-et vár el paraméterként ahol ha helyes tokent adunk meg akkor token alapján kijelentkezteti a felhasználót és törli a toként logIn\_felhasznalo\_procedure eljárás meghívásával.*

*Request adatok:*

* *token (str)*

*visszatérési adatok:*

* *nincs*

*getFelhasznalok*

*A getFelhasznalok metodus /* *getfelhasznalok végpont meghívásával indul el Request-et vár el paraméterként ahol ha helyes admin tokent adunk meg akkor lekéri az összes felhasználó adatát a felhasznalok táblából get\_felhasznalok\_procedure eljárás meghívásával.*

*Request adatok:*

* *Bearer Token (str)*

*visszatérési adatok:*

* *felhasznalo\_id (int)*
* *felhasznalo\_nev (str)*
* *vezetek\_nev (str)*
* *kereszt\_nev (str)*
* *vasarlo\_telefonszama (str)*
* *email (str)*
* *jelszo (str)*
* *szalitasi\_cime (str)*
* *jogosultsag (int)*
* *regisztracio\_ideje*
* *viszaigazolas\_statusz*
* *aktiv*
* *viszaigazolo\_kod\_id (int)*
* *viszaigazolo\_kod (str)*

*updateFelhasznalo*

*A updateFelhasznalo metodus /* *updatefelhasznalo végpont meghívásával indul el Request-et vár el paraméterként ahol ha helyes admin tokent adunk meg akkor felhasználo id alapján frissíti egy felhasználó adatát a felhasznalok táblában update\_felhasznalok\_procedure eljárás meghívásával.*

*Request adatok:*

* *token*
* *felhasznalo\_id*
* *felhasznalo\_nev*
* *vezetek\_nev*
* *kereszt\_nev*
* *vasarlo\_telefonszama*
* *email*
* *szalitasi\_cime*
* *jogosultsag*

*visszatérési adatok:*

* *felhasznalo\_id (int)*
* *felhasznalo\_nev (str)*
* *vezetek\_nev (str)*
* *kereszt\_nev (str)*
* *vasarlo\_telefonszama (str)*
* *email (str)*
* *jelszo (str)*
* *szalitasi\_cime (str)*
* *jogosultsag (int)*
* *regisztracio\_ideje*
* *viszaigazolas\_statusz*
* *aktiv*
* *viszaigazolo\_kod\_id (int)*
* *viszaigazolo\_kod (str)*

*updateOneFelhasznalo*

*A updateOneFelhasznalo metodus /* *updateonefelhasznalo végpont meghívásával indul el Request-et vár el paraméterként ahol token alapján frissíti a felhasználó adatát a felhasznalok táblában update\_one\_felhasznalo\_procedure eljárás meghívásával.*

*Request adatok:*

* *token*
* *felhasznalo\_nev*
* *vezetek\_nev*
* *kereszt\_nev*
* *vasarlo\_telefonszama*
* *email*
* *szalitasi\_cime*
* *jogosultsag*

*visszatérési adatok:*

* *felhasznalo\_id (int)*
* *felhasznalo\_nev (str)*
* *vezetek\_nev (str)*
* *kereszt\_nev (str)*
* *vasarlo\_telefonszama (str)*
* *email (str)*
* *szalitasi\_cime (str)*

*getOneFelhasznalo*

*A getOneFelhasznalo metodus /* *getonefelhasznalo* *végpont meghívásával indul el Request-et vár el paraméterként ahol token alapján lekéri egy felhasználó adatát a felhasznalok táblábol get\_one\_felhasznalo\_procedure eljárás meghívásával.*

*Request adatok:*

* *token*

*visszatérési adatok:*

* *felhasznalo\_id (int)*
* *felhasznalo\_nev (str)*
* *vezetek\_nev (str)*
* *kereszt\_nev (str)*
* *vasarlo\_telefonszama (str)*
* *email (str)*
* *jelszo (str)*
* *szalitasi\_cime (str)*
* *jogosultsag (int)*
* *regisztracio\_ideje*
* *viszaigazolas\_statusz*
* *aktiv*
* *viszaigazolo\_kod\_id (int)*
* *viszaigazolo\_kod (str)*

*genToken*

A genToken metodus a Random osztály használatával egy 60 karakteres tokent generál amit tartalmazhat számokat kis és nagy betűt

*visszatérési adatok:*

* token (str)

*hashPass*

A hashPass metodus a hash\_hmac metódus segítségével le kódolja a felhasználó jelszavát

SHA256 kodlolást használ és „salt” ként a saláta szöveg van neki megadva

*bemeneti paraméter:*

* jelszó (str)

*visszatérési adatok:*

* kódolt jelszó (str)

*genEmailCode*

A genEmailCode metodus a Random osztály segítségével egy 5 karakter hosszú számot ad vissza

*visszatérési adatok:*

* kód (str)

**ErtekelesekController**

*getErtekelesek*

*A getErtekelesek metódus /* *ertekelesek végpont meghivásával indul el az adatbázis Kategoriak tábla rekordjait adja vissza get\_ertekelesek\_procedure eljárás meghívásával*

*visszatérési adatok:*

* ertekeles\_id (int)
* ertekeles (str)

**GyartokController**

*addGyarto*

*A addGyarto metódus /* *addgyarto végpont meghivásával indul el egy Request-et vár el paraméternek amiból egy új gyártót add hozzá a gyartok táblához, ha a token admin tokenje add\_gyartok\_procedure* *eljárás meghívásával*

*Request adatok:*

* *token (str)*
* *gyarto\_neve (str)*
* *telefonszama (str)*
* *webhely (str)*

*visszatérési adatok:*

* result

*getGyartok*

*A getGyartok metódus /* *gyartok végpont meghivásával indul el az adatbázis Gyartok tábla összes rekordját agya vissza get\_gyartok\_procedure eljárás meghívásával*

*visszatérési adatok:*

* *gyarto\_id (int)*
* *gyarto\_neve (str)*
* *telefonszama (str)*
* *webhely (str)*

*updateGyarto*

*A updateGyarto metódus /* *updategyarto végpont meghívásával indul el egy Request-et vár el paraméternek amiból egy gyártót változtat meg id alapján ha a token admin tokenje update\_gyartok\_procedure eljárás meghívásával*

*Request adatok:*

* *token (str)*
* *gyarto\_id (int)*
* *gyarto\_neve (str)*
* *telefonszama (str)*
* *webhely (str)*

*visszatérési adatok:*

* result

**KosarazotTermekekController**

*getOneKosar*

*getOneKosarOriginal*

*addToKosar*

*updateKosar*

*deleteKosar*

*deleteKosarOriginal*

**RendelesekController**

*addRendeles*

*getOneRendeles*

*getOneRendelesOriginal*

*getAllRendeles*

*getAllRendeltTermek*

*getOneRendelesTermekek*

*updateRendeltTermek*

**TermekKepController**

*addTermekKep*

*A addTermekKep metódus /* *addtermekkep végpont meghívásával indul el egy Request-et vár el paraméternek amiból egy új képet add hozzá a termek\_kepek táblához ha a token egy admin tokenje add\_termek\_kep\_procedure eljárás meghívásával*

*Request adatok:*

* *token (str)*
* *termek\_id (int)*
* *kep\_helye (str)*

*visszatérési adatok:*

* *result (int)*

*deleteTermekKep*

*A deleteTermekKep metódus /* *deletetermekkep végpont meghívásával indul el egy Request-et vár el paraméternek amiból egy képet töröl ki a termek\_kepek táblábol ha a token egy admin tokenje delete\_termek\_kep\_procedure eljárás meghívásával*

*Request adatok:*

* *token (str)*
* *termek\_id (int)*

*visszatérési adatok:*

* *result (int)*

**TomegTulajdonsagokController**

*getTomegTulajdonsagok*

*A getTomegTulajdonsagok metódus /* *tomegtulajdonsagok végpont meghívásával indul el*

*adatbázis Tomeg\_tulajdonsagai tábla összes rekordját agya vissza get\_tomeg\_tulajdonsagok\_procedure eljárás meghívásával*

*visszatérési adatok:*

* *tomeg\_tulajdonsaga\_id (int)*
* *mertek\_egysege (str)*

*addTomegTulajdonsagok*

*A addTomegTulajdonsagok metódus /* *addtomegtulajdonsagok végpont meghívásával indul el egy Request-et vár el paraméternek amiból egy új tömeg tulajdonságot add hozzá a Tomeg\_tulajdonsagai táblához add\_tomeg\_tulajdonsagok\_procedure eljárás meghívásával*

*Request adatok:*

* *token (str)*
* *mertek\_egysege (str)*

*visszatérési adatok:*

* tomeg\_tulajdonsaga\_id (int)
* *mertek\_egysege (str)*
* *result (int)*

**VelemenyekController**

*getVelemenyek*

*A getVelemenyek metódus /* *getvelemenyek végpont meghívásával indul el*

*adatbázis velemenyek tábla összes rekordját adja vissza get\_velemenyek\_procedure eljárás meghívásával*

*visszatérési adatok:*

* *velemeny\_id (int)*
* *felhasznalo\_id (int)*
* *ertekeles\_id (int)*
* *velemeny (str)*
* *lathato (int)*

*addVelemenyek*

*A addVelemenyek metódus /* *addvelemenyek végpont meghívásával indul el egy Request-et vár el paraméternek amiból egy új véleményt add hozzá a velemenyek táblához add\_velemenyek\_procedure eljárás meghívásával*

*Request adatok:*

* *token (str)*
* *ertekeles\_id (int)*
* *velemeny (int)*

*visszatérési adatok:*

* *result (int)*

*deleteVelemenyek*

*A deleteVelemenyek metódus /* *deletevelemenyek végpont meghívásával indul el egy Request-et vár el paraméternek amiból egy véleménynek a láthatósági státuszát változtatja a velemenyek táblába ha a token egy admin tokenje delete\_termek\_kep\_procedure eljárás meghívásával*

*Request adatok:*

* *velemeny\_id*

*visszatérési adatok:*

* *result (int)*

**Response controllerek**

A response controlerek felelösek a végpontok által hivot metodusok viszatérési értékének a formázására

* *FelhasznalokResponseController*
* *GyartokResponseController*
* *KosarazotTermekekResponseController*
* *RendelesekResponseController*
* *TermekekResponseController*
* *TermekKepResponseController*
* *TomegTulajdonsagokResponseController*
* *VelemenyekResponseController*

**Továbbfejlesztési lehetőségek:**

Jelszó módosítás, emlékeztető

*ez a funkció még egyáltalán nem működik a banckend részén tehát ha a felhasználó elfelejti a jelszavát vagy meg szeretné változtatni arra nincs lehetőség jelenleg*

Felhasználó törlés

*jelenleg a felhasználó törlése nem opció hivatalosan de a backend felépítéséből adódóan már jelenleg is meg lehet oldani a felhasználó végleges in aktivizálását*

Kosarazót termék módosítása

*A kosárba rakót termék darabszámának módosítása jelen pillanatban is működő képes viszont nincs rajta ellenőrzés úgy mint a termék kosarazásánál tehát több darabszámban is lehet a kosárban tartani egy terméket mint amennyi a termékek táblában megtalálható*

Akciók, kedvezmények, kupon kódok

*jelen pillanatban ezek a funkciók nem elérhetőek de az adatbázis felépítéséből adódóan már most is kötelezően meg kell adni egy kedvezményt ami a nincs kedvezmény paraméter*

Desktop Aplication

Asztali alkalmazás fejlesztői dokumentáció:

# Osztályok:

## DbConnect:

**Osztály leírása**

A DbConnect osztály az Isabike alkalmazás adatbázis elérését kezelő osztálya. Az osztály felel az alkalmazás és a szerver közötti kommunikációért, a felhasználó bejelentkezésért, a termékek lekérdezéséért és a szűrésért.

## Függvények

**getData(string uri):**

* Leíró szöveg: Ez a függvény egy megadott végpontról kér le adatokat a backend szerverről Json formátumba.
* Paraméterek:
  + uri: A lekérdezni kívánt végpontból címe.
* Visszatérési érték:
  + A függvény a visszatérő Json adatból egy Jarray objektumot

készít siker esetén. Sikertelen lekérdezés esetében **null** értékkel tér vissza.

## getUserData(string uri):

* Leíró szöveg: Ez a függvény a felhasználói adatokat kér le a backend szerver megadott végponotjáról. A függvény

a Login.getToken() metódus által visszaadott tokennel hitelesíti a felhasználót.

* Paraméterek:
  + uri: A lekérdezni kívánt végpont címe.
* Visszatérési érték:
  + A függvény a visszatérő Json adatból egy Jarray objektumot

készít siker esetén. Sikertelen lekérdezés esetében **null** értékkel tér vissza.

## getFilteredData(string uri, string termeknev, double suly, string gyarto):

* Leíró szöveg: Ez a függvény egy megadott végpontról kér le adatokat a backend szerverről Json formátumba, majd szűri az adatott a megadott termék neve, súlya, és gyártója alapján
* Paraméterek:
  + uri: A lekérdezni kívánt URL címe.
  + termeknev: A keresett termék neve.
  + suly: A keresett termék súlya.
  + gyarto: A keresett termék gyártója.
* Visszatérési érték:
  + A függvény a visszatérő Json adatból egy Jarray objektumot

készít a szűrt termékek adataival siker esetén, sikertelen lekérés esetén **null** értékkel tér vissza

## loginToProg(string email, string password, string url):

* Leíró szöveg: Ez a függvény a felhasználót bejelentkezteti az alkalmazásba a megadott e-mail-cím és jelszó páros alapján.
* Paraméterek:
  + email: A felhasználó email címe.
  + password: A felhasználó jelszava.
  + url: A bejelentkezési API végpont URL címe.
* Visszatérési érték:
  + A függvény true értéket ad vissza sikeres bejelentkezés esetén,

egyébként false értéket.

## logOut(string token, string url):

* Leíró szöveg: Ez a függvény kijelenti a felhasználót az alkalmazásból a megadott felhasználói tokennel.
* Paraméterek:
  + token: A felhasználó bejelentkezési tokenje.
  + url: A kijelentkezési API végpont URL címe.
* Visszatérési érték:
  + A függvény nem térít vissza értéket (void).

## Hibakezelés

A DbConnect osztály minden függvénye try-catch blokkot használ a hibák kezelésére. Sikertelen lekérdezés vagy hitelesítési hiba esetén a függvények a MessageBox osztály segítségével megjelenítenek egy üzenetablakot amelyben részletes információt ad a hibáról. A hibaüzeneteket a backend szolgálja.

## DbOperations

**Osztály leírása**

Az Isabike.DbOperations osztály az Isabike alkalmazás adatbázis műveleteit kezelő

osztály. Az osztály felel a termékek és felhasználók felvételéhez, törléséhez és frissítéséhez a REST API segítségével.

## Függvények

**getKey(string res):**

* Leíró szöveg: Ez a függvény egy stringből próbálja kinyerni a token

kulcsot. A függvény a megadott válasz (res) stringből eltávolítja a felesleges karaktereket (",", "{", "}") és kettőspontot (':') használva szétválasztja a kulcs-érték párokat. A token kulcs pozíciója alapján (a második elem az érték) téríti vissza a megtalált token értéket.

* Paraméterek:
  + res: A token értékét tartalmazó válasz string.
* Visszatérési érték:
  + A függvény a megtalált token értéket adja vissza stringként,

vagy üres stringet, ha a művelet sikertelen volt.

## addProduct(string jsondata, string url):

* Leíró szöveg: Ez a függvény egy új terméket ad hozzá az adatbázishoz. A termék adatai JSON formátumban kerülnek átadásra a jsondata paraméterben. Az url paraméter pedig az API végpontot tartalmazza, ahova a HTTP POST kérés küldendő.
* Paraméterek:
  + jsondata: JSON formátumú string, amely a hozzáadandó

termék adatait tartalmazza.

* + url: A termékek hozzáadásához használt API végpont URL címe.
* Visszatérési érték:
  + A függvény nem térít vissza értéket (void). A művelet

eredménye a MessageBox osztály segítségével jelenik meg.

## deleteProduct(string jsondata, string url):

* Leíró szöveg: Ez a függvény töröl egy terméket az adatbázisból. A termék azonosítója a JSON formátumú jsondata paraméterben kerül átadásra. Az url paraméter pedig az API végpontot tartalmazza, ahova a HTTP DELETE kérés küldendő. A függvény a felhasználó megerősítését kéri a törlés előtt.
* Paraméterek:
  + jsondata: JSON formátumú string, amely a törölni kívánt

termék azonosítóját tartalmazza.

* + url: A termékek törléséhez használt API végpont URL címe.
* Visszatérési érték:
  + A függvény nem térít vissza értéket (void). A művelet

eredménye a MessageBox osztály segítségével jelenik meg.

## updateProduct(string jsondata, string url):

* Leíró szöveg: Ez a függvény frissít egy terméket az adatbázisban. A termék módosított adatai JSON formátumban kerülnek átadásra

a jsondata paraméterben. Az url paraméter pedig az API végpontot tartalmazza, ahova a HTTP PATCH kérés küldendő.

* Paraméterek:
  + jsondata: JSON formátumú string, amely a módosítani kívánt

termék adatait tartalmazza.

* + url: A termékek frissítéséhez használt API végpont URL címe.
* Visszatérési érték:
  + A függvény nem térít vissza értéket (void). A művelet

eredménye a MessageBox osztály segítségével jelenik meg.

## updateUser(string jsonData, string url):

* Leíró szöveg: Ez a függvény frissít egy felhasználót az adatbázisban. A felhasználó módosított adatai JSON formátumban kerülnek átadásra a jsonData paraméterben. Az url paraméter pedig az API végpontot tartalmazza, ahova a HTTP PUT kérés küldendő.
* Paraméterek:
  + jsonData: JSON formátumú string, amely a módosítani kívánt

felhasználó adatait tartalmazza.

* + url: A felhasználók frissítéséhez használt API végpont URL címe.
* Visszatérési érték:
  + A függvény nem térít vissza értéket (void). A művelet

eredménye a MessageBox osztály segítségével jelenik meg.

## Hibakezelés

Az DbOperations osztály minden függvénye try-catch blokkot használ a hibák kezelésére.

Sikertelen művelet esetén a függvények a MessageBox osztály segítségével jelenítik meg a hibaüzenetet.

## Termekek

**Osztály leírása**

Az Isabike.Termekek osztály egy egyszerű adatstruktúra, amely a termékek adatainak tárolására szolgál. Az osztály három tulajdonsággal rendelkezik:

* termek\_nev: A termék neve.
* gyarto\_neve: A termék gyártójának neve.
* tomeg\_erteke: A termék tömege.

## Tulajdonságok

### termek\_nev:

* + Típus: string
  + Leírás: A termék neve.
  + Példa: "Kerékpár"

### gyarto\_neve:

* + Típus: string
  + Leírás: A termék gyártójának neve.
  + Példa: "Trek"

### tomeg\_erteke:

* + Típus: float
  + Leírás: A termék tömege.
  + Példa: 10.5

## Használat

Az Isabike.Termekek osztályt a termékek adatainak tárolására és

továbbítására lehet használni. Például egy listában tárolhatjuk a termékeket, és megjeleníthetjük őket egy táblázatban.

## Függőségek

**Newtonsoft.Json**

A Newtonsoft.Json csomag függőségének indoklása

A **Newtonsoft.Json** csomag használata az Isabike programban a következő okok miatt indokolt:

### JSON adatcsere:

A program API-hívásokat hajt végre, melyek JSON formátumban továbbítanak és fogadnak adatokat. A Newtonsoft.Json csomag hatékony eszköz a JSON objektumok szerializálására és deszerializálására, ami egyszerűsíti az API-hívások implementációját.

### Objektum-adatmodell leképezése:

A program adatmodellje JSON struktúrákon alapul. A Newtonsoft.Json csomag lehetővé teszi a JSON objektumok egyszerű leképezését a program objektummodelljére, minimalizálva a manuális adatkonverzió szükségességét.

### Kompatibilitás:

A Newtonsoft.Json széles körben elterjedt és számos más .NET-es programmal kompatibilis. Ez megkönnyíti a program integrálását más rendszerekkel, és növeli a hordozhatóságát.

### Teljesítmény:

A Newtonsoft.Json gyors és hatékony, minimalizálva a JSON adatcserével kapcsolatos teljesítménybeli overhead-et.

### Egyszerű használat:

A Newtonsoft.Json API-ja könnyen tanulható és használható, egyszerűsítve a fejlesztési folyamatot.

### Alternatívák:

Léteznek más JSON-kezelő csomagok is .NET-hez, mint a **System.Text.Json**. Azonban a Newtonsoft.Json továbbra is az egyik legnépszerűbb és legmegbízhatóbb megoldás, és a fenti előnyök miatt ideális választás az Isabike programhoz.

### Összefoglalva:

A Newtonsoft.Json csomag használata az Isabike programban számos előnnyel jár, beleértve a hatékony JSON adatcserét, az egyszerű adatmodell- leképezést, a kompatibilitást, a teljesítményt és az egyszerű használatot. A csomag használata megkönnyíti a program fejlesztését és karbantartását, és javítja a program integrálhatóságát és hordozhatóságát.

# Formok

## MainForm

**Osztály leírása**

A MainForm osztály az Isabike alkalmazás főablakát reprezentáló osztály. Az osztály felel a felhasználói felület kezeléséért, a termékek és

vélemények megjelenítéséért, valamint a szűrésért és a termékekkel kapcsolatos műveletekért (törlés, frissítés).

## Vezérlők

* **manufactererBox** (ComboBox): A gyártók legördülő listája. A lista a DbConnect.getData függvény segítségével lekéri a gyártókat

a http://127.0.0.1:8000/api/gyartok URL-ről. A ValueMember property beállítása a gyártó azonosító (gyarto\_id) mezőre történik, a DisplayMember property pedig a gyártó nevének (gyarto\_neve) megjelenítését határozza meg.

* **categoryBox** (ComboBox): A kategóriák legördülő listája. A lista

a DbConnect.getData függvény segítségével lekéri a kategóriákat

a http://127.0.0.1:8000/api/kategoria URL-ről. A ValueMember property beállítás a kategória azonosító (kategoria\_id) mezőre történik, a DisplayMember property pedig a kategória nevének (kategoria\_neve) megjelenítését határozza meg.

* **viewGrid** (DataGridView): A termékek vagy vélemények megjelenítésére szolgáló

táblázat. A táblázat oszlopainak fejlécei a lekérdezett adatok mezőneveinek megfelelően alakulnak. A függvény a lekérdezés eredményét a táblázat adatforrásaként állítja be (viewGrid.DataSource).

* **termeknevTextbox** (TextBox): A szűréshez használt terméknév beviteli mezője. A

felhasználó által beírt terméknév alapján történik a szűrés.

* **suly\_textbox** (TextBox): A szűréshez használt súly beviteli mezője. A felhasználó által beírt súly alapján történik a szűrés.
* **ReviewsBtn** (Button): A termékek és a vélemények közötti váltásra szolgáló gomb. A gridState változó alapján állapítható meg az aktuális nézet (true: vélemény, false: terméke). A gomb megnyomása során a connectToDB függvény meghívódik a megfelelő adattípus betöltésére.
* **refreshBtn** (Button): Az adatok frissítése gomb. A gomb megnyomása az aktuális nézet (termékek vagy vélemények) frissítését indítja el. A connectToDB függvény újrahívásával lekéri a legfrissebb adatokat a szerverről.
* **SalesBtn** (Button): Az eladások gomb ( jelenleg nincs implementálva). A gomb feladata egy jövőbeli eladások modul megjelenítése lenne.
* **filterBtn** (Button): A szűrés gomb. A gomb megnyomása után a szűrőfeltételek (terméknév, súly, gyártó) alapján történik az adatok lekérdezése. A szűrést

a DbConnect.getFilteredData függvény valósítja meg.

* **updateButtonColumn** (DataGridViewButtonColumn): A táblázat egy oszlopa, amely "Módosítás" gombokat tartalmaz minden sorban. A gomb megnyomása a gomb sorában lévő cellákban átírt adatokkal módosítja a sor adatbázisban eltárolt adatait. A műveletet a viewGrid\_CellClick eseménykezelő függvény valósítja meg.
* **deleteButtonColumn** (DataGridViewButtonColumn): A táblázat egy oszlopa, amely "Törlés" gombokat tartalmaz minden sorban. A gomb megnyomása törli a kiválasztott terméket. A felhasználó megerősítése után

a DbOperations.deleteProduct függvény segítségével történik a törlés a megadott API végpontra (http://127.0.0.1:8000/api/deletetermek) küldött HTTP DELETE kéréssel.

## Eseménykezelők

* **Main\_Load:** Az űrlap betöltésekor meghívódik. A connectToDB függvényt hívja meg a termékek betöltésére. Ellenőrzi, hogy a "Törlés" és "Módosítás" oszlopok léteznek-e a táblázatban, ha nem, akkor beszúrja őket.
* **ReviewsBtn\_Click:** A "Vélemények" gomb megnyomásakor meghívódik.

A connectToDB függvényt hívja meg a vélemények betöltésére, és megváltoztatja a gomb háttérszínét.

* **refreshBtn\_Click:** A "Refresh" gomb megnyomásakor meghívódik.

A connectToDB függvényt hívja meg az aktuális nézet (termékek vagy vélemények) frissítésére.

* **filterBtn\_Click:** A "Filter" gomb megnyomásakor meghívódik.

A DbConnect.getFilteredData függvényt hívja meg a szűrőfeltételek alapján szűrt termékek lekérdezésére.

* **viewGrid\_CellClick:** A táblázat celláinak kattintásakor meghívódik. A kattintás

helyétől függően (Törlés/Módosítás oszlop) a DbOperations.deleteProduct illetve a DbOperations.updateProduct függvényeket hívja meg a termék törlésére vagy módosítására.

* **UserBtn\_Click:** A "User Operations" gomb megnyomásakor meghívódik.

Megjeleníti a UserForm űrlapot.

***Adatbázis műveletek***

* **connectToDB:** Csatlakozik az adatbázishoz és lekéri a termékeket vagy a véleményeket a DbConnect.getData függvény segítségével. Beállítja a táblázat adatforrását.
* **deleteProduct:** A DbOperations.deleteProduct függvényt hívja meg a kiválasztott termék törlésére.
* **updateProduct:** A DbOperations.updateProduct függvényt hívja meg a kiválasztott termék módosítására.

## Login

**Osztály leírása**

Az Isabike.Login osztály az Isabike alkalmazás bejelentkezési űrlapját reprezentáló osztály. A felhasználó bejelentkezési adatainak megadása és a bejelentkezés kezelése történik az osztályon keresztül. Az osztály statikus token attribútumot és metódusokat is tartalmaz a bejelentkezett felhasználó tokenjének kezelésére.

## Tulajdonságok

### token (static string):

* + - Leírás: A bejelentkezett felhasználó tokenje. A token egy titkos érték,

amely a felhasználó azonosítására szolgál a szerver oldali műveleteknél.

* + - Hozzáférés: A getToken metódus segítségével olvasható,

a setToken metódus segítségével állítható be.

## Metódusok

### Login():

* + - Leírás: Az osztály konstruktora. Inicializálja az űrlap elemeit.

### Login\_Load(object sender, EventArgs e):

* + - Leírás: Az űrlap betöltésekor lefutó eseménykezelő. Az űrlapot az

előtérbe helyezi.

### loginBtn\_Click(object sender, EventArgs e):

* + - Leírás: A "Login" gomb megnyomásakor lefutó eseménykezelő. A

megadott e-mail cím és jelszó alapján próbálja megvalósítani a bejelentkezést. Sikeres bejelentkezés esetén a főablak (MainForm) példányosodik és megjelenik, a bejelentkezési űrlap pedig eltűnik.

### closeBtn\_Click(object sender, EventArgs e):

* + - Leírás: A "Cancel" gomb megnyomásakor lefutó eseménykezelő.

Beállítja az űrlap bezárását.

* + **getToken():** (static)
    - Leírás: Visszatér a beállított token értékkel.
  + **setToken(string userToken):** (static)
    - Leírás: Beállítja a bejelentkezett felhasználó token értékét.

## ProductOperationsForm

**Osztály áttekintése**

Az Isabike.ProductOperationsForm osztály az Isabike alkalmazás egy termék hozzáadására szolgáló űrlapot reprezentáló osztály. Az űrlap segítségével új termékek adatai hozhatók létre és továbbíthatók az adatbázisba.

## Metódusok

### populateTomeg(string uri):

* + Leírás: Beállítja a tömegegység legördülő menü elemeit

az uri paraméterben megadott API végpontból lekérdezett adatok alapján. A lekérdezés

a DbConnect.getData metódus segítségével történik.

* + Paraméterek:
    - uri: Az API végpont URL címe, amely a

tömegegységekre vonatkozó adatokat tartalmazza.

### populateGyarto(string uri):

* + Leírás: Beállítja a gyártó legördülő menü elemeit

az uri paraméterben megadott API végpontból lekérdezett adatok alapján. A lekérdezés

a DbConnect.getData metódus segítségével történik.

* + Paraméterek:
    - uri: Az API végpont URL címe, amely a gyártókra

vonatkozó adatokat tartalmazza.

### populateKategoria(string uri):

* + Leírás: Beállítja a kategória legördülő menü elemeit

az uri paraméterben megadott API végpontból lekérdezett adatok alapján. A lekérdezés

a DbConnect.getData metódus segítségével történik.

* + Paraméterek:
    - uri: Az API végpont URL címe, amely a kategóriákra

vonatkozó adatokat tartalmazza.

### ProductOperationsForm():

* + Leírás: Az osztály konstruktora. Inicializálja az űrlap elemeit,

beleértve a termék nevét, gyártóját, kategóriáját, mennyiségét, tömegét, tömegegységét, színét, leírását, árát és elérhetőséget jelző beviteli mezőket, valamint a legördülő menüket és a "Mentés" és "Bezárás" gombokat.

### closeBtn\_Click(object sender, EventArgs e):

* + Leírás: A "Bezárás" gomb megnyomásakor lefutó

eseménykezelő. Beállítja az űrlap bezárását.

### okBtn\_Click(object sender, EventArgs e):

* + - * Leírás: A "Mentés" gomb megnyomásakor lefutó

eseménykezelő. A következő lépéseket hajtja végre:

* + - * + Létrehoz egy JSON objektumot a beírt termékadatokból, beleértve a bejelentkezett felhasználó tokenjét (Login.getToken()), a kategóriát,

gyártót, mennyiséget, tömeget, tömegegységet, színét, leírását, árát és elérhetőség értékét.

* + - * + A JSON objektumot string formátumba alakítja (JsonConvert.SerializeObject).
        + A DbOperations.addProduct metódust meghívja a termék hozzáadására a megadott API végponton (http://127.0.0.1:8000/api/addtermek) a JSON stringgel.
        + Sikeres művelet esetén nem történik külön visszajelzés. Sikertelen művelet esetén a kivételt kiírja egy Message box segítségével.

### ProductOperationsForm\_Load(object sender, EventArgs e):

* + - * Leírás: Az űrlap betöltésekor lefutó eseménykezelő.

Inicializálja az űrlap legördülő menüit a gyártók, kategóriák és tömegegységek adataival a megadott API végpontokból.

# UserForm

## Osztály áttekintése

Az Isabike.UserForm osztály az Isabike alkalmazás felhasználó kezelő űrlapát reprezentáló osztály. Az űrlap segítségével lehet lekérdezni a felhasználók adatait, és adminisztrátori jogosultságot adni nekik.

## Attribútumok

### makeadminButtonColumn (DataGridViewButtonColumn):

* + Leírás: Egy dinamikusan létrehozott DataGridView gomb oszlop,

amely a "Felhasználók" táblázat első oszlopában jelenik meg, és a "Makeadmin" feliratú gombot tartalmazza.

## Metódusok

### UserForm():

* Leírás: Az osztály konstruktora. Inicializálja az űrlap elemeit, beleértve a "Felhasználók" táblázatot és a "Frissítés" gombot.

### UserForm\_Load(object sender, EventArgs e):

* Leírás: Az űrlap betöltésekor lefutó eseménykezelő. A következőket hajtja végre:
  + Lekéri a felhasználók adatait a megadott API végpontról

(http://127.0.0.1:8000/api/getfelhasznalok) a GetRESTData metódus segítségével.

* + Létrehozza a "Makeadmin" gombot a táblázat első oszlopában

a adminButtonCreate metódus segítségével.

### GetRESTData(string uri):

* Leírás: Lekéri a felhasználók adatait az uri paraméterben megadott API végpontból a DbConnect.getUserData metódus segítségével. A lekérdezett adatokat a "Felhasználók" táblázat adatforrásává állítja be.

### adminButtonCreate():

* Leírás: Ellenőrzi, hogy a "Makeadmin" gomb oszlop létezik-e a táblázatban. Ha nem létezik, létrehozza és hozzáadja az első oszlopként a táblázathoz.

### userDataGridView\_CellClick(object sender, DataGridViewCellEventArgs e):

* Leírás: A "Felhasználók" táblázat celláira kattintáskor lefutó eseménykezelő.
* Funkció: Ha a felhasználó a "Makeadmin" gombra kattint (az első oszlopban lévő

gombra), akkor a következőket hajtja végre:

* + Létrehoz egy JSON objektumot a kiválasztott felhasználó adataival, kiegészítve az "adminisztrátori" jogosultsággal (3).
  + A JSON objektumot string formátumba alakítja.
  + A DbOperations.updateUser metódust meghívja a felhasználó adatainak

frissítésére a megadott API végponton (http://127.0.0.1:8000/api/updatefelhasznalo).

### refreshBtn\_Click(object sender, EventArgs e):

* Leírás: A "Frissítés" gombra kattintáskor lefutó eseménykezelő. Frissíti a "Felhasználók" táblázatot a legfrissebb adatokkal a GetRESTData metódus újrahívásával, és szükség esetén újra létrehozza a "Makeadmin" gombot

a adminButtonCreate metódus segítségével.

# Isabike Program Felhasználói Dokumentáció

## Bevezetés

Ez a dokumentáció az Isabike program felhasználói számára készült. A dokumentáció célja, hogy bemutassa a program különböző űrlapjainak és osztályainak használatát. A dokumentáció a következőket tartalmazza:

* + **Áttekintés:** Rövid leírás a program céljáról és funkcióiról.
  + **Információ a használt eszközökről:** A programban használt fejlesztői

eszközök és technológiák felsorolása (pl. .NET Windows Forms, Newtonsoft.Json).

* + **Telepítési útmutatás:** Útmutató az Isabike alkalmazás telepítéséhez
  + **Űrlapok dokumentációja:** Részletes leírás minden egyes űrlapról, beleértve:
  + Az űrlap célja és funkciója
  + Az űrlap elemei (gombok, szövegmezők, táblázatok stb.) és azok használata
  + Az űrlapon elérhető műveletek végrehajtásának lépései
  + **Osztályok dokumentációja:** Rövid ismertetés minden egyes osztályról,

beleértve:

* + Az osztály célja és funkciója
  + Az osztály metódusainak listája, paramétereik és a visszatérési érték leírása
  + Az osztály attribútumainak (ha vannak) leírása

## Használt eszközök:

* .NET Windows Forms: Az asztali alkalmazás grafikus felhasználói felületének létrehozásához használt keretrendszer.
* Newtonsoft.Json: Egy népszerű könyvtár JSON adatcseréhez .NET alkalmazásokban.

## Fontos megjegyzés:

A dokumentáció nem tartalmazza a program belső működésének részletes ismertetését, hanem a felhasználói szempontból fontos információkra koncentrál.

## Áttekintés

Az Isabike egy kerékpár áruház számára fejlesztett program, amely a következő funkciókat biztosítja:

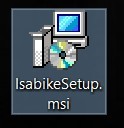
* Felhasználó bejelentkezés
* Termékek kezelése (hozzáadás, módosítás, törlés)
* Termékek listázása
* Vélemények megtekintése
* Felhasználók kezelése (adminisztrátor jogosultság esetén)

## Telepítési útmutató az Isabike programhoz

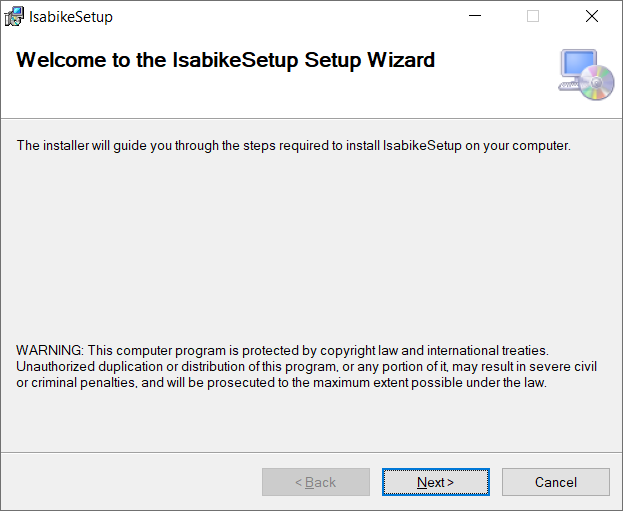
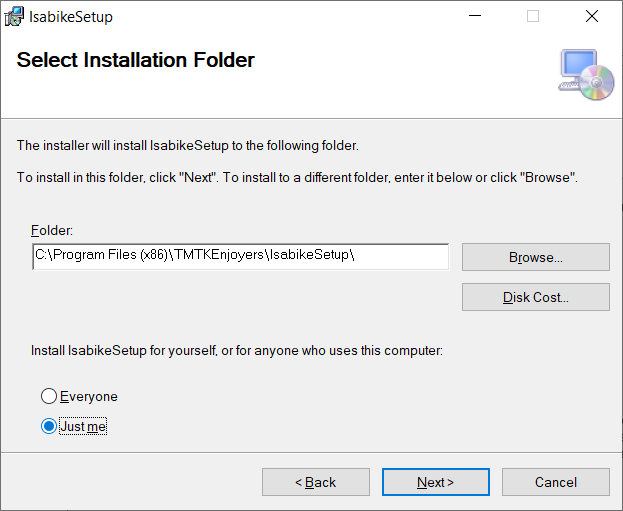
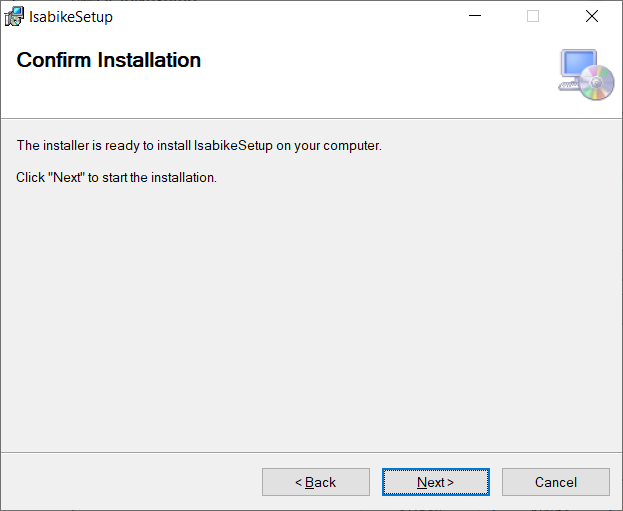
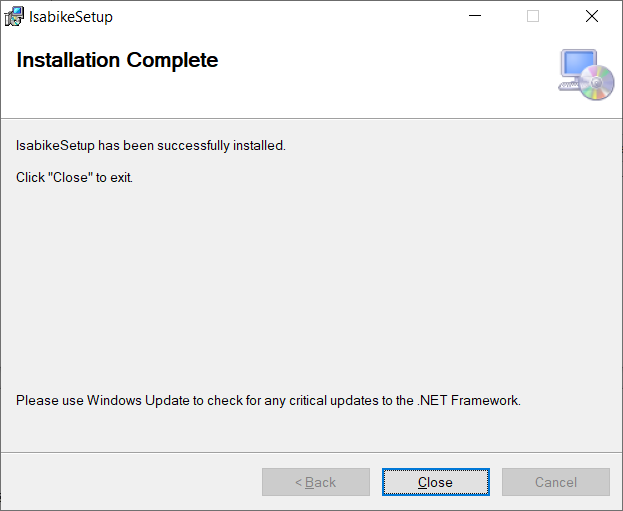
### Előfeltételek:

* A program futtatásához .NET Framework 4.7.2 vagy újabb verzió szükséges.
* A program telepítéséhez adminisztrátori jogosultsággal kell rendelkeznie.

### Telepítési lépések:

1. Töltse le a telepítőfájlt a link a telepítőfájlhoz(Névszerint: Isabikesetup.msi). Kép róla:
2. Futtassa a telepítőfájlt.
3. A telepítő varázsló lépésein haladva kövesse a képernyőn megjelenő utasításokat.

Képek róla:

1. ​
2. ​
3. ​
4. A „Next” gomb megnyomása után várja meg míg települ a program
5. ​
6. A close gomb lenyomása után az asztalon megtalálja a program indító icon-át

### A program használata:

* + A program elindításához kattintson az Isabike ikonra az asztalon.
  + A program bejelentkezési űrlappal nyílik meg.
  + Adja meg az e-mail címét és jelszavát a bejelentkezéshez.
  + A bejelentkezés után a program fő űrlapja jelenik meg.

## Űrlapok dokumentációja

**Login űrlap**

* + **Cél:** A Login űrlap lehetővé teszi a felhasználók számára a programba való bejelentkezést.

### Elemek:

* + - Email Textfield: A felhasználó e-mail címének megadása.
    - Password Textfield: A felhasználó jelszavának megadása

(jelszómező).

* + - Login gomb: A bejelentkezési adatok elküldése a programnak.
    - Close gomb: Az űrlap bezárása.

### Műveletek:

* + - Írd be az e-mail címed az Email Textfield mezőbe.
    - Írd be a jelszavad a Password Textfield mezőbe.
    - Kattints a Login gombra a bejelentkezéshez.

### Hibaüzenetek:

* + - A program megjelenít egy hibaüzenetet, ha a bejelentkezési adatok

hibásak.

* + - A program megjelenít egy hibaüzenetet, ha az adatbázis szerver nem elérhető

## MainForm űrlap

* + - **Cél:** A MainForm űrlap a program fő kezelőfelülete. Itt lehet megtekinteni a termékeket és a véleményeket, valamint hozzáférni egyéb funkciókhoz.

### Elemek:

* + - * Menüsor:
        + UserBtn gomb: A felhasználó kezelő űrlap megnyitása.
        + Termékek szűréséhez:

termeknevTextbox: Terméknév szűrő mező.

suly\_textbox: Súly szűrő mező.

manufactererBox: Gyártó szűrő legördülő lista.

filterBtn gomb: A szűrés végrehajtása.

* + - * Termékek táblázata (viewGrid): A termékek adatait

megjelenítő táblázat. A táblázat oszlopai a következők lehetnek:

* + - * + Termék azonosító
        + Termék kategória
        + Termék neve
        + Gyártó neve
        + Árkategória
        + Raktárkészlet
        + Tömeg típusa
        + Tömeg értéke
        + Szín
        + Leírás
        + Egységár
        + Elérhető

### Művelet gombok:

* + - * + Delete gomb: A kiválasztott termék törlése

(adminisztrátori jogosultság esetén).

* + - * + Update gomb: A kiválasztott termék szerkesztése (adminisztrátori jogosultság esetén).
        + ReviewsBtn gomb: A termékvélemények táblázatának megjelenítése/elrejtése.
        + RefreshBtn gomb: A táblázat frissítése a legfrissebb adatokkal.

### Műveletek:

* + - * + A termékek szűréséhez:

Írd be a keresett termék nevét a

termeknevTextbox mezőbe.

(Opcionális) Írd be a keresett termék súlyát a suly\_textbox mezőbe.

Válaszd ki a keresett termék gyártóját a manufactererBox legördülő listából.

Kattints a filterBtn gombra a szűrés végrehajtásához.

* + - * + A termékvélemények megtekintéséhez:

Kattints a ReviewsBtn gombra és a táblázatban

megjelennek a termékhez kapcsolódó vélemények.

* + - * + A táblázat frissítéséhez:

Kattints a RefreshBtn gombra.

### Hibaüzenetek:

* + - * + A program hibaüzenetet jelenít meg, ha a szűrési

feltételek nem teljesülnek.

* + - * + A program megjelenít egy hibaüzenetet, ha az adatbázis szerver nem elérhető

## ProductOperationsForm űrlap

* + **Cél:** A ProductOperationsForm űrlap lehetővé teszi új termékek hozzáadását a programhoz.

### Elemek:

* + - Termék kategória (productCategoryBox): Legördülő lista a termék

kategóriák kiválasztásához.

* + - Termék név (productNameText): Szövegmező a termék nevének megadásához.
    - Gyártó (productManufactererBox): Legördülő lista a termék gyártójának kiválasztásához.
    - Raktárkészlet (productCountText): Szövegmező a termék raktárkészletének megadásához.
    - Tömeg típusa (productWeightclassBox): Legördülő lista a termék tömeg típusának kiválasztásához.
    - Tömeg értéke (productWeightText): Szövegmező a termék tömeg értékének megadásához.
    - Szín (productColorText): Szövegmező a termék színének megadásához.
    - Leírás (productDescText): Szövegmező a termék leírásának megadásához.
    - Ár (productPriceText): Szövegmező a termék árának megadásához.
    - Elérhető (productAvelableBox): Jelölőnégyzet a termék

elérhetőségének jelzésére.

* + - OK gomb: Az új termék adatainak elküldése a programnak.
    - Close gomb: Az űrlap bezárása.

### Műveletek:

* + - Válaszd ki a termék kategóriáját a productCategoryBox legördülő

listából.

* + - Írd be a termék nevét a productNameText mezőbe.
    - Válaszd ki a termék gyártóját a productManufactererBox legördülő

listából.

* + - Írd be a termék raktárkészletét a productCountText mezőbe.
    - Válaszd ki a termék tömeg típusát a productWeightclassBox legördülő

listából.

* + - Írd be a termék tömeg értékét a productWeightText mezőbe.
    - Írd be a termék színét a productColorText mezőbe.
    - Írd be a termék leírását a productDescText mezőbe.
    - Írd be a termék árát a productPriceText mezőbe.
    - Jelöld be a productAvelableBox jelölőnégyzetet, ha a termék elérhető.
    - Kattints az OK gombra az új termék hozzá
    - A program megjelenít egy üzenetet az új termék sikeres

hozzáadásáról.

### Hibaüzenetek:

* + - A program hibaüzenetet jelenít meg, ha az űrlapon nem adtunk meg

minden szükséges adatot.

* + - A program megjelenít egy hibaüzenetet, ha az adatbázis szerver nem elérhető

## UserForm űrlap

* + - **Cél:** A UserForm űrlap lehetővé teszi a program felhasználóinak listázását és adminisztrátorrá való kinevezését (adminisztrátori jogosultság esetén).

### Elemek:

* + - * Felhasználók táblázata (userDataGridView): A program

felhasználóinak adatait megjelenítő táblázat. A táblázat oszlopai a következők lehetnek:

* + - * + Makeadmin gomb: A kiválasztott felhasználó

adminisztrátorrá kinevezése (adminisztrátori jogosultság esetén).

* + - * + Felhasználó azonosító
        + Felhasználónév
        + Vezetéknév
        + Keresztnév
        + Telefonszám
        + E-mail cím
        + Számlázási cím
        + Szállítási cím
        + Jogosultsági szint
      * RefreshBtn gomb: A táblázat frissítése a legfrissebb adatokkal.

### Műveletek:

* + - * A felhasználó adminisztrátorrá való kinevezéséhez:
        + Válaszd ki a kívánt felhasználót a userDataGridView

táblázatban.

* + - * + Kattints a Makeadmin gombra.
      * A táblázat frissítéséhez:
        + Kattints a RefreshBtn gombra.

### Hibaüzenetek:

* + - * A program hibaüzenetet jelenít meg, ha a művelet nem

hajtható végre.

* + - * A program megjelenít egy hibaüzenetet, ha az adatbázis szerver nem elérhető

## Osztályok dokumentációja

### DbConnect:

* + - * Cél: Adatbázishoz való kapcsolódás és lekérdezések

végrehajtása.

* + - * Metódusok:
        + getData(uri): Lekérdezi a JSON formátumú adatokat a

megadott URL-ről.

* + - * + getFilteredData(uri, terméknév, súly, gyártó): Lekérdezi a szűrt termékek adatait a megadott URL-ről.
        + getUserData(uri): Lekérdezi a felhasználók adatait a megadott URL-ről.
        + loginToProg(email, password, uri): Bejelentkezés a programba a megadott e-mail címmel és jelszóval.
        + logOut(token, uri): Kijelentkezés a programból a megadott tokennel.
        + addProduct(json, uri): Új termék hozzáadása a programban a JSON formátumú adatokkal.
        + updateProduct(json, uri): Termék frissítése a programban a JSON formátumú adatokkal.
        + deleteProduct(json, uri): Termék törlése a programból a JSON formátumú adatokkal.
        + updateUser(json, uri): Felhasználó frissítése a programban a JSON formátumú adatokkal.

### DbOperations:

* + - * Cél: Adatbázis műveletek végrehajtása.
      * Metódusok:
        + addProduct(json, uri): Új termék hozzáadása a

programban a JSON formátumú adatokkal.

* + - * + updateProduct(json, uri): Termék frissítése a programban a JSON formátumú adatokkal.
        + deleteProduct(json, uri): Termék törlése a programból a JSON formátumú adatokkal.
        + updateUser(json, uri): Felhasználó frissítése a programban a JSON formátumú adatokkal.

### További információk:

* A program forráskódja elérhető a <https://github.com/PlesovszkiIstvan/TMTK_enjoyer_vizsgarmek> linken.

Fontend

**Felhasználói Dokumentáció**

**Célok**

A webalkalmazás célközönsége kerékpárok iránt érdeklődő felhasználók, akik szeretnék böngészni a kínálatot, részletes információkat szeretnének kapni a különböző modellekről, valamint vásárolni szeretnének a platformon keresztül.

**Program leírása:**

Az ISABIKE Kerékpárüzlet alkalmazás egy webes platform, amely lehetővé teszi kerékpárok böngészését, részletes termékinformációk megtekintését és vásárlását. Az alkalmazás célja kényelmes és intuitív felhasználói élmény biztosítása a kerékpárok iránt érdeklődő vásárlók számára.

**Futási környezet:**

Az ISABIKE Kerékpárüzlet alkalmazás webes környezetben fut, így elérhető bármilyen modern webböngészőn keresztül, például Google Chrome, Mozilla Firefox, vagy Safari.

**Használat:**

Az alkalmazás használata egyszerű és intuitív. A felhasználók a böngészőben navigálva érik el az alábbi oldalakat:

* **Főoldal:** Az ISABIKE főoldalára érkeznek, ahol láthatják az elérhető márkákat és 6 ajánlott terméket. Az egyes biciklik kártyáiban található részletek gombra kattintva átvisz a termek oldalra.
* **Biciklik:** Megtekinthetik a termékkínálatot, információkat kaphatnak a az áról és elérhetőségről. Az előző és következő gombokal lehet a temékeket lapozni. Az egyes biciklik kártyáiban található részletek gombra kattintva átvisz a termek oldalra.
* **Termék:** Ez a oldal felelős egy adott termék egyébb részleteinek megjelenítésért mint kategoria, mennyiség rakáron, súly, szín, elérhetőség. A hozzáadás a kosárhoz gombra kattintva felveszi a kosárba a terméket. Ha a másik termék megtekintése gombra kattintunk visza irányit a bicikli oldalra.
* **Szerviz:** A kinált szervizszolgáltatásokl kapcsolatban informálódhatnak.
* **Rólunk:** További információkat találnak a vállalkozásról és annak történetéről.
* **Bejelentkezés:** Felhasználói fiókóba lehet bejelenkezni e-mail cím és jelszó megadásával. Amikor rányomunk a bejelenkezés gombra akkor elküdi a bejelenkezés a backendre és átvizs a főoldlra .
* **Regisztráció:** Felhasználói fiókót lehet létrehozni e-mail cím, név(Felhasználónév), keresztnév, vezetéknév, és jelszó megadásával. Amikor rányomunk a regisztráció gombra akkor elküdi a regisztrációt a backendre és átvizs a viszaigazolás oldara.
* **Viszaigazolás:** Felhasználói fiók aktiválása az e-mailben kapot 5 számjegyű kódal. Ha a megerősités gombra katintunk akkor átvizs a bejelenkezés oldara.
* **Kosár:** A kiválasztott termékeket itt tárolodnak vásárlás előtt, láthato termék neve:, menyisége, a kosárhoz adás ideje adatokat. Ha a megrendelés gombra katintunk akkor átvizs a rendelés oldara.
* **Rendelés:** Felhasználók megadják az adataikat (szálitó és fizetés opció nem mükődik) és leadják a rendelés amiről e-mail értesitést kapnak. Ha a küldés gombra katintunk akkor felveszi a rendelést és átvizs a főoldalra oldara.
* **Felhasználó-Rendelések:** Felhasználók visszatekinthetnek korábbi rendeléseikre és nyomon követhetik azok állapotát. Rendelés adatai gombra katintva megjeleniti rendelés ideje, megjegyzés, szálitó és fizetés opció adatokat.

**Fejlesztői Dokumentáció**

**Home komponens**

A Home komponens felelős a kezdőoldal tartalmának megjelenítéséért az ISABIKE Kerékpárüzlet webhelyén.

**Feladatok**

* Betölti a termékek adatait a backend szolgáltatásból.
* Megjeleníti a kerékpár márkákat logók formájában.
* Megjeleníti a kezdőoldalon a legnépszerűbb termékeket kártyák formájában.
* Biztosítja a felhasználó számára, hogy a termékek részleteit megtekinthesse a "Részletek" gombra kattintva.

**Metódusok**

**ngOnInit()**

* **Leírás**: Ez a metódus az Angular életciklusának része, és a komponens inicializálásakor fut le.
* **Feladat**: Betölti a termékeket és a márkákat a backend szolgáltatásból.

**getProducts()**

* **Leírás**: Ez a metódus lekéri a termékek adatait a backend szolgáltatásból.
* **Feladat**: Frissíti a products tömböt a lekért termékekkel.

**showProductDetails()**

* **Leírás**: Ez a metódus hívódik meg, amikor a felhasználó a termék részleteit kívánja megtekinteni.
* **Feladat**: Elmenti a kiválasztott termék azonosítóját a localStorage-be, majd navigál a termék részletek oldalra.

**Biciklik komponens**

A Biciklik komponens felelős a kerékpártermékek megjelenítéséért az ISABIKE Kerékpárüzlet webhelyén.

**Feladatok**

* Betölti a kerékpártermékeket a backend szolgáltatásból.
* Megjeleníti a kerékpártermékeket kártya formájában, amelyek tartalmazzák a termék nevét, gyártóját, leírását, árát és egy részletek gombot.
* Lehetővé teszi a felhasználó számára, hogy részleteket lásson a kiválasztott termékről.
* Biztosítja a lapozást, hogy a felhasználó könnyen navigálhasson az oldalak között.

**Metódusok**

**ngOnInit()**

* **Leírás**: Ez a metódus az Angular életciklusának része, és a komponens inicializálásakor fut le.
* **Feladat**: Betölti a kerékpártermékeket és beállítja az oldalak számát a lapozáshoz.

**getProducts()**

* **Leírás**: Ez a metódus lekéri a kerékpártermékeket a backend szolgáltatásból.
* **Feladat**: Frissíti a products tömböt a lekért termékekkel, valamint beállítja az összes termék számát a lapozáshoz.

**onPageChange()**

* **Leírás**: Ez a metódus hívódik meg, amikor a felhasználó vált az oldalak között.
* **Feladat**: Beállítja az aktuális oldalszámot a kapott oldalszámra.

**showProductDetails()**

* **Leírás**: Ez a metódus hívódik meg, amikor a felhasználó részleteket kér egy termékről.
* **Feladat**: Elmenti a termék azonosítóját a localStorage-ben, majd navigál a részletek oldalra.

**Register komponens**

A regisztráció komponens lehetővé teszi a felhasználók számára, hogy regisztráljanak az ISABIKE Kerékpárüzlet webhelyére.

**Felhasználás**

A komponens egy regisztrációs űrlapot tartalmaz, amely lehetővé teszi a felhasználók számára, hogy megadják a regisztrációhoz szükséges adatokat.

**Alkalmazás**

1. **Felhasználónév**: A felhasználó teljes nevét kell megadnia.
2. **Keresztnév**: A felhasználó keresztnevét kell megadnia.
3. **Vezetéknév**: A felhasználó vezetéknevét kell megadnia.
4. **E-mail cím**: A felhasználó érvényes e-mail címét kell megadnia.
5. **Jelszó**: A felhasználó biztonságos jelszavát kell megadnia.

**Folyamat**

1. A felhasználó kitölti az összes szükséges mezőt az űrlapon.
2. A felhasználó elküldi az űrlapot a "Regisztráció" gombra kattintva.
3. Az alkalmazás ellenőrzi az adatok helyességét és regisztrálja a felhasználót.

**Jellemzők**

* A komponens használja a Reactive Forms modellt a form validációjához.
* Az űrlap ellenőrzi a megadott adatok formátumát és érvényességét.
* Ha sikeres a regisztráció, a felhasználó további tevékenységeket végezhet az alkalmazáson belül.

**Megjegyzések**

* Ha bármely mezőt helytelenül töltik ki, hibaüzenet jelenik meg a felhasználónak.
* Az űrlap elküldése előtt győződj meg róla, hogy minden mezőt helyesen töltöttél ki.

**ngOnInit()**

* **Leírás**: Ez a metódus az Angular életciklusának részeként fut le, amikor a komponens inicializálódik.
* **Feladat**: Létrehozza a regisztrációs űrlapot és beállítja a szükséges validátorokat a form mezőihez.

**registerAction()**

* **Leírás**: Ez a metódus az űrlap elküldésekor fut le, amikor a felhasználó megpróbál regisztrálni.
* **Feladat**: Ellenőrzi az összes form mezőt, és ha minden mező megfelelően van kitöltve, elküldi a regisztrációs adatokat a backend szolgáltatásnak.

**registerForm**

* **Leírás**: Ez egy FormGroup objektum, amely a regisztrációs űrlapot reprezentálja.
* **Feladat**: Ez a FormGroup objektum kezeli az összes form mezőt és validációt biztosít a megfelelő adatbevitelhez.

**validationErrors**

* **Leírás**: Ez egy objektum, amely a validációs hibák tárolására szolgál.
* **Feladat**: Ha bármelyik mező hibásan van kitöltve, ez az objektum tárolja és megjeleníti a hibaüzeneteket a felhasználónak.

**isSubmitting**

* **Leírás**: Ez a változó jelzi, hogy éppen folyamatban van-e az űrlap elküldése.
* **Feladat**: Amikor az űrlap elküldése folyamatban van, ez a változó igaz értéket kap, és letiltja az űrlap újbóli elküldését.

**Login komponens**

Ez a komponens felelős a felhasználó bejelentkezési felületének megjelenítéséért az ISABIKE Kerékpárüzlet webhelyén.

**Feladatok**

* Megjeleníti az e-mail cím és jelszó beviteli mezőket.
* Biztosítja a bejelentkezés gomb funkcióját.
* Amennyiben a felhasználó még nem rendelkezik fiókkal, lehetőséget biztosít a regisztrációra.

**Metódusok**

**ngOnInit()**

* **Leírás**: Az Angular életciklusának része, a komponens inicializálásakor fut le.

**loginAction()**

* **Leírás**: Ez a metódus hívódik meg, amikor a felhasználó be akar jelentkezni.
* **Feladatok**: 3. Ellenőrzi a beviteli mezők validitását. 2. Elküldi a bejelentkezési adatokat a backend szolgáltatásnak. 3. Amennyiben a bejelentkezés sikeres, átirányítja a felhasználót a kezdőoldalra. 4. Mentésre kerül a token a localStorage-be. 5. Lekéri a felhasználó adatait a localStorage-ból és elmenti őket.

**Verify komponens**

Ez a komponens felelős a felhasználói fiók megerősítéséért az ISABIKE Kerékpárüzlet webhelyén.

**Tartalom**

**Sikeres Megerősítés**

Ha a fiók megerősítése sikeres volt, megjelenik egy üzenet, amely megerősíti a sikert, és lehetővé teszi a továbblépést.

**Megerősítő Kód Bevitel**

Ezen részben a felhasználó beírhatja az e-mailben kapott 5 számjegyű megerősítő kódot.

**Műveletek**

**verifyAccount()**

* **Leírás**: A fiók megerősítése.
* **Feladatok**: Elküldi a beírt megerősítő kódot a megfelelő szolgáltatásnak. Ha a megerősítés sikeres volt, átirányítja a felhasználót a bejelentkező oldalra.

**Változók**

**code**

* **Típus**: string
* **Leírás**: A felhasználó által beírt megerősítő kód.

**isSubmitting**

* **Típus**: boolean
* **Leírás**: Megadja, hogy éppen folyamatban van-e a megerősítés folyamata.

**validationErrors**

* **Típus**: any
* **Leírás**: Az esetleges validációs hibák tárolására szolgáló változó.

**verificationSuccess**

* **Típus**: boolean
* **Leírás**: Megadja, hogy a fiók megerősítése sikeres volt-e vagy sem.

**Uer-rendeles komponens**

Ez a komponens felelős egy adott termék részleteinek megjelenítéséért az ISABIKE Kerékpárüzlet webhelyén.

**Tartalom**

**Kép**

Az adott termék képe megjelenítésre kerül, ha rendelkezésre áll.

**Termek Információk**

A termék alábbi információi kerülnek megjelenítésre:

* **Termék neve**: Az adott termék neve.
* **Kategória**: Az adott termék kategóriája.
* **Gyártó**: Az adott termék gyártója.
* **Mennyiség rakáron**: Az adott termék raktáron lévő mennyisége.
* **Súly**: Az adott termék súlya és mértékegysége.
* **Szín**: Az adott termék színe.
* **Leírás**: Az adott termék leírása.
* **Ár**: Az adott termék egységára.
* **Elérhetőség**: Az adott termék elérhetősége, ami lehet "Elérhető" vagy "Nem Elérhető" attól függően, hogy van-e készleten.

**Műveletek**

* **Másik termék megtekintése**: Visszatérés a terméklista oldalra.
* **Hozzáadás a kosárhoz**: Ha a felhasználó be van jelentkezve, lehetőség van a termék hozzáadására a kosárhoz.

**Metódusok**

**ngOnInit()**

* **Leírás**: Az Angular életciklusának része, a komponens inicializálásakor fut le.
* **Feladatok**: Betölti a termék részleteit az URL-ben megadott azonosító alapján.

**goBack()**

* **Leírás**: Visszatérés a terméklista oldalra.
* **Feladatok**: Navigál a terméklista oldalra és törli az esetlegesen eltárolt termék azonosítót.

**addCart()**

* **Leírás**: A termék hozzáadása a kosárhoz.
* **Feladatok**: Elküldi a kosárba helyezendő termék azonosítóját a kosár szolgáltatásnak, majd kezeli az esetleges válaszüzeneteket.

**isLoggedIn()**

* **Leírás**: Ellenőrzi, hogy a felhasználó be van-e jelentkezve.
* **Feladatok**: Visszaadja a bejelentkezés állapotát.

**Rolunk komponens**

Ez a komponens az ISABIKE kerékpárüzlet bemutatására szolgál. Az oldal három fő részre van osztva: az üdvözlő szakaszra, a történetre és az elérhetőségre.

**Tartalom**

**Bevezető szakasz**

Ez a rész üdvözli a látogatókat az ISABIKE-ban. Tartalmazza az üzlet nevét, egy rövid üzenetet a minőségi kerékpárokról és kiegészítőkről, valamint egy képet a kerékpárüzletről.

**Történet**

Itt található a kerékpárüzlet bemutatása és az ISABIKE-hoz való csatlakozás értékrendje. Részletesen ismerteti, hogy az ISABIKE nem csupán egy kerékpárüzlet, hanem egy közösség, ahol a kerékpározás iránti szenvedély megosztható és átélhető.

**Elérhetőség**

Ebben a szakaszban megtalálhatók az ISABIKE elérhetőségi adatai, mint például a telefonszám, az email cím és a cím. Ezek segítségével a látogatók könnyen kapcsolatba léphetnek az üzlettel.

**Szerviz komponens**

Ez az oldal a szerviz kínálatokat mutatja be az ISABIKE kerékpárüzletben. A felsorolt szolgáltatásokat és azok árait tartalmazza.

**Állapotfelmérés**

Ez a szolgáltatás magában foglalja a fékrendszer vizsgálatát, szekunder áttétel vizsgálatát, gumiabroncsok állapotának ellenőrzése.A munkadíj: 8 000 Ft.

**Olaj- és szűrőcsere**

Ez a szolgáltatás az olaj- és szűrőcserét jelenti a kezelési útmutatóban javasolt periódus alapján. A munkadíj: 8 000 Ft.

**Láncszett csere**

Ez a szolgáltatás a jó állapotú szekunder áttétel cseréjét szettben tartalmazza. A munkadíj: 14 000 Ft-tól.

**Fékszerviz**

Fontos a biztonságos közlekedés érdekében. Ez a szolgáltatás magában foglalja a fékalkatrészek ellenőrzését és szükség esetén cseréjét. A munkadíj: 12 000 Ft-tól.

**Teleszkópszerviz**

A kényelmes használat érdekében elengedhetetlen a tökéletes állapotú futómű. A munkadíj: 40 000 Ft-tól.

**Gumiabroncs csere**

A motorkerékpár legfontosabb alkatrésze a gumiabroncs. Ez a szolgáltatás a gumiabroncs cseréjét jelenti. A munkadíj: 10 000 Ft-tól.

**Rendeles komponens**

Ez a komponens felelős a vásárlók rendeléseinek rögzítéséért az ISABIKE Kerékpárüzlet webhelyén.

**Űrlapmezők**

1. **Vásárló telefonszáma**: A vásárló telefonszámának megadása kötelező. Az érvényes telefonszám formátumának meg kell felelnie.
2. **Szállítási cím**: A rendelés szállítási címének megadása kötelező.
3. **Megjegyzés**: Opcionális mező, ahol a vásárló megjegyzéseket fűzhet a rendeléshez.
4. **Szállító**: Az alapértelmezett szállító cég, amely a rendelést teljesíti. A felhasználó nem módosíthatja ezt a mezőt.
5. **Fizetési opció**: Az alapértelmezett fizetési mód, amelyet a vásárló választott. A felhasználó nem módosíthatja ezt a mezőt.
6. **Kedvezmény**: Az esetleges kedvezmények megjelenítése. A felhasználó nem módosíthatja ezt a mezőt.

**Metódusok**

**ngOnInit()**

* **Leírás**: Az Angular életciklusának része, a komponens inicializálásakor fut le.
* **Feladatok**: Létrehozza és validálja az űrlapot.

**onSubmit()**

* **Leírás**: A rendelés űrlap elküldésekor hívódik meg.
* **Feladatok**:
  1. Ellenőrzi az űrlap validitását.
  2. Elküldi a rendelés adatait a szervernek.
  3. Megjelenít egy üzenetet a sikeres vagy sikertelen rendelésről.

**Fontos megjegyzés**

* Ellenőrizd az összes kötelező mező kitöltését a rendelés elküldése előtt.
* Kezeld a szerver által visszaadott hibákat a rendelés küldésekor.

**Kosár komponens**

Ez a komponens felelős a felhasználó bevásárlókosarában található termékek megjelenítéséért az ISABIKE Kerékpárüzlet webhelyén.

**Tartalom**

A komponens egy listát jelenít meg a felhasználó bevásárlókosarában található termékekről. Minden termékhez tartozik a termék neve, mennyisége és hozzáadásának ideje. Minden termékhez egy gomb is tartozik, amely lehetővé teszi a termék törlését a kosárból.

**Metódusok**

**getItems()**

* **Leírás**: A felhasználó bevásárlókosarában található termékek lekérdezése.
* **Feladatok**: Lekéri a felhasználó bevásárlókosarában található termékeket a megfelelő szolgáltatástól, majd megjeleníti azokat a felületen. Ha a kosár üres, kiír egy figyelmeztetést és visszairányítja a felhasználót a biciklik oldalra.

**deleteItem()**

* **Leírás**: Termék törlése a kosárból.
* **Feladatok**: Törli az adott terméket a kosárból a megfelelő szolgáltatás segítségével. A törlés után frissíti a kosár tartalmát.

**Változók**

**shoppingCartData**

* **Típus**: any[]
* **Leírás**: A felhasználó bevásárlókosarában található termékek tömbje.

**Felhasznalo-Rendeles komponens**

Ez a komponens felelős a felhasználó rendeléseinek megjelenítéséért az ISABIKE Kerékpárüzlet webhelyén.

**Tartalom**

A komponens egy listát jelenít meg a felhasználó rendeléseiről, minden rendeléshez tartozik a rendelés ideje, megjegyzés, szállító és fizetési opció. Minden rendeléshez tartozik egy gomb is, amely lehetővé teszi a rendelés részletes adatainak megtekintését.

**Metódusok**

**getItems()**

* **Leírás**: A felhasználó rendeléseinek lekérdezése.
* **Feladatok**: Lekéri a felhasználó rendeléseit a megfelelő szolgáltatástól, majd megjeleníti azokat a felületen. Ha a lekérdezés sikertelen, visszairányítja a felhasználót a biciklik oldalra, és kiír egy figyelmeztetést.

**getRendelesTermekek()**

* **Leírás**: Egy rendeléshez tartozó termékek lekérdezése.
* **Feladatok**: Lekéri az adott rendeléshez tartozó termékeket a megfelelő szolgáltatástól, majd megjeleníti azokat egy felugró üzenetben. Az üzenet tartalmazza a termék nevét, darabszámát, egységárát és állapotát.

**kezbesitve()**

* **Leírás**: Állapot visszaadása szöveges formában.
* **Feladatok**: Az adott szám alapján visszaadja a megfelelő állapot szöveges formában.

**getSzalito()**

* **Leírás**: Szállító nevének visszaadása szöveges formában.
* **Feladatok**: Az adott szám alapján visszaadja a megfelelő szállító nevét szöveges formában.

**getFizOp()**

* **Leírás**: Fizetési opció nevének visszaadása szöveges formában.
* **Feladatok**: Az adott szám alapján visszaadja a megfelelő fizetési opció nevét szöveges formában.

**Változók**

**rendelesData**

* **Típus**: any[]
* **Leírás**: A felhasználó rendeléseinek tömbje.

**rendelesTermekek**

* **Típus**: any[]
* **Leírás**: A kiválasztott rendeléshez tartozó termékek tömbje.

**UserAuthService Szolgáltatás**

Ez a szolgáltatás felelős a felhasználók autentikációjával és az azokhoz kapcsolódó műveletekkel.

**Metódusok**

**login():**

* **Leírás**: Felhasználó bejelentkezésének kezelése.
* **Végpont**: POST /api/login
* **Paraméter**: loginObj - Bejelentkezési adatokat tartalmazó objektum.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP POST kérést a bejelentkezési végpontra a megadott adatokkal. Visszatérési értékként egy Observable-t kapunk, amely tartalmazza a bejelentkezési választ.

**getUserData():**

* **Leírás**: Felhasználó adatainak lekérése a token alapján.
* **Végpont**: POST /api/getonefelhasznalo
* **Paraméter**: token - Felhasználóhoz tartozó token.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP POST kérést a felhasználó adatainak lekérdezésére a token alapján. Visszatérési értékként egy Observable-t kapunk, amely tartalmazza a felhasználó adatait.

**register():**

* **Leírás**: Felhasználó regisztrációjának kezelése.
* **Feladatok**: Ez a metódus küld egy regisztrációs kérést a megadott adatokkal a backend számára.

**verifyAccount():**

* **Leírás**: Felhasználó fiókjának ellenőrzése a megadott kóddal.
* **Végpont**: PUT /api/verify
* **Paraméter**: data - Ellenőrzési kódot tartalmazó objektum.
* **Visszatérési érték típusa**: Promise
* **Feladatok**: Ez a metódus küld egy HTTP PUT kérést a felhasználó fiókjának ellenőrzésére a megadott kóddal. A visszatérési érték egy Promise, amely tartalmazza az ellenőrzés eredményét.

**isLoggedIn():**

* **Leírás**: Ellenőrzi, hogy a felhasználó be van-e jelentkezve.
* **Visszatérési érték típusa**: boolean
* **Feladatok**: Ez a metódus ellenőrzi, hogy a felhasználó be van-e jelentkezve a helyi tárolt tokennel.

**logout():**

* **Leírás**: Felhasználó kijelentkezése.
* **Végpont**: POST /api/logout
* **Paraméter**: asd - Felhasználóhoz tartozó token.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP POST kérést a kijelentkezési végpontra a felhasználóhoz tartozó tokennel. Visszatérési értékként egy Observable-t kapunk, amely tartalmazza a kijelentkezési választ.

**Cart Szolgáltatás**

Ez a szolgáltatás felelős a felhasználó kosarával kapcsolatos műveletekkel.

**Metódusok**

**addToCart():**

* **Leírás**: Termék hozzáadása a kosárhoz.
* **Végpont**: POST /api/addkosar
* **Paraméterek**:
  + productId: A termék azonosítója, amelyet a kosárhoz szeretnénk adni.
  + quantity: A termék mennyisége.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP POST kérést a kosárhoz adás végpontra a megadott adatokkal.

**getCartItems():**

* **Leírás**: Kosár elemeinek lekérése.
* **Végpont**: POST /api/getonekosar
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP POST kérést a kosár elemeinek lekérdezésére.

**updateCartItem():**

* **Leírás**: Kosárban lévő elem mennyiségének frissítése.
* **Végpont**: PUT /api/updatekosar
* **Paraméterek**:
  + cartItemId: A kosárban lévő elem azonosítója.
  + quantity: Az új mennyiség.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP PUT kérést a kosárban lévő elem mennyiségének frissítésére.

**removeCartItem():**

* **Leírás**: Elem eltávolítása a kosárból.
* **Végpont**: DELETE /api/deletekosar/:cartItemId
* **Paraméter**: cartItemId: A kosárból eltávolítandó elem azonosítója.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP DELETE kérést a megadott elem eltávolítására a kosárból.

**submitOrder():**

* **Leírás**: Rendelés leadása.
* **Végpont**: POST /api/addrendeles
* **Paraméter**: orderData: A rendelés adatait tartalmazó objektum.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP POST kérést a rendelés leadás végpontra a megadott adatokkal.

**getOrder():**

* **Leírás**: Felhasználó rendeléseinek lekérése.
* **Végpont**: POST /api/getonerendeles
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP POST kérést a felhasználó rendeléseinek lekérdezésére.

**getOrderData():**

* **Leírás**: Egy rendelés részletes adatainak lekérése.
* **Végpont**: GET /api/getonerendelestermekek/:id
* **Paraméter**: id: A rendelés azonosítója.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP GET kérést a megadott rendelés részletes adatainak lekérdezésére.

**Authentication-client Szolgáltatás**

Ez a szolgáltatás felelős az autentikációs műveletekhez kapcsolódó ügyféloldali feladatok végrehajtásáért.

**Metódusok**

**register():**

* **Leírás**: Regisztrálja az új felhasználót az adott névvel, e-mail címmel és jelszóval.
* **Paraméterek**:
  + felhasznalo\_nev: string - A regisztrált felhasználó teljes neve.
  + kereszt\_nev: string - A felhasználó keresztneve.
  + vezetek\_nev: string - A felhasználó vezetékneve.
  + email: string - A felhasználó e-mail címe.
  + password: string - A felhasználó jelszava.
* **Visszatérési érték**: Egy Observable, amely tartalmazza a regisztrációval kapcsolatos válaszobjektumot.

**Pagination Szolgáltatás**

Ez a szolgáltatás felelős az aktuális oldalszám tárolásáért és frissítéséért.

**Metódusok**

**setCurrentPage():**

* **Leírás**: Beállítja az aktuális oldalszámot.
* **Paraméter**:
  + page: Az új aktuális oldalszám.
* **Feladatok**: Ez a metódus frissíti az aktuális oldalszám értékét a megadott értékre.

**Product szolgáltatás**

Ez a szolgáltatás felelős a termékekkel kapcsolatos adatok lekérdezéséért.

**Főbb funkciók**

**getProducts():**

* **Leírás**: Termékek lekérése.
* **Végpont**: GET /api/termekek/
* **Visszatérési érték típusa**: Observable<any[]>
* **Feladatok**: Ez a metódus küld egy HTTP GET kérést a termékek lekérdezésére.

**getProduct():**

* **Leírás**: Egy termék lekérése az azonosítója alapján.
* **Végpont**: GET /api/onetermekek/:id
* **Paraméter**: id: A termék azonosítója.
* **Visszatérési érték típusa**: Observable
* **Feladatok**: Ez a metódus küld egy HTTP GET kérést a megadott termék azonosítója alapján.