# Backend dokumentáció

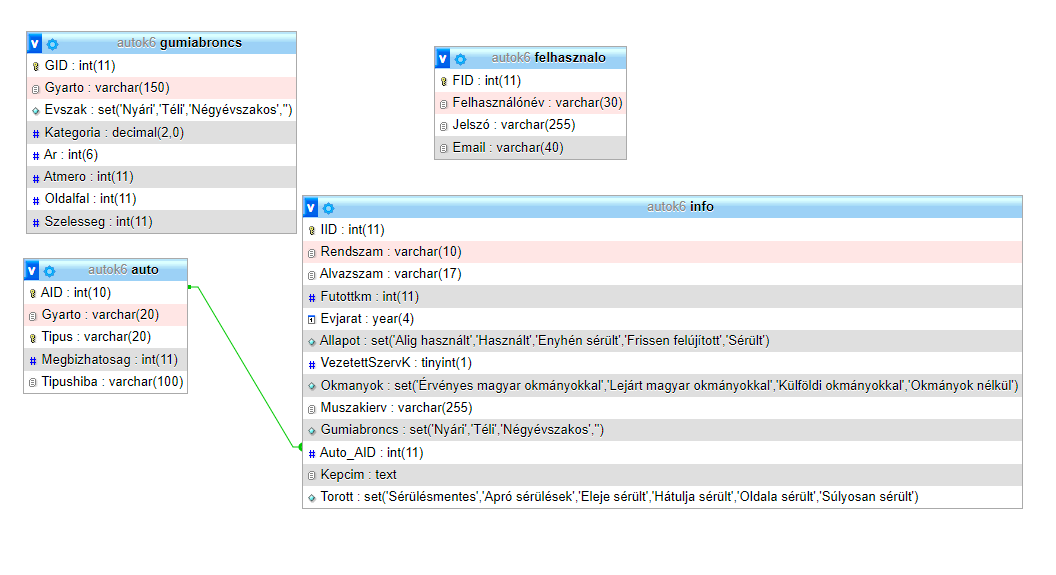
## Fejlesztői környezet ismertetése:

Fejlesztő program: Visual Studio Code 2019

### Használt technológiák:

Node.js, Vue.js, MySql

### Adatbázis, kapcsolati diagram:

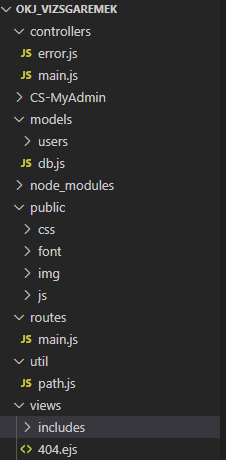
****

Három részre bontható az adatbázis, az első a személygépjármű lekérdezésére szolgáló adatokat tartalmazza két áblában, a második a gumiabroncsok értékesítésével kapcsolatos. A harmadik pedig a regisztrált feelasználók adatait tartalmazza.

Az **„auto”** táblában találhatók egy autó általános adatai, így egy adott típusú és évjáratú járműből egy rekordot tartalmaz. Az **„AID”** idegenkulcs az **„info”** táblában ami tartalmazza az egyedi információkat az adott járműről.

A **„felhasznalo”** táblában kerül rögzítésre minden felhasználó sikeres regisztrációja, **md5** titkosított jelszava, emailcíme, és felhasználóneve. A **„gumiabroncs”** tábla pedig tartalmazza a webshopunkban elérhető gumiabroncsok minden adatát.

### Könyvtár struktúra:

**controllers** – a **routes** mappából meghívott metódusok és eljárások találhatóak ebben a mappában

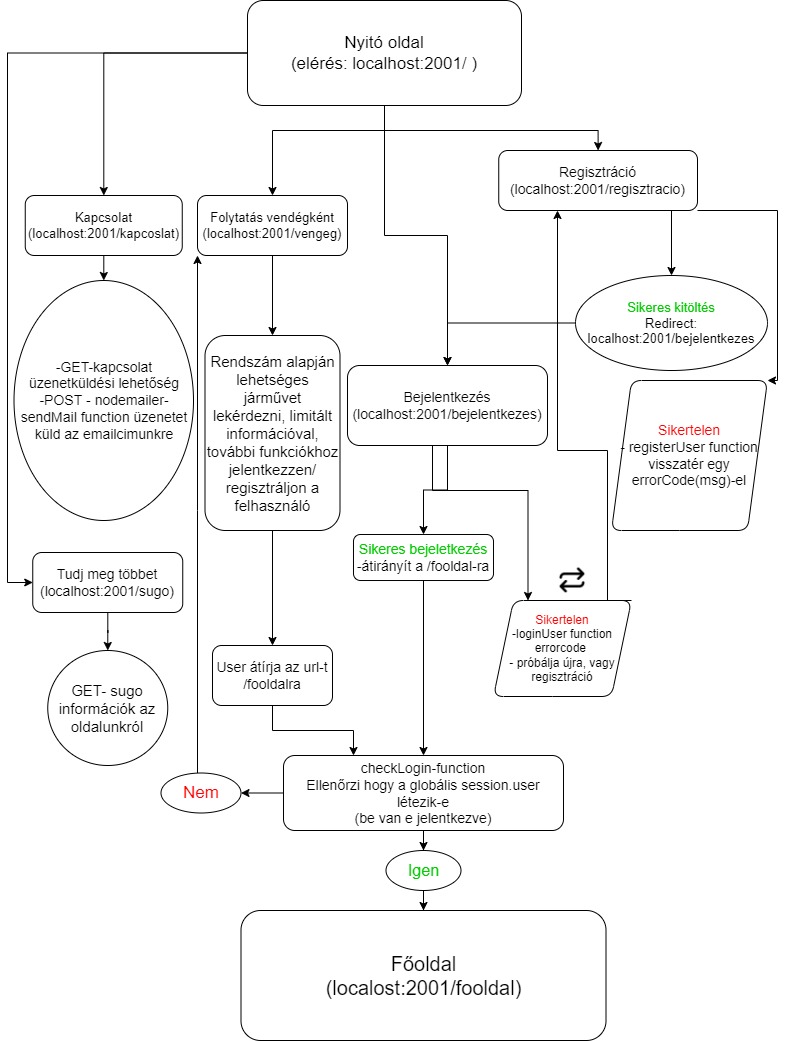
**models** – a felhasználók regisztrációját, autentikációját, és az adatbázisból való lekérdezések kódjait tartalmazza melyeket exportálás után a **/controllers/main.js** fájlba inportálunk

**public** – a frontendhez szükséges fájlokat tartalmazza a megfelelő mappákban

**routes** – a szerver útvonalait tartalmazza

**views** – tartalmazza a renderelésre kész .ejs fájlokat, a **/include** a views .ejs fájlaiba beépített kódrészleteket ezzel elkerülve az ismétlődő kódokat

## A weboldal működése:



## Nyitó oldal

A szerver a **localhost:2001**-es porton fut, ha megkapja böngészőből a „**/**” kérést akkor a megfelelő oldalt legenerálja a felhasználónak egy .ejs állományban, a megadott paraméterekkel.

A **/routes/main.js** állomány felelős a kérések irányításáért:

(kódrészlet:)

router.get('/bejelentkezes',

mainController.getBejelentkezes);

router.post('/bejelentkezes', mainController.postLogin);

router.get('/bejelentkezes', mainController.checkLogin);

router.get('/regisztracio', mainController.getRegisztracio);

router.post('/regisztracio', validationForm.form ,mainController.validateRegistration);

router.get('/sugo', mainController.getSugo);

router.get('/kapcsolat', mainController.getContactus);

router.post('/contactuskuld', mainController.postContactus);

A fent látható módon kerülnek meghívásra az adott útvonalakhoz rendelt metódusok a **/controllers/main.js** fájlból.

A POST kéréseket mindig egy **post** típusú **form** küldi a megfelelő **.ejs** állományból és ennek megfelelően kerülnek meghívásra a POST metódusok.

### Kezdő oldal

router.get('/', mainController.getIndex);

Ezzel meghívásra kerül a **/controllers/main.js** fájlban található függvény ami lerendereli a **/views/kezdolap.ejs** fájlt a user paraméterrel:

//Kezdő oldal

exports.getIndex = (req, res, next) => {

    res.render("kezdolap", {

        pageTitle: "CarScope",

        path: "/",

        user: req.session.user,

    });

};

## Paraméterek:

A kezdő oldal megjelenítésekor az .ejs fájlnak átadásra kerül a fenti példában látható módon egy „user” nevű paraméter ami egy globális változótól kapja értékét (tartalmazza, hogy a felhasználó be van-e jelentkezve)  
Az .ejs kódrészletben egy if() elágazásban ennek a változónak az értéke határozza meg a navigációs sáv kinézetét a következő módon:

<% if(user == undefined) { %>

          <form class="form-inline my-2 my-lg-0">

            <a class="btn btn-outline-light" href="bejelentkezes"

            type="submit">Bejelentkezés</a>

            &nbsp;&nbsp;

            <a href="regisztracio" class="btn btn-outline-light"

            type="submit">Regisztráció</a>

          </form>

      <% }else{ %>

          <form class="form-inline my-2 my-lg-0" method="post"

          action="/kijelentkezes">

            <button class="btn btn-outline-light"

            type="submit">Kijelentkezés</button>

          </form>

     <% } %>

Amennyiben a felhasználó nincs bejelentkezve (a user paraméter üres), a **„Bejelentkezés”**, **„Regisztráció”** gombok lesznek láthatóak, ellenkező esetben a **”Kijelentkezés”** gomb.

## Regisztráció

A regisztrációs űrlap kitöltése után a felhasználó rákattint a **„Regisztráció”** gombra és lefut a **/regisztracio** form, egy post kérés.  
A **/controllers/main.js** - meghívja a **validateRegistration** metódust:

exports.validateRegistration = (req, res, next) => {

    let msg = [];

    const error = validationResult(req);

    const errorList = error.array();

    errorList.forEach((element) => {

        msg.push({ param: element.param, msg: element.msg });

    });

    if (!error.isEmpty()) {

        res.render("regisztracio", {

            pageTitle: "CarScope - Regisztráció",

            path: "/regisztracio",

            errorCode: msg,

            user: req.session.user,

        });

    } else {

        register.registerUser(req, res, function(err, data) {

            if (err) {

                res.json({ error: true });

            } else {

                res.redirect("/");

            }

        });

    }

Az **express-validator** importálása után feliratkozunk a csomagokra és egy **„error”** változóban tároljuk a **/models/users/validate.js** által visszaadott hibaüzenetet.

Ha a felhasználó valamit rosszul töltött ki, a hibaüzenet kiírásra kerül a regisztrációs oldalon.

Ellenkező esetben lefut a **/models/users/register.js – registerUser** metódusa

(kódrészlet:)

register.prototype.registerUser = function(req, res, next) {

        secret = "!%/=.\_"+req.body.jelszo+"!%/.\_";

        jelszo = req.body.jelszo;

        md5Hasher = crypto.createHmac("md5", secret);

        jelszo = md5Hasher.update(jelszo).digest("hex");

        params = [req.body.felhasznalonev, req.body.email,jelszo, req.body.password\_again,0],

        checkAvailabilityQuery = 'SELECT Felhasználónév, Email FROM felhasznalo WHERE Felhasználónév = ? OR Email = ?',

        registerUserQuery = 'INSERT INTO felhasznalo(Felhasználónév, Email, Jelszó) VALUES (?,?,?)',

A felhasználó által megadott jelszó először titkosításra kerül az **md5** egyirányú kódolási szabvány szerint.

A **checkAvailabilityQuery** mysql lekérdezés ellenőrzi, hogy a felhasználó már szerepel-e az adatbázisban, ha nem, a **registerUserQuery** insert into beszúrja az adatokat az adatbázis **„felhasznalo”** táblájába.

Ha sikeres a regisztráció a **Nodemailer** használatával emailt küldünk a felhasználónak a sikeres regisztrációról:

var nodemailer = require('nodemailer');

var transporter = nodemailer.createTransport({

    service: 'gmail',

    auth: {

        user: 'carscope.site@gmail.com',

        pass: 'Carscope2022'

    }

});

A nodemailer levelező szerverként használja a gmail fiókunkat. A működése érdekben azonban szükséges volt a CarScope gmail fiók biztonsági beállításait átállítani, hogy engedélyezze külső alkalmazások számára alevelező használatát.

var mailOptions = {

                       to: req.body.email,

                       subject: 'SIKER! - CarScope',

                       text: 'Sikeresen regisztrál a CarScopeoldalon!'

             };

             transporter.sendMail(mailOptions,function( error, info){

               if (error) {console.log(error);

                    } else {

                       connection.release();

                       callback(null,null);

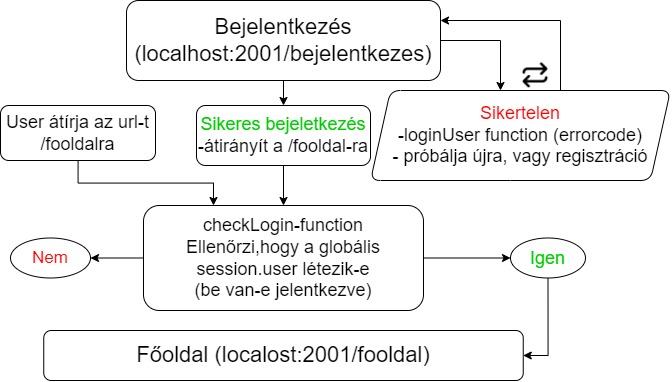
                       res.redirect('/bejelentkezes')

                    }

         });

Ha sikerült a felhasználó regisztrálása és az email elküldése, a **/bejelentkezes** oldal jelenik meg a felhasználó számára.

## Bejelentkezés



### Honnan tudjuk, hogy tényleg be van-e jelentkezve a felhasználó?

Amennyiben sikeres a bejelentkezés, vagy a felhasználó olyan url-t ír be a keresőbe melynek eléréséhez szükséges a bejelentkezés, az oldal betöltése előtt a szerver ellenőrzi, hogy az adott helyről érkező kérést végrehajtó felhasználó be van-e jelentkezve.

**/controllers/main.js** (122.sor)

//Be van-e jelentkezve

exports.checkLogin = (req, res, next) => {

    if (req.session.user != null) {

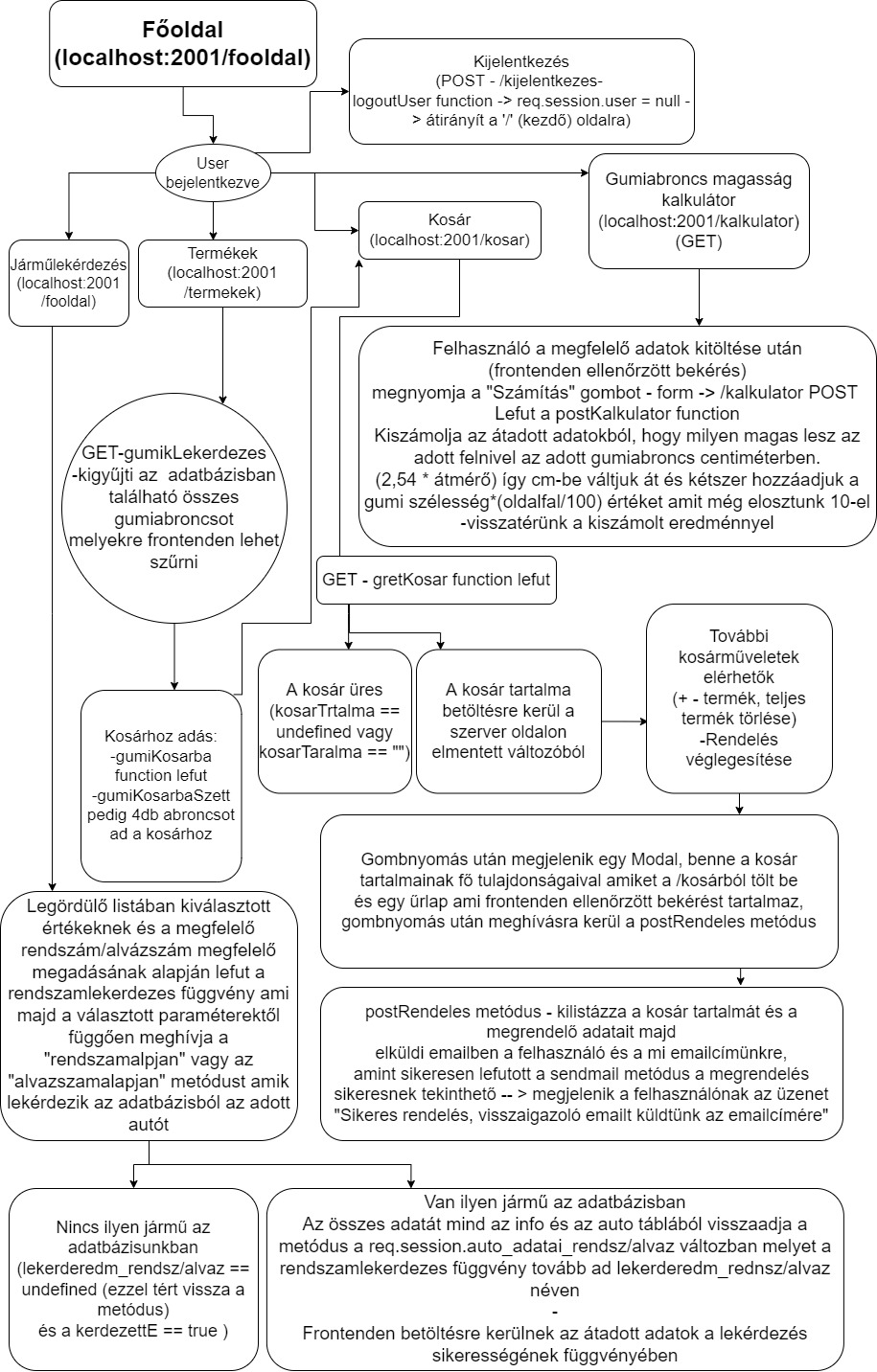
        res.redirect("/fooldal");

    } else {

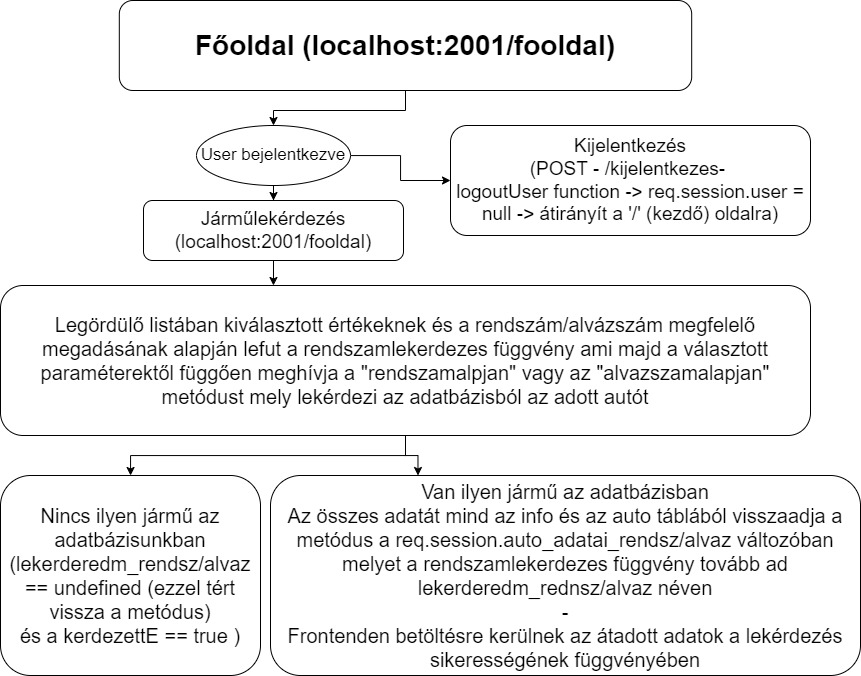
        res.redirect("/vendeg");

Amennyiben a **session** által létrehozott **user** változó létezik, tehát nem „null” vagy „undefined”, a szerver átirányít a **/fooldal** url-re, és a felhasználó bejelentkezve folytathatja tovább az oldal használatát.

## Főoldal működését bemutató ábra



## Jármű lekérdezés



Miután a felhasználó megadta milyen paraméter alapján szeretne jármű adatokat lekérdezni és helyesen kitöltötte a beviteli mezőt (a hibás adatok bevitelét frontenden ellenőrizzük, így backenden már nem szükséges az ellenőrzés), a „Keresés” gomb lenyomására a **/fooldal** form **POST** kérése lefut - **/routes/main.js** felelős az irányításért – meghívásra kerül a **/controllers/main.js** – rendszamlekerdezes metódusa:

exports.rendszamlekerdezes = (req, res, next) => {

    if (req.body.R == "True") {

        lekerdR.rendszamalapjan(req, res, function(err, data) {

A legördülő listából átadott érték alapján (**body.R** vagy **body.A**) eldől, hogy milyen paraméter alapján kérdezünk le járművet és ennek függvényében meghívásra kerül a **/models/users/auto.js** -ből importált „rendszamalapjan” vagy „alvazszamalapjan” függvény.

lekerd.prototype.rendszamalapjan = function(req, res, callback){

    rendszam = req.body.rendszam,

    params = [rendszam],

    lekerdquery = 'SELECT auto.Gyarto, auto.Tipus, info.Evjarat, info.Allapot, info.Futottkm, info.VezetettSzervK, auto.Megbizhatosag, auto.Tipushiba, info.Okmanyok, info.Muszakierv, info.Alvazszam, info.Gumiabroncs, info.Kepcim, info.Torott FROM info INNER JOIN auto ON info.Auto\_AID = auto.AID WHERE info.Rendszam = ?';

    mysqlPool.getConnection(function(err, connection){

        connection.query(lekerdquery, params, function(err, rows, fields){

            if(rows.length <= 0){

                connection.release();

                console.log(rendszam);

                console.log("Nincs ilyen rendszámú");

                req.session.auto\_adatai\_rendsz = null;

                req.session.auto\_adatai\_alvaz = null;

                callback(null,undefined);

            }else{

                req.session.auto\_adatai\_rendsz = rows[0];

                req.session.auto\_adatai\_alvaz = null;

                console.log(req.session.auto\_adatai\_rendsz)

                connection.release();

                callback(null,rows[0]);

            }

        });

    });

}

Definiálájuk a mySql lekérdezést (a fenti példában rendszám alapján), majd meghívjuk a **/models/db.js**-fájlból importált **getConnection** függvényt, ami ha üres tömbbel tér vissza akkor nincs a feltételnek megfelelő jármű az adatbázisban, ellenkező esetben átadásra kerül az **.auto\_adatai\_rendsz** tömbben.

Visszatérünk a **/controllers/main.js** fájl „rendzsamlekerdezes” metódusába ami a visszatérési értékeknek megfelelően le rendereli a **fooldal.ejs** fájlt és átadásra kerülnek a megfelelő paraméterek.  
Ha volt az adatbázisban a kérésnek megfelelő jármű, a **/views/fooldal.ejs**  fájlban ejs kód segítségével kiírjuk az autó adatait, pl.:

<h1 class="card-title"><%=lekerderedm\_rendsz.Gyarto %>

                      <%=lekerderedm\_rendsz.Tipus %> </h1>

     <h5>Évjárat: <%= lekerderedm\_rendsz.Evjarat %></h5>

     <h5>Állapot: <%= lekerderedm\_rendsz.Allapot %></h5>

     <h4>Futoásteljesítmény:</h4>

     <h5><%= lekerderedm\_rendsz.Futottkm %> </h5>

## Kosár

### Termékek kilistázása

Az adatbázisban szereplő gumiabroncsokat egy SQL lekérdezést végrehajtó függvény segítségével kiírjuk. Ezeket frontenden szűrni lehet.

### Termékek kosárhoz adása

Ha a felhasználó kosárhoz ad egy vagy több terméket akkor a következő metódus fut le:

**/controllers/main.js**

//Gumik kosárba helyezése

exports.gumiKosarba = (req, res, next) => {

    function kosarhozad() {

        termek = req.body.termek\_id;

        if (req.session.user.kosar == undefined) {

            req.session.user.kosar = [];

            var kosar = { termek\_id: termek, qty: 1 };

            req.session.user.kosar.push(kosar);

        } else {

            var contains = false;

            for (let termek of req.session.user.kosar){

                if (termek.termek\_id == req.body.termek\_id) {

                    termek.qty += 1;

                    var contains = true;}

            }

         if (contains == false){

         req.session.user.kosar.push({ termek\_id:termek, qty: 1});}}

        res.redirect("/termekek");

}

    setTimeout(kosarhozad, 950);

};

A **„kosarhozad”** függvény ellenőrzi, hogy létezik-e már a jelenleg bejelentkezett felhasználó számára egy globális **„session.user.kosar”** változó. Ha nem akkor létrehozzuk és eltároljuk azt a termék id-ját ahol a függvény meghívásra került és a mennyiségét 1 re állítjuk.  
Ha már van a kosárban ilyen id-vel rendelkező termék akkor csak a mennyiségét növeljük. – Később ezt töltjük be a **/kosár** oldalon.

## Tesztelés

Teszteléshez a **Jest testert** használtuk a következő módon:

**/public/test/server.test.js**

const login = require("../../models/users/login.js");

const req = require("express/lib/request");

test("login-jo", () => {

    var req = [];

    req.body = [];

    req.body.email = "proba@proba.com";

    req.body.jelszo = "Proba1122";

    req.session = [];

    var res = [];

    var rows = [];

    login.loginUser(req, res, function(err, data) {

        expect(err).toBe(null)

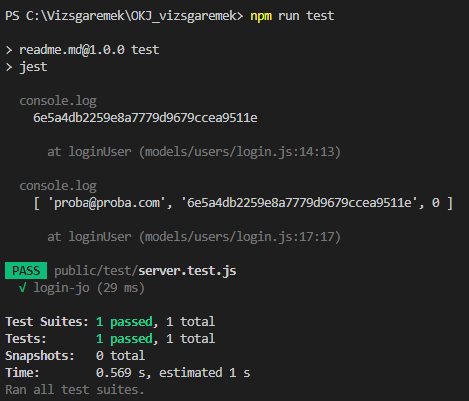
    })

    req.session.user = null;

});

A kódban látott felhasználó már regisztrálva van, így ezekkel a paraméterekkel meghívott **loginUser** metódus hiba nélkül fut le.

A konzolba beírt **npm run test** parancs után a teszt lefut és a következő üzenetet látjuk:



A teszt sikeresen lefutott és a console.log ban látható, hogy a **loginUser** metódus lefutott és az md5 titkosítást is elvégezte az általunk megaddott jelszón.

Ha a tesztet nem regisztrált felhasználóval futtatjuk le akkor errort fogunk kapni, tehát a tesztet is ennek megfelelően kell megírni:

expect(err).toBe(true)

### Regisztráció tesztelése

test("register-jo", () => {

    var req = [];

     req.body = [];

     req.body.felhasznalonev = "test";

     req.body.email = "test@test.com";

     req.body.jelszo = "Test11222";

     req.body.password\_again = "Test11222";

     req.session = [];

     var res = [];

     var res = function(){};

     res.render = function(){};

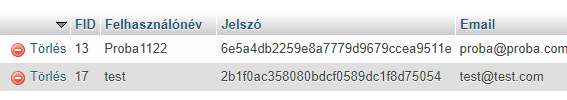
     register.registerUser(req, res, function(err, data) {

         expect(err).toBe(null)

    })

});

A paraméterekben átadott felhasználó még nincs regisztrálva.   
A teszt sikeresen lefut, tehát nem hibával tér vissza a **registerUser** metódus és az adatbázisban is megjelenik az új felhasználó, és a titkosított jelszava:



Rossz adatok esetén a tesztelés szintén lefut amennyiben az **err** paraméter **true**-ra van állítva, így értelemszerűen az adatbázisban nem lesz rögzítve a felhasználó.