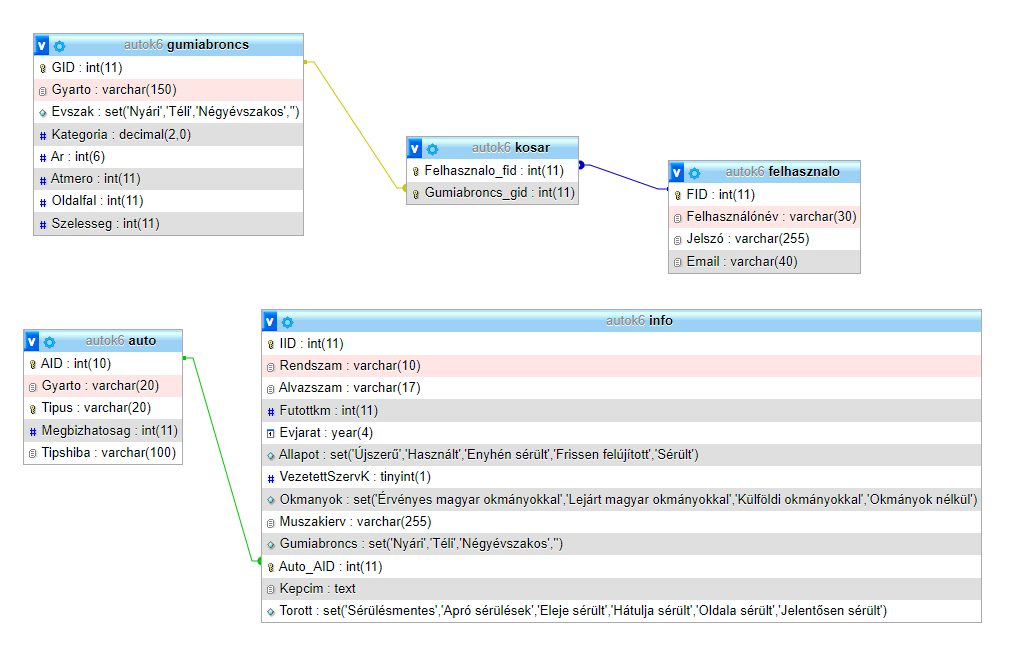
# Fejlesztői környezet ismertetése:

Fejlesztő program: Visual Studio Code 2019

### Használt technológiák:

Node.js Vue.js, mySql

### Adatbázis, kapcsolati diagram:

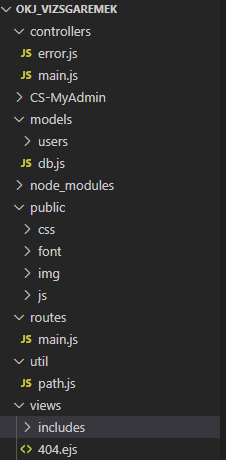
****

Két fő részre bontható az adatbázis, az egyik a személygépjármű lekérdezésére szolgáló adatokat tartalmazza a második pedig a gumiabroncsok értékesítésével kapcsolatos.

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, titkosított jelszava, emailcíme, és felhasználóneve. A „gumiabroncs” tábla pedig tartalmazza a webshopunkban elérhető gumiabroncsok minden adatát.

### Könyvtárstruktúra:

**controllers** -

**models** – a felhasználók regisztrációját, autentikációját és az adatbázisból való lekérdezések kódjait tartalmazza, továbbá

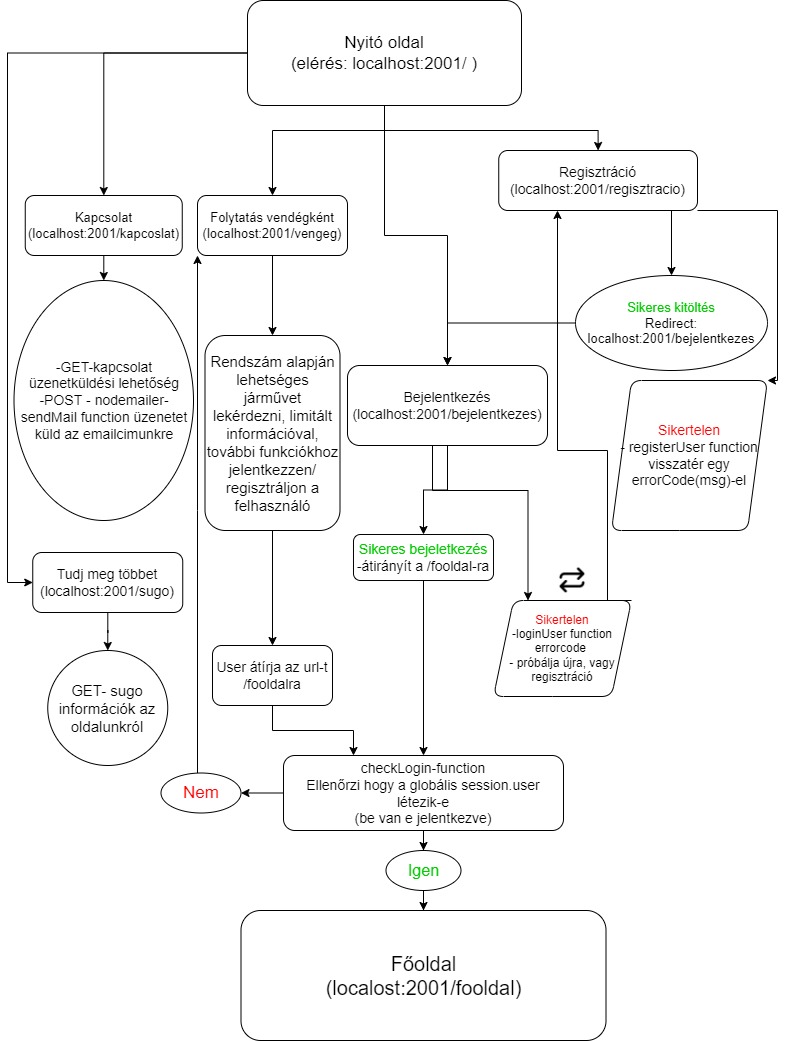
**public** – a frontendhez szükséges fájlokat tartalmazza a megfelelő mappákban

**routes** – a szerver útvonalait tartalmazza

**util** -

**views** – tartalmazza az .ejs fájlokat és a(z) /include a views ejs fájlaiba beépített kódrészleteket

# Weboldal működésének ábrái



# 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, (jelen esetben GET):

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

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

//Kezdő oldal

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

    res.render("kezdolap", {

        pageTitle: "CarScope",

        path: "/",

        user: req.session.user,

    });

};

Az összes GET kérés hasonló módon történik az oldalon, több paraméterrel.

## 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)  
Egy .ejs kódrészletben az if() eljárásban ennek a változónak az értéke határozza meg a navigációs sáv kinézetét.

<% 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 **„Bejelentkezés”**, **„Regisztráció”** gombok lesznek láthatóak, ellenkező esetben a **”Kijelentkezés”** gomb.

# Regisztráció

Regisztrációs űrlap kitöltése után a felhasználó rányom a **„Regisztráció”** gombra és lefut a **/regisztracio** form, egy post kérés.  
**/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** beimportálása 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** metódus

(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 egy md5 egyirányú kódolási szabvány szerint.  
A **checkAvailabilityQuery** mysql lekérdezés ellenőrzi, hogy a felhasználó már szerepel-e az adatbázisban, amennyiben 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 szükséges volt a gmail fiók biztonsági beállításait átállítani.

var mailOptions = {

                       to: req.body.email,

                       subject: 'SIKER! - CarScope',

                       text: 'Sikeresen regisztrál a CarScope oldalon!'

             };

             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.

# C:\Users\pusoma.gergo\Downloads\elsooldaldrawabrabejelent.jpgBejelentkezés

### 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 {

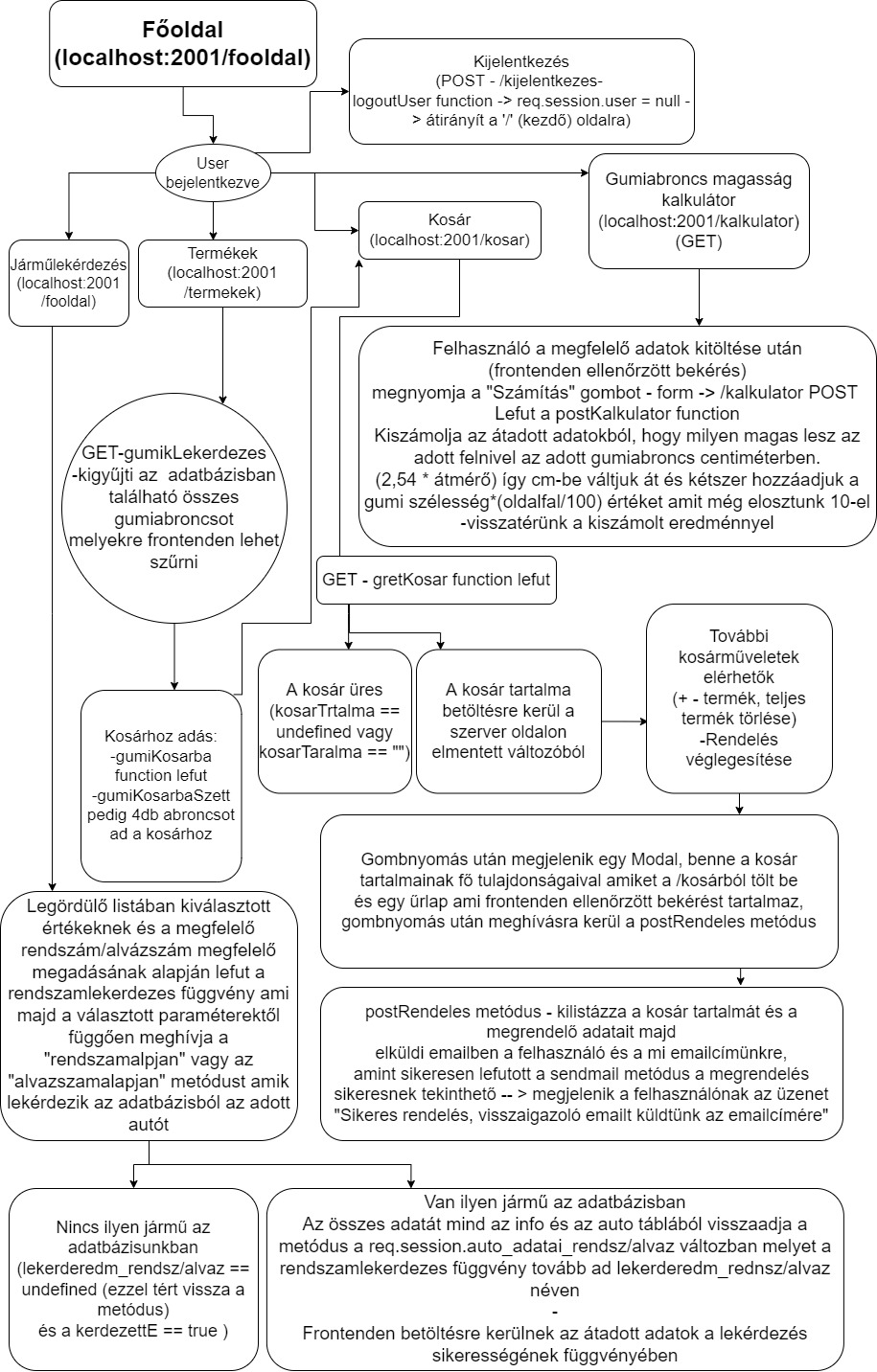
        res.redirect("/vendeg");

    }

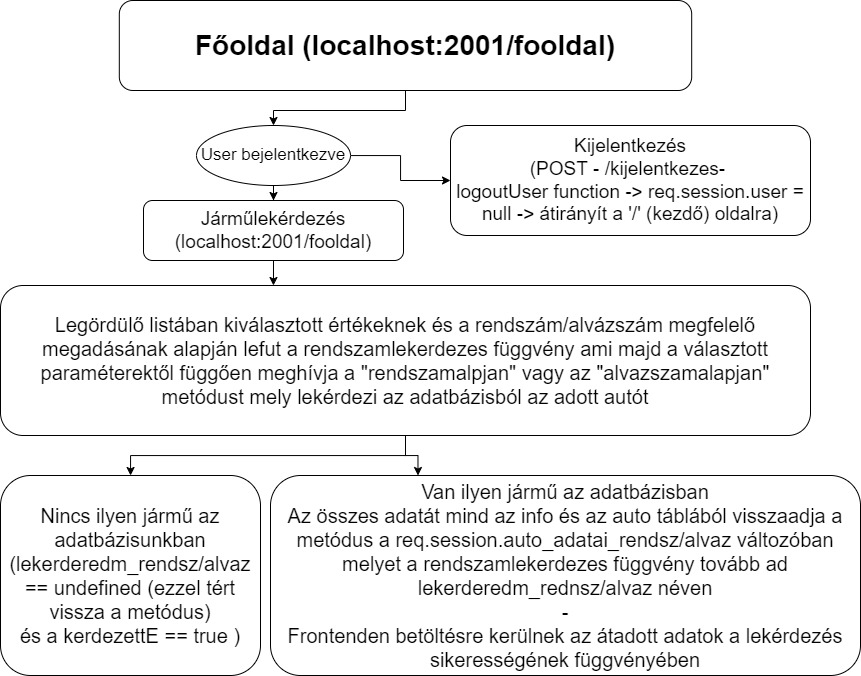
};

Amennyiben a **session** által létrehozott **user** változó létezik, tehát nem „null”, 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 „Keresés” gomb lenyomására **/fooldal** form **post** kérése lefut - /routes/main.js felelős az irányításért – meghívásra kerül a /controllers/main.js –

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

    if (req.body.R == "True") {

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