**Подготовка за финалниот проект**

**Имплементација на рестрикција на одредени улогу во нашиот случај Админ**

Имаме некоја рута која сакаме да дозволиме некоја функција која би била возможна само за админ, администратор, или која улога и да ја кажеме, во случајов сакаме да имплементираме рута со која рута се бриши документ од дата базата

app.delete( "/avtomobili/:id", auth.protect,auth.restrict("administrator", "admin"),avtoHandler.deletе);

Сакаме да дозволиме само админстраторот или админот во нашиот случај да можат да избришат документ од автомобили колекцијата, пишуваме уште еден middleware функција

exports.restrict = (...roles) => {

return (req, res, next) => {

if (!roles.includes(req.user.role)) {

return res.status(403).send("You dont have access");

}

next();

};

};

Оваа функција како параметри ги зема во заградата запиашаните атрибути во нашиот случај администратор и админ и проверуваме дали во нашиот корисник има таква улога, каде што во случајот би провериле во req.user.role = administrator , ако не е администратор тогаш ќе неможе во нашиов случај да избрише документ од автомобили колекцијата, а ова req.user.role го добиваме од req.user = userTrue; што го запишавме во протект middelwarot, и затоа restrict middelwarot го пишуваме после протект значи ако сакаме да извршиме рестрикција на некои улоги од корисникот мора прво во рутата да го пишиме протект миделварот, и на крај задаваме req.user = userTrue, и после сите функции после middlewarot што се ќе можат да пристапат за кој корисник станува збор дали има улога за админ или не   
  
exports.protect = async (req, res, next) => {

try {

let token;

if (

req.headers.authorization &&

req.headers.authorization.startsWith("Bearer")

) {

token = req.headers.authorization.split(" ")[1];

} else if (req.cookies.jwt) {

token = req.cookies.jwt;

}

if (!token) {

return res.status(500).send("You are not logged in! please log in");

}

const decoded = await promisify(jwt.verify)(token, process.env.JWT\_SECRET);

const userTrue = await User.findById(decoded.id);

if (!userTrue) {

return res.status(401).send("User doenst longer exist!");

}

req.user = userTrue;

next();

} catch (err) {

return res.status(500).send(err.message);

}

};

И еве како би изгледала шемата за да имплементираме улога како на пример администратор админ или што и да посакаме

const userSchema = new mongoose.Schema({

name: {

type: String,

required: [true, "Name is required"],

},

email: {

type: String,

required: [true, "Email is required"],

lowercase: true, //site bukvi da se mali

unique: true, // normalno sekoj email da ni e unikaten

validate: [validator.isEmail, "Please provide a valid email"], // i za validacija kje e iskorstiime bibliotekata

},

role: {

type: String,

enum: ["user", "administrator"],

default: "user",

},

password: {

type: String,

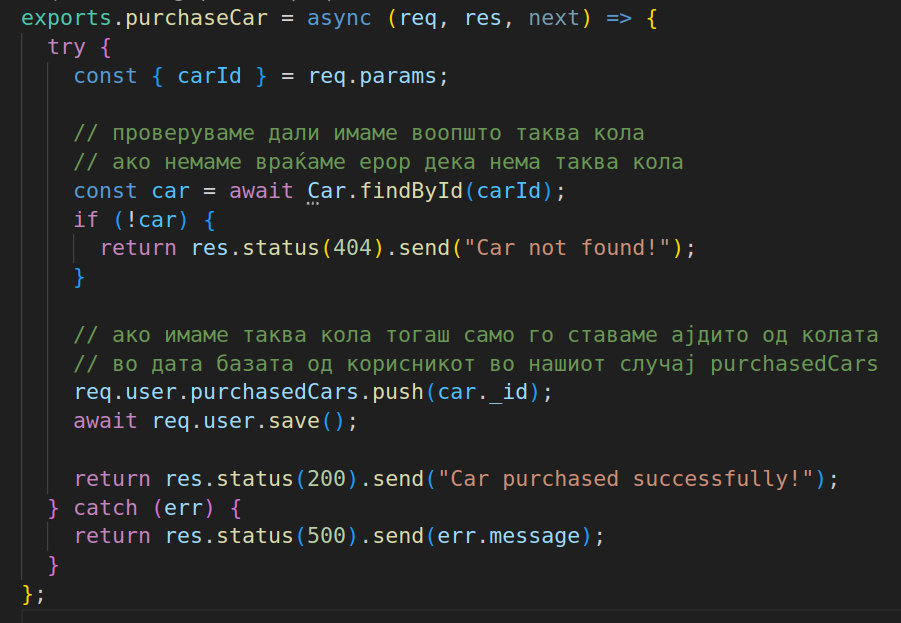
required: [true, "Password is required"],

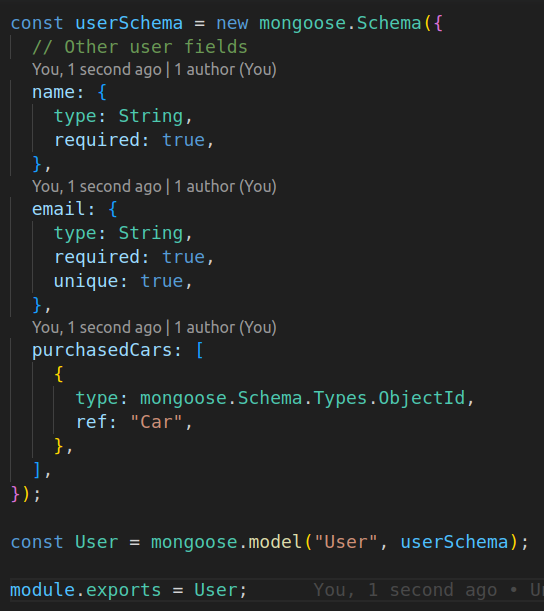
minlength: [8, "Password must be at least 8 characters long"],

// validate: [validator.isStrongPassword, "Please provide a stron password"]

});

**Имплементација на купување во случајов на Коли** (Multitenancy – со ова се кажува дека секој корисник ќе си има различен документ во нашиот случај ќе има различни купено коли од друг корисник)   
  
За купување кола ќе употребиме reference ради тоа што со embeding пристапот може да доведе до вишок на податоци и потенцијални проблеми со синхронизацијата. Ако очекувате информациите за автомобилот често да се менуваат, генерално се препорачува да се складира само референцата



Како шти гледаме req.user го имаме од протект миделварот значи што рековме ако прикачиме нешто на req сите наредни миделвери имаат пристап до модифицираниот риквест, што значи дека со оваа имплементација само логирани корисници можат да купат кола, значи  
  
а пак моделот на корисникот треба да биде вака  


и би имале рута со гет за да ги рендерираме колите во ejs, одкога ќе ги рендерираме колите во ejs со forEach ќе ги прикажиме и со форма со акција на рута “/cars/purchase/<%= car.\_id %>” со метода пост ќе може да купиме кола

и еве пример како би изгледала формата во ejs

