TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

Veebirakenduse loomine keemiliste elementide ja reaktiivide vastuvõtmiseks ja tarbimiseks,

Creating a web application to receive and consume chemical elements and reagents

ARUKAD SÜSTEEMID JA RAKENDUSINFOTEHNOLOOGIA ÕPPEKAVA
LÕPUTÖÖ

Üliõpilane: Kirill Botškov
Üliõpilaskood: 193094EDTR
Juhendaja:Natalja Ivleva lektor

Kohtla-Järve, 2024

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.

Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

"...." ..... 20.....

Autor: .....
/ allkiri /

Töö vastab rakenduskõrgharidusõppe lõputööle/magistritööle esitatud nõuetele
"...." ..... 20.....

Juhendaja: .....
/ allkiri /

Kaitsmisele lubatud
"...." ..... 20.....

Kaitsmiskomisjoni esimees .....
/ nimi ja allkiri /

LIHTLITSENTS LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS JA REPRODUTSEERIMISEKS

Mina Kirill Botškov (sünnikuupäev: 17.11.1998)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Veebirakenduse loomine keemiliste elementide ja reaktiivide vastuvõtmiseks ja tarbimiseks, mille juhendaja on Natalja Ivleva,

1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

TalTech Inseneriteaduskond Virumaa kolledž

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Kirill Botškov, EDTR71

Õppekava, peaeriala: EDTR17/18, Arukad süsteemid ja rakendusinfotehnoloogia

Juhendaja(d): Lektor, Natalja Ivleva, natalja.ivleva@taltech.ee

Lõputöö teema:

(eesti keeles) *Veebirakenduse loomine keemiliste elementide ja reaktiivide vastuvõtmiseks ja tarbimiseks*

(inglise keeles) Creating a web application to receive and consume chemical elements or reagents

Lõputöö põhieesmärgid:

1. Luua veebirakenduse keemiliste elementide hankimiseks ja tarbimiseks
2. Teha andmekaitse vahend

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Node js ja React js tehnoloogia õppimine	15.09.2023
2.	Projekti serveriosa (Backend) loomine, milles realiseeritakse ühendus andmebaasiga, samuti erinevad päringud andmebaasile, näiteks uue kasutaja registreerimise päring.	28.09.2023
3.	Projekti kliendiosa (Frontend) loomine, milles rakendatakse kasutajaliides ja funktsioonid, mis esitavad serverile päringuid, kui kasutajad kasutavad veebilehe erinevaid funktsioone.	20.10.2023
4.	Tutvumine Json Web Tokeniga	30.10.2023
5.	Json Web Token'i lisamine projektile	16.11.2023
6.	Kasutajaliidese täiustamine erinevate visuaalsete komponentide abil.	30.11.2023

Töö keel: eesti keel **Lõputöö esitamise tähtaeg:** "....."..... 20.....a

Üliõpilane: Kirill Botškov "....."..... 20.....a
/allkiri/

Juhendaja: Natalja Ivleva "....."..... 20.....a
/allkiri/

Programmijuht: Žanna Gratšjova "....."..... 20.....a
/allkiri/

SISUKORD

EESSÕNA 8

LÜHENDITE JA TÄHISTE LOETELU 9

SISSEJUHATUS 10

1. TEHNOLOOGIA VALIK 11

1.1. Andmebaasi valimine 11
1.2. Serveri tehnoloogia valik 11
1.3. Klienditehnoloogia valik 12
2. ANDMEBAAS 13

2.1 Andmebaasi struktuur 13
2.1.1. Tabel Users 13
2.1.2. Tabel Substance 14
2.1.3. Tabel SubstanceHistory 15
2.1.4. Tabel ChemicalElementsSpend 16
2.1.5. Substancedata vaade 17
2.1.6. Substancedeletedata vaade 17
3. SERVERI OSA 18

3.1. Serveri osa struktuur 18
3.2. Kasutatud moodulid serveriosas 18
3.3. Töötamine andmebaasiga 19
3.3.1. Andmebaasi ühendus 19
3.3.2. Serverist andmebaasi tehtavate päringute kirjeldus 19
3.3.3. JWT autentimine 20
3.4. Serveriosa ja kliendiosa vaheline teabevahetus 22
3.4.1. HTTP andmeedastusmeetodid 22
4. KLIENDI OSA 24

4.1. Kliendiosa struktuur 24
4.2. Kasutatud moodulid kliendiosas 24
4.3. Kasutaja registreerimine ja autoriseerimine 25
4.4. Veebilehe avaleht 28
4.4.1. Lisatud kemikaalide tabel 28
4.4.2. Tarbimise ajaloo tabel 31
4.4.3. Tabel tarbitud kemikaalide kohta 31
4.5. Veebilehtede ühendamine 32
5. ARENGUKAVAD 33

KOKKUVÕTTE.....34
SUMMARY.....35
KASUTATUD KIRJANDUSE LOETELU 36
LISA 1. TAGASISIDE VEEBIRAKENDUSE KOHTA KEEMIAINSENERIDELT 38

EESSÕNA

Lõputöö on tellitud TalTech Virumaa Kolledži poolt. Lõputöö eesmärgiks on luua veebirakendus sissetulevate ja väljaminevate keemiliste elementide, ainete jms. arvestuse pidamiseks.

Lõputöö teema pakkusid välja arukad süsteemide ja rakendusinfotehnoloogia lektor Natalja Ivleva, kes on ka juhendaja ning keemiainsenerid Moonika Ferschel ja Anna Mesilane.

Soovin öelda suured tänud oma juhendajale abi eest selle töö valmimisel.

Võtmesõnad: Backend, Frontend, Node.js, React, PostgreSQL, bakalaureusetöö.

LÜHENDITE JA TÄHISTE LOETELU

Bootstrap	tööriist kiireks veebilehe kujundamiseks
PostgreSQL	andmebaas
Node.js	JavaScripti käivituskeskkond võrgurakenduste loomiseks
Backend	serveriosa
Frontend	kliendiosa
JWT	avatud standard, mida kasutatakse teabevahetuseks kliendi ja veebirakenduste serveri vahel. See andmevahetus toimub hashed tokeni abil. (ingl k *JSON Web Token*, JWT)
HTTP	hüperteksti edastusprotokoll (ingl k *HyperText Transfer Protocol*, HTTP)
SQL	struktureeritud päringukeel (ingl k *Structured Query Language*, SQL)
React	JavaScripti käivituskeskkond kasutajaliideste loomiseks
HTML	hüperteksti märgistuskeel (ingl k *HyperText Markup Language*, HTML)

SISSEJUHATUS

TalTech Virumaa kolledžis peavad keemialabori töötajad keemiliste ainete sisestamise ja tarbimise üle arvestust käsitsi, st nad kirjutavad kõik paberile. Seetõttu tehti ettepanek luua neile veebirakendus, mis lihtsustab arvestuse pidamist ja võimaldab laboritöötajatel vabaneda paberkandjal arvestusest.

Veebirakendus pakub mitmeid võimalusi, näiteks: luua kasutajakonto, lisada teavet sissetuleva keemilise aine kohta, mis salvestatakse andmebaasis ja kuvatakse kasutajaliideses tabeli kujul, samuti on võimalik muuta keemilise aine kirjet, muuta keemilise aine kogust, mille muutused salvestatakse tarbimisajaloos eraldi tabeli vahekaardil, kustutada kasutatud keemilist ainet, mille andmed salvestatakse kasutajaliideses eraldi tabelis.

Lõputöös võrreldi erinevaid keskkondi andmebaasi loomiseks, luues serveripoolt (Backend) ja kliendipoolt (Frontend).

Peamised ülesanded, mis on püstitatud lõputöö täitmiseks:

1. Luua andmebaas PostgreSQL-is, et salvestada kõik andmed, mida kasutaja saab veebirakendusest kätte. Andmebaas ise on salvestatud kolledži serverisse.

2. Serveriosa (Backend) loomine, sealhulgas ühenduse loomine kolledži serverisse salvestatud andmebaasiga, samuti andmebaasi päringute loomine, näiteks kasutajate registreerimine ja autoriseerimine.

3. Kliendiosa (Frontend) loomine, sealhulgas kasutajaliidese loomine, samuti serverile erinevate toimingute tegemiseks esitatavate taotluste loomine.

1. TEHNOLOOGIA VALIK

Selles peatükis tutvustatakse erinevaid veebirakenduse loomiseks valitud tehnoloogiaid. Veebirakenduse võib jagada kolmeks peamiseks osaks andmebaas, kus andmeid hoitakse, ning serveriosa (Backend) ja kliendiosa (Frontend).

1.1. Andmebaasi valimine

Selles alapeatükis käsitletakse autori poolt veebirakenduse jaoks valitud andmebaasi. Võrreldakse PostgreSQL ja MYSQL, mis põhinevad SQL relatsioonilisel andmebaasi haldussüsteemil.

PostgreSQL on võimas avatud lähtekoodiga objekt-relatsiooniline andmebaasisüsteem, mis kasutab ja laiendab SQL-keelt koos paljude funktsioonidega, mis võimaldavad turvaliselt salvestada ja skaleerida kõige keerukamaid andmetöötluskoormusi. [1]

MySQL on Oracle'i poolt välja töötatud avatud lähtekoodiga relatsiooniline andmebaasi haldussüsteem, mis põhineb struktureeritud päringukeelel. Seda kasutatakse paljudes tuntud rakendustes, nagu Facebook, Twitter, Netflix, Uber, Airbnb, Shopify ja Booking.com. [2]

Lõpuks valis autor neist kahest andmebaasist PostgreSQLi, sest selles andmebaasis oli juba olemas kogemus tänu kolledži erinevate andmebaaside õppimise kursusele ning kolledži soovitustele.

PostgreSQL andmebaasis töötamiseks kasutati programmi pgAdmin. pgAdmin on programm, mis on kõige populaarsem ja multifunktsionaalsem platvorm PostgreSQL andmebaasi arendamiseks. [3]

1.2. Serveri tehnoloogia valik

Maailmas on palju serverite tehnoloogiaid veebirakenduste loomiseks, millest kõige kuulsamad on Node.js, Django, millest on juttu järgmises alapeatükis.

Node.js on avatud lähtekoodiga keskkond JavaScripti keeles kirjutatud serveripoolsete ja võrgurakenduste arendamiseks, mis sobib ideaalselt andmemahukate rakenduste jaoks, sest kasutab sündmusepõhist asünkroonset mudelit. [4]

Django on Pythoni keelel põhinev serveripoolne platvorm, mis uhkeldab sellega, et võtab enda peale suurema osa veebiarendusest, võimaldades teil keskenduda oma veebirakenduse arendamisele seal, kus seda vaja on. [5]

Lõppkokkuvõttes valis autor Node.js keskkonna tänu sellele, et tal oli selles keskkonnas juba kogemusi tänu veebirakenduste arendamise kolledži kursusele.

1.3. Klienditehnoloogia valik

Nagu serveripoolseid tehnoloogiaid, on ka kliendipoolseid tehnoloogiaid palju. Kõige tuntumad on React ja Angular, mida võrreldakse järgmises alapeatükis.

React - võimaldab luua kasutajaliideseid eraldi osadest, mida nimetatakse komponentideks, mis on JavaScripti funktsioonid. [6]

Angular on platvorm ja raamistik ühe lehekülje kliendirakenduste loomiseks HTML-i ja TypeScripti abil. Angular on kirjutatud TypeScriptis. [7]

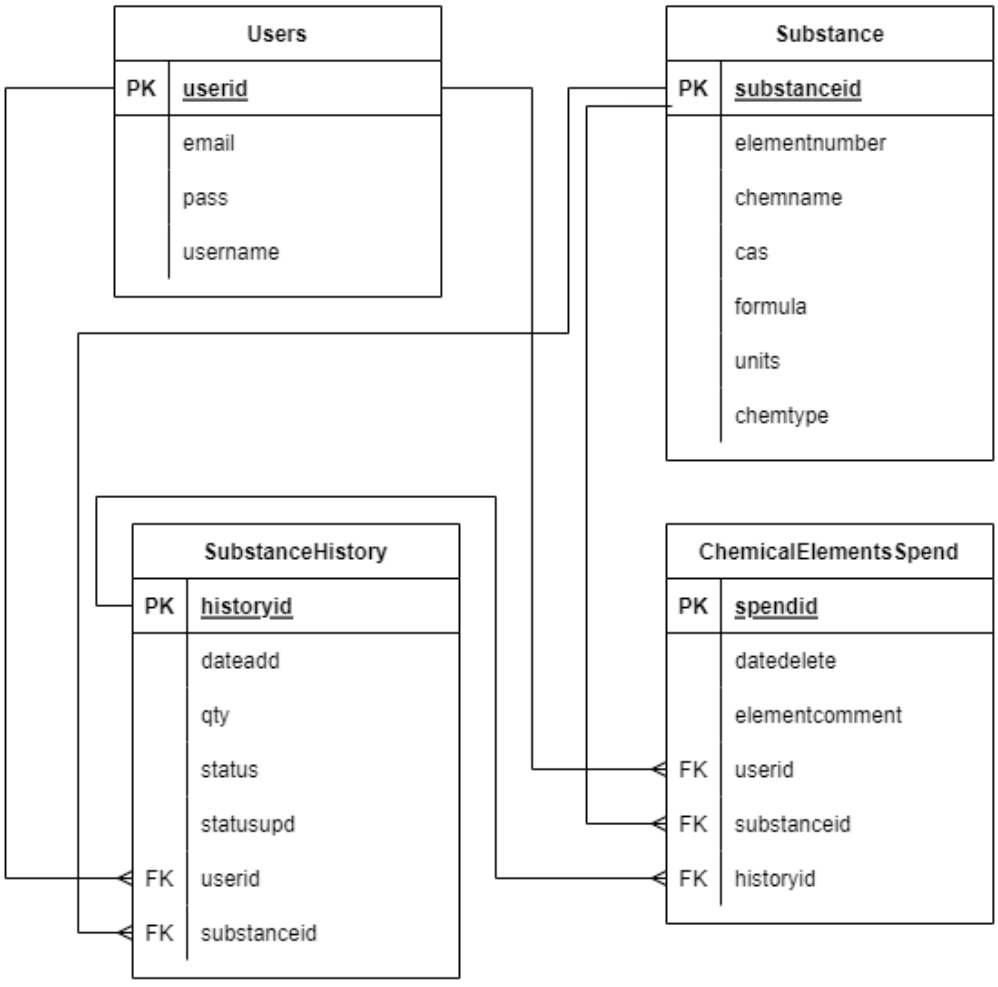
Lõpuks valis autor Reacti keskkonna, sest tal oli selles keskkonnas juba kogemusi tänu veebirakenduste arendamise kolledžikursusele ja JavaScripti programmeerimiskeele tundmisele.

2. ANDMEBAAS

Selles peatükis ja selle alapeatükkides käsitletakse veebirakenduse andmebaasi loomist, selle struktuuri ning tabelite ja vaadete kirjeldusi.

2.1 Andmebaasi struktuur

Veebirakenduse andmebaas koosneb neljast tabelist ja kahest vaatest. Tabelites hoitakse kogu veebirakenduse toimimiseks vajalikku teavet ning järgnevates alapeatükkides käsitletakse tabeleid ja vaateid (Joonis 2.1).



Joonis 2.1 ERD diagramm

2.1.1. Tabel Users

Seda tabelit kasutatakse kasutajaandmete salvestamiseks. Selle tabeli kaudu toimub kasutaja autoriseerimine, kontrollides, kas selline kasutaja on andmebaasis olemas, ning uue kasutaja registreerimine (Joonis 2.2).

Tabel koosneb neljast väljast (Tabel 2.1):

Tabel 2.1 Tabeli Users väljade kirjeldus

Põllu tüüp	Nimetus	Andmetüüp	Kirjeldus
PK	userID	serial	See on selle tabeli esmane võti
	email	varchar	Väli salvestab kasutaja postiandmeid ja sellel on UNIQUE piirang, mis muudab välja ainulaadseks
	pass	varchar	Välja salvestatakse parooli andmed
	username	varchar	Väljale salvestatakse andmed kasutaja nime kohta

```
CREATE TABLE Users (
    userID serial PRIMARY KEY,
    email varchar NOT NULL UNIQUE,
    pass varchar NOT NULL,
    username varchar NOT NULL
);
```

Joonis 2.2 Tabel Users

2.1.2. Tabel Substance

See tabel on vajalik keemilise elemendi või aine andmete säilitamiseks ja sisaldab põhiteavet kasutaja poolt veebirakenduse kaudu lisatud keemilise elemendi või aine kohta (Joonis 2.3).

See tabel koosneb kaheksast väljast (Tabel 2.2):

Tabel 2.2 Tabeli väljade kirjeldus Substance

Põllu tüüp	Nimetus	Andmetüüp	Kirjeldus
PK	substanceID	serial	See on selle tabeli esmane võti
	elementnumber	int	Väli salvestab selle elemendi numbri, mille all see andmebaasi lisati, ja sellel on UNIQUE piirang, mis muudab välja ainulaadseks
	chemname	varchar	Väli sisaldab keemilise elemendi nime
	casnumber	varchar	Väli sisaldab keemilise elemendi CAS-numbrit

	cas	varchar	Väli sisaldab veebiviidet keemilise elemendi CAS-numbrile
	formula	varchar	Väljal salvestatakse keemilise elemendi valemiga seotud nimi
	units	varchar	Väljal salvestatakse andmed kemikaali mõõtühikute kohta
	chemtype	varchar	Väli sisaldab andmeid keemilise aine liigi kohta

```
CREATE TABLE Substance (
    substanceID serial PRIMARY KEY,
    elementnumber int NOT NULL UNIQUE,
    chemname varchar NOT NULL,
    casnumber varchar NOT NULL,
    cas varchar NOT NULL,
    formula varchar NOT NULL,
    units varchar NOT NULL,
    chemtype varchar NOT NULL
);
```

Joonis 2.3 Tabel Substance

2.1.3. Tabel SubstanceHistory

Selles tabelis hoitakse andmeid kemikaali tarbimise kohta (Joonis 2.4).

See tabel koosneb seitsmest väljast (Tabel 2.3):

Tabel 2.3 Tabeli väljade kirjeldus SubstanceHistory

Põllu tüüp	**Nimetus**	**Andmetüüp**	**Kirjeldus**
PK	historyID	serial	See on selle tabeli esmane võti
	dateadd	varchar	Väljale salvestatakse andmed keemiliste elementide lisamise/kustutamise aja kohta andmebaasis
	qty	int	Väli sisaldab teavet kemikaali koguse kohta
	status	boolean	Väljal on kaks olekut TRUE ja FALSE ning seda on vaja elemendi kustutamisel. Algseisund TRUE

	statusupd	boolean	Väljal on kaks olekut TRUE ja FALSE ning seda on vaja kemikaali tarbimisajaloo koostamiseks. Algseisund TRUE
FK	userID	int	On välisvõti ja seostab seda tabelit tabeliga Users
FK	substanceID	int	On samuti välisvõti ja seob selle tabeli Substance tabeliga.

```
CREATE TABLE SubstanceHistory (
    historyID serial PRIMARY KEY,
    dateadd varchar NOT NULL,
    qty int NOT NULL,
    status boolean DEFAULT '1',
    statusupd boolean DEFAULT '1',
    userID int,
    FOREIGN KEY (userID) REFERENCES Users (userID),
    substanceID int,
    FOREIGN KEY (substanceID) REFERENCES Substance (substanceID)
);
```

Joonis 2.4 Tabel AddChemicalElements

2.1.4. Tabel ChemicalElementsSpend

Selles tabelis on andmed kasutatud kemikaalide kohta (Joonis 2.5).

Tabel koosneb kuuest väljast (Tabel 2.4):

Tabel 2.4 Tabeli väljade kirjeldus ChemicalElementsSpend

Põllu tüüp	**Nimetus**	**Andmetüüp**	**Kirjeldus**
PK	spendID	serial	See on selle tabeli esmane võti
	datedelete	varchar	Väljale salvestatakse kemikaali eemaldamise kuupäev
	elementcomment	varchar	Väljale salvestatakse teave kemikaali eemaldamise põhjuse kohta
FK	userID	int	On välisvõti ja seostab seda tabelit tabeliga Users
FK	substanceID	int	On samuti välisvõti ja seob selle tabeli Substance tabeliga

FK	historyID	int	On samuti võõrvõti ja seob seda tabelit tabeliga SubstanceHistory

```
CREATE TABLE ChemicalElementsSpend(
    spendID serial PRIMARY KEY,
    datedelete varchar NOT NULL,
    elementcomment varchar NOT NULL,
    userID int,
    FOREIGN KEY (userID) REFERENCES Users (userID),
    substanceID int,
    FOREIGN KEY (substanceID) REFERENCES Substance (substanceID),
    historyID int,
    FOREIGN KEY (historyID) REFERENCES SubstanceHistory (historyID)
);
```

Joonis 2.5 Tabel ChemicalElementsSpend

2.1.5. Substancedata vaade

See vaade ühendab teavet kolmest tabelist Substance, SubstanceHistory ja Users ning kuvab kasutajale andmeid lisatud kemikaali kohta (Joonis 2.6).

```
CREATE VIEW public.substancedata as
SELECT users.userid, substance.substanceid, substancehistory.historyid, substance.elementnumber,
substance.chemname, substance.formula,substance.chemtype, substancehistory.qty,substance.units,
substance.casnumber, substance.cas, substancehistory.dateadd, users.username,substancehistory.status,
substancehistory.statusupd FROM SubstanceHistory
INNER JOIN substance ON substancehistory.substanceid = substance.substanceid
INNER JOIN users ON substancehistory.userid = users.userid
WHERE substancehistory.status ='1'
```

Joonis 2.6 Substancedata vaade

2.1.6. Substancedeletedata vaade

See vaade ühendab kõigi nelja tabeli Substance, AddChemicalElements, Users ja ChemicalElementsSpend andmed, et näidata kasutajale teavet tarbitud kemikaalide kohta (Joonis 2.7).

```
CREATE VIEW public.substancedeletedata as
SELECT users.userid, substance.substanceid,substancehistory.historyid,chemicalelementsspend.spendid,
substance.elementnumber,substance.chemname,chemicalelementsspend.datedelete,chemicalelementsspend.elementcomment,
users.username,substancehistory.status FROM substancehistory
INNER JOIN substance ON substancehistory.substanceid = substance.substanceid
INNER JOIN users ON substancehistory.userid = users.userid
INNER JOIN chemicalelementsspend ON chemicalelementsspend.historyid = substancehistory.historyid
WHERE substancehistory.status ='0'
```

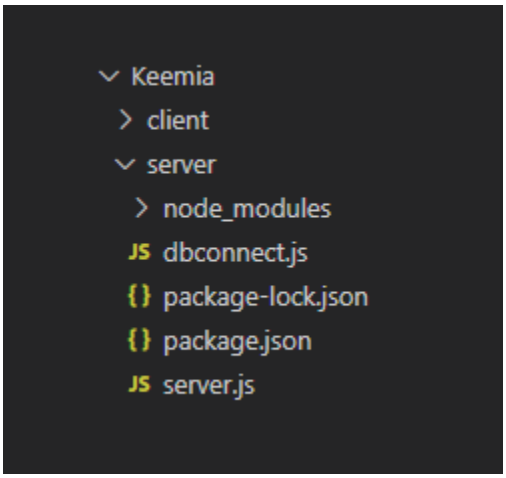
Joonis 2.7 Substancedeletedata vaade

3. SERVERI OSA

Selles peatükis ja selle alapeatükkides räägime veebirakenduse serveripoole (Backend) loomisest, mis on veebirakenduse kasutaja eest varjatud täidis, mis suhtleb nii andmebaasi kui ka veebirakenduse kliendipoolega (Frontend), samuti serveripoole struktuurist, JWT autentimisest, koodi kirjeldusest ja kaasatud moodulitest.

3.1. Serveri osa struktuur

Selles alapeatükis käsitletakse serveriosa struktuuri, mis koosneb järgmistest failidest ja kaustadest: dbconnect.js ja server.js failid, kaks json-faili nagu package.json ja package-lock.json, mis kujutavad endast projekti seadete kogumikku, ning kaust node-modules, kus hoitakse paigaldatud lisamooduleid. (Joonis 3.1).



Joonis 3.1 Serveri poolne kataloog

3.2. Kasutatud moodulid serveriosas

Serveriosa korrektseks toimimiseks on vaja paigaldada mõned kasulikud moodulid nagu express, cors, body-parser (Joonis 3.2).

- Express on Node.js-i raamistik, mis rakendab veebirakenduste loomiseks vajalikke funktsioone. Sellel on ka valmis funktsioonid HTTP-päringute käsitlemiseks ja iga HTTP-meetodi jaoks on olemas funktsioon. [8]
- Cors on HTTP-pealkirjadel põhinev mehhanism, mis võimaldab serveril või API-l määrata mis tahes muud allikad kui allikas, millest tundmatu allikas saab loa ressurssidele juurdepääsuks ja nende allalaadimiseks. [9]
- Body-parser on moodul, mida kasutatakse HTTP päringu keha (req.body) kaudu saadetud andmete töötlemiseks. [10]

```
const express = require('express')
const cors = require('cors')
const bodyParser = require('body-parser')
```

Joonis 3.2 Mooduli andmete väljakutsumine

3.3. Töötamine andmebaasiga

Fail dbconnect.js ühendub andmebaasiga ja loob päringuid andmebaasile, et teha erinevaid toiminguid, mida kasutaja teeb kliendiosas.

3.3.1. Andmebaasi ühendus

Kõigepealt tuleb paigaldada pg-moodul. See moodul sisaldab kogumikku node.js-i mooduleid PostgreSQL-andmebaasiga töötamiseks. Pärast seda saate kasutada Connection Pool'i, mis sisaldab teavet serveriga ühenduse kohta. (Joonis 3.3) [11]

```
const Pool = require('pg').Pool

const pool = new Pool({
    user: 't193094',
    host: 'dev.vk.edu.ee',
    database: 'KeemiaProject2',
    password:          ,
    port: 5432,
});
```

Joonis 3.3 Andmebaasi ühendus

3.3.2. Serverist andmebaasi tehtavate päringute kirjeldus

Kõik veebirakenduses tehtavad kasutaja toimingud käivad läbi serveri poole ja siin toimuvad sellised toimingud nagu kasutaja autoriseerimine ja registreerimine, uue kemikaali lisamine.

Näide kolmest järgmisest taotlusest:

- Uue kasutaja registreerimine. Selles andmebaasi päringus registreeritakse uus kasutaja INSERT SQL-käsu abil (Joonis 3.4).

```
const UserRegistration = async (req, res) => {
    const { emailReg, passReg, usernameReg } = req.body;
    try {
        await pool.query("INSERT INTO users (email,pass,username) VALUES ($1,$2,$3)",
            [emailReg, passReg, usernameReg]);
    } catch (err) {
        res.json(err.message);
    }
};
```

Joonis 3.4 Uue kasutaja registreerimise taotlus

- Lisatud keemiliste ainete toodang. See andmebaasi päring väljastab andmed aineandmete vaates, mida kasutaja näeb pealehel pärast autoriseerimist (Joonis 3.5).

```
const GetAllElements = async (req, res) => {
    try {
        const response = await pool.query("SELECT * FROM substancedata ORDER BY elementnumber ASC");
        res.json(response.rows)
    } catch (err) {
        res.json(err.message);
    }
};
```

Joonis 3.5 Andmete väljastamise taotlus lisatud kemikaalide kohta

- Uue kemikaali lisamine. See päring lisab andmebaasi uue kemikaali, kasutades INSERT SQL käsku (Joonis 3.6).

```
const AddNewElement = async (req, res) => {
    try {
        const { elementnumAdd, elementnameAdd, casnumberAdd, casAdd, formulaAdd, unitsAdd, typeAdd,
            qtyAdd, useridAdd } = req.body;
        await pool.query("INSERT INTO substance VALUES (default,$1,$2,$3,$4,$5,$6,$7)",
            [elementnumAdd, elementnameAdd, casnumberAdd, casAdd, formulaAdd, unitsAdd, typeAdd]);
        await pool.query("INSERT INTO substancehistory(historyid,dateadd,qty,userid,substanceid)" +
            "VALUES (default,TO_CHAR(NOW()::date, 'DD/MM/YY'),$1,$2,(SELECT max(substanceid) FROM substance))",
            [qtyAdd, useridAdd]);
    } catch (err) {
        res.json(err.message);
    }
};
```

Joonis 3.6 Uue kemikaali lisamise taotlus

3.3.3. JWT autentimine

Selles veebirakenduses kasutatakse kasutaja autentimiseks ja andmekaitseks JSON Web Tokenit või lühendatult JWT-d. Selle toimimiseks kasutatakse jsonwebtoken-moodulit.

JWT on avatud standard, mida kasutatakse teabe vahetamiseks kliendi ja veebirakenduste serveri vahel. See andmevahetus toimub hashed tokeni kaudu. Token ise on teave, mida saab kergesti kontrollida ja mis võib olla juhuslik tähtnumbriliste märkide kogum. [12]

JWT koosneb kolmest osast: päis, kasulikud andmed ja allkiri:

- Pealkiri - koosneb tavaliselt kahest osast: JWT-tokendi tüübist ja kasutatud allkirjaalgoritmist.
- Kasulikud andmed - see on teine osa, mis koosneb avaldustest või objektidest, näiteks kasutajaandmetest.
- Allkiri - viimane osa on krüptograafilise algoritmi abil loodud string, mida saab kasutada kasutatava teabe terviklikkuse kontrollimiseks. [12]

Veebirakenduse serveris on päring, mis autoriseerib kasutaja, kontrollides, kas kasutaja on andmebaasis olemas, ja kui kasutaja on leitud, luuakse sellele kasutajale unikaalne sümbol (Joonis 3.7).

```
const UserAuthentication = async (req, res) => {
    try {
        const { emailLog, passLog } = req.body;
        const response = await pool.query("SELECT userid, username, email FROM users WHERE email = $1 AND pass = $2",
            [emailLog, passLog]);
        const result = response.rows;
        if (result.length != 0) {
            jwt.sign({ result }, jwtaccesskey, (err, accesstoken) => {
                if (err) {
                    res.json("Something went wrong")
                }
                res.send({ result, access: accesstoken });
            });
        }
        else {
            res.send(result)
        }
    } catch (err) {
        res.json(err.message);
    }
};
```

Joonis 3.7 Kasutaja autoriseerimistaotlus

Veebirakenduse serveris on ka funktsioon, mis kontrollib, kas kasutajal on märgis, sest ilma märgiseta ei saa kasutaja veebirakenduses andmeid vaadata ega mingeid toiminguid teha (Joonis 3.8).

```
function verityToken(req, resp, next) {
    let token = req.headers['authorization'];
    if (token) {
        token = token.split(' ')[1];
        jwt.verify(token, jwtaccesskey, (err) => {
            if (err) {
                resp.status(401).send({ result: 'Please provide valid token' })
            } else {
                next();
            }
        })
    } else {
        resp.status(401).send({ result: 'Please provide token' })
    }
}
```

Joonis 3.8 Tokeni kontrollimise funktsioon

3.4. Serveriosa ja kliendiosa vaheline teabevahetus

Server.js failis toimub serveri ja kliendi suhtlus tänu HTTP andmeedastusmeetoditele, mis omakorda täidavad teatud funktsioone, mis on loodud dbconnect.js failis.

3.4.1. HTTP andmeedastusmeetodid

Expressi moodulil on valmis meetodid HTTP-päringute töötlemiseks. Hypertext Transfer Protocol ehk lühendatult HTTP töötab kliendi ja serveri vahelise taotlus-vastusprotokolliga. Kõige sagedamini kasutatavad meetodid on GET, POST, PUT ja DELETE. [13]

- GET - kasutatakse andmete taotlemiseks määratud ressursilt. [13]
- POST - kasutatakse andmete saatmiseks serverile ressursi loomiseks/uuendamiseks. [13]
- PUT - kasutatakse andmete saatmiseks serverile ressursi loomiseks/uuendamiseks. Erinevus POST ja PUT vahel seisneb selles, et sama PUT päringu korduv kutsumine annab alati sama tulemuse, samas kui POST päringu korduval kutsumisel on kõrvalmõjud sama ressursi korduva loomise näol. [13]
- DELETE - kustutab määratud ressursi. [13]

Taotluse app.METHOD(PATH, HANDLER) marsruut ise on järgmise struktuuriga:

- app on ekspressi näide. [14]
- METHOD - HTTP päringumeetod. [14]
- PATH on URL-tee serveris. [14]

- HANDLER - funktsioon, mis täidetakse marsruudi sobitamise korral. [14]

Käesolevas projektis on kasutatud järgmisi meetodeid, millest mõned on näited:

- See POST päring teostab uue kasutaja registreerimise, kutsudes registreerimisfunktsiooni dbconnect.js failist (Joonis 3.9).

```
// http://localhost:5000/keemia/user/registration
app.post('/keemia/user/registration', db.UserRegistration);
```

Joonis 3.9 POST taotlus registreerimiseks

- See GET-päring täidab lisatud kemikaalide andmete väljastamisega seotud toimingu, kutsudes dbconnect.js-failist keemilise väljundi funktsiooni ja JWT-tokeni valideerimisfunktsiooni (Joonis 3.10).

```
// http://localhost:5000/keemia/get/all/elements
app.get('/keemia/get/all/elements',db.verityToken, db.GetAllElements);
```

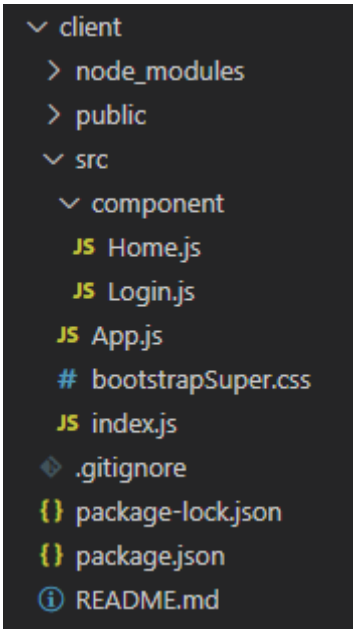
Joonis 3.10 Kemikaalide väljundi GET päring

4. KLIENDI OSA

Selles peatükis ja selle alapeatükkides räägime veebirakenduse kliendipoolest (Frontend), mis on kasutajaliides, ning räägime ka kliendipoole struktuurist, kasutatavatest moodulitest ja koodikirjeldustest.

4.1. Kliendiosa struktuur

Selles alapeatükis käsitletakse kliendiosa struktuurist, mis koosneb järgmistest kaustadest ja failidest: package.json ja package-lock.json failid, mis vastutavad projekti seadete eest, public ja src kaustad, mis on peamised, kus kogu kliendiloogika rakendatakse, ja node-modules kaust, kus hoitakse installeeritud lisaraamatukogusid. Enamik neist failidest paigaldatakse käsuga npx create-react-app, millega luuakse põhiprojekt (Joonis 4.1).



Joonis 4.1 Kliendiosa struktuur

4.2. Kasutatud moodulid kliendiosas

Lisaks Reactile kasutas autor kliendiosa loomisel kasutajaliidese rakendamiseks täiendavaid mooduleid (Joonis 4.2).

- react-router-dom - see pakett võimaldab rakendada dünaamilist marsruutimist veebirakenduses. See võimaldab teil kuvada lehekülgi ja võimaldada kasutajatel nendes navigeerida. [15]
- react-toastify on üks parimaid pakette pop-up-teavituste lisamiseks React'ile. [16]

- react-bootstrap-table-next - see pakett võimaldab luua mitmeotstarbelisi tabeleid Bootstrapi abil. [17]
- react-bootstrap-table2-paginator - see pakett täiendab react-bootstrap-table-next paketti ja võimaldab jagada tabelit lehekülgedeks. [18]
- react-bootstrap-table2-filter - see pakett täiendab paketti react-bootstrap-table-next ja lisab võimaluse veergude filtreerimiseks. [19]
- react-bootstrap-table2-toolkit - see pakett täiendab react-bootstrap-table-next paketti ja võimaldab lisada tabeliotsingu ja muud. [20]

```
import React, { Fragment, useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import BootstrapTable from "react-bootstrap-table-next";
import paginationFactory from "react-bootstrap-table2-paginator";
import filterFactory, { selectFilter } from 'react-bootstrap-table2-filter';
import ToolkitProvider, { Search } from 'react-bootstrap-table2-toolkit/dist/react-bootstrap-table2-toolkit';
```

Joonis 4.2 Mooduli andmete väljakutsumine

Samuti, mida veebirakendus saab teha ilma ilusa kujunduseta. Seetõttu kasutati selles veebirakenduses valmis Bootstrap tööriistakomplekti. Bootstrap on raamistik kohanduva veebilehe kujunduse kiireks loomiseks. [21]

4.3. Kasutaja registreerimine ja autoriseerimine

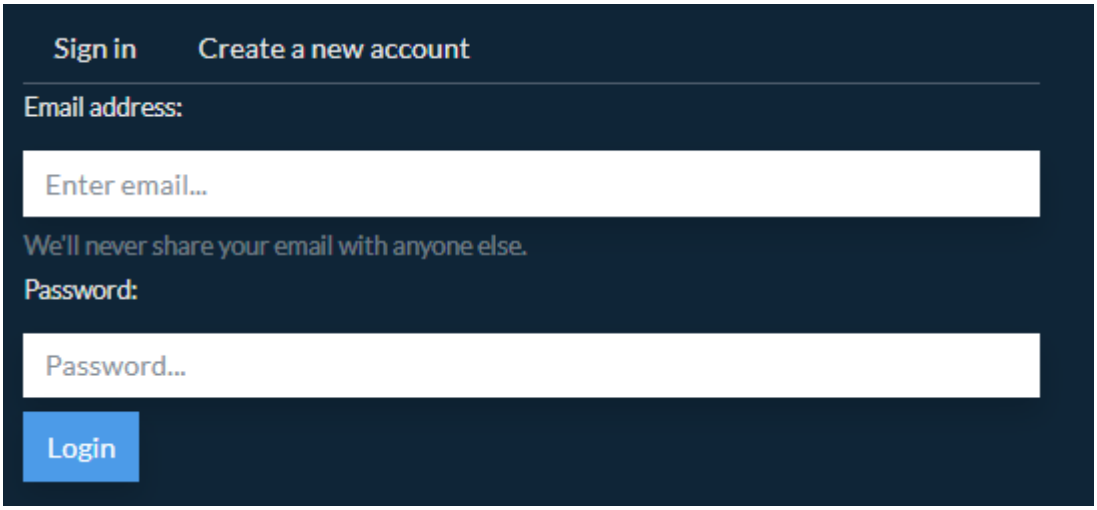
Fail login.js rakendab kasutaja autoriseerimise ja registreerimise kasutajaliidese, mis on eraldi veebileht.

Kõik toimingud, mida kasutaja teeb kliendiosas, läbivad serveriosa, selleks on rakendatud funktsioonid, et sooritada mõningaid toiminguid kliendilt serverile.

Vaatleme näide uue kasutaja autoriseerimisest ja registreerimisest:

Kasutaja autoriseerimine kliendiosas võib jagada kaheks etapiks.

- Esimene samm on kasutajaliides, mida kasutaja näeb, kui ta külastab veebilehte (Joonis 4.3). Kasutajaliides luuakse HTML-keeles (Joonis 4.4). HTML on hüperteksti märgistuskeel veebirakenduste loomiseks. [22]



Joonis 4.3 Andmete sisestamise väljad autoriseerimiseks

```
<div className="tab-pane fade active show" id="Authorization">
  <form>
    <div className="form-group">
      <label className="form-label">Email address:</label>
      <input type="email" className="form-control my-1 shadow" aria-describedby="emailHelp"
        placeholder="Enter email..." value={emailLog} onChange={e => setEmailLog(e.target.value)} />
      <div>
        {errorLog && !emailLog && <small className="text-danger">The field is empty</small>}
      </div>
      <small className="form-text text-muted">We'll never share your email with anyone else.</small>
    </div>
    <div className="form-group">
      <label className="form-label">Password:</label>
      <input type="password" className="form-control my-1 shadow" placeholder="Password..."
        value={passLog} onChange={e => setPassLog(e.target.value)} />
      {errorLog && !passLog && <small className="text-danger">The field is empty</small>}
    </div>
    <button className="btn btn-primary my-1 shadow" onClick={UserAuthentication}>Login</button>
  </form>
</div>
```

Joonis 4.4 Kasutaja autoriseerimisvälja HTML-koodi fragment

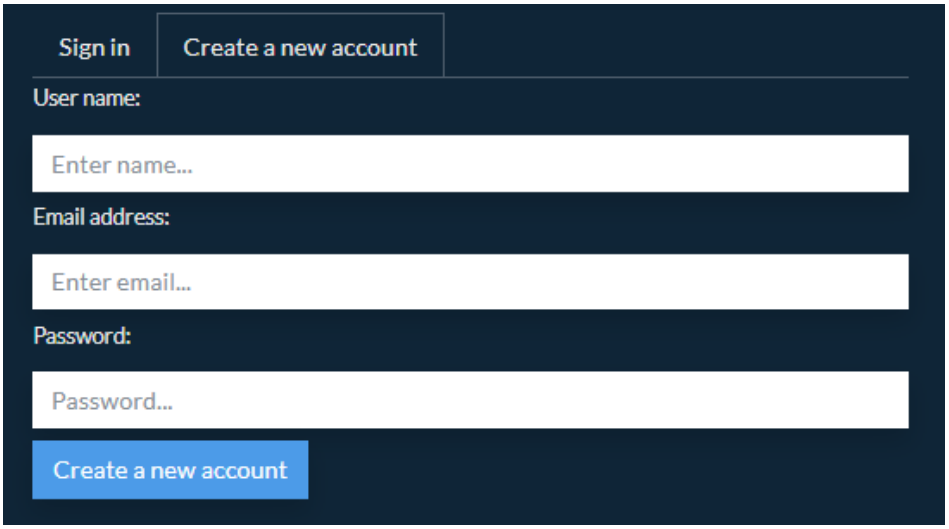
- Teine samm on autoriseerimistaotluse eest vastutava funktsiooni täitmine serveri poolel (Joonis 4.5). Näiteks, nagu joonisel (Joonis 4.3) on näidatud, on olemas nupp "Login", mis tänu Reacti sündmusele onClick käivitab funktsiooni, mis autoriseerib kasutaja, samuti veeväljale sisestatud andmed.

```
const UserAuthentication = async e => {
  e.preventDefault();
  try {
    const logbody = { emailLog, passLog };
    let response1 = await fetch(`http://localhost:5000/keemia/user/authentication`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(logbody)
    });
    response1 = await response1.json()
    const result = response1.result;

    if (emailLog.length === 0 || passLog.length === 0) {
      setErrorLog(true);
      return (false)
    } else if (result === undefined) {
      LoginErrorMessage();
    }
    else if (response1.access) {
      localStorage.setItem("user", JSON.stringify(result[0]))
      localStorage.setItem("token", JSON.stringify(response1.access))
      navigate('/home');
    }

  } catch (err) {
    console.error(err.message);
  }
}
```

Joonis 4.5 Funktsioon kasutaja autoriseerimiseks

Uue kasutaja registreerimise võib samuti jagada kaheks etapiks.

- Esimene samm on andmete sisestamine kasutajaliidese sisendväljale kogu registreerimisprotsess on sarnane autoriseerimisega (Joonis 4.6).



Joonis 4.6 Andmete sisestamise väljad registreerimiseks

- Teine samm on käivitada funktsioon, mis vastutab uue kasutaja registreerimise taotlemise eest serveri poolel (Joonis 4.7).

```
const UserRegistration = async e => {
  e.preventDefault();
  try {
    const regbody = { emailReg, passReg, usernameReg };
    const checkbody = { emailReg };

    let res2 = await fetch(`http://localhost:5000/keemia/checkemail`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(checkbody)
    });
    res2 = await res2.json()

    if (emailReg.length === 0 || passReg.length === 0 || usernameReg.length === 0) {
      setErrorReg(true);
      return (false)
    }
    else {
      if (res2.length > 0) {
        AddressErrorMessage();
      }
      else if (res2.length === 0) {
        RegisterSuccessMessage();
        setEmailReg('');
        setPassReg('');
        setUsernameReg('');
        await fetch(`http://localhost:5000/keemia/user/registration`, {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify(regbody)
        });
      }
    }
  } catch (err) {
    console.error(err.message);
  }
};
```

Joonis 4.7 Funktsioon kasutaja registreerimiseks

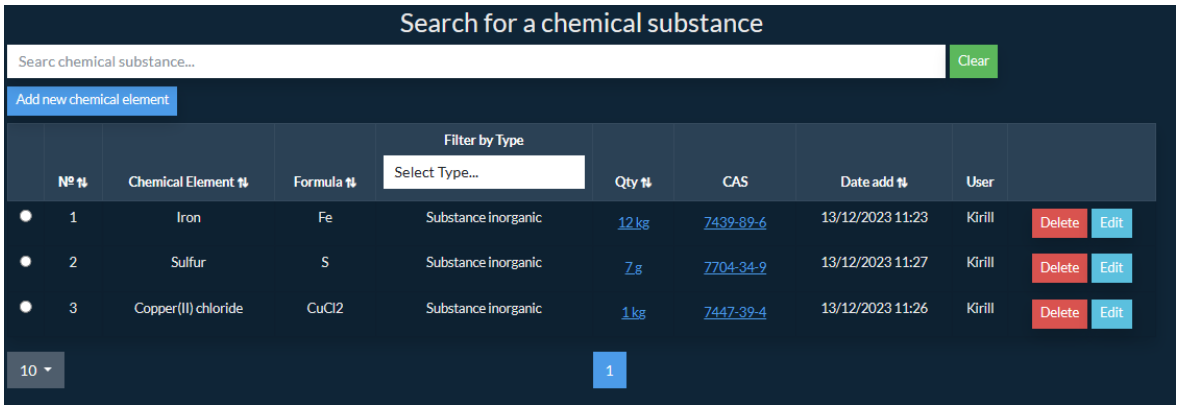
4.4. Veebilehe avaleht

See home.js fail vastutab kodulehe eest, kuhu kasutaja jõuab pärast autoriseerimist. Seal toimub kogu veebirakenduse peamine funktsionaalsus. Lehekülg on jagatud kolmeks vahekaardiks: tabel lisatud kemikaalidega, tarbimise ajalugu ja tabelid tarbitud kemikaalidega.

4.4.1. Lisatud kemikaalide tabel

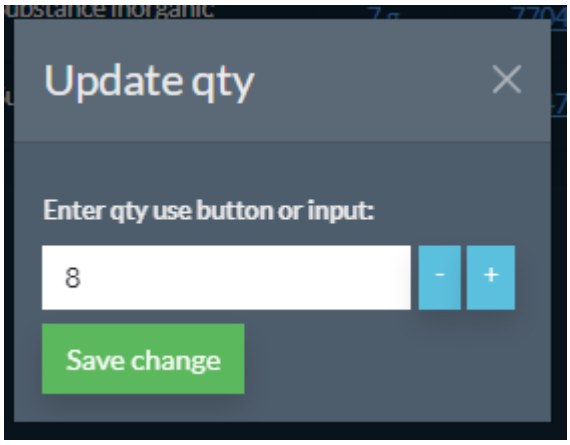
See on vahekaart, mida kasutaja näeb pärast autoriseerimist. Sellel näete tabelit keemiliste ainete ja selles olevate lisafunktsioonide kohta, aineotsingu rida ja nuppu uue keemilise aine lisamiseks. Lugege selle vahekaardi funktsionaalsuse kohta lähemalt:

1. Tabelis on suur hulk funktsioone, näiteks andmete sorteerimine ja filtreerimine, kemikaali koguse muutmine, CAS-numbri lingi jälgimine, andmete redigeerimine reas, kemikaali kustutamine ja lehekülgedel navigeerimine (Joonis 4.8).



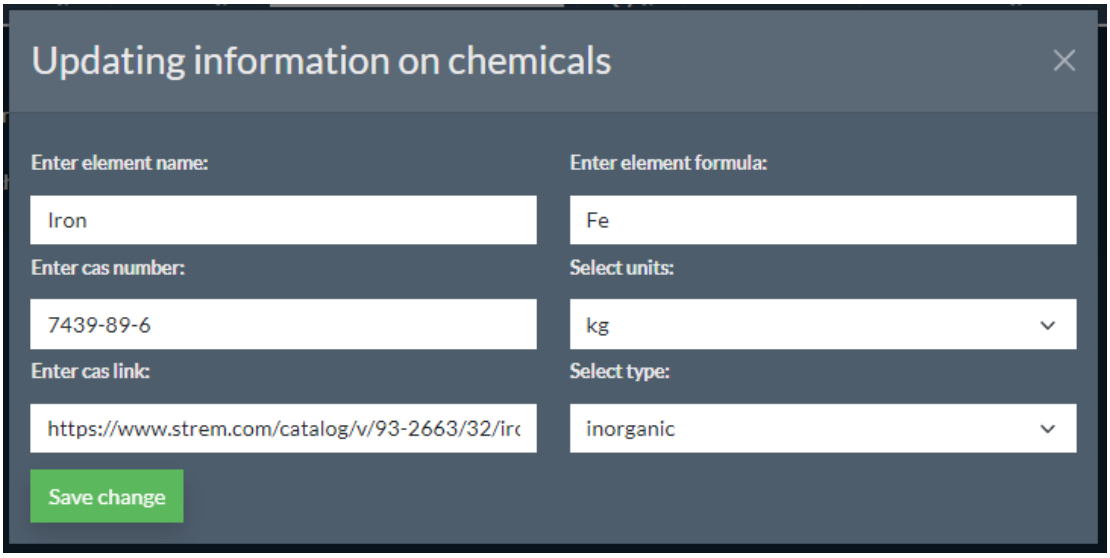
Joonis 4.8 Lisatud kemikaalide tabel

- Muutused aine koguses avatakse eraldi hüpikaknas (Joonis 4.9).



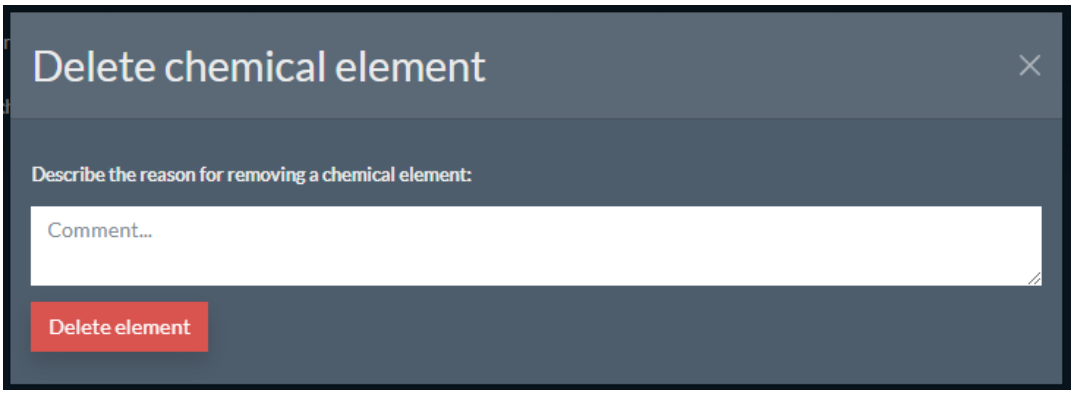
Joonis 4.9 Koguse muutmise hüpikaken

- Ridade redigeerimine avaneb ka eraldi hüpikaknas (joonis 4.10).



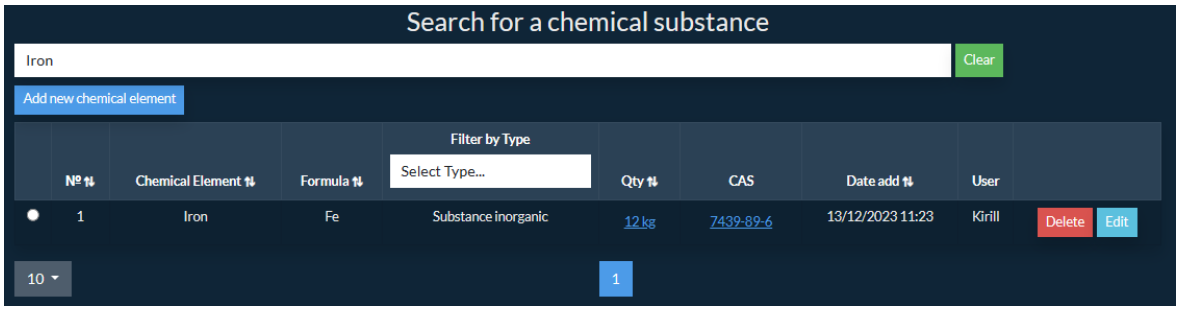
Joonis 4.10 Avanev aken keemilise teabe muutmiseks

- Ka kustutamine avaneb eraldi hüpikaknas (Joonis 4.11).



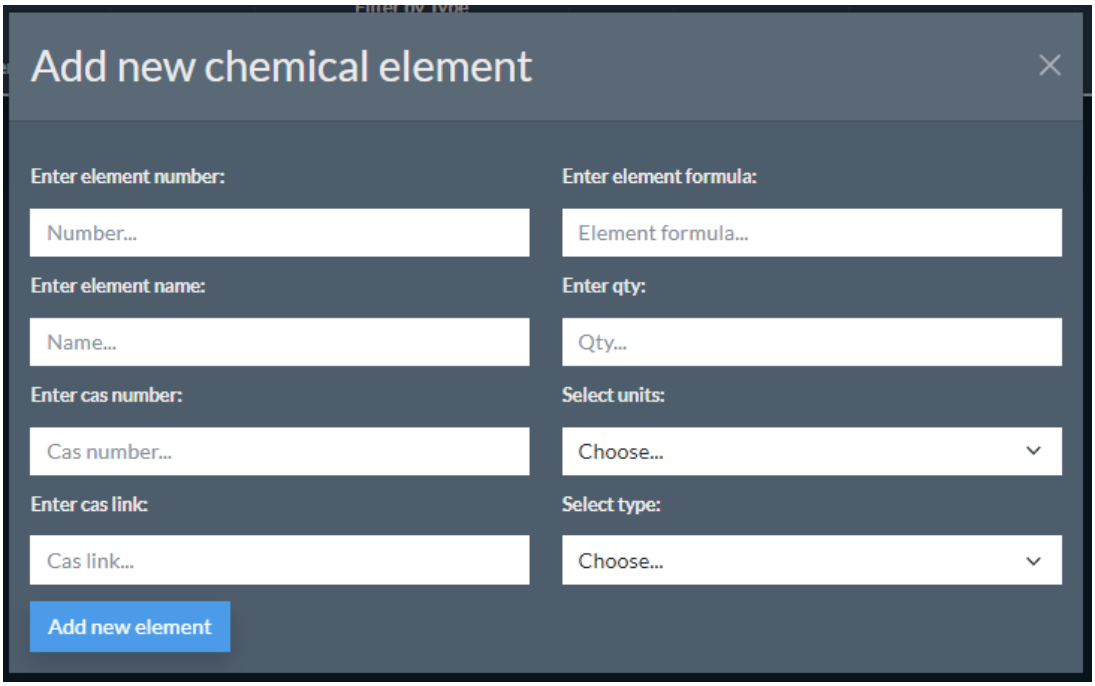
Joonis 4.11 Avanev aken kemikaali eemaldamiseks tabelist

2. Otsinguriba võimaldab otsida keemilist elementi selle nime, valemi ja kasutajanime järgi, millega see lisati või redigeeriti (joonis 4.12).



Joonis 4.12 Keemilise elemendi otsinguriba

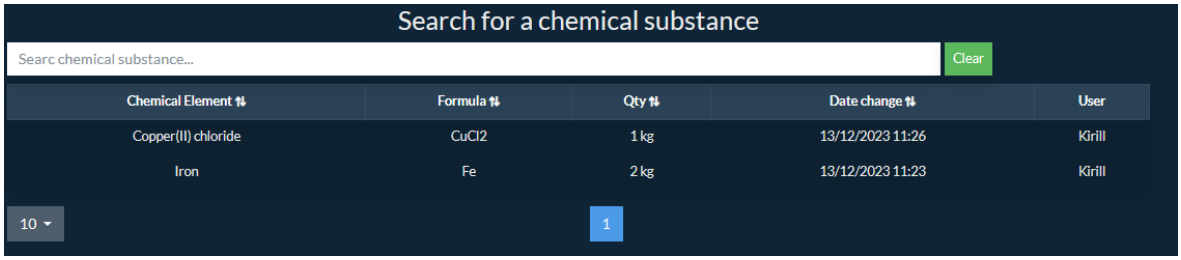
3. Nupp "uue kemikaali lisamine" võimaldab kasutajal lisada uue kemikaali eraldi hüpikaknas, kuhu lisatakse automaatselt kasutaja nimi ja lisamise kuupäev (Joonis 4.13).



Joonis 4.13 Kemikaali hüpikakna lisamine

4.4.2. Tarbimise ajaloo tabel

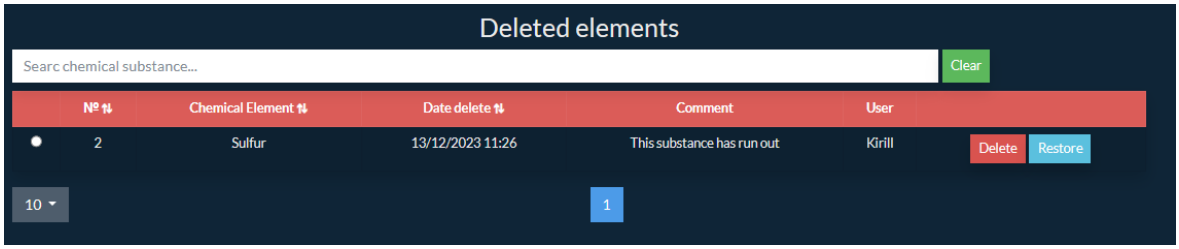
Sellel vahekaardil on tabelis kirjas tarbitud kemikaali kogus, selle sisend ja väljund ning võimaldab otsida tabelist kemikaali nime, valemi või kemikaali kogust muutnud kasutaja nime järgi (Joonis 4.14).



Joonis 4.14 Tabel, mis näitab tarbimise ajalugu

4.4.3. Tabel tarbitud kemikaalide kohta

Sellel vahekaardil kuvatakse tabelis andmed kasutatud kemikaalide kohta, mis on mingil põhjusel siia paigutatud. Siin saab kasutaja otsida kemikaali selle nime ja kustutanud kasutaja nime järgi. Samuti on võimalik taastada kemikaali, kui see on kogemata kustutatud, ja kustutada kirje lõplikult, st see võimaldab aeg-ajalt andmeid puhastada (Joonis 4.15).



Joonis 4.15 Tabel, mis näitab eemaldatud keemilisi elemente.

4.5. Veebilehtede ühendamine

See fail ühendab kaks faili Login.js ja Home.js, et moodustada konteiner erinevate komponentide jaoks, antud juhul eraldi leht kasutaja autentimiseks ja registreerimiseks ning pealeht (Joonis 4.16).

```
import React, { Fragment } from "react";
import { Navigate, Outlet, BrowserRouter as Router, Routes, Route } from "react-router-dom";
import './bootstrapSuper.css';
import Login from "./component/Login"
import Home from "./component/Home"

const Private = () => {
  const auth = localStorage.getItem('user');
  return auth ? <Outlet/>:<Navigate to="/"/>
};

function App() {
  return (
    <Fragment>
      <Router>
        <Routes>
          <Route element={<Private/>}>
            <Route path="/home" element={<Home />} />
          </Route>
            <Route path="/" element={<Login />} />
        </Routes>
      </Router>
    </Fragment>
  );
}

export default App;
```

Joonis 4.16 Kahte faili ühendav kood

5. ARENGUKAVAD

Selles peatükis keskendutakse veebirakenduse edasise arendamise ja tulevase testimise plaanidele..

Selle töö tellisid Tallinna Tehnikaülikooli Virumaa kolledži keemiainsenerid eesmärgiga luua veebirakendus, mis võimaldab pidada arvestust erinevate kemikaalide sisse- ja väljaminekute kohta. Veebirakendusel on kaks veebilehte. Esimesel lehel saab kasutaja registreerida uue konto ja autoriseerida olemasoleva konto, teisel veebilehekülgel näeb kasutaja veebirakenduse põhifunktsioone: ta saab tabelis näha juba olemasolevaid keemilisi aineid, lisada uusi, muuta keemilise aine kogust, st sisestada selle sisse- ja väljavoolu ja näha seda kõike eraldi tabelis, samuti kustutada keemilise aine kirje ja näha selle kohta infot vastavas tabelis.

Hetkel saab veebirakendust käivitada ainult lokaalselt, st seadmes, millel veebirakenduse kood loodi, ning plaanis on panna veebirakendus oma domeenile.

Samuti plaanime luua rollid kasutajatele nagu admin ja moderaator, kellel on rohkem võimalusi veebirakendusega manipuleerimiseks, ning tavakasutaja, kes saab kasutada ainult veebirakenduse tavapäraseid funktsioone.

Testimist ei ole veel toimunud keemialaboratooriumi töötajatega, kuid juba veebirakenduse esialgsete demonstratsioonide põhjal on neil ideid uute funktsioonide lisamiseks (vt Lisa 1). Kuid nagu iga muu veebirakenduse puhul, on alati uusi ideid, mida tuleb veel täiustada ja täiustada, kuid praegu täidab veebirakendus kõik põhiülesanded.

KOKKUVÕTTE

Lõputöö teema on "Veebirakenduse loomine keemiliste elementide ja reaktiivide vastuvõtmiseks ja tarbimiseks".

Käesoleva lõputöö eesmärgiks on luua veebirakendus Tallinna Tehnikaülikooli Virumaa kolledži keemialabori töötajatele kemikaalide sisse- ja väljaminekute jälgimiseks. Enne veebirakenduse loomise ettepanekut pidasid keemialabori töötajad arvestust paberkandjal.

Veebirakenduse loomisel õppis autor erinevaid tehnoloogiaid veebirakenduse loomiseks, lõi Backend serveripoole ja Frontend kliendipoole kasutades JWT autentimist.

Lõputöö võib jagada kolmeks peamiseks osaks. Esimeses osas kirjeldatakse andmebaasi loomist ja tutvustatakse erinevaid andmebaase võrreldes neid ning antakse ülevaade veebirakenduse jaoks valitud andmebaasist ja tutvustatakse ka selle struktuuri. Teises osas kirjeldatakse Backend-serveri poole loomist ja JWT-autentimist ning tutvustatakse erinevaid Backend-serveri poolseid tehnoloogiaid, võrreldes neid ja andes ülevaate Backend-serveri poole jaoks valitud tehnoloogiast ning selle struktuurist, samuti esitatakse koodinäide. Kolmas osa vastutab Frontend-kliendipoole loomise eest ning tutvustab erinevaid Frontend-kliendipoolseid tehnoloogiaid ja nende struktuuri ning esitab koodinäite.

Veebirakenduse testimist keemialabori töötajatega ei ole veel tehtud, kuid tutvumiseks on esitatud veebirakendus ise. Edasise arenduse plaanis on testida veebirakendust keemialabori töötajatega, kuid mitte lõpliku töö raames, vaid selle rakenduse edasise arendamise käigus, mille käigus võidakse tuvastada erinevaid puudusi või soovida olemasolevat funktsionaalsust parandada või lisada midagi uut, sest nagu iga veebirakenduse puhul, on alati midagi, mida saab muuta, parandada või täiustada.

SUMMARY

The topic of the graduation thesis is "Creating a web application to receive and consume chemical elements and reagents".

The aim of this graduate thesis is to create a web application to record chemical incoming and outgoing chemical substances for the employees of the chemistry laboratory of Virum College of Tallinn University of Technology. Before the proposal to create the web application, the employees of the chemistry laboratory kept records in paper form.

When creating the web application, the author familiarized himself with different technologies for creating a web application, created a Backend server side and Frontend client side using JWT authentication.

The graduate thesis can be divided into three main parts. The first part describes the creation of the database and introduces the different databases comparing them and gives an idea of the selected database for the web application and also introduces its structure. The second part describes the creation of the Backend server side and JWT authentication and introduces the different Backend server side technologies by comparing them and giving an idea of the selected technology for the Backend server side as well as its structure and presents a code example. The third part is responsible for creating the Frontend client side and introduces the different Frontend client side technologies as well as its structure and presents a code example.

Testing of the web application with the chemical laboratory staff has not been done yet, but the web application itself has been presented for familiarization. In the plans for further development to test the web application with the employees of the chemical laboratory, but not as part of the final work, but in the further development of this application, which may be identified various shortcomings or wishes to improve the existing functionality or add something new, because as with any web application there is always something that can be changed, improved or finalized.

KASUTATUD KIRJANDUSE LOETELU

1. Postgresql. What is PostgreSQL? [Online] https://www.postgresql.org/about/ (04.12.2023).
2. Oracle. What is MySQL? [Online] https://www.oracle.com/mysql/what-is-mysql/ (04.12.2023).
3. pgAdmin. pgAdmin [Online] https://www.pgadmin.org/ (04.12.2023).
4. Taha Sufiyan. What is Node.js? [Online] https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs (04.12.2023).
5. Djangoproject. Meet Django [Online] https://www.djangoproject.com/ (04.12.2023).
6. react. Create user interfaces from components [Online] https://react.dev/ (04.12.2023).
7. angular. Introduction to Angular concepts [Online] https://angular.io/guide/architecture (04.12.2023).
8. nodejsdev. Express [Online] https://nodejsdev.ru/guides/webdraftt/express/#express (04.12.2023).
9. bestprogrammer. Использование CORS в Node.js [Online] https://bestprogrammer.ru/izuchenie/ispolzovanie-cors-v-node-js (04.12.2023).
10. educative. What is Express body-parser? [Online] https://www.educative.io/answers/what-is-express-body-parser (04.12.2023).
11. node-postgres. Welcome [Online] https://node-postgres.com/ (04.12.2023).
12. Rishabh Poddar. What is a JWT? Understanding JSON Web Tokens 24.03.2022 [Online] https://supertokens.com/blog/what-is-jwt (04.12.2023).
13. w3schools. HTTP Request Methods [Online] https://www.w3schools.com/tags/ref_httpmethods.asp#:~:text=The%20PUT%20Method&text=The%20difference%20between%20POST%20and,the%20same%20resource%20multiple%20times. (04.12.2023).
14. expressjs. Основы маршрутизации [Online] https://expressjs.com/ru/starter/basic-routing.html (04.12.2023).
15. geeksforgeeks. What is react-router-dom ? [Online] https://www.geeksforgeeks.org/what-is-react-router-dom/ (04.12.2023).
16. npmjs. React-Toastify [Online] https://www.npmjs.com/package/react-toastify (04.12.2023).
17. react-bootstrap-table. react-bootstrap-table2 [Online] https://react-bootstrap-table.github.io/react-bootstrap-table2/docs/about.html (04.12.2023).
18. react-bootstrap-table. Pagination [Online] https://react-bootstrap-table.github.io/react-bootstrap-table2/docs/basic-pagination.html (04.12.2023).

19. react-bootstrap-table. Column Filter [Online] https://react-bootstrap-table.github.io/react-bootstrap-table2/docs/basic-filter.html (04.12.2023).
20. react-bootstrap-table. Getting Started [Online] https://react-bootstrap-table.github.io/react-bootstrap-table2/docs/toolkits-getting-started.html (04.12.2023).
21. Сергей Почекутов. Знакомство с Bootstrap: установка и подходящие сценарии использования 09.09.2020 [Online] https://timeweb.com/ru/community/articles/znakomstvo-s-bootstrap (04.12.2023).
22. w3schools. HTML Introduction [Online] https://www.w3schools.com/html/html_intro.asp (04.12.2023).

LISA 1. TAGASISIDE VEEBIRAKENDUSE KOHTA KEEMIAINSENERIDELT

Tudeng on koostanud suurepärase andmebaasi rakendusliku uurimisrühma reaktiivide loendamiseks. Andmebaasi on võimalik sisestada kõik vajalikud reaktiivid, numereerida need vastavalt vajadusele, kategoriseerida ühendi järgi, sisestada reaktiivi kogus ja lisada ohutuskaart. Lisaks kõigele võimaldab veebileht igale tööliikmele individuaalset ligipääsu ning kajastab kõik tehingud ajaloos. Antud andmebaas kergendab reaktiivide inventuuri ja otsingut, võimaldab loobuda vanadest Exceli ja paberkandjal tabelitest ning on kättesaadav igast arvutist. Koostatud andmebaas on igati abiks töörühma jaoks, vastab kõigile läbirääkimistel esitatud nõuetele ja kergendab uurimisrühma reaktiivide loendamist. Taolist andmebaasi võiks edaspidi laiendada ka seadmete ja tarvikute jaoks. Kindlasti oleks andmebaas kasulik ka TalTech Virumaa kolledži ja Põlevkivi Kompetentsikeskuse laborites.