TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Aivar Loopalu 164416IAPB

TÖÖLAUARAKENDUS KAPILLAARELEKTROFOREESI TEOSTAMISEKS

bakalaureusetöö

Juhendaja: Evelin Halling, PhD

Kaasjuhendaja: Jelena Gorbatšova, PhD

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Aivar Loopalu

20.05.2019

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua Tallinna Tehnikaülikooli Keemiainstituudis asuvale kapillaarelektroforeesi seadmele töölauarakendus, mis saaks kätte Arduino mikrokontrolleri peal töötavalt seadmelt signaali, kuvaks signaali reaalajas graafikul ja teised katseandmed töölauarakenduses ning katse lõppedes salvestaks graafikul olevad punktid tekstifaili, katse seadistused metadatana teise faili, graafiku pildifaili ning interneti olemasolul saadaks kogu katse ajal kogutud informatsiooni läbi REST API andmebaasi.

Töö tulemusena valmis Java programmeerimiskeeles kirjutatud töölauarakendus, mida on laborikatsete jaoks võimalik seadistada; mis saab Arduino mikrokontrolleri peal töötavalt kapillaarelektroforeesi seadmelt kätte signaalid analüütide ioonide elektrilise juhtivuse kohta, informatsiooni elektrivoolu kohta; kuvab kogu informatsiooni graafilises töölauarakenduses (analüütide signaalid reaalajas muutuval graafikul) ning katse lõppedes salvestab singaalide info ühte tekstifaili, katse seadistused teise tekstifaili, kuvatud graafiku pildifaili ning interneti olemasolul saadab kogu katse ajal kogutud informatsiooni JSON formaadis Spring Booti peale kirjutatud REST API-le. REST API omakorda töötleb saadud informatsiooni ning seejärel salvestab andmed vastavatesse tabelitesse PostgreSQL andmebaasis.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 7 peatükki, 17 joonist, 0 tabelit.

Abstract

Desktop Application For Performing Capillary Electrophoresis

The aim of current thesis is to make a desktop application for the instrument of capillary electrophoresis that resides in the Department of Chemistry in Tallinn University of Technology. The application should be able to receive a signal from the instrument that is built on Arduino circuit board and make the signal visible on live chart. Also all of the other information should be presented on the graphical interface. In the end of laboratory experiment all the information about the experiment should be saved into a text file, the signals should be saved to another text file and the chart itself in an image file. If the Internet is available, all the information, that is collected during the experiment, should be sent to the database through REST API.

By the end of this thesis a desktop application written in Java was made. This application can be modified for laboratory experiments; it receives signals about the conductivity of the ions of chemical analytes and the information about the experiment from the instrument that is built on Arduino circuit board; presents them on graphical interface (the signals of analytes on the real-time chart); and in the end of the experiment it saves all the data of signals into one text file, information about the experiment into another file and chart itself into an image file. If the Internet is available, all the information, that is collected during the experiment, will be sent in JSON format to REST API that is built on Spring Boot framework. REST API will process the information and then will save it into the tables of PostgreSQL database.

The thesis is in Estonian language and contains 28 pages of text, 7 chapters, 17 figures, 0 tables.

Lühendite ja mõistete sõnastik

ADC	*Analog Digital Converter*, analoog-digitaal muundur
analüüt	testitavas proovis olev aine (komponent)
API	*Application Programming Interface*, rakendusliides
BGE	*Background Electrolyte*, katsetes kasutatav taustelektrolüüt
CLI	*Command Line Interface*, käsurealiides
debugimine	koodist vea põhjuse otsimine ja eemaldamine
dll	*Dynamik Link Library*, Windowsi operatsioonisüsteemile mõeldud teegi faili laiend
FXML	*JavaScript Object Notation*, andmevahetusvorming
getter	Javas meetod, mis tagastab mingi väärtuse või objekti
ID	*Inner Diametre*, kapillaari sisediameeter
IDE	*Integrated Development Environment*, integreeritud arenduskeskkond
jar	*Java ARchive*, Java paketifaili laiend
JDK	*Java Development Kit*, Java arenduskomplekt
JRE	*Java Runtime Environment*, Java käivituskeskkond
JSON	*JavaScript Object Notation*, andmevahetusvorming
kapillaar	väga väikese läbimõõduga peenike toru
kHz	kilohertz, sageduse mõõtühik
kV	kilovolt, pinge mõõtühik
maatriks	testitava proovi domeen (näiteks "merevesi")
meetod	katsemeetod koos kõikide vajalike seadistustega
min	minut, aja mõõtühik
MHz	megahertz, sageduse mõõtühik
mmol	millimool, ainehulga mõõtühik
mol	mool, ainehulga mõõtühik
OD	*Outer Diametre*, kapillaari välisdiameeter
plugin	programmiosa, mida on võimalik programmile lisada

port	rakenduses kasutatava andmesideühenduse lõpp-punkt, mille kaudu toimub suhtlus
ppb	*Parts Per Billion*, 1 mikrogramm liitri kohta, ainehulga mõõtühik
ppm	*Parts Per Million*, 1 milligramm liitri kohta, ainehulga mõõtühik
REST	*Representational State Transfer*, tarkvara arhidektuuri stiil
sec	sekund, aja mõõtühik
string	Javas teatud pikkusega tekst
SQL	*Structured Query Language*, programmeerimises kasutatav keel, mis on vajalik andmebaasidega suhtlemisel
teek	funktsioonide, klasside ja moodulite kogu
U	pinge sümbol elektroonikas
µm	mikromeeter, pikkuse mõõtühik
µmol	mikromool, ainehulga mõõtühik

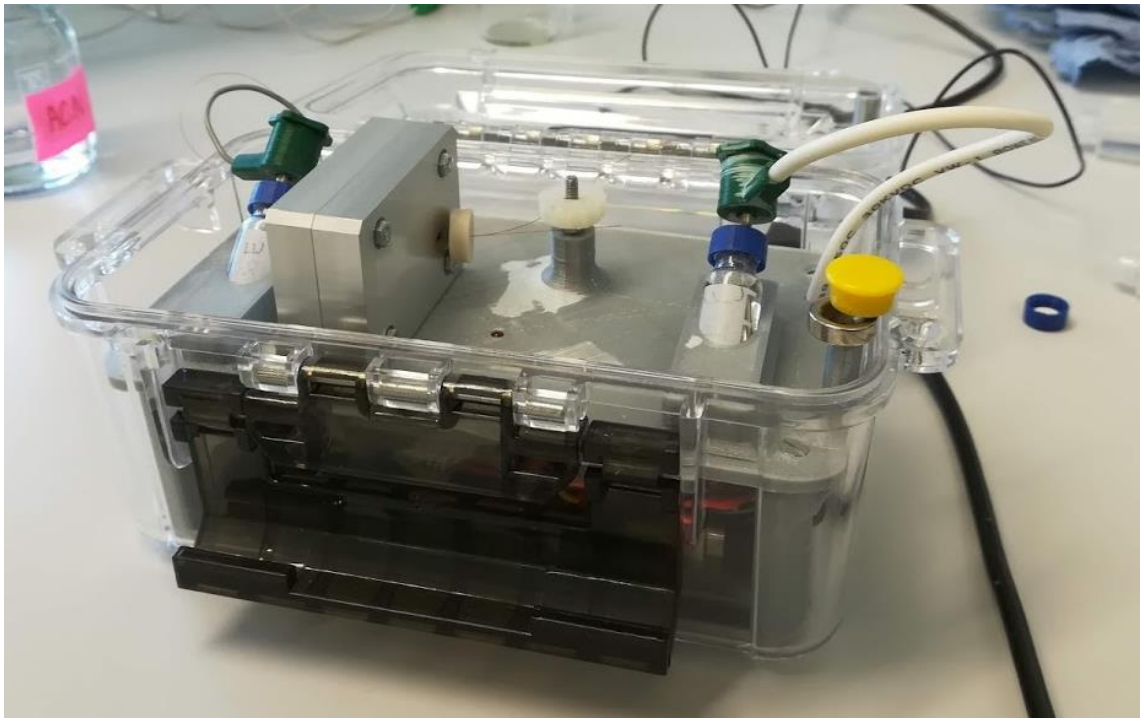
Sisukord

Jooniste loetelu

Tabelite loetelu

No table of figures entries found.

Sissejuhatus



Joonis 1. Kapillaarelektroforeesi seade

Analüütilises keemias kasutatakse erinevate keemiliste elementide ja keemiliste ühendite tuvastamiseks elektroforeesi. Käesolevas lõputöös keskendutakse elektroforeesi alammeetodile kapilaarsele elektroforeesile.

Elektroforees on analüütide lahutamise meetod, kus laetud osakesed liiguvad elektrijuhtivust omavas vedelas keskkonnas elektrivälja mõjul. Vastavalt osakeste laengule ja mõõtmetele liiguvad nad kapillaaris erineva kiirusega. Kapillaari lõpuosas asuvas detektoris registreeritakse komponentide kontsentratsioonile vastavad signaalid (piigid) ajalises järjestuses ning saadakse elektroferogramm [1].

Lõputöö eesmärgiks on luua Java baasil töölauarakendus, mis võtab Arduino mikrokontrolleri peale ehitatud kapillaarelektroforeesi seadmelt (Joonis 1) vastu signaali, töötleb seda ja kuvab reaalajas graafikule. Vastavalt töölauarakenduse seadistustele nii

signaal kui ka kuvatav graafik muutuvad. Laborikatse lõpus salvestatakse andmed kolme faili: metadata (töölauarakenduses seadistatud muutujad ning katse andmed) tekstifailis, graafik PNG pildifailis ning elektroforeesi seadmest kätte saadud signaalid tekstifailis. Samuti, võimalusel, saadetakse andmed edasi Spring Booti raamistiku peale ehitatud REST API-le, kust edasi andmebaasi.

Peatükis 2 kirjeldatakse kapilaarset elektroforeesi ja kapillaarelektroforeesi seadet.

Peatükis 3 esitatakse nõuded loodavale süsteemile.

Peatükis 4 kirjeldatakse olemasolevat töölauarakendust ning sõnastatakse lahti probleem, mida see lõputöö püüab lahendada.

Peatükis 5 kirjeldatakse rakenduses kasutatavat metoodikat, teeke ja raamistikke.

Peatükis 6 kirjeldatakse loodud süsteemi struktuuri.

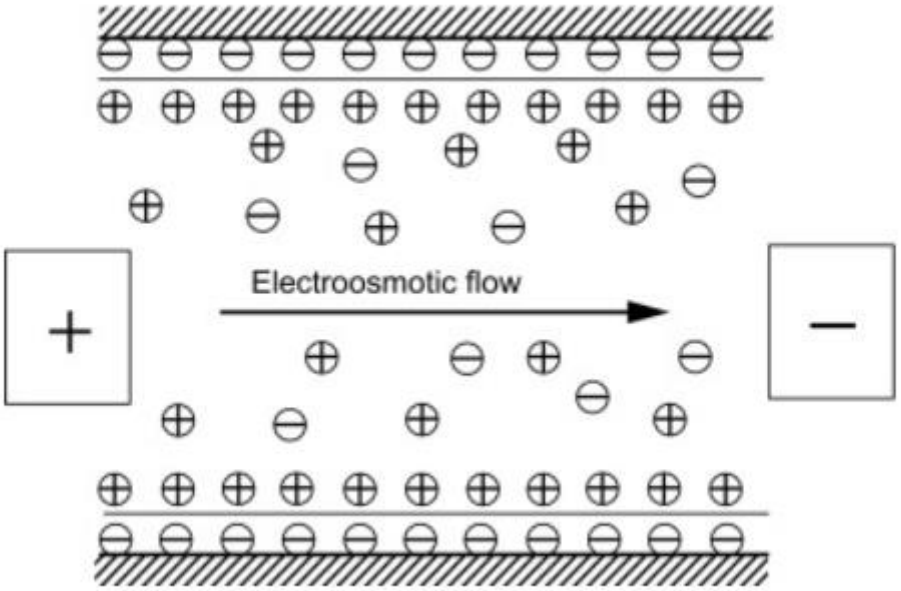
Peatükis 7 kirjeldatakse loodud süsteemi testimist.

1 Kapillaarelektroforees

1.1 Kapillaarelektroforeesi mõiste

Kapillaarelektroforees on analüütilises keemias kasutatav elektriliselt laetud (ioonsete) osakeste lahutamise meetod, mis töötati välja 1960. aastatel [1].

Elektroforees on analüütide lahutamise meetod, kus laetud osakesed liiguvad elektrijuhtivust omavas vedelas keskkonnas elektrivälja mõjul. Laengut kandva osakese liikumise kiirus on võrdeline elektrivälja tugevusega ja osakese ioonlaenguga ning pöördvõrdeline osakese raadiusega ja keskkonna viskoossusega [1].

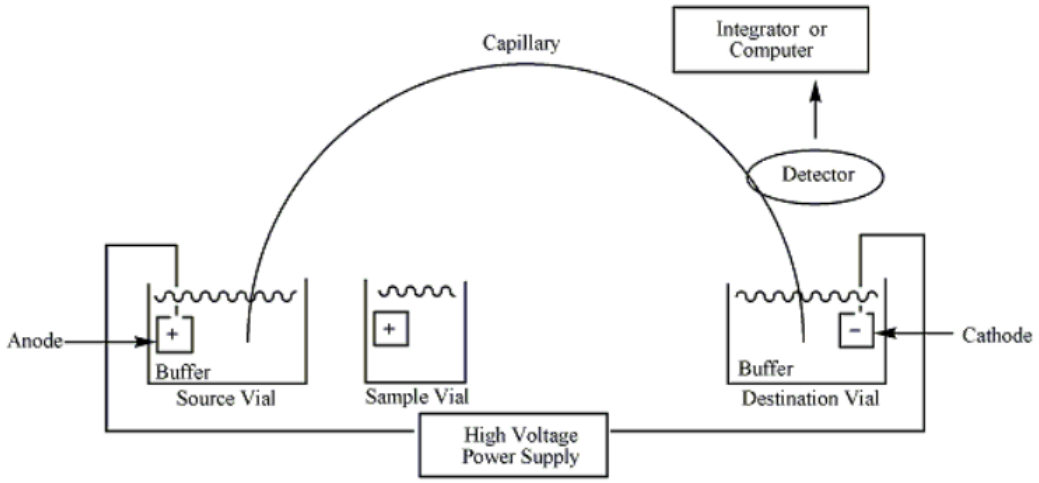


Joonis 2. Elektroosmootne voog kapillaaris [2]

Kapillaarelektroforeesi korral viiakse elektroforeesi protsess läbi 25–100-mikromeetrise sisediameetriga kvarts-, klaas- või polümeerses kapillaaris, mis on täidetud töölahusega (elektrolüüdiga) ja millele rakendatakse kõrgepinge 10–30 kV. Proovi sisestamise järel hakkavad selle komponendid elektriväljas elektroosmootse voo (Joonis 2) mõjul liikuma

erineva kiirusega vastavalt osakeste laengule ja mõõtmetele, mistõttu toimub nende lahutamine tsoonideks. Kapillaari lõpuosas olevas elektrijuhtivusdetektoris registreeritakse komponentide kontsentratsioonile vastavad signaalid (piigid) ajalises järjestuses ning saadakse elektroferogramm [1].

1.2 Kapillaarelektroforeesi seade



Joonis 3. Kapillaarelektroforeesi süsteem skemaatiliselt

Kapillaarelektroforeesi seade (Joonis 3) koosneb kõrgepinge allikast, mis on ühendatud elektrolüüdi lahust (puhverlahust) sisaldavate anumatega, millest ühes on anood ja teises katood. Neid anumaid ühendab sama elektrolüüdilahusega täidetud kapillaar, selle katoodipoolne osa läbib elektrijuhtivusdetektori, mille signaal saadetakse registreerimisseadmesse. Analüüsitava proovi kapillaari sisestamiseks ühendatakse selle anoodipoolne ots ja pingeallikas lühiajaliselt proovi sisaldavasse anumasse. Kõrgepinge toimel liiguvad kapillaaris molekulid ja ioonid katoodi poole, ehkki erinevate kiirustega. Kapillaari efektiivne pikkus, kus toimub analüütide lahutamine, on anoodist kuni detektorini [1].

2 Nõuded loodavale süsteemile

Töös saadakse kapillaarelektroforeesi seadmelt kätte signaal. See kuvatakse elektroferogrammina töölauarakenduses. Peale katset salvestatakse andmed failidesse ja saadetakse võimaluse ka andmebaasi.

Järgnevalt tuuakse välja nõuded süsteemi iga alamosa kohta. Need on kooskõlastatud süsteemi lõppkasutajaga konsulteerides.

2.1 Nõuded töölauarakendusele

2.1.1 Nõuded graafikule

- Graafik peab kujutama kapillaarelektroforeesi seadmelt saadud signaali
- Kas sama graafik või eraldi graafik (valikuliselt) peab kujutama seadmelt saadud voolutugevust
- Graafik peab muutuma ajas
- Graafiku skaala peab olema sõltuv hetkel nähtavast graafikuosast
- Erineva ajaskaala puhul peab graafik horisontaalselt olema kas välja venitatud või kokku surutud

2.1.2 Nõuded valikmenüüdele

- Valida peab saama töölauarakenduse kasutajat
- Menüüde funktsionaalsus peab sõltuma kasutajale antud volitustest (administraatoril on õigus kõike muuta, samas tavakasutaja saab valida ainult eelseadistatud valikute vahel)
- Menüü peab kuvama kõikide katses kasutatavate meetodite nimekirja
- Meetodite nimekirja peab saama lisada uut meetodit
- Menüü peab kuvama kõikide katses kasutavate maatriksite nimekirja

- Maatriksite nimekirja peab saama lisada uut maatriksit
- Menüü peab kuvama kõikide otsitavate analüütide valik-nimekirja
- Analüütide valik-nimekirja peab saama lisada uut analüüti
- Katses kasutavatele analüütidele peab saama lisata nende kontsentratsiooni vedelikus
- Menüü peab kuvama analüütide mõõtühikute nimekirja
- Menüü peab kuvama kõikide kasutavate BGE-de valik-nimekirja
- BGE-de valik-nimekirja peab saama lisada uut BGE-d
- Katses kasutavatele BGE-dele peab saama lisada nende kontsentratsiooni vedelikus
- Menüü peab kuvama BGE-de mõõtühikute nimekirja (mol, mmol, µmol, ppm, ppb)
- Menüü peab kuvama katses kasutavate kapillaaride ID-de ja OD-de nimekirja (25/150 µm, 25/350 µm, 50/150 µm, 50/350 µm, 75/175 µm, 75/350 µm, 100/175 µm, 100/350 µm)
- Menüü peab kuvama katses kasutavate kapillaaride täispikkuste nimekirja (20-75 cm, 5 cm pikkuste vahedega)
- Menüü peab kuvama katses kasutavate kapillaaride effektiivpikkuste nimekirja (10-60 cm, 5 cm pikkuste vahedega)
- Menüü peab kuvama vedeliku süstimismeetodite nimekirja (rõhk, vaakum, elekter)
- Menüü peab kuvama vedeliku süstimismeetodi võrdlusdimensiooni (erinevus, pinge)
- Menüüs peab saama sisestada vedeliku süstimismeetodi võrdlusdimensiooni väärtust

- Menüü peab kuvama vedeliku süstimismeetodi võrdlusdimensiooni väärtuse mõõtühikute nimekirja (cm/mbar, kV)

- Menüüs peab saama sisestada vedeliku süstimismeetodi kestvust (sekundites)

2.1.3 Nõuded nuppudele

- Valikus peavad olema elektrivoolu sagedust muuta võimaldavad nupud: 300 kHz, 400 kHz, 500 kHz, 660 kHz, 800 kHz, 880 kHz, 1 MHz, 1,3 MHz, 1,6 MHz, 2 MHz

- Valikus peavad olema ajaliselt vaadeldava graafiku osa muuta võimaldavad nupud: 30 sec, 1 min, 2 min, 3 min, 5 min, 10 min

- Töölauarakendusel peavad olema nupud katse alustamiseks, peatamiseks, graafiku puhastamiseks ning katseandmete ja graafiku salvestamiseks

- Töölauarakendusel peab olema nupp, mis lülitab kapillaarelektroforeesi seadmes kõrgepinge sisse/välja

- Sõltuvalt kõrgepinge olekule (sees/väljas) peab kõrgepinge lüliti värvus muutuma rohelise ja punase vahel

- Töölaual peab olema taimerit sisse/välja lülitav nupp

2.1.4 Üldised nõuded

- Töölauarakendusel peab olema tekstiväli, kuhu saab kirjutada katsega seotud kommentaare

- Töölauarakendus peab kuvama hetke voolutugevuse

- Töölauarakendus peab võimaldama saata kapillaarelektroforeesi seadmesse protsentuaalset kõrgepinge suurust

- Töölauarakendus peab kuvama hetkel kasutuses oleva kõrgepinge protsentuaalse suuruse

- Töölauarakendusel peab olema sisse- ja väljalülitatav taimer, peale mida katse lõpetatakse, salvestatakse ning kõrgepinge lülitatakse välja

- Töölauarakendusel peab saama valida taimeri kestvust (1-60 minutit, 1 minutiliste vahedega)
- Töölauarakendus peab kuvama katse kestust formaadis HH:mm:ss:fff
- Töölauarakendus peab kuvama kapillaarelektroforeesi seadmelt kättesaadud signaali muutmata kujul tekstiväljal
- Kõrgepinge protsentuaalne väärtus peab jääma 0 ja 100 vahele

2.2 Nõuded salvestatavatele andmetele

2.2.1 Nõuded salvestatavale graafiku failile

- Graafiku fail peab olema PNG formaadis
- Graafiku failil peab olema kogu katse kestel salvestunud signaalidest moodustunud elektroferogramm

2.2.2 Nõuded salvestatavale andmefailile

- Salvestama peab ainult kapillaarelektroforeesi seadmest saadud pinge väärtused
- Salvestada ei tohi esimest 100 väärtust (kuna nende seas tekib kõrgepinge sisse lülitamisest põhjustatud piik, mis pole seotud analüütidega)

2.2.3 Nõuded salvestatavale metadata failile

Salvestama peab

- katse teinud isiku nime
- katse teinud koodi, mis näitab katse teinud isiku volitusi
- meetodi nime
- maatriksi
- kapillaari ID ja OD
- kapillaari täispikkuse

- kapillaari effektiivse pikkuse
- elektrivoolu sageduse
- süstimismeetodi
- süstimismeetodi võrdlusdimensiooni koos selle väärtuse ja mõõtühikuga
- süstimismeetodi võrdlusdimensiooni kestuse sekundites
- kõrgepinge protsentuaalse suuruse
- katses kasutatavad analüüdid ja nende kontsentratsioonid koos mõõtühikuga
- katses kasutatavad BGE-d ja nende kontsentratsioonid koos mõõtühikuga
- katse kohta käiva kommentaari
- katse ajalise kestvuse formaadis HH:mm:ss:fff

2.2.4 Üldised nõuded

- Katse lõppedes tekkinud failid tuleb panna eraldi kausta
- Nii failide kui ka kausta nimes peab olema ajatempel formaadis kuupäev-kuu nimi-aasta-tund-minut

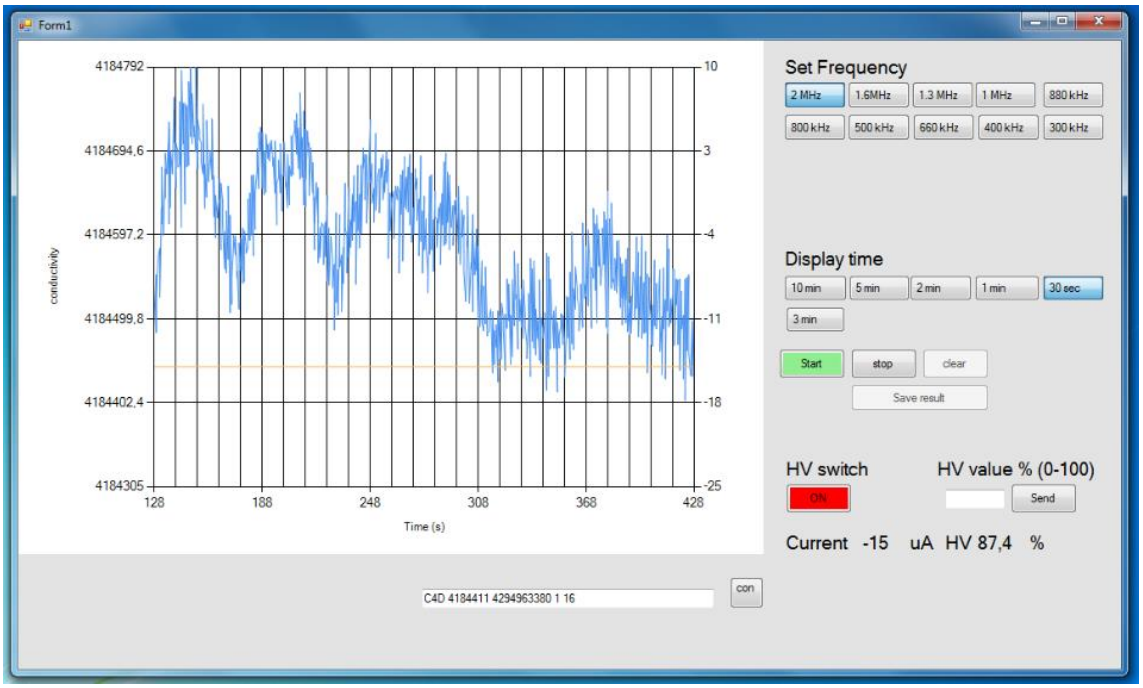
2.3 Nõuded REST API-le

- REST API peab olema sama jar faili sees, kus on ka veebirakendus (osa teise tudengi bakalaureusetööst)

2.4 Nõuded andmebaasile

- Kasutama peab PostgreSQL andmebaasi. Seda põhjusel, et ülikooli serveris, kus edaspidi rakendus kasutusele võetakse, kasutatakse samuti PostgreSQL andmebaasisüsteemi.

3 Olemasolev töölauarakendus ja probleem



Joonis 4. Olemasolev töölauarakendus Visual Basicu baasil

Seni kasutati katsete tegemisel Visual Basicus kirjutatud töölauarakendust (Joonis 4). Sellel olid osad praegustest nõuetest juba täidetud.

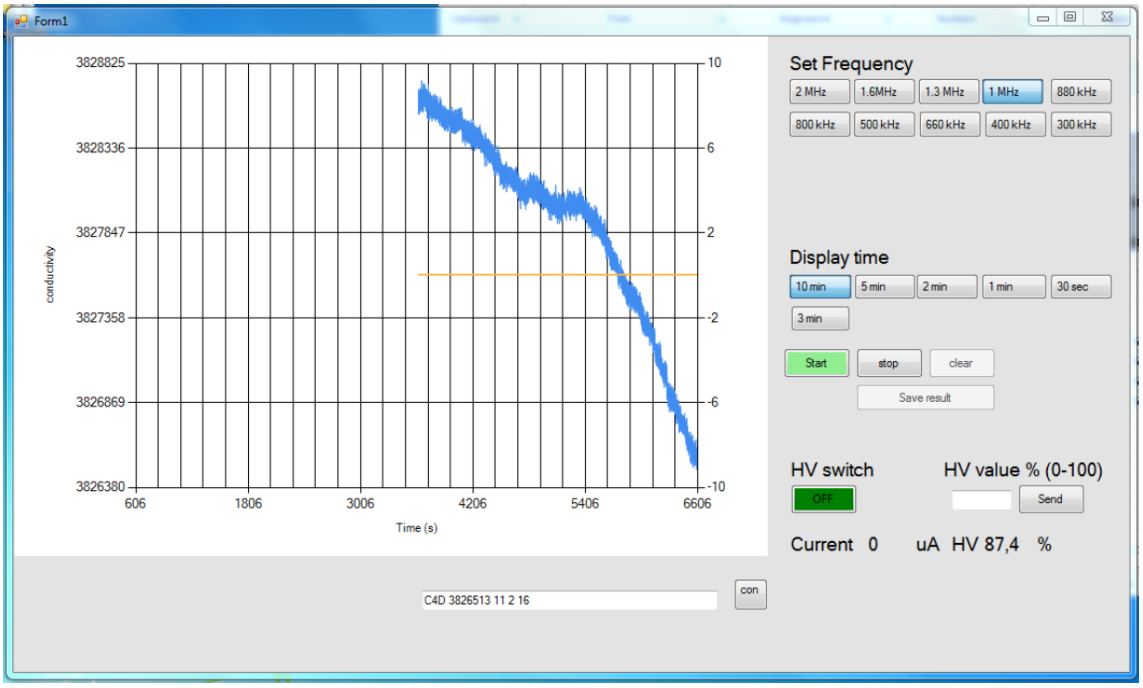
Olemas olid:

- elektrivoolu sagedust muutvad nupud
- graafiku vaadeldava osa suurust muutvad nupud
- ajaliselt muutuv signaali graafik
- ajaliselt muutuv voolutugevuse graafik
- nupp kõrgepinge sisse/välja lülitamiseks
- kõrgepinge protsentuaalset suuruse sisestamise võimalus
- hetke voolutugevust näitav tekstiväli

- hetkel kasutatav protsentuaalne kõrgepinge suurus
- kapillaarelektroforeesi seadmelt kättesaadud signaal muutmata kujul tekstiväljal
- andmete faili salvestamine.

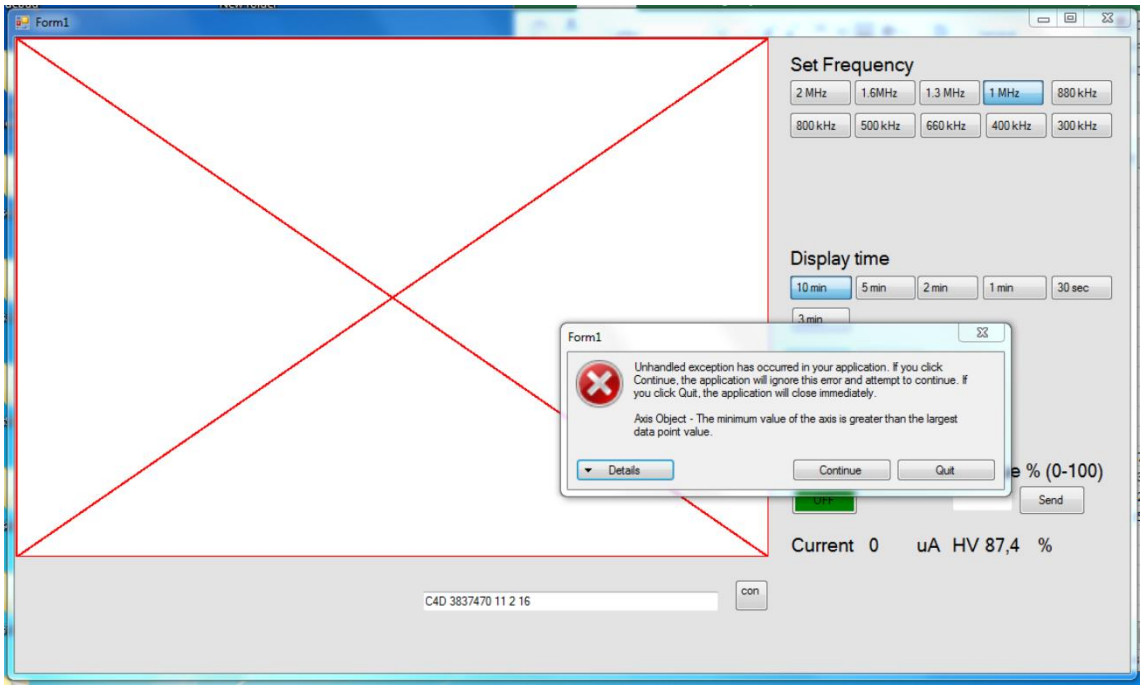
3.1 Probleem

Vajadus oli katsesse puutuvate andmete valikumenüü järele.



Joonis 5. Viga 10 minuti laiuses graafikus

10 minuti laiuses aknas oli näha ainult 5 minuti jagu graafikut. Kõik, mis läks üle 5 minuti, polnud enam nähtav (Joonis 5).



Joonis 6. *Crashinud* töölauarakendus

Töölaua rakendus lõpetas aegajalt täiesti töötamise ning *crashis* (sulgus plaaniväliselt). Peale seda oli sage selline olukord, kui rakendus *crashis* mitu korda järjest (Joonis 6). Enne kui õnnestus selle töökorras avamine.

Salvestada sai ainult kapillaarelektroforeesi seadmest saadud signaali. Graafiku ja metadata salvestamise võimalus puudus.

Katseandmeid ei saadetud andmebaasi, vaid sisestati käsitsi Exceli failidesse.

3.2 Probleemi lahendamise eesmärk

Vajadus oli uue töölauarakenduse järgi, mis pidi töötama ilma *crashimiseta*. Rakenduses oli vajalik menüü olemasolu, kus pidi saama teha erinevaid katsesse puutuvaid valikuid. Kuna Exceli failidesse käsitsi andmete sisestamine oli tülikas ning failidega ümberkäimine polnud kasutajasõbralik, oli vajadus andmete andmebaasi salvestamise järele.

4 Kasutatud metoodika

Selles peatükis kirjeldatakse kasutatud metoodikat, mida lõputöö autor loodud süsteemis kasutas või proovis kasutada. Mainitakse erinevaid programme, teeke, raamistikke kui ka andmeformaate.

4.1 Esialgne kasutatav töölauarakendus

4.1.1 Microsoft Visual Studio

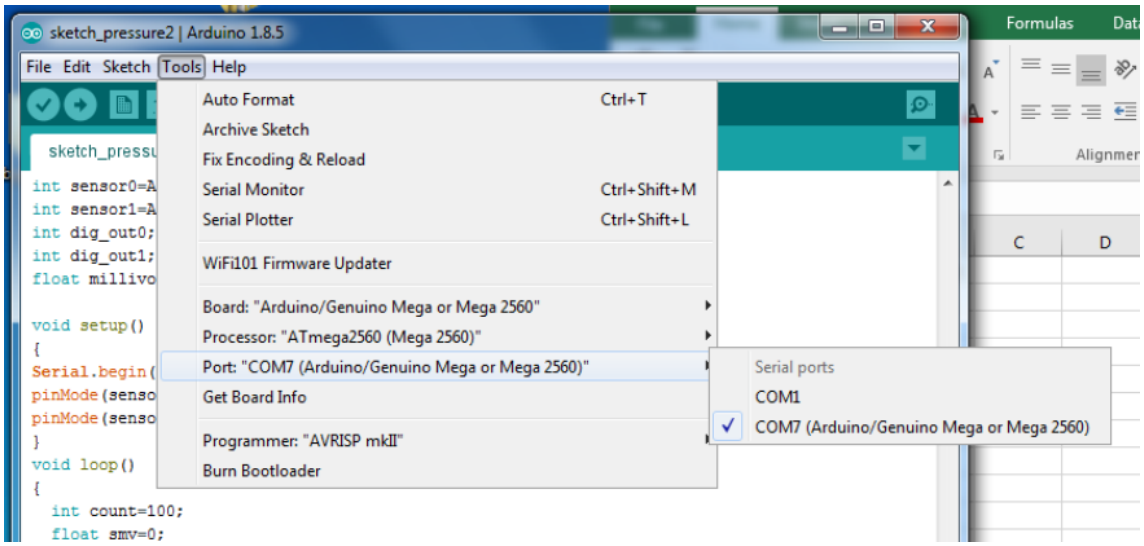
Keemikutel oli juba kasutuses töölauarakendus, millele tuginedes tudeng pidi omapoolset süsteemi hakkama üles ehitama. See oli kirjutatud Visual Basicus. Et mõista, mis funktsionaalsus oli ühe või teise rakenduse nupu taga peitus, pidi uurima Visual Basicu projekti. Selleks kasutas tudeng Microsoft Visual Studiot.

Tegu on Microsofti poolt loodud arenduskeskkonnaga, mis sobib peale teiste keelte ka Visual Basicu projektide kirjutamiseks. Microsoft Visual Studiol on olemas nii tekstiredaktor kui ka graafilise liidese disainimise funktsionaalsus.

4.2 Arduino

4.2.1 Arduino IDE

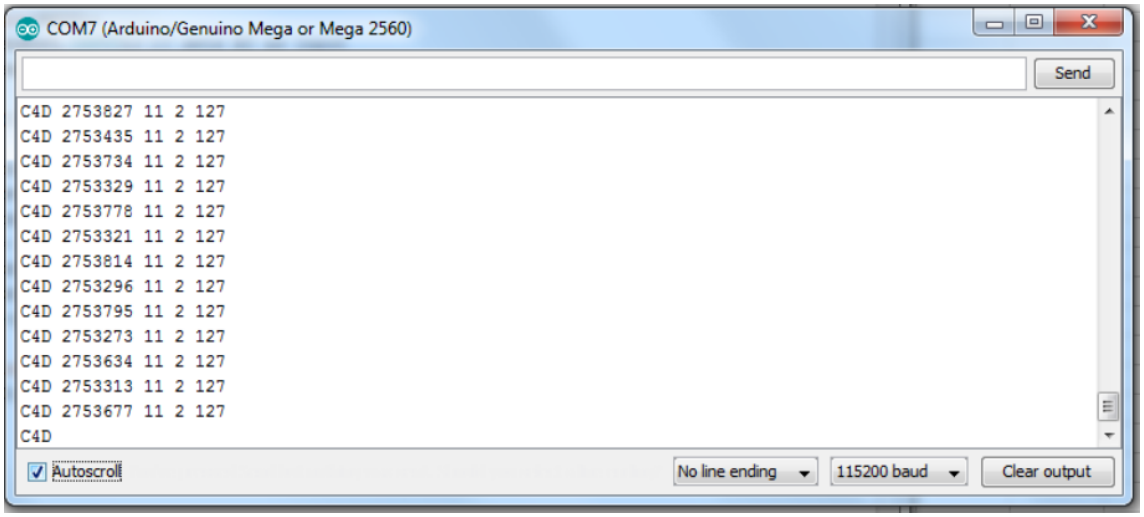
Java rakenduse ja Arduino vahelise suhtluse üles seadmiseks uuris tudeng kapillaarse elektroforeesi seadme koodi. Selleks kasutas ta Arduino IDE-t.



Joonis 7. Arduino mikrokontrolleri, protsessori ja pordi valimine

Arduino IDE võimaldab kirjutada koodi Arduino mikrokontrolleri jaoks, seda mikrokontrollerile laadida ning seejärel käivitada.

Et saada kätte informatsioon, tuleb kõigepealt valida sobiv Arduino mikrokontrolleri mudel. Seejärel tuleb valida sobiv protsessor. Ning kõige lõpus tuleb valida sobiv port, mille kaudu Arduino IDE infot saab/saadab (Joonis 7). Koodi kirjutamisel tuleb arvestada sellega, et pordi numbrid vahest muutuvad.



Joonis 8. Saadud info Arduino mikrokontrollerilt

Kui mikrokontroller, protsessor ja port on paika seatud, on võimalik Arduino mikrokontrollerile käske saata ning sealt infot saada (Joonis 8).

Käesoleval juhul tuleb informatsioon kapillaarelektroforeesi seadmest kujul "C4D 2753321 11 2 127". Seda informatsiooni hakkab töölauarakendus ka töötlema.

Informatsiooni lahtiseletus:

C4D - kapillaarelektroforeesi seadmele pandud "nimi".

2753321 - signaal, mis näitab analüüdi iooni elektrilist juhtivust. See kantakse ka graafikule.

11 - voolutugevus mikroamprites.

2 - pinge oleku staatus (1 - kõrgepinge on sisse lülitatud, 2- kõrgepinge on välja lülitatud).

127 - vabade ADC sammude arv.

4.2.1.1 ADC

Töölauarakenduses sisestatakse kõrgepinge protsentuaalse suurus. Kuna kapillaarelektroforeesi seadmes muudetakse kõrgepinge väärtust digitaalsete sammudega, mitte analoogilisel kujul, siis konverteeritakse kõrgepinge protsentuaalne väärtus digitaalselt kasutatavatesse ühikutesse. Seadmes kasutatakse 128 digitaliseerimise ühikut. Kasutatavat pinget (U_kasutatav) arvutatakse valemiga

U_kasutatav = x / 127 * U_kogupinge, kus x on kasutatavate ADC sammude arv. 128 asemel 127, kuna esimene digitaalne samm on 0 ja siis pole pinget. Pingel olemasolul on kasutusel 127 digitaalset sammu.

1% -le vastab järelikult 1,27 sammu.

Näiteks kui sisestati 87%, siis selle põhjal saadakse arvutamisel 127 - 87 * 1,27 = 16,51 See ümardatakse täisarvuni ning saadakse 17. Seega arvutamisel kasutatakse 17 digitaliseerimise sammu. Kui kasutusel on kõrgepinge 20 kV, siis 87% puhul kasutatakse 17 / 127 * 20 ≈ 13,7 kV kõrgepinget.

4.3 Java

Kuna tudeng on kolme aasta jooksul õppinud kõige rohkem Java programmeerimiskeelt, siis graafilise kasutajaliidese tegemisel otsustati kasutada just seda programmeerimiskeelt.

4.3.1 IntelliJ

Tegu on Java IDE-ga. IntelliJ-s on olemas tekstiredaktor, graafilise liidese disainimise võimalus, *debugimise* võimalus kui ka erinevate teekide ja *pluginite* lisamise võimalus. IntelliJ abil on lihtne koodi kirjutada, seda jooksutada ning kompileerida jar faili.

4.3.2 Gluon Scene Builder

Gluon Scene Builder on graafiliste liideste disainimisrakendus, mis töötab JavaFX teegis kasutusel olevate objektidega. Selles on olemas "võta-ja-lohista" funktsionaalsus, mis tähendab seda, et võimalik on juba valmis objekte (näiteks nuppe, tekstikaste, menüüsid) disainiaknasse õigele kohale lohistada.

Kuigi IntelliJ pakub ka graafilise liidese disainimise võimalust, kasutas tudeng käesolevas töös Gluon Scene Builderit, kuna see on spetsiaalselt mõeldud JavaFX teegile. Samas IntelliJ kõiki võimalusi, mis Gluon Scene Builderis olemas on, ei paku, kuna IntelliJ disainimisrakendus on pigem üldine, mitte suunatud teatud kindlale teegile.

4.4 Tudengi poolt loodud töölauarakendus

4.4.1 RXTX

RXTX on Java teek, mis on mõeldud tööks JDK-le jada- ja paralleelühendusega portidega [3] .

Kuigi funktsionaalselt oli see loodavale süsteemile küll sobiv, otsustas tudeng seda teeki mitte kasutada. Nimelt probleem oli RXTX teegi installeerimises. See eeldas endas kahe dll faili asetamist Java süsteemikataloogidesse. Lõppkasutaja jaoks poleks see aga olnud kasutajasõbralik lahendus. Lõppkasutaja soovis lihtsalt üht faili oma töölaual käivitada, mitte tegeleda erinevate lisafailide ja kaustadega. Tudeng küll mõtles ka installeeritava programmi loomisele, aga lihtsam oli kasutada teist teeki. Samuti RXTX teeki enam ei

arendata edasi. Viimane uuendus tehti https://github.com/rxtx/rxtx keskkonnas 6 aastat tagasi.

4.4.2 JSSC

Nagu ka RXTX on JSSC teek, mis on mõeldud tööks JDK-le jada- ja paralleelühendustega portidega. See teek on uuem ning ka stabiilsem. RXTX teegi kasutamisega on erinevatel kasutajatel ilmnenud erinevaid probleeme (vood lukustavad üksteist, vood jäetakse teegi sees lahti ning seetõttu pole võimalik programmi sulgeda normaalselt, teegi normaalne töö sõltub erinevate protsesside õigest ajastusest, jne) [4].

Kuna JSSC teegi installeerimiseks ei ole vaja kasutada eraldi lisafaile, otsustas tudeng oma töös kasutada just seda teeki. Samuti ilmnes, et JSSC pakub lihtsalt võimalust portide automaatseks tuvastamiseks. Samas kui RXTX teegi puhul pidi koodi kirjutama, millist porti soovitakse kasutada.

4.4.3 libGDX

libGDX on Javapõhiste mängude loomise raamistik. See pakub ühtset API-t, mis sobib igale platformile. Raamistik pakub keskkonda kiireks prototüüpimiseks ja kiireks iteratsiooniks. Töölaua JVM-i funktsioonid, nagu reaalajas koodivahetus, vähendavad oluliselt vajadust koodi pidevalt kompileerida erinevate platformide jaoks [5].

Kuigi libGDX pakub palju graafilisi võimalusi, ei ole selles juba valmis mooduleid graafikute tegemiseks. Kuna kapillaarelektroforeesi töölauarakenduses tuleb kasutajale kuvada reaalajas muutuvat graafikut, siis libGDX ei olnud sobilik valik. libGDX puhul oleks küll olnud võimalik implementeerida graafikute kuvamine, aga see oleks eeldanud sellise funktsionaalsuse nullist üles ehitamist. Samas kui on olemas alternatiivsed lahendused, kus graafikute moodul on juba valmis arendatud.

4.4.4 Swing

Swing on Java arenduskomplekt graafiliste liideste loomiseks. Swing arenduskomplektis on juba valmis loodud erinevad komponendid (näiteks nupud, nimekirjad, tekstiväljad, menüüd, jne [6]), mida on võimalik kiiresti ja lihtsalt oma koodis kasutada.

Kuigi Swing on libGDX-ga võrreldes palju sobivam käesoleva projekti arendamisel, otsustas tudeng enamikku funktsionaalsust sellest mitte kasutada. Nimelt leidis tudeng uuema alternatiivi (JavaFX).

Küll on Swing abil tehtud kapillaarelektroforeesi töölauarakenduses graafiku salvestamise funksionaalsus. Nimelt vastav lahendus JavaFX puhul võttis märgatavalt kauem aega ja nõudis suuremat mälukasutust (olukorrani, kus reaalselt polnud võimalik enam mõistliku aja jooksul graafikut salvestada, kui sellel oli üle kümne tuhande mõõtepunkti). Swing abil õnnestus graafiku salvestamine loetud sekundite jooksul.

4.4.5 JavaFX

JavaFX on graafiliste liideste arendamisplatform, mis pakub mitmekesiseid valmis arendatud graafilisi komponente. JavaFX on ka pidevas arenduses (võrreldes Swing-iga, mida enam edasi ei arendata).

Kuigi Swing puhul on olemas rohkem dokumentatsiooni ja rohkem kasutajaid, otsustas tudeng kasutada just JavaFX teeki selle jätkusuutliku arenduse tõttu.

4.4.6 ControlsFX

ControlsFX on avatud lähtekoodiga projekt JavaFX teegile, mille eesmärk on pakkuda kõrgekvaliteedilisi kasutajaliideste kontrollelemente ja muid töövahendeid, et täiendada JavaFX põhifunktsionaalsust. See on loodud JavaFX 8.0+ versioonile [7].

4.4.7 JFreeChart

JFreeChart on Java teek, mis on mõeldud professionaalsel tasemel graafikute loomiseks ja kuvamiseks. See pakub hästi dokumenteeritud API-t, mis toetab paljusi erinevaid graafikutüüpe; paindlikku disaini, mida lihtne laiendada nii serveripoolsetele kui ka kasutajapoolsetele rakendustele; tuge erinevatele väljunditüüpidele (pildifailid, JavaFX komponendid, Swing komponendid, jne) [8].

JFreeChart on see Swingi põhine teek, mida tudeng kasutas graafiku pildifaili salvestamisel.

4.4.8 JCommon

JCommon on teek, mis on vajalik JFreeChart teegi töös. Selles on tekstiutiliidid, serialisatsiooni utiliidid ja muud JFreeChart jaoks vajalikud klassid [9].

4.4.9 JSON

JSON on andmevahetuse formaat. JSON-is kasutatakse võtme-väärtuse paare, kus ühele võtmele määratatakse mingi väärtus või väärtuste jada.

Järgnevalt on näide lõputöös JSON formaadi kasutamisest:

```
{
  "nameOfTest": "11 jaanuar",
  "nameOfMethod": "mingi labTest",
  "nameOfUser": "Aivar",
  "userClass": "1",
  "analytes": [
    {
      "analyte": {
        "name": "",
        "value": "uraan",
        "valid": true
      },
      "concentration": {
        "name": "",
        "value": "11",
        "valid": true
      }
    },
    {
      "analyte": {
        "name": "",
        "value": "plaatinum",
        "valid": true
      },
      "concentration": {
        "name": "",
        "value": "100",
        "valid": true
      }
    }
  ],
  "analyteUnit": "mol",
  "matrix": "kraanivesi",
  "bgeUnit": "ppb",
  "capillary": "50/150 mm",
  "capillaryTotalLength": "20 cm",
  "capillaryEffectiveLength": "10 cm",
  "injectionMethod": "Pressure",
  "injectionChoice": "Difference",
  "injectionChoiceValue": "20",
  "injectionChoiceUnit": "cm",
  "injectionTime": "5 s",
  "current": "-15 µA",
  "frequency": "2 MHz",
```

```
  "description": "mingi test",
  "hvValue": "87 %",
  "testTime": "00:00:23:231"
}
```

JSON formaadis saadeti infot REST API-le, mis võtmete põhjal võttis välja vajaliku info ning salvestas vastavasse andmebaasi tabelisse. Samuti toimus info töölauarakendusele tagasi saatmisel JSON formaadis.

4.4.10 Gson

Gson on java teek, mida saab kasutada Java objektide konverteerimiseks nende JSON kujule. Seda saab kasutada ka JSON stringi konverteerimiseks vastavasse Java objekti. Gson võib töötada suvaliste Java objektidega, sealhulgas olemasolevate objektidega, mille lähtekoodi ei omata [10].

4.5 REST API

REST API, mis võttis töölauarakenduselt info vastu, töötles seda ja suhtles andmebaasiga kirjutamisel kasutati sarnaselt töölauarakenduse kirjutamisele Java programmeerimiskeelt. API kirjutati Spring Booti rakendusse.

4.5.1 Spring Boot

Spring Boot on Spring raamistikul (Java raamistik) põhinev lahenus, mis aitab luua rakendusi, mida saab lihtsalt käivitada. Spring Boot rakendusse saab juba eelnevalt https://start.spring.io/ lehel lisada kõik vajalikud teegid, sõltuvused ja metadata. Seega juba algusjärgus vajab Spring Boot projekt väga vähest konfigureerimist.

Spring Boot hõlmab endas Tomcat veebiserverit, mille peal Java saab töötada. Samuti võimalust valida, kas kasutatakse Gradle või Maven automatiseerimissüsteemi.

4.5.2 PostgreSQL JDBC Driver

PostgreSQL JDBC Drive on Java draiver PostgreSQL andmebaasidega ühendumiseks ja nendega töötamiseks.

4.5.3 Gson

Gson on java teek, mida saab kasutada Java objektide konverteerimiseks nende JSON kujule. Seda saab kasutada ka JSON teksti konverteerimiseks vastavasse Java objekti.

Gson võib töötada suvaliste Java objektidega, sealhulgas olemasolevate objektidega, mille lähtekoodi ei omata [10].

4.6 Andmebaas

Kuna eelnevalt on lõputööd kirjutav tudeng õppinud mitme õppeaine raames PostgreSQL andmebaasi kasutamist ning andmebaas, kus lõplik rakendus hakkab edaspidi jooksma, on samuti PostgreSQL põhine, siis lõputöös kasutati samuti just PostgreSQL andmebaasisüsteemi.

Andmebaasi salvestati katseandmed, metadata ja analüüsi tulemused, mis saadi laboris tehtud katsetelt kapillaarelektroforeesi seadmega.

4.6.1 Microsoft Access

Esmane andmebaasi disainimine toimus Microsoft Accessis, kus loodi vajalikud andmebaasi tabelid, nendevahelised suhted ning kuhu sisestati ka algne informatsioon.

Microsoft Access on andmebaasi halduse süsteem, mis hõlmab endas graafilist kasutajaliidest, tarkvara arendamise töövahendeid, võimalust käivitada SQL koodi kui ka võimalust luua andmebaasi.

4.6.2 Bullzip MS Access to PostgreSQL

Bullzip MS Access to PostgreSQL on programm, mis konverteerib MS Access andmebaasi faili SQL koodi (Lisa 1), mida saab PostgreSQL süsteemi toetavas serveri jooksutada. Peale konverteerimist on vajalik mõningane faili modifitseerimine (jutumärkide eemaldamine, tabelite õigesse järjekorda panek) enne kui seda saab serveris käivitada.

4.6.3 pgAdmin

pgAdmin on PostgreSQL andmebaaside administreerimis- ja arendusvahend. Sellega saab hallata mitut andmebaasi korraga, luua uusi ühendusi, uusi andmebaase, uusi andmebaasi tabeleid ja ka lisada informatsiooni tabelitesse. Samuti on sellega võimalik käivitada SQL lauseid.

pgAdmin töötab ka lokaalselt, eeldusel, et arvutisse on installeeritud PostgreSQL andmebaas.

4.7 Versioonihaldus

Et säilitada varasemaid versioone loodud koodist kasutati versioonihalduseks GitLab versioonihalduse keskkonda.

4.7.1 GitLab

GitLab on versioonihalduse keskkond, kus on võimalik säilitada varasemaid versioone oma loodud koodist, luua uusi projekte, töötada projekti kallal kogu meeskonnaga, pidada nimekirja tehtud ja tegemata ülesannetest ja palju muud.

Tudengi failid ja kood on https://gitlab.cs.ttu.ee/water-analyzer repositooriumis. Repositooriumi omanik on juhendaja Evelin Halling.

4.7.2 Git Bash CLI

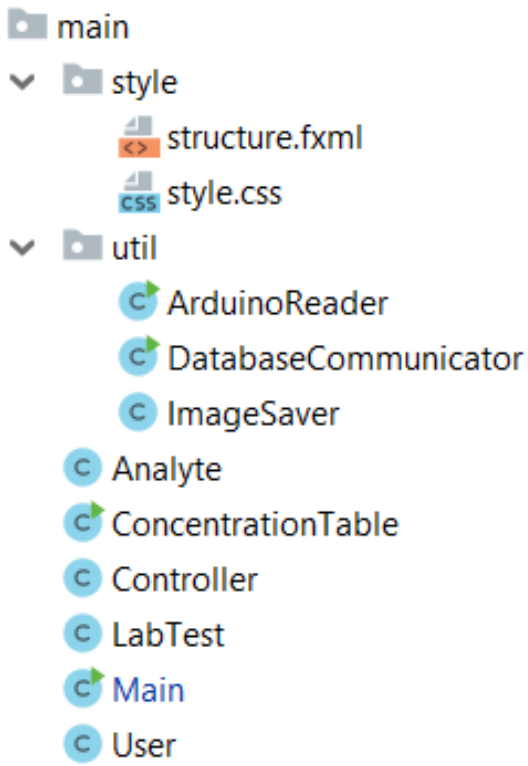
GitLabi repositooriumisse uuenduste saatmiseks kasutati Git Bash konsooliprogrammi, mis on mõeldud Git-iga Windowsi operatsioonisüsteemis töötamiseks. Git Bash hõlmab endas peale Giti käskude ka mõningaid Linuxi operatsioonisüsteemi käske.

5 Süsteemi struktuur

Selles peatükis kirjeldatakse rakendustes olevaid klasse ja andmebaasis olevaid tabeleid. Samuti seletatakse lahti tegevusvoos protsesside loogiline ja ajaline järjestus.

5.1 Töölauarakendus

5.1.1 Kasutatud klassid



Joonis 9. Töölauarakenduses kasutatud klassid

Töölauarakenduses kasutas tudeng kolme paketti: "main" pakett, "style" pakett ja "util" pakett.

5.1.1.1 "style" pakett

Paketis hoitakse FXML faili ja CSS faili. Ehk seda osa, mis on oluline graafilise osa disainis.

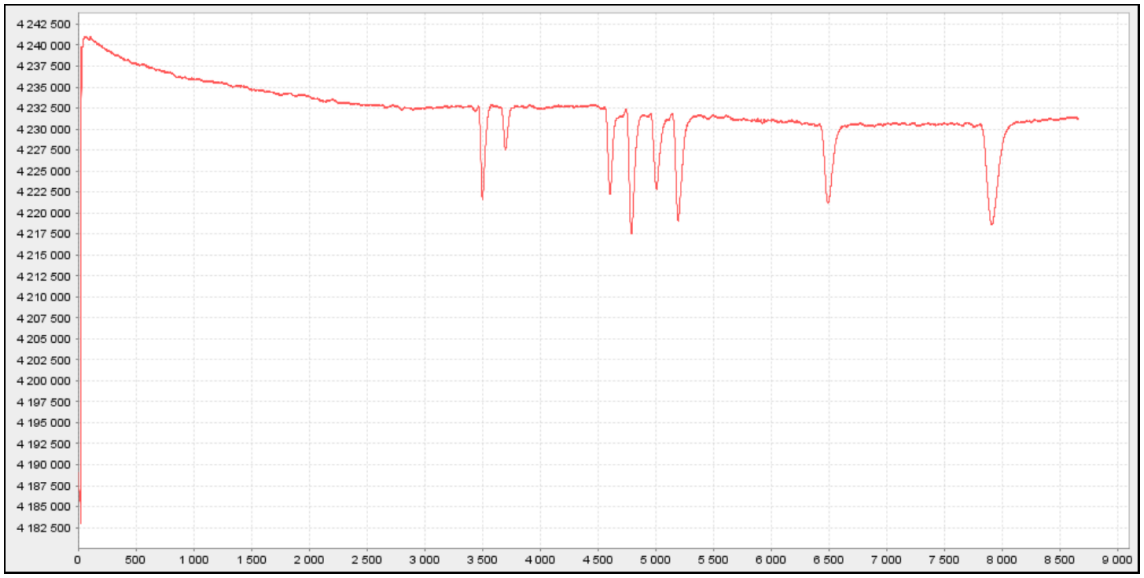
"structure.fxml" failis on töölauarakenduse graafilise liidese struktuur (nupud, tekstiväljad, menüüd, jne).

"style.css" failis oli nuppude ja graafikuga seotud stiili-info.

5.1.1.2 "util" pakett

ArduinoReader klass tegeleb Arduinoga ühenduse loomisega, sellest info kätte saamise ja Arduinole käskude saatmisega.

DatabaseCommunicator klass võtab salvestatud katseandmed ning saadab need REST API poole teele. REST API-s omakorda salvestatakse andmed andmebaasi. Samuti see klass teeb päringuid REST API peale, et saada andmebaasist andmeid.



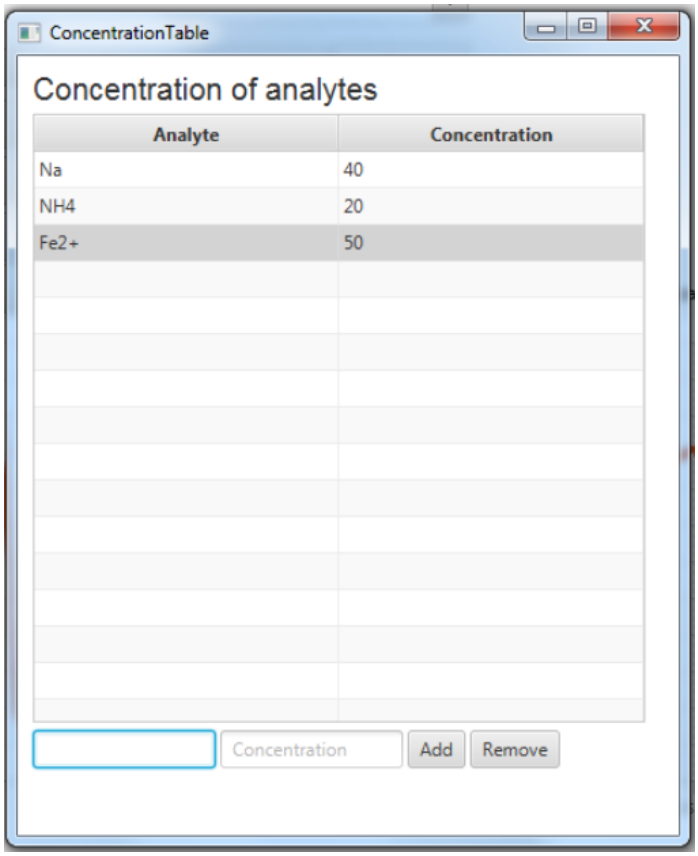
Joonis 10. Pildifailis olev graafik

ImageSaver klass salvestab graafiku pildifaili (Joonis 10).

5.1.1.3 "main" pakett

"main" paketis on nii "util" pakett, "style" pakett kui ka klassid.

Analyte klass on analüüdi objekt, mida kasutatakse kontsentratsioonitabelis. Seal on nii analüüdi nimi kui ka analüüdi kontsentratsiooni väärtus.



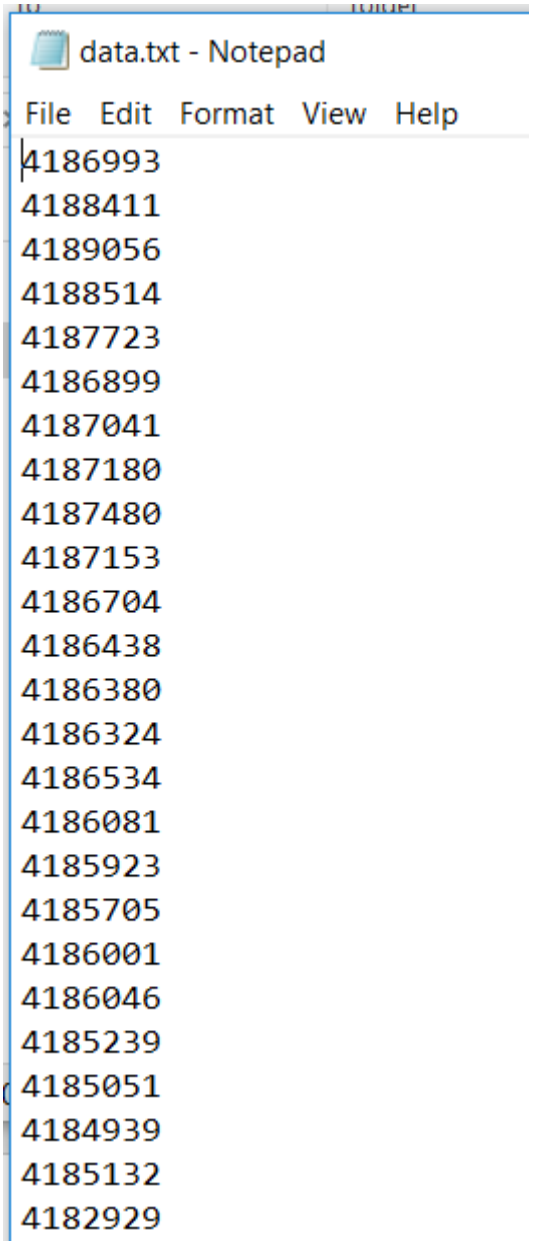
Joonis 11. Nimekiri analüütidest ja nende kontsentratsioonidest

ConcentrationTable klass on kontsentratsioonitabeli objekt. See kasutab Analyte klassi ning on omakorda välja kutsutud Main klassis. ConcentrationTable klass kuvab info valiknimekirjades ära märgitud analüütidest/BGE-dest. Samuti on seal võimalik lisada/eemaldada nii analüüte kui ka BGE-sid. Peale analüüdi/BGE kuvamist saab lisada ka selle kontsentratsiooni väärtuse (Joonis 11).

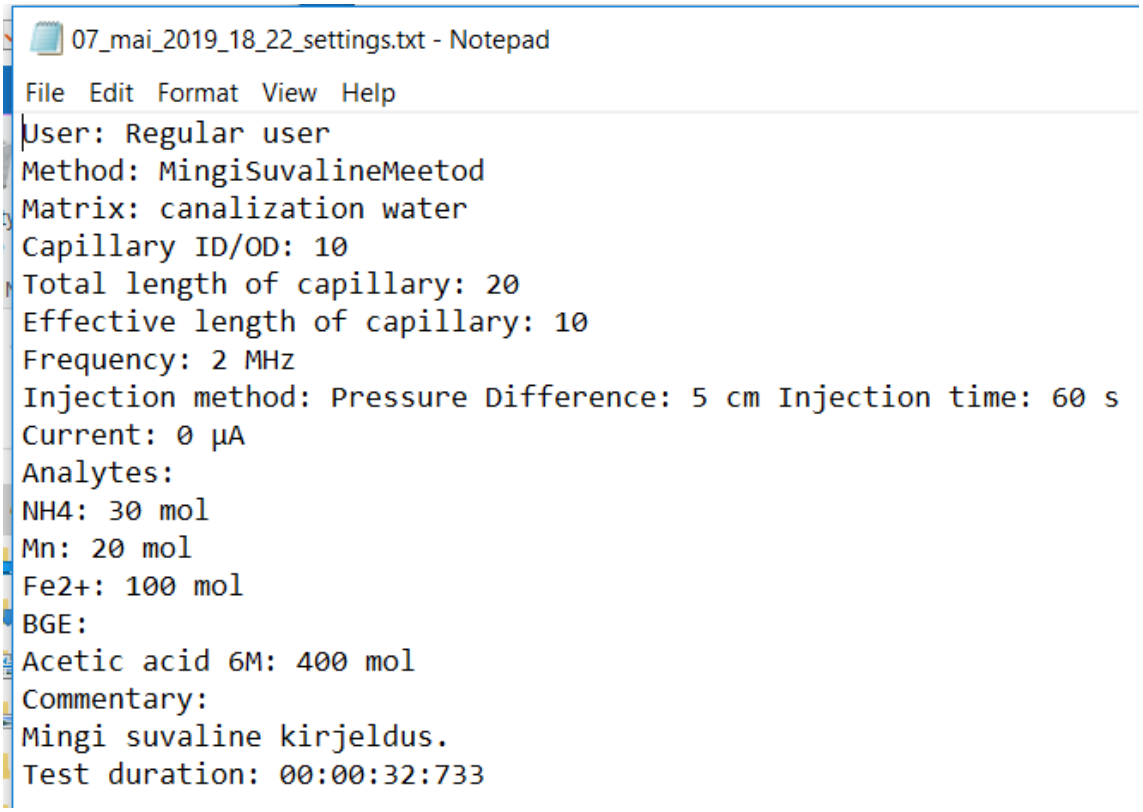
Controller klass on JavaFX projekti poolt loodud klass, mida kasutatakse projekti käivitamisel.

LabTest klass on objekt, milles hoitakse kõiki testiga seotud andmeid. Sellest objektist luuakse Gson abil JSON string, mis omakorda REST API-le saadetakse. Osa sellest JSON-ist on välja toodud lõputöö lehekülgedel 30-31.

User klass on objekt, milles hoitakse rakenduse kasutaja nime ja ka tema staatust (administraator/keemik/tavakasutaja).



Joonis 12. Arduinolt saadud signaalid tekstifailis



Joonis 13. Metadata tekstifailis

Main klassis luuakse graafiline liides, tehakse ühendus Arduinoga, kutsutakse välja ülejäänud klassid ja katse lõpus salvestatakse signaalid tekstifaili (Joonis 12) ja metadata tekstifaili (Joonis 13).

5.1.2 Tegevusvoog

Kõigepealt luuakse JavaFX listener, mis graafilise liidese sulgumisel kõik protsessid suleb. Vastasel juhul jäävad osad protsessid ikka tagaplaanil tööle. Seejärel loetakse sisse FXML fail ja CSS fail ning hakatakse looma graafilist liidest. Valmis luuakse graafilise liidese põhi (ehk aken).

Enne graafilise liidese elementide loomist tehakse interneti olemasolul REST API peale GET päringud, et andmebaasist uut informatsiooni alla laadida ja uuendada seda informatsiooni, mille põhjal luuakse graafilises liideses nimekirjad. Interneti puudumisel lihtsalt järgmised sammud võtavad paar sekundit kauem aega (kuna rakendus üritab REST API-t jooksutava serveriga ikkagi paar korda ühenduda).

Seejärel luuakse kõik nupud, valikmenüüd ja nimekirjad ning kuvatakse need lõppkasutajale.

Peale graafilise liidese valmis saamist üritab rakendus ühenduda Arduino peal jooksva kapillaarelektroforeesi seadmega. Kui ühendumine õnnestub, siis rakendus hakkab reaalajas kuvama seadmelt saadud signaali tekstilahtris. Kui ühendumine ei õnnestu (seade on välja lülitatud või seadmes on rike), siis rakendus küll avaneb, aga tekstilahtrisse ei ilmu signaaliga seotud infot.

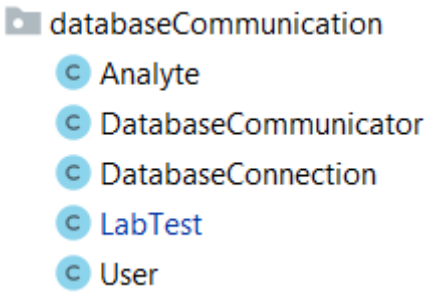
Juhul kui Arduinoga on ühendus olemas, siis võib soovi korral paika panna katse seadistused ja soovi korral ka taimeri. Seejärel "Start" nupule vajutades saadetakse Arduino mikrokontrollerile signaal kõrgepinge sisse lülitamiseks ja signaal sobiva voolu sageduse valikuks. Voolu sagedus muutub kapillaarelektroforeesi seadmes kuna seal on erinevad takistid ja erinevate takistite valikul muudetakse ka sagedust erinevaks. Kätte saadud signaali hakatakse siis kuvama graafikule ja samal ajal salvestama mällu.

Katse läbi saamisel (kas "Stop" nupule vajutades või taimeri aja läbi saamisel) saadetakse Arduino mikrokontrollerile käsk kõrgepinge välja lülitada ning mällu salvestatud testandmed salvestatakse tekstifailidesse, pildifailidesse. Samuti tehakse testandmetest LabTest objekt, mis Gson abil JSON stringiks muudetakse ja interneti olemasolul REST API-le saadetakse.

Kõige lõpus tehakse uuesti REST API peale GET päringud, et andmebaasist uut informatsiooni alla laadida.

5.2 REST API

5.2.1 Kasutatud klassid



Joonis 14. REST API-s kasutatud klassid

REST API asub databaseCommunication paketi sees.

Analyte klass on analüüdi objekt, mida kasutatakse kontsentratsioonitabelis. Seal on nii analüüdi nimi kui ka analüüdi kontsentratsiooni väärtus. REST API-s saadakse Gson abil sealt kätte analüüdi nime ja tema kontsentratsiooni.

DatabaseCommunicator klass tegeleb REST päringutega. Kasutatakse järgnevaid päringuid: getAnalytes, getBges, getMatrixes, getMethods, getUsers, postTest (postTest on REST API-le katseandmete saatmiseks). REST API kuulab päringuid samanimeliste ühenduspunktide pealt.

DatabaseConnection teeb ühenduse PostgreSQL andmebaasiga ning tagastab Connection objekti, mille kaudu SQL päringuid käivitama hakatakse ja mille kaudu SQL päringute vastus tuleb.

LabTest klass on objekt, milles hoitakse kõiki testiga seotud andmeid. Selle klassi abil luuakse JSON stringist LabTest objekt, kust saab erinevat katsega seotud infot getter meetoditega välja võtta.

User klass on objekt, milles hoitakse rakenduse kasutaja nime ja ka tema staatust (administraator/keemik/tavakasutaja).

5.2.2 Tegevusvoog

Enamike GET päringute puhul (/getUsers, /getAnalytes, /getBges, /getMatrixes) teeb REST API ühe SQL päringu andmebaasi, et sisse lugeda kõik vastava andmebaasi tabeli read. Need read lisatakse Arraylisti, mis Gson abil JSON stringiks muudetakse ja tagasi töölauarakendusele saadetakse.

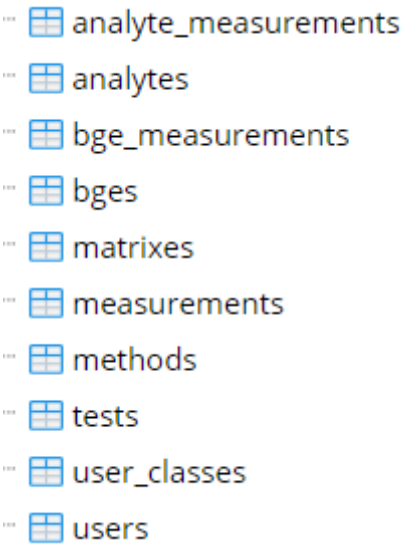
Samas kuna methods tabelis on viited omakorda matrixes, analytes_measurements ja bge_measurements tabelitele ja nendes omakorda viited analytes ja bges tabelitele, siis /getMethods requestide puhul tuleb teha iteratiivselt iga SQL päringu abil saadud method_id abil omakorda päringuid matrixes tabelisse, et igale meetodile vastav maatriks saada. Samuti tehakse iteratiivselt päringuid analytes_measurements ja bge_measurements tabelitesse ja nendest omakorda analytes ja bges tabelitesse, et saada igale meetodile vastavad analüüdid ja nende kontsentratsioonid ning BGE-d ja nende kontsentratsioonid. Saadud informatsiooni abil luuakse LabTest objekt, mis Gson abil JSON stringiks konverteeritakse ja tagasi töölauarakendusele saadetakse.

/postTest päring on sarnane /getMethods päringule selles osas, et iteratiivselt tehakse SQL päringuid andmebaasidesse. Alguses nendesse tabelitesse, millel puuduvad välised viited omakorda teistele tabelitele. Kõigepealt tehakse SELECT * päring ja loetakse ArrayListi sisse kõik tabeli read. Kui tabeli ridade sees on lisatav objekt, siis seda teist korda enam ei lisata, vaid tagastatakse selle objekti kood. Kui aga see objekt puudub ArrayListis, siis tehakse INSERT INTO päring ning tagastatakse selle objekti kood. Seejärel aja seejärel nendesse, millel on juba välised viited olemas. Ja samamoodi: kõigepealt SELECT * päring ja seejärel, vajadusel, INSERT INTO päring. Nii käiakse läbi kõik võtmed ja nende väärtused JSON stringis, mis /postTest päringuga REST API-le saadeti.

5.3 Andmebaas

Järgnevalt iseloomustatakse andmebaasis olevaid tabeleid ning nendevahelisi seoseid.

5.3.1 Kasutatud tabelid



Joonis 15. Andmebaasi loodud tabelid

PostgreSQL andmebaasis kasutatakse 10 tabelit, mis on seotud töölauarakenduse kasutajate ja laborikatsetest saadud andmetega.

analyte_measurements tabelis hoitakse testi tegemise aega, meetodi koodi, analüüdi koodi, analüüdi kontsentratsiooni ja analüüdi mõõtühikut.

analytes tabelis hoitakse kõiki analüüte.

bge_measurements tabelis hoitakse testi tegemise aega, meetodi koodi, BGE koodi, BGE kontsentratsiooni ja BGE mõõtühikut.

bges tabelis hoitakse kõiki taustaelektrolüüte.

matrixes tabelis hoitakse kõiki maatrikseid.

measurements tabelis hoitakse testi koodi ja kapillaarelektroforeesi seadmest saadud signaali väärtust.

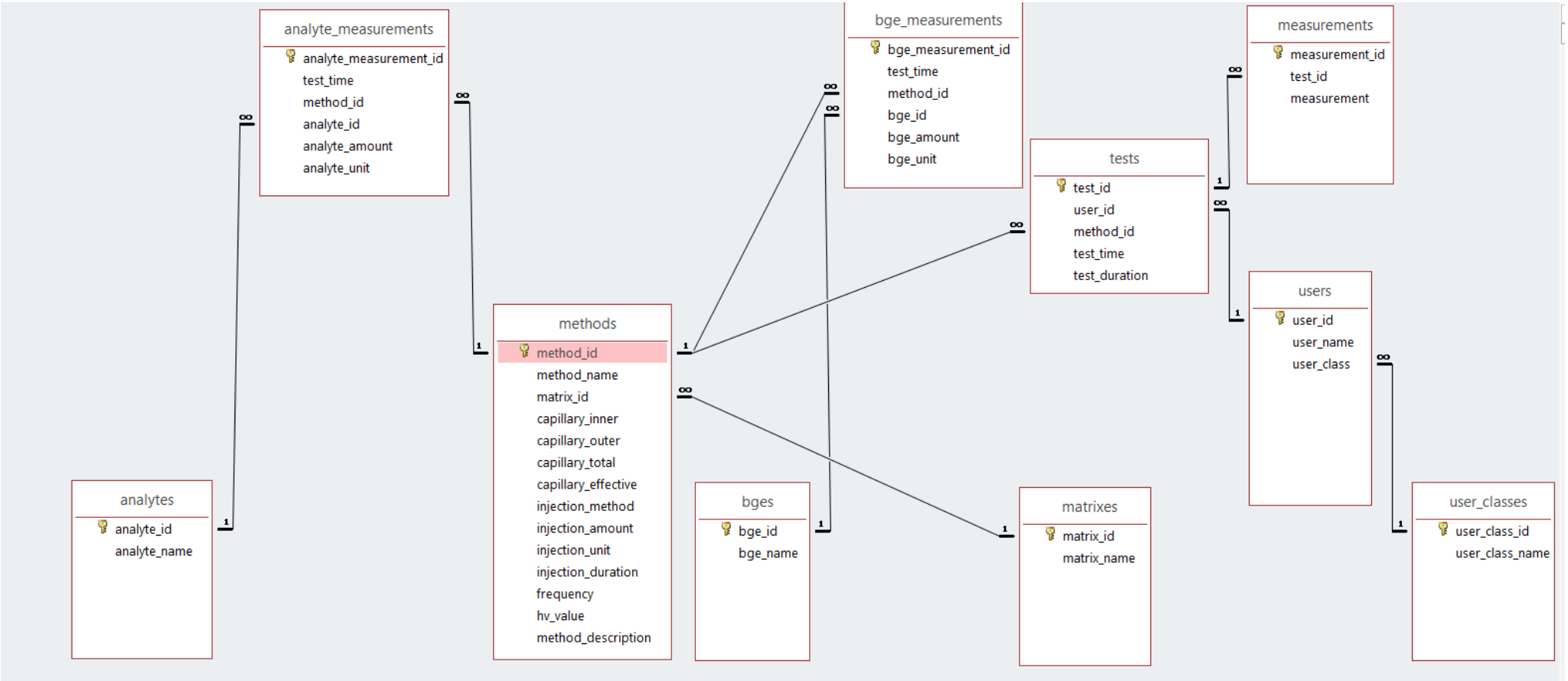
methods tabelis hoitakse meetodi nime, maatriksi koodi, kapillaari sisediameetrit, kapillaari välisdiameetrit, kapillaari täispikkust, kapillaari effektiivpikkust, analüüdi sisestamisviisi, analüüdi sisestamisviisi võrdlust taustsüsteemiga, analüüdi sisestamisviisi võrdluse mõõtühikut, analüüdi sisestamise aega, elektrivoolu sagedust kilohertzides, kõrgepinge protsentuaalset väärtust ja katse kirjeldust.

tests tabelis hoitakse kasutaja koodi, meetodi koodi, testi tegemise aega, testi kestvust sekundites.

user_classes tabelis hoitakse kasutaja klassi nime (näiteks "administraator").

users klassis hoitakse kasutaja nime ja kasutaja klassi koodi.

5.3.2 Tabelitevahelised seosed



Joonis 16. Tabelitevahelised seosed

6 Testimine

Testimisel kasutati erinevaid meetodeid: manuaalset testimist, andmete võrdlemist varasemate katseandmetega kui ka kasutajakogemuse testimist.

6.1 Manuaalne testimine

Manuaalne testimine toimus nii laboris kapillaarelektroforeesi seadmega, kui ka tudengi kodus testandmetega. Kuna mõlemal juhul oli andmete allikas erineva olemusega, siis kasutati kaht erinevat versiooni töölauarakendusest: 1)versioon, mis üritas infot kätte saada Arduino peal jooksvalt kapillaarelektroforeesi seadmelt, 2)versioon, mis luges signaali sisse tekstifailist. Kuna tesktifailist andmete lugemine ei taganud töökindlust Arduinoga, oli vajalik ka laboris kohapealne testimine.

Andmebaasi töökindluse testimiseks jooksutas tudeng oma arvutis PostgreSQL andmebaasi ning tegi ühenduse localhosti pihta.

6.2 Andmete võrdlemine varasemate katseandmetega

Kaasjuhendaja Jelena Gorbatšova võimaldas ligipääsu varasematele katseandmetele. Seega võimalik oli kontrollida, kas graafikud ning nii kapillaarelektroforeesi seadmelt kui ka töölauarakenduselt saadav informatsioon on kooskõlas selle informatsiooniga, mis saadi Visual Basicus kirjutatud töölauarakendust kasutades. Samuti oli võimalus võrrelda graafikute pildifaile (Visual Basicu andmetest genereeriti pildifail Excelis, tudengi töölauarakendus salvestas pildifaili automaatselt).

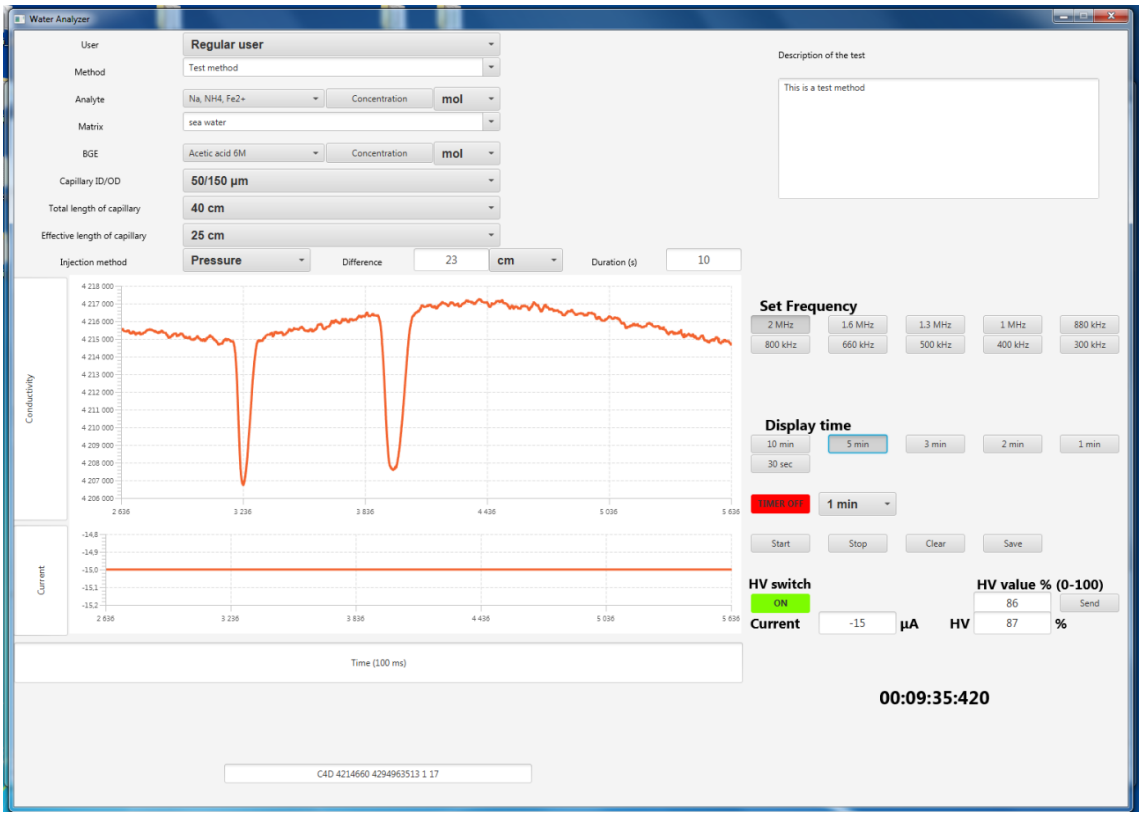
6.3 Kasutajakogemuse testimine

Tudeng tegi oma koodist aegajalt töövalmis versioone ning lasi nii kaasjuhendajal Jelena Gorbatšoval kui ka tema bakalaureuse tööd tegeval keemiatudengil oma rakendust testida ja kasutada ning vastavalt tagasisidele tegi oma koodis muudatusi. Mingi hetk oli tudengi

koostatud töölauarakendus paremal tasemel kui varasem Visual Basicu programm ning siis hakati laboris kasutama ainult tudengi arendatavat programmi.

Kokkuvõte

Töö eesmärgiks oli luua kapillaarelektroforeesis kasutatav töölauarakendus, mis pidi välja vahetama seni kasutatava, defektse ja mittetäieliku rakenduse. See rakendus pidi info kätte saama Arduino mikrokontrolleri peal jooksvalt kapillaarelektroforeesi seadmelt, infot töötlema, kuvama signaali graafikul ning peale katse lõppu salvestama katse tulemused tekstifailidesse, graafik pildifaili ning internetiühenduse olemasolul saatma andmed ka REST API kaudu andmebaasi.



Joonis 17. Loodud töölauarakendus

Töölauarakenduses (Joonis 17) on lõppkasutajal võimalik valida erinevate katse seadistuste vahel: katse tegija nimi, kapillaari mõõtmed, testitavad analüüdid, testitava analüüdi sisestamismeetod, testimiskeskkond, elektrivoolu sagedus, kõrgepinge ning selle protsentuaalne kasutus, katse kestvus ja ka graafikul nähtav ajaline osa.

Graafik on reaalajas muutuv ning sõltuvalt ajalise osa valikust on võimalik näha erinevat osa graafikust (näiteks viimase 1 minuti jooksul nähtav osa graafikust).

Samuti on võimalik seadistada, kas on soov kasutada taimerit või mitte. Taimeri kasutamisel lõpetab töölauarakendus peale aja läbi saamist ise töö, salvestab andmed ja lülitab kõrgepinge kapillaarelektroforeesi seadmes välja.

Kui internetiühendus on olemas, siis andmed saadetakse JSON formaadis Spring Booti peale ehitatud REST API rakendusele, mis vastavalt JSON-is olevale võtmetele andmed tabelitesse salvestab. Peale salvestamist saadetakse uued andmed tagasi töölauarakendusele. Töölauarakenduses olev informatsioon uuendatakse nii rakenduse käivitamisel kui ka peale andmebaasi info saatmist. Uuendamist vajavad analüütide nimekiri, tausta-elektrolüütide nimekiri, meetodite nimekiri kui ka maatriksite nimekiri. Seda põhjusel, et keemik võis rakenduses luua eelmainitud nimekirja uue nime, mida andmebaasis ei ole. Seega et see nimi oleks ka edaspidi töölauarakenduses, on uuendamine vajalik.

Loodud süsteemi on arvatavasti plaanis tulevikus täiustada, kuna kaasjuhendaja Jelena Gorbatšoval on uusi ideid, mida ta ideaalis sooviks näha rakenduses, aga mis bakalaureusetöö raamidesse ei jäänud. Kuna see rakendus on nii Jelena Gorbatšova kui ka tema bakalaureuse töö tudengi poolt sagedasti kasutatav, siis ilmneda võib olukordi, mis vajavad täiendamist.

Kuna süsteemi arendamine oli seotud kapillaarelektroforeesiga, siis sai lõputöö kirjutaja palju uusi teadmisi analüütilisest keemiast, sai ise kätt proovida laborikatsetes ning õppis omavahel ühendama Arduino mikrokontrollerit ja Java rakendust. Kuna varasemalt kasutatav rakendus oli kirjutatud Visual Basicus, siis tekkis kogemus ka Visual Basicu keelega.

Kasutatud kirjandus

[1] „Wikipedia,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Kapillaarelektroforees. [Kasutatud 16 Mai 2019].

[2] „LibreTexts,“ 11 May 2019. [Võrgumaterjal]. Available: https://chem.libretexts.org/Bookshelves/Analytical_Chemistry/Supplemental_Modules_(Analytical_Chemistry)/Instrumental_Analysis/Capillary_Electrophoresis. [Kasutatud 16 Mai 2019].

[3] „RXTX Wiki,“ [Võrgumaterjal]. Available: http://rxtx.qbang.org/wiki/index.php/FAQ. [Kasutatud 16 Mai 2019].

[4] „stack overflow,“ [Võrgumaterjal]. Available: https://stackoverflow.com/questions/12317576/stable-alternative-to-rxtx. [Kasutatud 16 Mai 2019].

[5] „libGDX,“ [Võrgumaterjal]. Available: https://libgdx.badlogicgames.com/features.html. [Kasutatud 16 Mai 2019].

[6] „Wikibooks,“ [Võrgumaterjal]. Available: https://en.wikibooks.org/wiki/Java_Swings/Java_Swing#Swing_GUI_Components. [Kasutatud 16 Mai 2019].

[7] „GitHub,“ [Võrgumaterjal]. Available: https://github.com/controlsfx/controlsfx. [Kasutatud 16 Mai 2019].

[8] „JFreeChart,“ [Võrgumaterjal]. Available: http://www.jfree.org/jfreechart/. [Kasutatud 16 Mai 2019].

[9] „JFree,“ [Võrgumaterjal]. Available: http://www.jfree.org/jcommon/. [Kasutatud 16 Mai 2019].

[10] „GitHub,“ [Võrgumaterjal]. Available: https://github.com/google/gson. [Kasutatud 16 Mai 2019].

Lisa 1 – SQL kood andmebaasi tabelite ja nendega seonduva loomiseks

```
DROP DATABASE IF EXISTS vee_andmebaas;
CREATE DATABASE vee_andmebaas;

--
-- Table structure for table 'analytes'
--

DROP TABLE IF EXISTS analytes;
CREATE TABLE analytes (
  analyte_id SERIAL NOT NULL,
  analyte_name VARCHAR(255),
  PRIMARY KEY (analyte_id)
);

--
-- Dumping data for table 'analytes'
--

INSERT INTO analytes (analyte_id, analyte_name) VALUES (1, 'Na');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (2, 'K');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (3, 'Li');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (4, 'NH4');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (5, 'Ba');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (6, 'Mg');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (7, 'Mn');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (8, 'Fe2+');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (9, 'Br');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (10, 'Cl');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (11, 'SO4');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (12, 'SO3');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (13, 'NO3');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (14, 'NO2');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (15, 'F');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (16, 'PO4');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (17, 'Thiamin');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (18, 'Nicotinic acid');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (19, 'Nicotinamid');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (20, 'Pyridoxid');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (21, 'Ascorbic acid');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (22, 'GABA');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (23, 'Arginin');
```

```
INSERT INTO analytes (analyte_id, analyte_name) VALUES (24, 'Lysin');
INSERT INTO analytes (analyte_id, analyte_name) VALUES (25, 'Phenylalanin');
-- 25 records
SELECT setval('analytes_analyte_id_seq', MAX(analyte_id)) FROM analytes;

--
-- Table structure for table 'matrixes'
--

DROP TABLE IF EXISTS matrixes;
CREATE TABLE matrixes (
  matrix_id SERIAL NOT NULL,
  matrix_name VARCHAR(255),
  PRIMARY KEY (matrix_id)
);

--
-- Dumping data for table 'matrixes'
--

INSERT INTO matrixes (matrix_id, matrix_name) VALUES (1, 'soil');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (2, 'sand');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (3, 'rocks');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (4, 'tap water');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (5, 'rain water');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (6, 'spring water');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (7, 'aquarium water');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (8, 'sea water');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (9, 'canalization water');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (10, 'saliva');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (11, 'blood');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (12, 'urine');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (13, 'plant extract');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (14, 'juice');
INSERT INTO matrixes (matrix_id, matrix_name) VALUES (15, 'drink');
-- 15 records
SELECT setval('matrixes_matrix_id_seq', MAX(matrix_id)) FROM matrixes;

--
-- Table structure for table 'bges'
--

DROP TABLE IF EXISTS bges;
CREATE TABLE bges (
  bge_id SERIAL NOT NULL,
  bge_name VARCHAR(255),
  PRIMARY KEY (bge_id)
);

--
```

```
-- Dumping data for table 'bges'
--

INSERT INTO bges (bge_id, bge_name) VALUES (1, 'Acetic acid 1M');
INSERT INTO bges (bge_id, bge_name) VALUES (2, 'Acetic acid 2M');
INSERT INTO bges (bge_id, bge_name) VALUES (3, 'Acetic acid 3M');
INSERT INTO bges (bge_id, bge_name) VALUES (4, 'Acetic acid 6M');
INSERT INTO bges (bge_id, bge_name) VALUES (5, 'Mes');
INSERT INTO bges (bge_id, bge_name) VALUES (6, 'His');
-- 6 records
SELECT setval('bges_bge_id_seq', MAX(bge_id)) FROM bges;

--
-- Table structure for table 'user_classes'
--

DROP TABLE IF EXISTS user_classes;
CREATE TABLE user_classes (
  user_class_id SERIAL NOT NULL,
  user_class_name VARCHAR(255),
  PRIMARY KEY (user_class_id)
);

--
-- Dumping data for table 'user_classes'
--

INSERT INTO user_classes (user_class_id, user_class_name) VALUES (1,
'Administrators');
INSERT INTO user_classes (user_class_id, user_class_name) VALUES (2, 'Scientists');
INSERT INTO user_classes (user_class_id, user_class_name) VALUES (3, 'Regular
users');
-- 3 records
SELECT setval('user_classes_user_class_id_seq', MAX(user_class_id)) FROM
user_classes;

--
-- Table structure for table 'users'
--

DROP TABLE IF EXISTS users;
CREATE TABLE users (
  user_id SERIAL NOT NULL,
  user_name VARCHAR(255),
  user_class INTEGER DEFAULT 0,
  PRIMARY KEY (user_id)
);

--
-- Dumping data for table 'users'
```

```
--

INSERT INTO users (user_id, user_name, user_class) VALUES (1, 'Administrator', 1);
INSERT INTO users (user_id, user_name, user_class) VALUES (2, 'Scientist', 2);
INSERT INTO users (user_id, user_name, user_class) VALUES (3, 'Regular user', 3);
-- 3 records
SELECT setval('users_user_id_seq', MAX(user_id)) FROM users;

--
-- Table structure for table 'methods'
--

DROP TABLE IF EXISTS methods;
CREATE TABLE methods (
  method_id SERIAL NOT NULL,
  method_name VARCHAR(255),
  matrix_id INTEGER DEFAULT 0,
  capillary_inner INTEGER DEFAULT 0,
  capillary_outer INTEGER DEFAULT 0,
  capillary_total INTEGER DEFAULT 0,
  capillary_effective INTEGER DEFAULT 0,
  injection_method VARCHAR(255),
  injection_amount INTEGER DEFAULT 0,
  injection_unit VARCHAR(255),
  injection_duration INTEGER DEFAULT 0,
  frequency INTEGER DEFAULT 0,
  hv_value INTEGER DEFAULT 0,
  method_description VARCHAR(255),
  PRIMARY KEY (method_id)
);

--
-- Dumping data for table 'methods'
--
-- 0 records

SELECT setval('methods_method_id_seq', MAX(method_id)) FROM methods;
CREATE INDEX methods_capillary_effective_id ON methods (capillary_effective);
CREATE INDEX methods_capillary_id ON methods (capillary_inner);
CREATE INDEX methods_capillary_total_id ON methods (capillary_total);
CREATE INDEX methods_frequency_id ON methods (frequency);
CREATE INDEX methods_injection_method_id ON methods (injection_method);
CREATE INDEX methods_matrix_id ON methods (matrix_id);

--
-- Table structure for table 'analyte_measurements'
--

DROP TABLE IF EXISTS analyte_measurements;
```

```
CREATE TABLE analyte_measurements (
  analyte_measurement_id SERIAL NOT NULL,
  test_time VARCHAR(255),
  method_id INTEGER DEFAULT 0,
  analyte_id INTEGER DEFAULT 0,
  analyte_amount INTEGER DEFAULT 0,
  analyte_unit VARCHAR(255),
  PRIMARY KEY (analyte_measurement_id)
);

--
-- Dumping data for table 'analyte_measurements'
--
-- 0 records

SELECT setval('analyte_measurements_analyte_measurement_id_seq',
MAX(analyte_measurement_id)) FROM analyte_measurements;
CREATE INDEX analyte_measurements_analyte_id ON analyte_measurements
(analyte_id);
CREATE INDEX analyte_measurements_method_id ON analyte_measurements
(method_id);
CREATE INDEX analyte_measurements_test_id1 ON analyte_measurements
(test_time);

--
-- Table structure for table 'bge_measurements'
--

DROP TABLE IF EXISTS bge_measurements;
CREATE TABLE bge_measurements (
  bge_measurement_id SERIAL NOT NULL,
  test_time VARCHAR(255),
  method_id INTEGER DEFAULT 0,
  bge_id INTEGER DEFAULT 0,
  bge_amount INTEGER DEFAULT 0,
  bge_unit VARCHAR(255),
  PRIMARY KEY (bge_measurement_id)
);

--
-- Dumping data for table 'bge_measurements'
--
-- 0 records

SELECT setval('bge_measurements_bge_measurement_id_seq',
MAX(bge_measurement_id)) FROM bge_measurements;
CREATE INDEX bge_measurements_bge_id ON bge_measurements (bge_id);
CREATE INDEX bge_measurements_method_id ON bge_measurements (method_id);
CREATE INDEX bge_measurements_test_id1 ON bge_measurements (test_time);
```

```
--
-- Table structure for table 'measurements'
--

DROP TABLE IF EXISTS measurements;
CREATE TABLE measurements (
  measurement_id SERIAL NOT NULL,
  test_id INTEGER DEFAULT 0,
  measurement INTEGER DEFAULT 0,
  PRIMARY KEY (measurement_id)
);

--
-- Dumping data for table 'measurements'
--
-- 0 records

SELECT setval('measurements_measurement_id_seq', MAX(measurement_id)) FROM
measurements;
CREATE INDEX measurements_test_id ON measurements (test_id);

--
-- Table structure for table 'tests'
--

DROP TABLE IF EXISTS tests;
CREATE TABLE tests (
  test_id SERIAL NOT NULL,
  user_id INTEGER DEFAULT 0,
  method_id INTEGER DEFAULT 0,
  test_time VARCHAR(255),
  test_duration INTEGER DEFAULT 0,
  PRIMARY KEY (test_id)
);

--
-- Dumping data for table 'tests'
--
-- 0 records

SELECT setval('tests_test_id_seq', MAX(test_id)) FROM tests;
CREATE INDEX tests_method_id ON tests (method_id);
CREATE INDEX tests_teststest_time ON tests (test_time);
CREATE INDEX tests_user_id ON tests (user_id);
```