TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Henri-Martin Jaakson

Toitumisnõustaja veebirakendus

Bakalaureusetöö (9 EAP)

Juhendaja: Ahti Peder, PhD

Tartu 2018

Toitumisnõustaja veebirakendus

Lühikokkuvõte:

Bakalaureusetöö eesmärgiks on luua veebirakendus, mis muudab toitumisnõustamise nii nõustaja kui ka kliendi jaoks mugavamaks ja lihtsamaks.

Rakendus on jaotatud kliendi ja nõustaja osadeks. Kliendi osa võimaldab kliendil sisestada tarbitud toiduainete andmeid, uuendada ja jälgida keha mõõtmeid ning näha kalorite ja makrotoitainete lubatud koguste piiride soovitusi. Nõustaja osa ülesandeks on vaatlusaluse kohta kasuliku teabe kuvamine, mille abil saaks lihtsalt jälgida kliendi progressi ning vajadusel teha tema toitumises muudatusi või täiendusi.

Töös tutvustatakse kasutatud tehnoloogiaid ning erinevate rakenduse osade arhitektuuri ja tööpõhimõtet.

Võtmesõnad:

Veebirakendus, klient–server arhitektuur, andmebaas.

CERCS: P175 Informaatika, süsteemiteooria

Nutrition Advisor Web Application

Abstract:

The aim of this bachelor's thesis is to create a web application that makes nutrition advising more convenient and easier for both the advisor and the client.

The application is divided into the client and advisor parts. The client part allows the user to enter consumed food information, update and monitor body dimensions, and see recommendations for calorie and macronutrient quantities. The advisor part is responsible for displaying useful data about clients to the advisor, and, if necessary, allow them to make changes or additions to client's diet.

The thesis includes an overview of the technologies used, introduces application's architecture and operation of the various parts.

Keywords:

Web application, client–server model, database.

CERCS: P175 Informatics, systems theory

Sisukord

1 Sissejuhatus **5**

2 Funktsionaalsed nõuded **6**
2.1 Kliendi osa 6
2.2 Nõustaja osa 7

3 Kasutatud tehnoloogiad **8**
3.1 Node.js 8
3.2 React 9
3.3 React-router 9
3.4 Styled-components 9
3.5 MobX 9
3.6 Chart.js 10
3.7 Express 10
3.8 JSON Web Token 10
3.9 Pg-promise 11
3.10 PostgreSQL 11

4 Arhitektuur **12**
4.1 Andmebaas 12
4.1.1 Tabelid 12
4.1.2 Andmete pärimine 13
4.2 Serveripoolne osa 14
4.2.1 Autentimine 15
4.2.2 Volitamine 15
4.2.3 Valideerimine 15
4.2.4 Andmete lugemine ja muutmine 16
4.2.5 Vastamine 16
4.3 Kasutajapoolne osa 17
4.3.1 Ehitus 17
4.3.2 Suhtlus serveriga 19
4.3.3 Mudelid 19
4.3.4 Oleku haldamine 20
4.3.5 Mitmekeelsus 20

5 Kokkuvõte **21**

Viidatud kirjandus **22**

Lisad **23**
I. Paigaldusjuhend 23
II. Litsents 24

1 Sissejuhatus

Üha rohkem inimesi üritab järgida tervislikku eluviisi ning selle üheks oluliseks osaks on toitumine. Kuna inimestel on vähe aega või neil pole piisavalt motivatsiooni ise oma toitumist jälgida ja parimaid toitumistavasid uurida, siis kasutavad nad nõustajate abi. Nõustajate ülesandeks on uurida nõustatavate toitumist ja elustiili ning anda selle põhjal soovitusi.

Töö autoriga võttis ühendust klient, kes tegeleb toitumisnõustamisega ning kelle nõustamise protsess oli ebamugav nii tema kui ka ta klientide jaoks. Iga nädala lõpus saatsid nõustaja kliendid e-posti teel keha mõõtmeid ning nädala jooksul tarbitud toiduainete nimekirju koos toitumisalase infoga, mille seas olid tarbitud kaloraaž ning süsivesikute, valkude ja rasvade kogused. Kliendid kasutasid erinevaid rakendusi toitumise jälgimiseks, kus pole võimalik andmeid jagada, ja seega pidid nad andmed enne saatmist välja kirjutama käsitsi.

Sellise süsteemi tõttu olid nõustajale saadetud andmed kuni nädal aega vanad ning nõustaja ei saanud õigeaegselt tagasisidet anda. Kuna iga klient saatis andmed erinevas formaadis, siis oli tarbitud koguseid ja kehamõõtmete muutust keeruline jälgida ja visualiseerida. Kehamassiindeksi ja päevaste lubatud piiride arvutamiseks pidi aga kasutama teist rakendust, täites iga kord arvutusteks vajalikke välju.

Bakalaureusetöö eesmärgiks on luua veebirakendus, mis muudab toitumisnõustamise nii nõustaja kui ka kliendi jaoks mugavamaks ja lihtsamaks lahendades toodud probleemid. Töös antakse ülevaade kasutatud tehnoloogiatest ja kirjeldatakse rakenduse tööd.

Töö teises peatükis tuuakse välja funktsionaalsed nõuded, kolmandas kirjeldatakse kasutatud tehnoloogiaid ning neljandas osas antakse ülevaade rakenduse arhitektuurist ja tööpõhimõttest.

2 Funktsionaalsed nõuded

Funktsionaalsed nõuded on loodud koostöös kliendiga sõltuvalt rakenduse ülesannetest. Nõuded on jagatud kliendi ja nõustaja osadeks. Kliendi osa ülesanneteks on võimaldada kasutajal sisestada tarbitud toiduainete andmeid, uuendada ja jälgida keha mõõtmeid ning näha kalorite ja makrotoitainete lubatud koguste piiride soovitusi. Nõustaja osa ülesanneteks on vaatlusaluse kohta tarbitud toiduainete ja kehamõõtmete andmete kuvamine ning nõustajal kliendi toitumises muudatuste tegemise võimaldamine.

2.1 Kliendi osa

1. Registreerimine ja sisse logimine.
   (a) Kasutaja peab saama registreeruda kehtiva registratsiooni koodi olemasolul.
   (b) Kasutaja peab saama sisse logida e-posti aadressi ja parooliga.
   (c) Kasutaja peab saama välja logida.
2. Tarbitud toiduainete lisamine.
   (a) Kasutaja peab saama luua hommiku, lõuna, õhtu või vahepealse söögikorra.
   (b) Kasutaja peab saama lisada söögi söögikorda.
   (c) Kasutaja peab saama kustutada söögi söögikorrast.
   (d) Kasutaja peab saama lisada toiduaine koos selle andmetega.
3. Kehamõõtmete lisamine ja haldamine.
   (a) Kasutaja peab saama lisada uusi suurusi.
   (b) Kasutaja peab saama uuendada suuruste väärtusi.
   (c) Kasutaja peab saama kustutada mõõtusid.
4. Mõõtmete muutuste jälgimine ja toitumise muutmine.
   (a) Nõustaja peab saama muuta kliendi kaloraaži muutu ning makrotoitainete osamäärasid.
   (b) Kasutaja peab saama näha kehamassiindeksit.
   (c) Kasutaja peab saama näha mõõtude väärtusi graafikul.

5. Tarbimisajaloo vaatamine.

   (a) Kasutaja peab saama näha viimase 4 nädala tarbimise ajalugu, mis sisaldab iga päeva kohta tarbitud kalorid, makrotoitainete (süsivesikud, valgud, rasvad) kogused ja nendele vastavad päevased normid.

   (b) Kasutaja peab saama näha viimase 4 nädala igal päeva tarbitud söökide nimekirja.

6. Sätete muutmine.

   (a) Kasutaja peab saama vahetada keelt eesti ja vene keele vahel.

   (b) Kasutaja peab saama muuta e-posti aadressit ja parooli.

   (c) Kasutaja peab saama sulgeda kontot.

2.2 Nõustaja osa

Nõustaja peab saama teha lisaks kliendi osa funktsionaalsustele järgnevat.

1. Klientide haldamine.

   (a) Nõustaja peab saama genereerida registratiooni koode, mille abil saavad kliendid registreeruda.

   (b) Nõustaja peab saama näha klientide nimekirja.

   (c) Nõustaja peab saama kliente eemaldada.

2. Kliendi mõõtmete muutuste jälgimine ja toitumise muutmine.

   (a) Nõustaja peab saama muuta kliendi kaloraaži muutu ning makrotoitainete osamäärasid.

   (b) Nõustaja peab saama näha kliendi kehamassiindeksit.

   (c) Nõustaja peab saama näha kliendi mõõtude väärtusi graafikul.

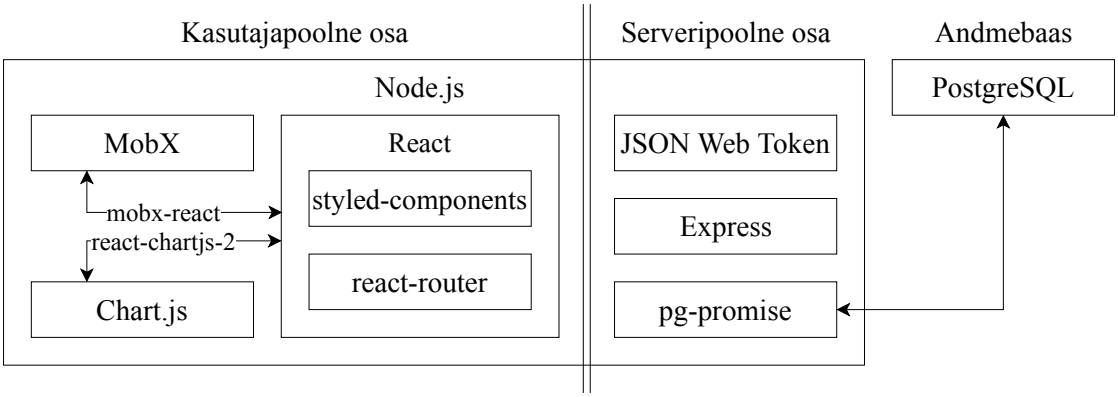
3. Kliendi tarbimisajaloo vaatamine.

   (a) Nõustaja peab saama näha kliendi viimase 4 nädala tarbimise ajalugu, mis sisaldab iga päeva kohta tarbitud kalorid, makrotoitainete (süsivesikud, valgud, rasvad) kogused ja nendele vastavad päevased normid.

   (b) Nõustaja peab saama näha kliendi viimase 4 nädala iga päeva tarbitud söökide nimekirja.

3 Kasutatud tehnoloogiad

Kasutatud tehnoloogiad nagu ka veebirakendus on jaotatud kolme osa vahel (joonis 1). Kasutajapoolne osa kasutab teeki React kasutajaliidese loomiseks ning teeki MobX selle olekuhalduseks. Serveripoolne osa põhineb JavaScripti käitlusmootoril Node.js ning kasutab veebirakenduste loomise raamistiku Express. Andmebaasiks on relatsioonilise andmebaasi haldamise süsteem PostgreSQL.



Joonis 1. Peamised kasutatud tehnoloogiad ja nende seosed.

3.1 Node.js

Node.js on programmeerimiskeele JavaScript käitusmootor, mis põhineb Chrome V8 mootoril ja mis kasutab mitteblokeerivat sündmusjuhitavat (ingl *event driven*) arhitektuuri [1]. Sündmuse toimumisel, näiteks serveriga ühendamisel, kutsutakse välja asünkroonne tagasihelistusfunktsioon (ingl *asynchronous callback function*), mille tõttu saab käidelda mitu sündmust korraga [2]. Käitlusmootorit Node.js kasutatakse peamiselt veebirakenduste loomiseks.

Node.js kasutab paketihaldussüsteemi npm, mis sisaldab ligikaudu 600 000 avatud lähtekoodiga paketti [3]. Süsteemi npm ülesandeks on pakettide vaheliste sõltuvuste haldamine. See on saavutatud projekti juurkaustas asuva faili *packages.json* abil, milles on defineeritud kõik paketid ja nende versioonid, millest tarkvara sõltub. Selle faili alusel installib npm kõik vajalikud tehnoloogiad.

Rakenduse kasutajaliides ja serveri tarkvaras on kasutuses Node.js kui ka npm. Seega on mõlemad osad programmeeritud samas keeles, mille tõttu saab osa koodist taaskasutada ning ei pea rakenduse erinevaid osasid arendama erinevates keeltes.

3.2 React

React on JavaScripti teek kasutajaliideste loomiseks. Kasutajaliides on jagatud taaskasutatavateks ühe põhilise ülesandega iseseisvateks osadeks - komponentideks. Komponendiks võib olla nupp, kalender ridadega, veergudega ja kuupäeva vahetajaga või terve vaade. Komponendid võivad koosneda HTML elementidest ja teistest komponentidest [7].

Komponentide struktuur ning funktsionaalsus on defineeritud JavaScripti laiendiga .js failides. Struktuur on loodud HTML-iga sarnases keeles JSX, mis muudab defineeritud struktuuri komponendi loomise funktsiooni väljakutseks [8]. Kuna komponendid on JavaScriptis defineeritud kui klassi isendid, võib neid salvestada muutujatesse, kasutada nende klassimeetodeid ning luua ja tagastada neid teistes funktsioonides.

3.3 React-router

React-router on teek, millega saab luua kasutajaliidese vaadete vahel navigeerimise. Selle teegiga saab defineerida kahte tüüpi komponente: lingid (ingl *link*) ja marsruudid (ingl *route*) .

Link on komponent, millele vajutades muutub brauseri internetiaadress ning lisandub kirje sirvimise ajalukku lehe värskendamiseta. Rakenduses on kõik navigeerimisriba elemendid lingid. Marsruudi komponent aga kuvab mingit komponenti olenevalt praegusest aadressist. Kui aadress vastab sellele argumendina antud teele, siis marsruut kuvab argumendina antud komponendi.

3.4 Styled-components

Styled-components on teek, mis lihtsustab Reacti komponentide kujundamist. Ilma teekideta Reactis tuleb komponendi kujundamiseks defineerida selle klassinimi (ingl *classname*) ja mujal CSS failis klassinimele viidates kirjeldada selle kujundust [10]. Teek styled-components võimaldab luua komponendi koos selle kujundusega lähtefailis, mille tõttu komponent, selle kujundus ja funktsionaalsus võib olla ühes ja samas failis.

3.5 MobX

MobX on olekuhalduse teek. Programmi olek on kõikide muutujate kogum, mis on vajalik programmi tööks. Näiteks kalendri komponent peab teadma valitud kuupäeva muutuja väärtust. Kui rakendus muudab selle väärtust ilma seda kasutava komponendi uuendamiseta, siis programmi olek ja kasutajale näidatavad andmed

võivad mitte olla enam kooskõlas. Teek MobX püüab teha sellise olukorra tekkimist võimatuks [9].

Süsteemi MobX kasutavat olekut võib võrrelda tabelarvutussüsteemiga, kus kõik oleku muutujad on lahtrid. Lahtrile, mille väärtus sõltub teistest lahtritest, vastab teegi MobX arvutatud (ingl *computed*) tüüpi muutuja. Lahtrile, millest sõltub arvutatud lahter, on jälgitavat (ingl *observable*) tüüpi, kuna selle muutmisel muutub ka arvutatud lahter ja seega tuleb selle muutumist jälgida. Jälgitava lahtri muutmisele vastab *action*, mis on olekut muutev funktsioon.

MobX ja React on seotud teegiga mobx-react. Sellega saab defineerida jälgijaid (ingl *observer*). Jälgija on komponent, mis joonistab ennast ümber siis, kui mingit tema poolt kasutatavat jälgitavat muutujat muudetakse.

3.6 Chart.js

Chart.js on graafikute loomise teek. Sellega saab luua nii joon-, tulp-, sektor- ja muid diagramme kui ka mitut tüüpi kombineerivaid graafikuid. Kuna Chart.js joonised ei põhine Reacti komponentidel, siis kasutajaliides kasutab teegi react-chartjs-2 poolt defineeritud graafikutele vastavad komponente.

3.7 Express

Express on käitusmootoril Node.js põhinev raamisitk serveri rakenduste loomiseks. See lihtsustab serveritarkvara marsruutide (ingl *route*) ja vahevarade (ingl *middleware*) loomist.

Marsruut seob interneti aadressi ja päringu meetodi (ingl *request method*) seda käitlevate vahevaradega [4]. Päringu meetodiks võib olla mistahes HTTP päringu meetod, näiteks GET, POST, DELETE ja teised [4].

Vahevara on funktsioon, mis muudab päringut ja selle vastust ning saadab nad vajadusel edasi järgmisele vahevarale [5]. Näiteks üks vahevara võib vormindada päringu väljad, peale mida järgmiste vahevarade sisendiks on juba vormindatud andmed.

Express on laialdaselt kasutuses (ligi 4.5 miljonit allalaadimist nädalas [6]) ning seega on korduma kippuvatele probleemidele olemas lahendused nii koodina kui ka teekidena. See sisaldab väikest arvu uusi klasse ja funktsioone, mille tõttu on selle kasutuselevõtt kerge.

3.8 JSON Web Token

JSON Web Token (JWT) on andmevahetusvormingul JSON põhinev avatud standard lubade (ingl *token*) loomiseks, mis võivad sisaldada tõendeid (ingl *claim*) [11]. Näiteks võib server genereerida loa tõendiga, et kasutaja identifikaator (ID) on 2

ning saata selle kliendile. Klient võib pärast kasutada seda luba et tõestada, et ID ongi 2.

3.9 Pg-promise

Pg-promise on node-postgres põhinev teek andmebaasiga ühenduste loomiseks ja haldamiseks ning päringute tegemiseks. Teek muudab andmete pärimise funktsioonid JavaScripti Promise objektideks.

Promise isend esindab mingi asünkroonse funktsiooni tulevikus lõpuleviimist ja selle tagastusväärtust [12]. Asünkroonseks funktsiooniks võib olla andmebaasile päringu tegemine, mis tagastab päritud andmed. Promise tagastusväärtuseks võib olla ka teine promise objekt, mis tähendab, et üks asünkroonne funksioon võib kasutada teise funksiooni poolt tagastatud andmeid. Promise objektide kasutuse peamiseks eeliseks on see, et see vähendab programmikoodi treppimise sügavust, millega kaasneb programmi parem loetavus ja haldus.

3.10 PostgreSQL

PostgreSQL on vabavaraline relatsioonilise andmebaasi haldamise süsteem. Selle eesmärgiks on andmete turvaline hoiustamine ning teiste rakenduste poolt tehtud päringute alusel andmete tagastamine.

PostgreSQL võimaldab tagastada andmeid andmevahetusvormingu JSON formaadis, mille tõttu on võimalik tagastada hierarhilise struktuuriga andmeid. Näiteks on võimalik tagastada ühe päringuga kõik toidukorrad ning nende ajal tarbitud söögid. Seega saab vähendada päringute arvu nii kliendipoolse osa ja serveri vahel kui ka serveri ja andmebaasi vahel.

4 Arhitektuur

Veebirakendus on jaotatud kolme iseseisva programmiüksuse vahel: andmebaas, serveripoolne osa ja kasutajapoolne osa. Kasutajapoolne osa teeb serverile päringuid, peale mida serveritarkvara kontrollib päringu õigsust, loeb ja muudab andmebaasi andmeid ning tagastab kasutajale vastuse.

4.1 Andmebaas

Rakenduses on andmebaasi ülesandeks andmete hoiustamine ning serveritarkvara päringute alusel andmete muutmine ja tagastamine. Kuna andmebaas on relatsiooniline, siis kasutatakse tabeleid andmete struktureerimiseks. Andmebaasis on ka defineeritud keerulisemad andmete lisamise ja tagastamise funktsioonid.

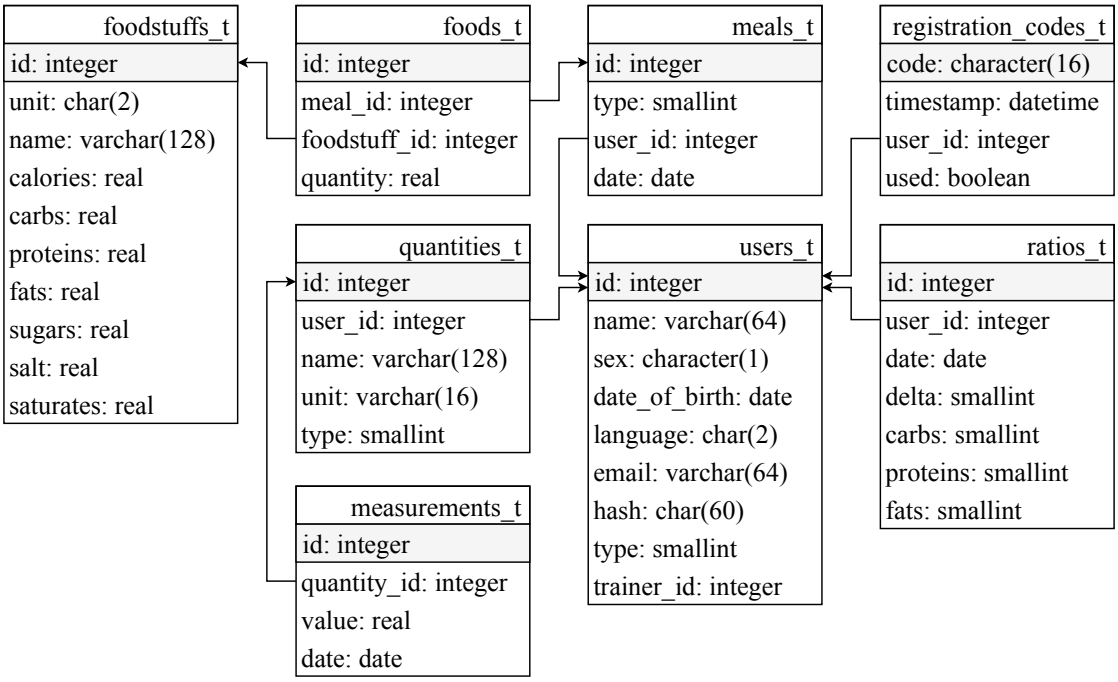
4.1.1 Tabelid

Andmebaasis on struktuurpäringukeelega SQL defineeritud hulk tabeleid (tabel 1). Kõik tabelid lõppevad sufiksiga `_t`, mille tõttu saab päringutes eristada tabeleid teistest tabeli sarnastest objektidest, peamiselt vaadetest ja ajutistest tabelitest. Üldiselt leidub igas tabelis ID veerg (joonis 2), mille abil saab kliendiosa serverile üksüheselt öelda, millise veeruga seotud andmeid tuleb muuta ja tagastada.

Tabel 1. Andmebaasi tabelid.

Tabel	**Kirjeldus**
users_t	Kasutajate andmete tabel. Sisaldab e-posti aadressi ja parooli väljasid autentimiseks ning üldiseid kasutaja andmeid, näiteks nimi, sünnikuupäev ja sugu.
foodstuffs_t	Toiduainele vastav tabel. Sisaldab toiduaine toitumisalast teadet - kaloraaži ja makrotoitainete koguseid ühiku kohta.
meals_t	Toidukorrale vastav tabel. Sisaldab andmeid tüübi, näiteks hommikusöök, ning kuupäeva kohta.
foods_t	Söögi tabel. Vastab mingile tarbitud toiduaine kogusele toidukorras. Näiteks kahe õuna söömine 21.03.2018 hommikusöögiks.
quantities_t	Suurusele vastav tabel. Igale kasutajale luuakse hulk vaikimisi suuruseid, mille seas on mass, pikkus, aktiivsus ja erinevad keha ümbermõõdud.
measurements_t	Vastab suususe väärtusele kindlal päeval.
ratios_t	Vastab kaloraaži muudule ja makrotoitainete osamääradele kindlal päeval.

Tabel	Kirjeldus
registration_codes_t	Registratsiooni koodide tabel. Registreerimisel kontrollitakse, et kood leidub ning kehtib. Uuele kasutajale seatakse nõustajaks registratsiooni koodi veerus olev kasutaja väärtus.



Joonis 2. Andmebaasi tabelid ja nende veergude seosed.

4.1.2 Andmete pärimine

Andmete lisamine ja pärimine on suuresti teostatud andmebaasis defineeritud funktsioonide abil. Peamiseks põhjuseks on, et mõningate tabelite puhul tuleb ridade lisamisel teha eelkontrolli või väärtustada teiste päringute abil andmete sisestamisel kasutatavaid muutujaid. Teiseks põhjuseks on see, et kõik andmete lisamise funktsioonid tagastavad sisestatud andmed.

Andmete pärimise funktsioonid aga võivad tagastada mitme tabeli andmed ühe päringuga. Näiteks kasutaja kindla päeva andmete päring tagastab kõikide söögikordade ja nende söökide, kliendi tähtsate suuruste (mass, pikkus ja aktiivsus) ning toitumise parameetrite andmed. Sellisel tagastatakse rida, mille veergudeks on kolm JSON objekti.

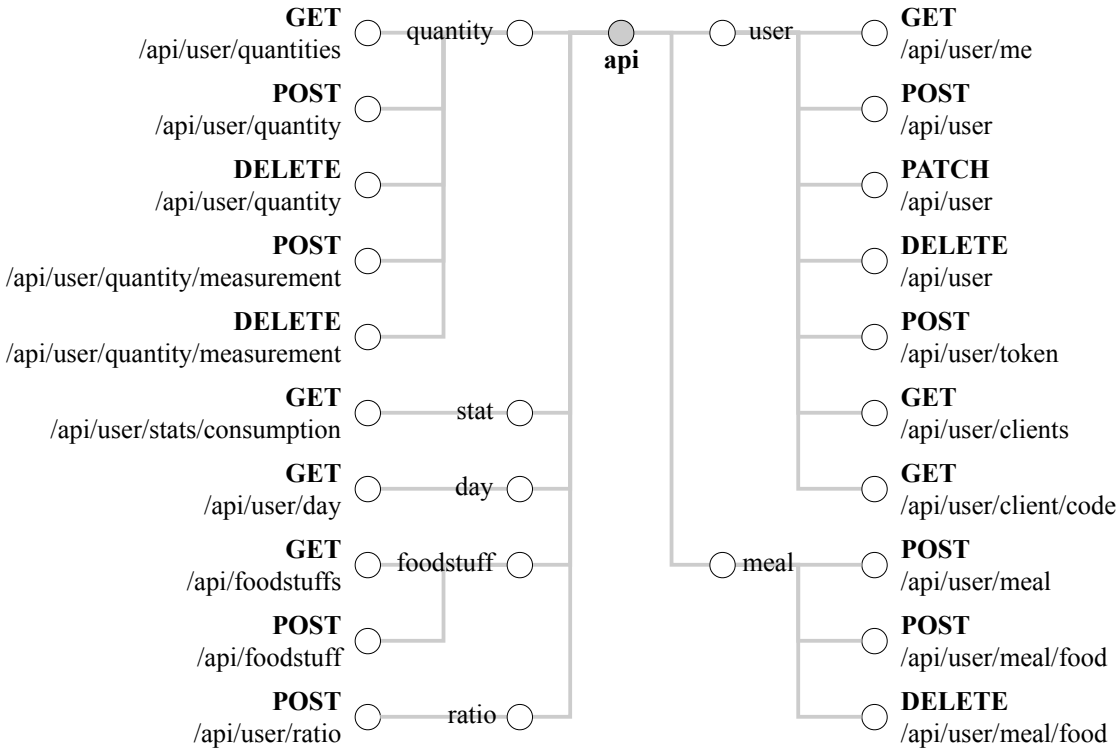
Kõik ülejäänud päringud, sealhulgas kustutamise päringud, on defineeritud serveritarkvaras.

4.2 Serveripoolne osa

Serveripoolseks osaks on kasutajaliidese ja andmete vaheline ühendus. See on eraldiseisev rakendus, mis olenevalt kliendipoolse osa päringutest lisab, muudab, kustutab või tagastab andmebaasist andmeid.

Serveritarkvaras on defineeritud hulk marsruute (ptk 3.7). Marsruudi aadress on hierarhiline ning püüab imiteerida andmebaasi tabelite seoseid. Päringu meetod oleneb serveri tegevustest sellele vastamisel. Üldjuhul GET päringu puhul loeb tarkvara andmebaasist andmed ilma neid muutmata ning tagastab kättesaadud andmed. POST päringu puhul lisab server andmebaasi andmeid ning tagastab äsja lisatud andmed. DELETE meetod aga kustutab andmeid, vastuseks on tõeväärtus `true`. Marsruut meetodiga POST ja aadressiga `/api/user/quantities/measurement` lisab mingi kasutaja mingile suurusele uue mõõtme väärtuse. Millisele kasutaja suurusele lisatakse väärtus, oleneb päringu väljadest. Kõik marsruudid on grupeeritud ruuteritesse (joonis 3).

Router on Expressi klass, mille abil saab marsruute grupeerida. Serveritarkvaras on iga ruuter koos selle marsruutidega defineeritud erinevas failis. Peafailis ühendatakse ruutereid serveri isendiga.



Joonis 3. Serveri ruuterid ning nende marsruudid.

Rakenduse serveripoolse osa ülesanneteks iga päringu puhul on:

1. kasutaja tuvastamine;
2. andmetele ligipääsu kontrollimine;
3. päringu väljade õigsuse kontrollimine;
4. andmebaasi andmete muutmine;
5. vastuse tagastamine.

Kõik ülesanded on defineeritud vähemalt ühe Expressi vahevara (ptk 3.7) abil ning need kutsutakse välja kindlale marsruudile päringu tegemisel.

4.2.1 Autentimine

Sisselogimisel saadab kasutaja serverile e-posti aadressi ja parooli. Serveritarkvara kontrollib nende õigsust ja saadab tagasi JWT (ptk 3.8) uue loa tõendiga, mis määrab kasutaja identifikaatori (ID). Uus luba tagastatakse ka uue kasutaja registreerumisel. Järgnevad kliendi päringud sisaldavad genereeritud luba, mille abil saab server kliendi ID järgi kindlaks teha.

Kui andmete ligipääs nõuab autentimist, aga päring ei sisalda luba või on luba aegunud, tagastab server HTTP 401 *Unauthorized* vea.

4.2.2 Volitamine

Andmetele saavad ligi neid andmeid omav kasutaja või selle kasutaja nõustaja. Näiteks kui klient pärib kõiki tema poolt tarbitud sööke, peab andmebaasist tagastama kõik söögid, mis kuuluvad selle kasutaja poolt loodud söögikordadesse. Seega peavad suurem osa päringutest sisaldama ka kasutaja ID välja. Erinevalt JWT loas olevast kasutaja ID-st, mida kasutatakse ainult ligipääsu kontrollimiseks, kasutatakse just päringus olevat ID välja andmete muutmiseks. Kui klient pärib andmeid, millele ligipääsu tal ei ole, tagastab server HTTP 403 *Forbidden* vea.

4.2.3 Valideerimine

Valideerimine on andmete kontrollimine eesmärgiga, et nad on korrektsed ja õiges vormingus. Rakendus valideerib kõiki päringu väljasid, kasutades teeki express-validator, mis defineerib hulk validaatoreid vahevaradena. Nende seast kasutab serveripoolne osa peamiselt välja tüübi, sõne pikkuse ning e-posti aadressi vormingu õigsuse kontrolle.

4.2.4 Andmete lugemine ja muutmine

Andmehalduse eest vastutavad mudelid ja kontrollerid. Mudeliteks on andmebaasi tabelite põhjal grupeeritud funktsioonide grupid. Nende funktsioonide ülesandeks on selle tabeliga seotud andmete pärimine struktuurpäringukeeles SQL. Näiteks kasutaja mudelis on defineeritud kasutajate lisamise, muutmise ja kustutamise, kõikide kasutaja klientide andmete tagastamise ning muude kasutajaga seotud tegevuste päringud.

Kontrolleriks on Expressi vahevara (ptk 3.7), mis loeb ja muudab andmebaasi andmeid mudelite abil ning tagastab pärijale mingi vastuse. Vahevara kasutab juba eelnevalt valideeritud päringu väljasid mudelite funktsioonide argumentidena. Üks kontroller võib välja kutsuda mitu erinevat funktsiooni erinevatest mudelitest.

Funktsionaalsuse jaotamine mudelite ja kontrollerite vahel eraldab SQL päringud ülejäänud koodist ning võimaldab taaskasutada päringuid teistes kontrollerites.

4.2.5 Vastamine

Server tagastab vastusena andmevahetusvormingu JSON objekti, millel on väli `data` või `error` olenevalt sellest, kas tekkis viga või mitte.

Välja `data` väärtuseks on üldjuhul päritud andmed. See võib olla üks väärtus, andmevahetusvormingu JSON objekt või nende loend. Seevastu välja `error` väärtuseks on objekt kuni kolme väljaga: HTTP vea kood, selle inimloetav alternatiiv ja vigaste väljade nimekiri, kui viga tekkis väljade õigsuse kontrolli ajal.

Sellise formaadi tõttu saab kasutajapoolse osa programm selgelt kontrollida, kas päring õnnestus või mitte. Ebaõnnestumise korral on võimalik järeldada, milles viga seisnes.

4.3 Kasutajapoolne osa

Kasutajapoolse osa ülesandeks on kasutajalt sisendi saamine ning selle alusel serveripoolsele osale päringu tegemine ning tulemuse näitamine. Sisendiks võib olla kustutamise nupule vajutamine, vormi väljade täitmine ja „Salvesta“ nupule vajutamine või mingi aadressiga lehele navigeerimine. Seega peab kasutajaliides võimaldama kasutajal mugavalt andmeid sisestada ja näitama arusaadavalt vajalikke andmeid.

4.3.1 Ehitus

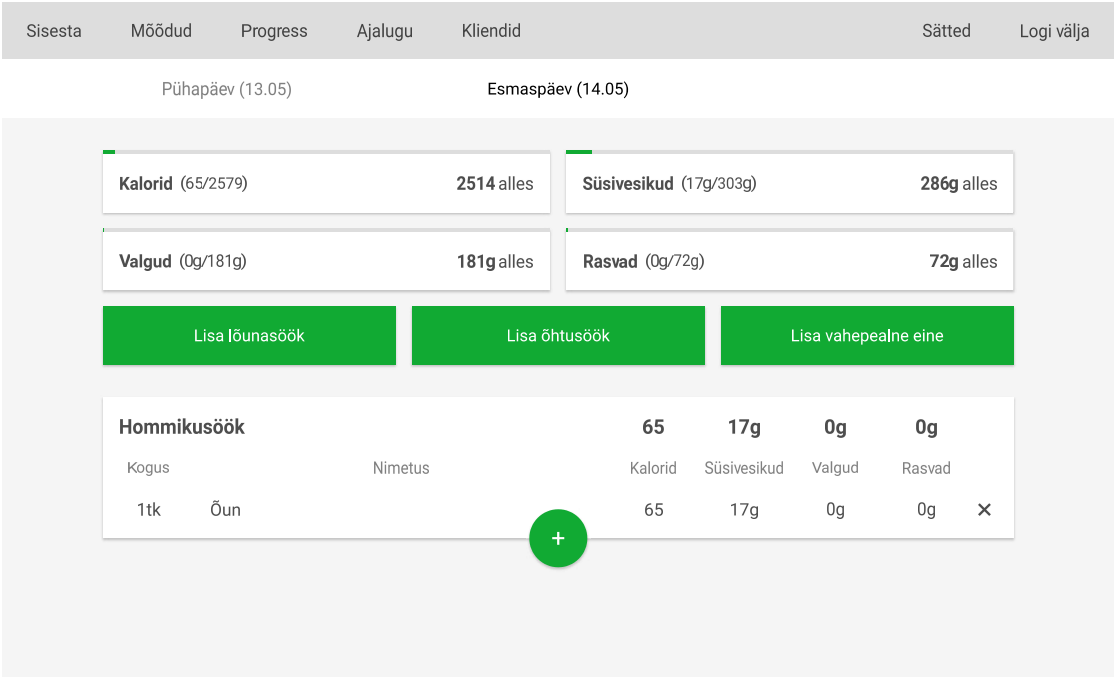
Kasutajaliides on jagatud kahte tüüpi komponentideks: stseenid ja lihtkomponendid. Stseen on osalehele, näiteks registratsiooni vormile või kliendihaldusele, vastav komponent. Stseeni, erinevalt komponendist, kuvatakse ainult siis, kui brauseri aadressiks on sellele vastav internetiaadress. Kõik stseenid on argumentideks teegi react-router marsruudi komponentidele (ptk 3.3), omavad unikaalset aadressit ja täidavad ühte kindlat ülesannet (tabel 2). Selle tõttu aadressi muutmisel muutub ka kuvatav stseen.

Tabel 2. Kasutajapoolse osa stseenid.

Stseen	**Aadress**	**Ülesanne**
Clients	`/clients`	Nõustaja kliendihaldamise stseen. Saab näha kliendi andmeid, mõõtude ja tarbimise ajalugu ning muuta toitumist.
Counter	`/counter`	Toiduainete sisestamise stseen (joonis 4). Saab lisada toidukordi ning toidukordadesse tarbitud toiduaineid. Näitab päeva jooksul tarbitud kaloraaži ja makrotoitainete koguseid ning vastavaid päevaseid norme.
History	`/history`	Tarbimise ajaloo stseen. Näitab viimase nelja nädala tarbitud koguste graafikuid, millel on kujutatud ka päevaste normide joondiagrammid. Tulbale vajutades näitab rakendus sellel päeval tarbitud toiduaineid.
Home	`/`	Selle stseeni ülesandeks on suunata kasutaja ümber õigesse stseeni. Kui kasutaja pole sisse logitud, on selleks Login stseen, vastasel juhul Counter.
Login	`/login`	Sisselogimise vormiga stseen.

Stseen	Aadress	Ülesanne
Progress	`/progress`	Stseen, mis näitab kehamassiindeksit kui ka keha suuruste muutusi. Nende alusel võib muuta toitumise parameetreid (kaloraaži, selle muutu ja makrotoitainete osamäärasid).
Quantities	`/quantities`	Suuruste lisamise ja suuruste mõõtmete uuendamise stseen.
Register	`/register`	Registratsiooni stseen.
Settings	`/settings`	Kasutaja sätete muutmise stseen. Muuta saab keelt, e-posti aadressit ja parooli.

Iga stseen koosneb omakorda komponentidest. Kuna komponendid on iseseisvad, võivad mitu stseeni taaskasutada sama komponenti. Stseeni ülesandeks on ka vajalike andmete pärimine ning nende edastamine osakomponentidele. Näiteks toiduainete sisetamise stseen pärib valitud päeva kohta kõik vajalikud andmed ning edastab need söögikorrale vastavale komponendile ja tarbitud koguste edenemisribadele (joonis 4).



Joonis 4. Toiduainete sisestamise stseen.

4.3.2 Suhtlus serveriga

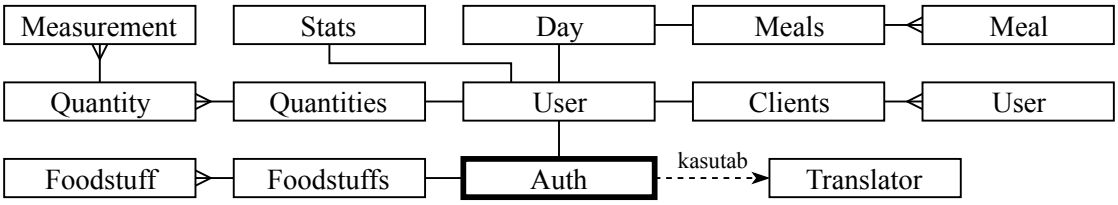
Tüüpilises veebirakenduses tagastab server HTML dokumendi koos andmetega, mille brauser kuvab kasutajale. Seda nimetatakse serveripoolseks renderdamiseks (ingl *server-side rendering*). Loodud rakendus aga kasutab kliendipoolset renderdamist, seega kasutajaliides loob ise HTML dokumendi serverilt päritud andmete abil.

Andmete kättesaamiseks peab server kontrollima kasutaja ligipääsu õigust. Selleks peab kasutajaliides iga päringu päisesse lisama serveri poolt genereeritud JWT (ptk 3.8) loa. Kui seda pole, peab kasutaja sisse logima, peale mida rakendus salvestab tagastatud loa.

Vajalike andmete kättesaamiseks teeb kasutajaliides teatud HTTP päringuid teatud marsruutidele. Näiteks sisselogitud kasutaja andmete puhul on see meetodi GET päring marsruudil `/api/user/me`. Tagastatud andmeid kasutatakse mudelite loomiseks.

4.3.3 Mudelid

Kasutajapoolsed mudelid on andmebaasi tabelite ja serveripoolsete mudelite hübriidid, mis tähendab, et nende ülesandeks on nii andmete struktureerimine kui ka kättesaamine. Mudelite isendiväljadeks on andmebaasi vastavate tabelite veerud ja teised seotud mudelid ning isendimeetoditeks lisaandmete pärimise ja õiges vormingus väljade andmete tagastamine.



Joonis 5. Kasutajapoolsed mudelid ja nende isendite seosed.

Rakendus kasutab mudeleid komponentide argumentidena. Tänu sellele saavad komponendid kasutada nii mudelite andmeid kui ka kõiki selle meetodeid. Kui komponent vajab mudeliga seotud lisaandmeid, siis kutsub komponent välja mudeli meetodi, mis pärib serverilt vajalikud andmed, loob vajadusel tagastatud andmete põhjal uue mudeli ja salvestab selle väärtuse isendimuutujasse. Seega päritakse serverilt ainult neid andmeid, mida tegelikult vaja on.

Nõustaja klientide andmed on kapseldatud User tüüpi mudelitesse ning omavad sama funktsionaalsust nagu nõustaja enda User mudeli isend. See tähendab, et kliendipoolne osa võib pärida samasuguseid andmeid klientide kohta nagu sisselogitud kasutaja kohta. Kuna igas päringus peab olema kasutaja ID väli, mille järgi

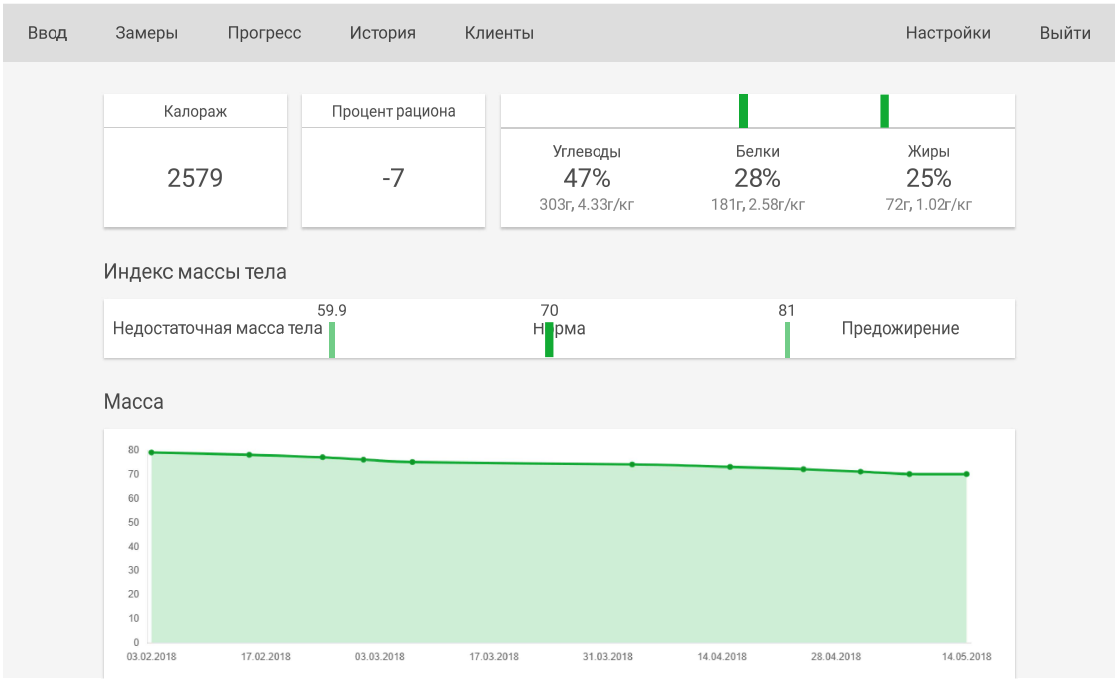
serveritarkvara andmeid loeb ja muudab, siis on selleks andmeid muutva mudeli lähima klassi User isendi ID väli.

4.3.4 Oleku haldamine

Päritud andmete salvestamisega ja mudelite väljade muutmisega tekib probleem, kuna mudeli andmed muutuvad, aga mudelit kasutavad komponendid võivad ikka kehtetuid andmeid kuvada. Sellepärast on kõik mudelite isendiväljad olekuhaldussüsteemi MobX poolt jälgitavad muutujad (ptk 3.5) ning kõik komponendid, mis kasutavad mudelit, on jälgija tüüpi (ptk 3.5). Kuna kõik mudelite isendid on omavahel seotud puusarnase struktuuriga (joonis 5), siis on kõik isendid jälgitavad muutujad.

4.3.5 Mitmekeelsus

Komponentide tõlkimine on teostatud mudeli Translator abil, milles on üks jälgitav väli. Selle väärtuseks on JavaScripti objekt, mille väljad on tõlgitud sõnad või fraasid, mida kasutavad komponendid tõlgitud teksti kuvamiseks (joonis 6). Keele vahetusel seatakse mudeli välja väärtuseks selline objekt, kus samadele väljadele vastavad teises keeles sõned. Kuna see objekt on jälgitav, siis selle muutmisel joonistavad kõik komponendid ennast ümber ja seega muutub kuvatav keel.



Joonis 6. Progressi stseen vene keeles.

5 Kokkuvõte

Bakalaureusetöö eesmärk oli luua veebirakendus, mis lihtsustaks toitumisnõustamist. Loodud tarkvara võimaldab klientidel sisestada tarbitud toiduaineid, lisada ja uuendada mõõtmeid ning jälgida oma toitumise ajalugu ja kehamõõtmete muutuste edenemist. Nõustaja saab lisaks lisada ja eemaldada kliente, jälgida vaatlusaluse toitumist ja mõõtmete muutusi ning vajadusel muuta toitumise parameetreid, mille seas on kaloraaž, selle muut ja maktrotoitainete osamäärad.

Töö esimeses osas toodi välja rakenduse nõuded, seejärel tutuvustati kasutatud tehnoloogiaid. Kolmandas osas kirjeldati erinevate rakenduse osade arhitektuuri ja tööpõhimõtet. Rakenduse paigaldusjuhend on kirjeldatud lisas I.

Edasiarendamise suundi on mitmeid. Neist suurimateks on lisada võimalus klientidel sisestada ja jälgida oma treenigute ning veetarbimise andmeid. Oluliseks edasiarenduseks oleks ka muuta rakendus mobiilisõbralikumaks, kas täiendades loodud veebirakendust või luues eraldiseisev programm nutitelefonidele, ning lisada sõnumite saatmise süsteem.

Viidatud kirjandus

[1] Node.js. https://nodejs.org/en/. (21.04.2018)

[2] About Node.js. https://nodejs.org/en/about/. (21.04.2018)

[3] What is npm? https://docs.npmjs.com/getting-started/what-is-npm. (21.04.2018)

[4] Routing. https://expressjs.com/en/guide/routing.html. (21.04.2018)

[5] Writing middleware for use in Express apps. https://expressjs.com/en/guide/writing-middleware.html. (21.04.2018)

[6] Express. https://www.npmjs.com/package/express. (13.05.2018)

[7] Components and Props. https://reactjs.org/docs/components-and-props.html. (25.04.2018)

[8] JSX In Depth. https://reactjs.org/docs/jsx-in-depth.html. (08.05.2018)

[9] Ten minute introduction to MobX and React. https://mobx.js.org/getting-started.html. (25.04.2018)

[10] Styling and CSS. https://reactjs.org/docs/faq-styling.html. (22.04.2018)

[11] Introduction to JSON Web Tokens. https://jwt.io/introduction/. (17.04.2018)

[12] Promise. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise. (06.05.2018)

Lisad

I. Paigaldusjuhend

1. Laadige alla ja installige: Node.js (vähemalt v9.8.0) ja PostgreSQL (vähemalt 10.0).
2. Looge uus PostgreSQL andmebaas.
3. Käivitage loodud andmebaasis fail `/api/sql/database.sql`, mis loob vajalikud tabelid ja funktsioonid ning lisab nõustaja kasutaja jaoks registratisooni koodi.
4. Muudke faili `/api/src/config/index.js` nii, et selle väljadeks oleks õiged andmebaasi andmed.
5. Avage konsooliaken kaustas `/api` ja sisestage järmised käsud:
   (a) `npm install` - Laeb alla kõik vajalikud paketid;
   (b) `npm start` - Käivitab serveritarkvara.
6. Brauseris navigeerige aadressile `localhost:3001/register` ja looge uus kasutaja registreerimis koodiga `CREATETHEADVISOR`.

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Henri-Martin Jaakson**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

   Toitumisnõustaja veebirakendus

   mille juhendaja on Ahti Peder

   1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i 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 teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 14.05.2018