TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut
Võrgutarkvara õppetool

ITV40LT

TOITUMISPÄEVIK SPORTLASTELE

Bakalaureusetöö

Üliõpilane: Hans-Toomas Saarest
Üliõpilaskood: 123998IAPB
Juhendaja: Jaagup Irve

Tallinn 2015

Autorideklaratsioon

Käesolevaga kinnitan, et esitatud bakalaureusetöö on minu isikliku töö tulemus. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud. Antud tööd ei ole kusagil varem kaitsmisele esitatud.

……..... …….....

(kuupäev) *(allkiri)*

Annotatsioon

Töö eesmärgiks on võimaldada sportlastele pidada toitumispäevikut. Toitmispäeviku korrektseks pidamiseks peab kasutaja sisestama kõik oma päevased toidukorrad, mis võivad koosneda piiramata arvust toitudest. Kasutaja peab saama ka toidukorda salvestada, et seda hiljem uuesti kasutada. Lisaks peab süsteem võimaldama vaadata sisestatud toidukordasid, muuta salvestatud toidukordade nime ja kuvama graafikutena kasutaja erinevate keha parameetrite muutumist ajas.

Taolise rakenduse vajadus pärineb sihtgrupilt. Kuna kasutajatel on erinevad seadmed, on töö üheks probleemiks leida sobiv lahendus ja õiged tehnoloogiad kavandatava prototüübi realiseerimiseks. Realiseeritav prototüüp peab töötama lisaks arvutile kõigil mobiilsetel seadmetel. Teiseks on vaja kaardistada süsteemi nõuded ja kasutusjuhud. Nõuetest ja kasutusjuhtudest lähtuvalt tuleb kavandada rakenduse arhitektuur ja disain ning see realiseerida. Töös kirjeldatakse prototüübi arengut alates süsteemi nõuete kirjeldamisest kuni valminud prototüübi testimiseni.

Töö tulemusel valmib prototüüplahendus kavandatavast süsteemist. Kõige otstarbekamaks lahenduseks süsteemi realiseerimisel on teha veebirakendus, mis on kasutatav iga operatsioonisüsteemiga seadmes. Prototüüpi tuleb testida, et leida võimalikke arendusprotsessis tekkinud vigu. Prototüübil on mitmeid edasiarenduse võimalusi mugavama kasutajakogemuse saavutamiseks. Selles töös realiseeritakse rakenduse põhifunktsionaalsus, mis võimaldab kasutajal hakata pidama toitumispäevikut.

Töö on kirjutatud eesti keeles ja sisaldab teksti 43 leheküljel, 5 peatükki, 9 joonist ja 4 tabelit.

Abstract

The thesis provides opportunity especially for atheletes to count their daily calories by putting all their daily meals into nutrition diary. For getting correct results user should add meals he/she consumes in a day. One meal can contain unlimited number of foods. Before entering the first meal of the day user must enter his/hers body weight and round of stomach. Furthermore, there must be an opportunity for user to save the meal he/she just composed for later usage. Moreover, system must have ability to present user the history of his/hers meals, have ability to change the name of saved meal and, furthermore, give a graph that shows the variations of user's body parameters in time.

There is a need for such system in Estonia. Users have different, therefore, one subject is selecting the most appropriate solution and suitable techonologies when developing the system. The prototype must be usable on all mobile devices. Next, system requirements and use cases have to be clearly formed. Then, based on system requirements and use cases the architecutre and design of the system will be planned and put into practice. The thesis describes the evolution of the system from describing system requirements up to testing the prototype.

The result of this work will be a prototye of planned system. Programming web-based application is the most appropriate solution to comprehend the variety of different operating systems, especially on mobile devices. The prototype has to be tested to find possible bugs that have appeared during development. The prototype has several further development options for better user experience. The thesis provides base functionality for user to start filling his/hers nutrition diary.

The thesis is written in Estonian. Thesis consists of text on 43 pages including 5 chapters, 9 figures and 4 tables.

Jooniste nimekiri

Sisukord

Sissejuhatus

Aktiivsetele sportlastele on õigesti toitumine oluline, et teha keha võimete kohta maksimaalne sooritus. Õigesti toitumiseks on vaja teada päevast soovitavat energiakogust ning makroparameetrite (valkude, rasvade ja süsivesikute) tasakaalu. Selle põhjal saab söögiks valida täpselt need toidud, mis annavad õige koguse vajalikke toitaineid.

Käesoleva bakalaureusetöö idee pärineb otse sihtgrupilt, võttes arvesse eelkõige kulturistide ja *fitness-ga* tegelevate sportlaste vajadusi. Toitumispäeviku pidamiseks on olemas mitmeid mobiilirakendusi ja veebilehti, kuid enamus neist on ingliskeelsed ja seetõttu eestlastele raskesti kasutatavad. Eestikeelse toitude nimistuga on praegusel hetkel ainult mõned veebikeskkonnad, kuid neid ei ole mobiilsetel seadmetel mugav kasutada. Lisaks ei arvesta olemasolevad süsteemid sportlaste vajadustega. Sõltuvalt inimese päevasest aktiivsusest vajab inimene teatud hulga energiat. Toitumispäeviku mugavaks pidamiseks on selles töös programmeeritud veebirakendus tervikuna eestikeelne. See tähendab, et on kasutatud eestikeelset toitude andmebaasi, mille põhjal saab kasutaja pidada toitumispäevikut.

Töö peaeesmärgiks on võimaldada eestikeelse toitumispäeviku pidamist igas seadmes. Rakenduses on rõhutud lihtsale kasutajaliidesele, sobivusele kõigi ekraani resolutsioonidega ja võimalusele toiminguid võimalikult kiiresti sooritada. Lisaks sellele saab kasutaja salvestada sisestatud toidukorra. Selleks annab kasutaja toidukorrale nime, mille põhjal saab seda tulevikus tervikuna uuesti kasutada. Enne antud päeva esimese toidukorra sisestamist peab kasutaja sisestama oma kehakaalu ja kõhu ümbermõõdu ning valima sobiva aktiivsuse astme. Rakendus koondab kasutaja isiklikud toitumisalased andmed koos sisestatud kehakaalu ja teiste parameetritega kronoloogiliseks ajalooks. Andmete ajalugu on võimalik vaadata päevapõhiselt. Andmeid kuvatakse graafikute ja diagrammidena.

1. Projekti taust ja nõuded

Selles peatükis tutvustatakse projekti nõudeid, mis peavad olema programmeeritavas süsteemis realiseeritud. Nõuded jagunevad funktsionaalseteks ja mittefunktsionaalseteks nõueteks. Funktsionaalsed nõuded kirjeldavad, mida süsteem peab tegema – mida kasutaja saab süsteemis teha. Mittefunktsionaalsed nõuded kirjeldavad üldisemaid süsteemiga seotud nõudeid nagu turvalisus, usaldusväärsus ja efektiivsus ega keskendu konkreetsetele süsteemi osadele. Lisaks nõuetele tutvustatakse programmis kasutatavate toitude andmete päritolu.

1.1. Funktsionaalsed nõuded

1. Kasutaja saab sisestada oma kehakaalu ja kõhu ümbermõõdu kord päevas.
2. Toidukorra lisamiseks peavad antud päeva kehakaal ja kõhu ümbermõõt olema sisestatud.
3. Kasutaja saab valida toidukorra liiki: hommikusöök, lõunasöök, õhtusöök, vahepala.
4. Kasutaja saab koostada toidukorra mitmetest toitudest.
5. Kasutaja saab toidukorra koostamisel sisestada valitud toidu kaalu grammides.
6. Kasutaja saab toidukorda valitud toidu sealt eemaldada.
7. Kasutaja saab toidukorda kinnitada.
8. Kasutaja saab toidukorrale nime anda ja seeläbi toidukorra salvestada.
9. Kasutaja saab toidukorra lisamise sammude vahel liikuda edasi-tagasi. Sammud on toidukorra liigi valimine, toitude sisestamine ja kinnitamine.
10. Kui kasutaja on sisestanud toitusid, läheb tagasi eelmisesse sammu ja valib uue toidukorra liigi, siis sisestatud toidud kustutatakse.
11. Kasutaja saab muuta salvestatud toidukorra nime.
12. Kasutaja saab kustutada salvestatud toidukorra.
13. Salvestatud toidukordasid kuvatakse toidukorra liigi põhiselt.
14. Kasutaja saab päevapõhiselt vaadata sisestatud toidukordasid.
15. Ajalugu sisaldab toidukorrapõhiselt tarbitud toitusid koos koguse ja saadud energiahulgaga.
16. Ajalugu võimaldab graafikuna vaadata kehakaalu ja kõhu ümbermõõdu muutust ajas.

17. Süsteem arvutab kehakaalu, pikkuse, soo ja vanuse põhjal kasutaja soovitusliku energiakoguse.

1.2. Mittefunktsionaalsed nõuded

1. Parool tuleb andmebaasi salvestada soolatud räsiväärtusena.
2. Peab olema tagatud elementaarne CSRF-rünnaku vastane kaitse. Vormid varustada *CSRF-token'ga*.
3. Toidu tekstiotsingus tuleb jooksvalt otsingutulemusi näidata alates 3-märgilisest otsingusõnest.
4. Andmebaasist tuleb andmeid lugeda virtuaalselt andmete kihilt. Iga tabel peab olema kaetud vaatega.
5. Süsteemi tavakasutajal on roll „ROLE_USER".
6. Rakendus peab järgima MVC arhitektuuri.
7. Kasutaja saab süsteemi sisse logida kasutades oma e-maili ja parooli.
8. Kasutaja saab süsteemis end registreerida.
9. Anonüümsele (sisse logimata) kasutajale kuvatakse ainult võimalust sisse logida või end registreerida. Muu funktsionaalsus on peidetud.
10. Rakenduse sisuline funktsionaalsus on kättesaadav ainult sisse loginud kasutajale, kes on rollis „ROLE_USER".

1.3. Toidu koostise andmebaas

Töös kasutatud toitude andmed on saadud Tervise Arengu Instituudi toidu koostise andmebaasist (versioon 6). Teaduspõhine toidu koostise andmebaasi pidamine algas 1990ndate alguses Soome Terviseinstituudist hangitud Micronutrica menüüde analüüsimise programmist. 2009. aastal viidi andmebaas vastavusse Eurofoods nõuetega ja alustati pidevat andmete uuendamist. Andmeid toidu koostise andmebaasi kogutakse kirjandusallikatest, teistest sarnastest andmebaasidest, toidutööstuse esindajatelt ja andmete analüüsimise teel. [1]

Antud töös kasutatav andmebaas sisaldab 2765 erinevat toorainet (sh mõned valmistoitudena), kus iga toidu juures on andmed 59 erineva toitaine kohta. Selles töös kasutatakse saadavat energiakogust 100g kohta ja eraldi saadavat energiakogust valkudest, rasvadest ja süsivesikutest (samuti 100g kohta).

2. Süsteemi arhitektuur ja disain

Selles peatükis tutvustatakse süsteemi arhitektuuri ja disaini ning käesoleval hetkel olemasolevaid sarnaseid lahendusi. Käsitlus on üldisema ja analüüsivama suunaga, kus keskendutakse programmeeritud rakenduse erinevatele kasutusjuhtudele, energia ja toitainete arvutamisele. Lisaks sellele kirjeldatakse toidukorra lisamiseks vajalike etappide läbimist ja tutvustatakse rakenduse üldist arhitektuuri ja disaini.

2.1. Alternatiivsed lahendused

Sarnaseid rakendusi on maailmas tehtud mitmeid, kuid suurem osa on neist ingliskeelsed. Eestikeelse toitude nimistuga rakendusi on vähe. Järgnevalt analüüsitakse kahte eestikeelset rakendust ja lisaks tutvustatakse ka üht ingliskeelset lahendust.

2.1.1. Tervise arengu instituudi toitumisprogramm

Toitumisprogramm põhineb Tervise Arengu Instituudi toidu koostise andmebaasi versioonil 5 (27.06.2013). Programmis on võimalik koostada isiklikke menüüsid, retsepte ja pidada toidupäevikut. Programm arvutab kasutaja kohta välja vajaliku päevase energiakoguse, toitainete koguse, aga ka tarbitud vitamiinide ja mineraalainete kogused. Lisaks võrdleb rakendus kasutaja kohta arvutatud toitumisalaseid andmeid vanusegrupile vastavalt soovitusliku Eesti keskmisega. Selle põhjal saab kasutaja teada, kas ta on antud toitainet tarbinud liiga palju või liiga vähe. Toitumisprogrammi kasutamiseks tuleb registreeruda. [2]

Toitumisprogrammi puuduseks on selle halb kasutajaliides. Programmi loomisel ei ole arvestatud eri suuruses ekraanidega ja seetõttu ei kohanda veebikuva end vastavalt ekraani suurusele.

2.1.2. Fitness.ee kaloriarvuti

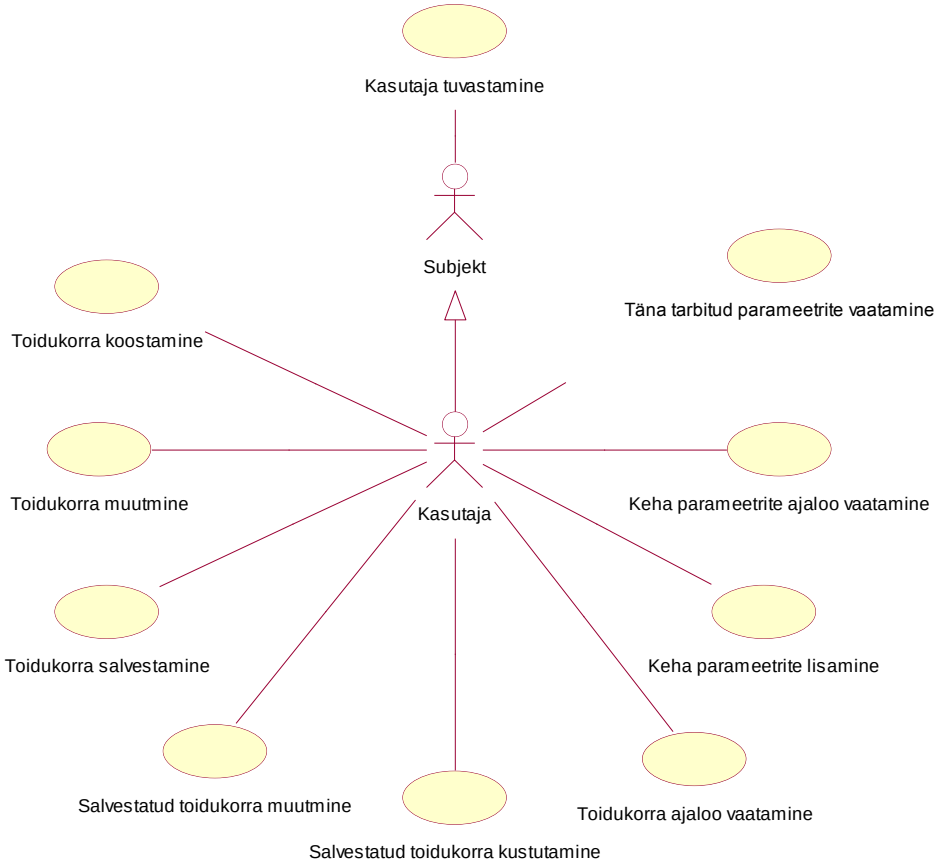
Fitness.ee kaloriarvuti võimaldab tarbitud toitude põhjal välja arvutada saadud energia ja toitainete koguse. Andmeid saab sisestada iga toidukorra kohta eraldi. Rippmenüüst tuleb valida toit ja seejärel sisestada tarbitud kogus grammides. Fitness.ee kaloriarvuti kasutamiseks ei pea registreeruma, kuid seetõttu ei salvesta süsteem kasutaja andmeid, vaid sisestatud toidukorra kohta saadavad andmed saab kasutaja lasta saata e-mailile. Selline lahendus on ebamugav, kuna kasutaja peab statistika tegemiseks ise saadud andmeid töötlema ja struktureerima. Lisaks sellele ei ole fitness.ee kaloriarvuti kohandatud töötama erineva resolutsiooniga ekraanidel. [3]

2.1.3. Myfitnesspal toitumispäevik

Myfitnesspal on laialdaste võimalustega ingliskeelne rakendus. See võimaldab pidada toitumispäevikut, sisestada treeningplaani ja pidada tulemuste üle arvestust. Rakenduse toitumispäeviku osa sarnaneb paljuski selles töös realiseeritava rakendusega. Siiski võib välja tuua mõned kitsaskohad: toitude nimed on ingliskeelsed ja seetõttu eestlasele raskesti leitavad, sisestatud toidu andmeid pole võimalik muuta (toit tuleb ära kustutada ja andmebaasiotsingut kasutades uuesti lisada), kohati ei kasutata euroopas levinud mõõtühikuid. Positiivse poole pealt võimaldab rakendus kirjeldada enda eesmärgid ja näitab selle põhjal kasutaja enda eesmärgini jõudmist. Lisaks kasutab rakendus suurt toitude andmebaasi, mis sisaldab üle 5 miljoni erineva toidu. Rakendusest on olemas nii mobiilirakendus kui ka veebileht. [4]

2.2. Programmi kasutusjuhud

Programmis realiseeritakse kokku üheksa erinevat sisulise funktsionaalsusega seotud kasutusjuhtu ja kümnendana kasutaja tuvastamine. Kasutusjuhtude struktuur on näha joonisel 1.



Joonis 1 Programmi kasutusjuhtude eskiismudel

Toidukorra salvestamine ja toidukorra koostamine on teineteisele sarnased, kuid samas erinevad kasutusjuhud. Salvestatud toidukorra tekitamiseks tuleb anda toidukorrale nimi. Kasutajale kuvatakse salvestatud toidukordasid vastavalt valitud toidukorra liigile. Toidukorra või salvestatud toidukorra muutmisel on kasutajal võimalik lisada või eemaldada toidukorrast toitusid ja muuta toidu kaalu.

Lisaks toidukordade lisamisele kuvatakse kasutajale toidukordade ajalugu. Päevane ajalugu sisaldab toidukorrapõhiselt tarbitud toitusid koos kogustega ja lisaks eraldi päeva jooksul saadud energiakogust koos süsteemi poolt soovitatud energiakogusega.

Keha parameetrite (kaal ja kõhu ümbermõõt) lisamine on kohustuslik enne esimese toidukorra lisamist antud päeval. Vastasel juhul on kasutajal võimalik vaadata vaid ajalugu ja muuta salvestatud toidukordasid.

2.3. Vajaliku päevase energia ja makroparameetrite arvutamine

Päevase energiakoguse arvutamiseks kasutatakse Harris-Benedicti valemit [5]. Energiakoguse (E) arvutamiseks korrutatakse inimese BMR *(basal metabolic rate)* aktiivsuskoefitsiendiga (A) [6]. BMR on inimese minimaalne energiakulu ajaühikus, kui inimene on puhkeasendis. BMR arvutamiseks on vaja teada inimese sugu, kaalu, pikkust ja vanust. BMR arvutamise valem on kirjeldatud lisas 1. Järgnev valem kirjeldab vajaliku energiakoguse arvutamist.

$$E = BMR * A$$

Aktiivsuskoefitsient A sõltub inimese aktiivsusest. Aktiivsuskoefitsient on võimalik määrata alljärgnevast tabelist 1.

Tabel 1 Aktiivsuskoefitsiendid

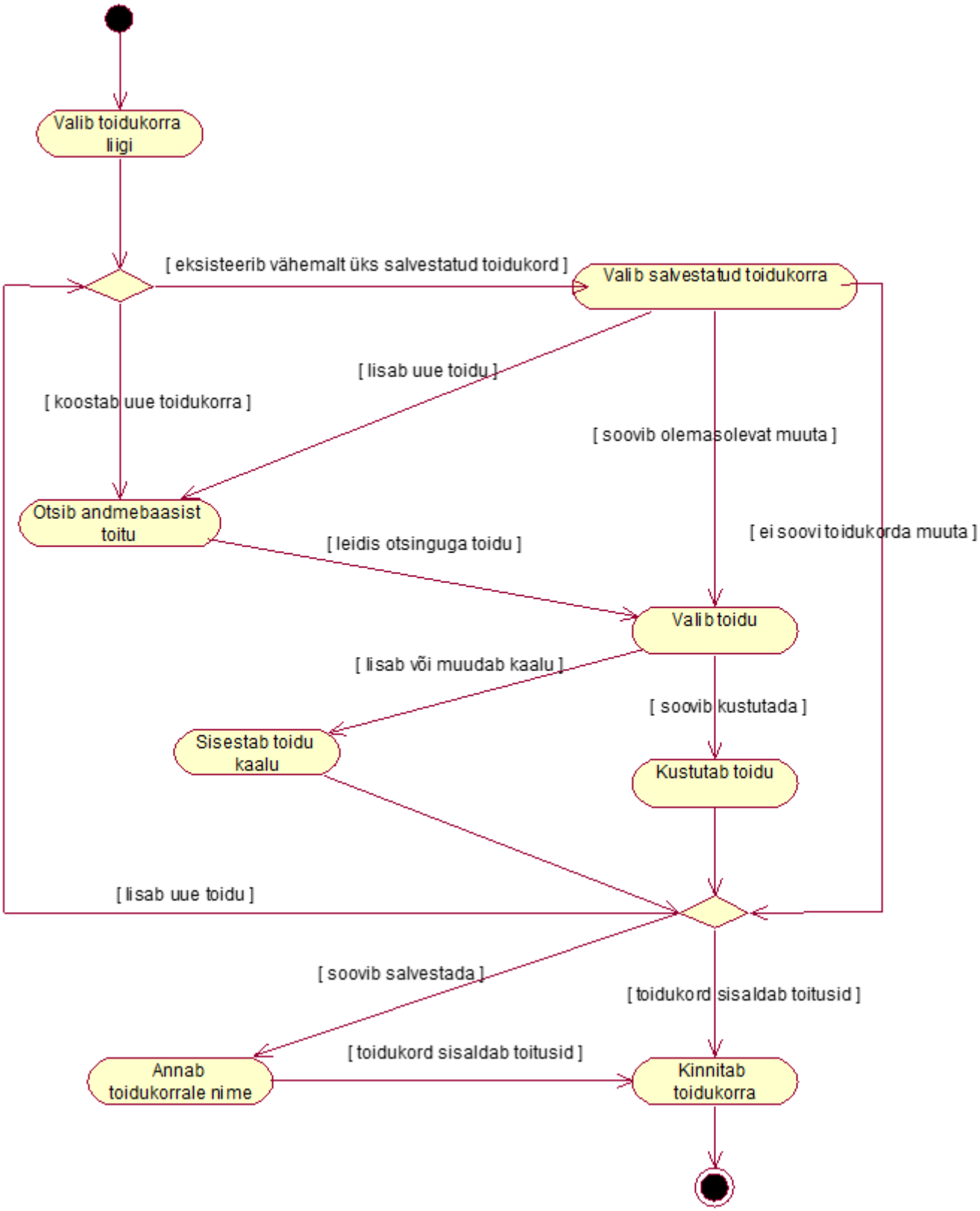
Aktiivsus	A
Üliaktiivne (Kõva füüsilist koormust pakkuv töö koos raske treeninguga samal päeval. Näiteks ratturid, triatleedid.)	1,9
Väga aktiivne (Kõva füüsiline aktiivsus suurem osa päevast. Näiteks ehitajad, jalgrattatakso juhid.)	1,725
Keskmiselt aktiivne (Füüsiline aktiivsus teatav osa päevast. Näiteks postiljonid, kelnerid.)	1,55
Väheaktiivne (Mitteistuv töö suuremal osal päevast. Näiteks õpetajad, müügimehed.)	1,375
Mitteaktiivne, istuv (Suurem osa päevast istuv. Näiteks kontoris töötavad inimesed.)	1,2

2.4. Toidukorra lisamine

Toidukorra lisamine on kolmeetapiline protsess, kus iga sammu vahel saab edasi-tagasi liikuda. Esimene samm on toidukorra liigi valimine, kus kasutaja peab määrama, kas tegu on hommiku-, lõuna-, õhtusöögi või vahepalaga. Teises sammus saab toitudest koostada toidukorra. Kolmas samm on toidukorra kinnitamiseks. Kolmandas sammus saab kasutaja veenduda, et sisestatud andmed on õiged. Pärast kinnitamist ei ole toidukorda enam võimalik muuta. Toidukorra kinnitamiseks peab toidukord sisaldama vähemalt ühte toitu.

Kui kasutaja vahetab toidukorra liiki pärast toitude sisestamist, siis valitud toidud kustutatakse. Kui kasutaja uut toidukorra liiki ei vali, aga liigub sammude vahel edasi-tagasi, hoiab süsteem kõiki kasutaja valikuid mälus.

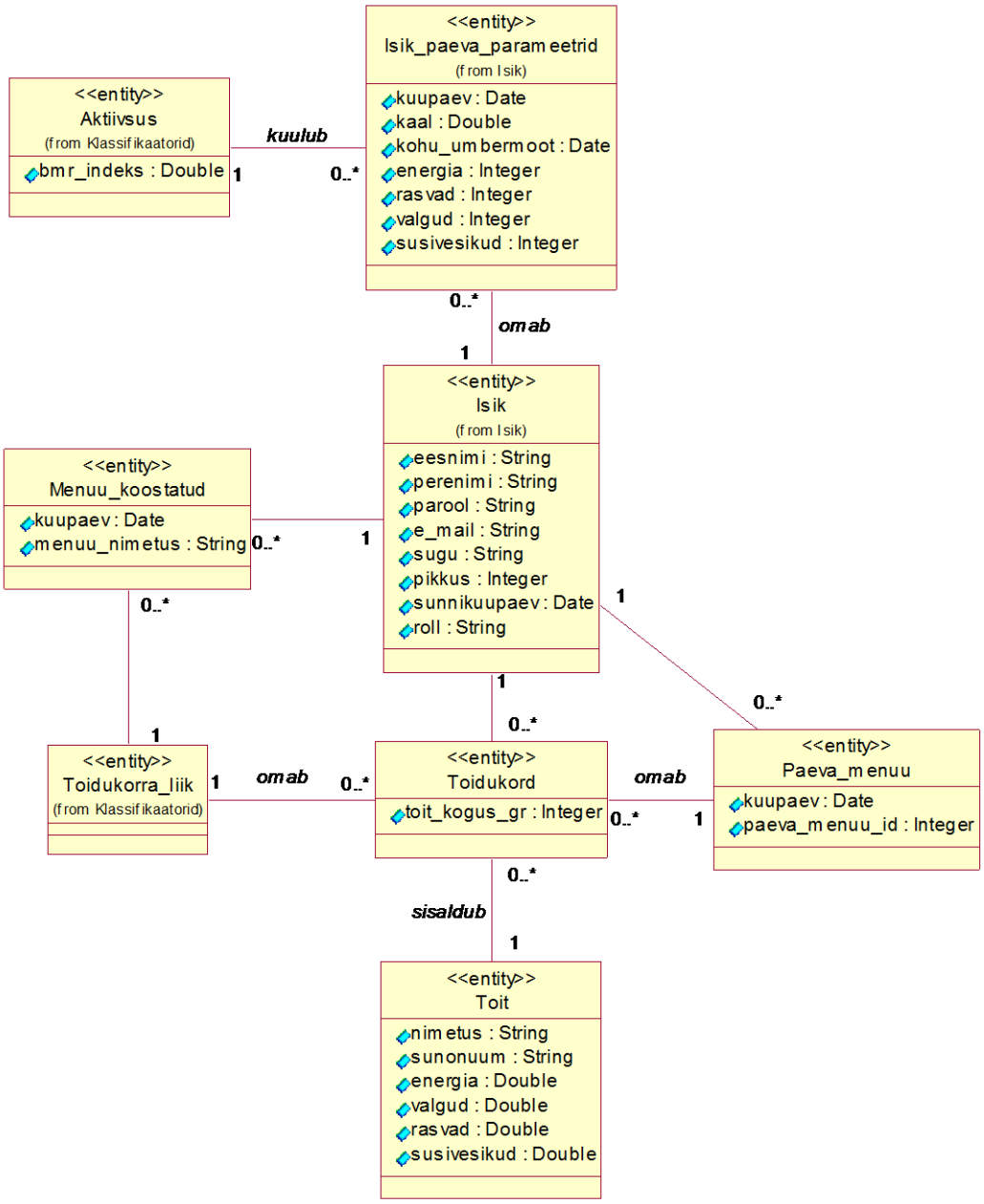
Toitude lisamiseks toidukorda on kolm võimalust. Esimene võimalus on koostada uus toidukord, otsides toite tekstiotsinguga andmebaasist. Teine võimalus on valida varasemalt salvestatud toidukord rippmenüüst. Kolmas võimalus on valida varem salvestatud toidukord ja seda seejärel muuta. Iga toidu valimisel tuleb sisestada toidu kaal. Salvestatud toidukorra toitudel on kaal varasemalt salvestatud ja seda saab soovi korral muuta. Toitude lisamist toidukorda illustreerib järgnev tegevusdiagramm (joonis 2).



Joonis 2 Toidukorra lisamise tegevusdiagramm

2.5. Andmebaasi olemi-suhte diagramm

Joonisel 3 kirjeldatakse olemid ja olemitevahelised võimsustikud, millest süsteemi realisatsioonis kujunevad andmebaasitabelid ja tabelite vahelised suhted. Olemi-suhte diagrammil on kasutatud terminit „menüü“, mis rakenduse kontekstis vastab salvestatud toidukorrale.



Joonis 3 Olemi-suhte diagramm

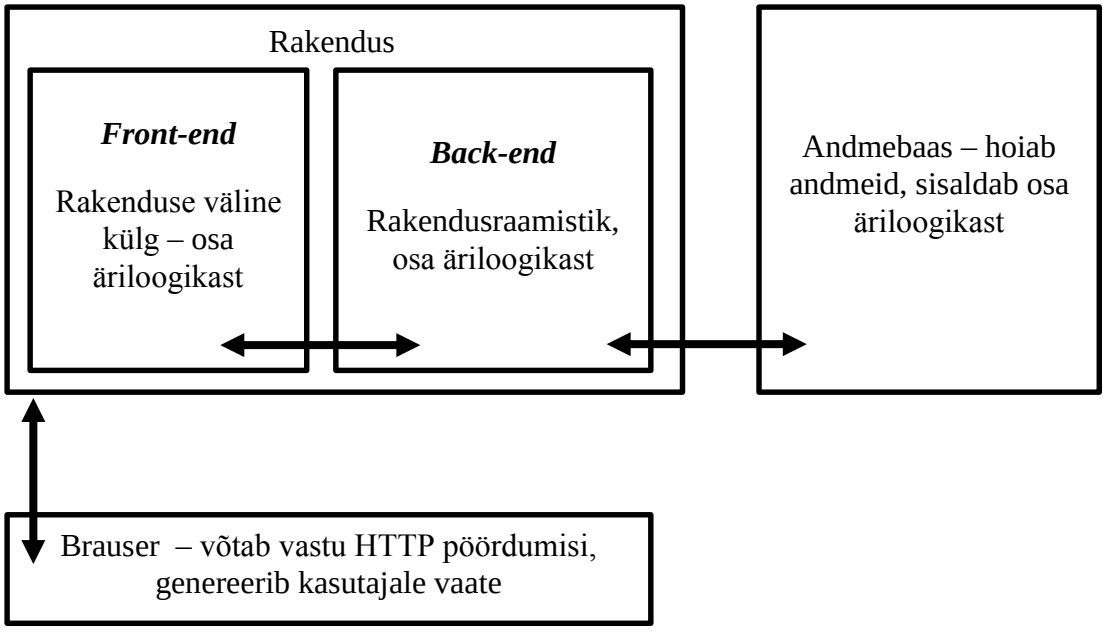
Järgnevas tabelis 2 kirjeldatakse olemitüüpe lähemalt.

Tabel 2 Olemitüüpide selgitused

Olemitüüp	Selgitus
Aktiivsus	Klassifikaator, mis kirjeldab isiku aktiivsust andes isikule konkreetse BMR-indeksi.
Isik_paeva_parameetrid	Kasutaja kohta käivad päevapõhised andmed nagu kehakaal, kõhu ümbermõõt ja tarbitud toitainete kogused.
Isik	Kasutaja kohta käivad üldised andmed nagu eesnimi, perenimi, e-mail jm.
Menuu_koostatud	Andmed salvestatud toidukorra kohta. Sisaldab salvestatud toidukorra nimetust.
Toidukord	Andmed toidukorras oleva ühe toidu kohta. Igale toidule vastab kaal grammides.
Toidukorra_liik	Klassifikaator, mis kirjeldab toidukorra liigi. Toidukorral on 4 võimalikku liiki: hommikusöök, lõunasöök, õhtusöök ja vahepala.
Toit	Toitudest moodustub toidukord. Sisaldab makroparameetrite väärtuseid 100g kohta.
Paeva_menuu	Hoiab isiku ja kuupäeva kombinatsiooni kohta unikaalset koodi, mille põhjal määratakse, mis toidukord kuulub millisele isikule ja vastab millisele kuupäevale.

2.6. Süsteemi arhitektuur

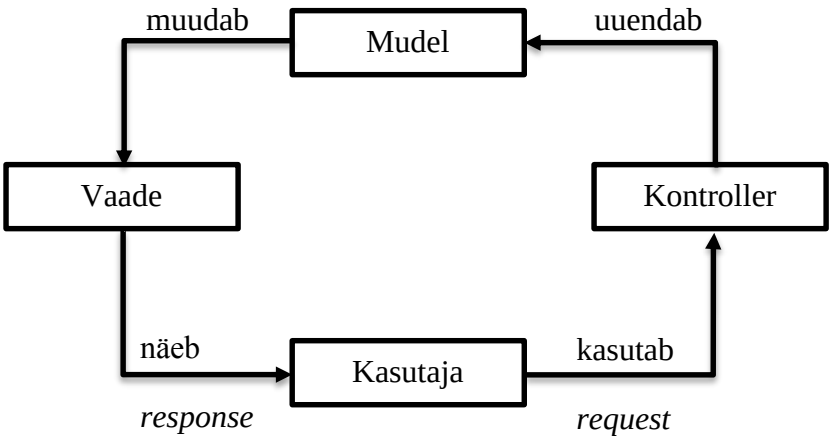
Infosüsteemi toimimiseks on vaja kolme osa: andmebaas, rakendus ja brauser. Joonis 4 illustreerib nende kolme komponendi omavahelist seotust. Brauser suhtleb HTTP päringute abil rakendusega. Rakendus jaguneb kaheks loogiliseks tükiks: *front-end* ja *back-end*. Esimeses genereeritakse HTML kujul vaade koos seda toetavate failidega (kujundus, skriptid, pildid). *Back-end* sisaldab endas põhilist osa äriloogikast. Lisaks võtab *back-end* vastu kasutaja pöördumisi (HTTP päringuid). *Back-end* sisaldab rakendusraamistikku, mis suhtleb andmebaasiga ja aitab genereerida vaadet.



Joonis 4 Lihtsustatud süsteemi ülesehitus

2.6.1. MVC arhitektuur

Rakenduse arhitektuuri disainil lähtutakse MVC *(model-view-controller)* arhitektuuri põhimõtetest. MVC arhitektuur koosneb kolmest komponendist: mudel, vaade ja kontroller. Joonis 5 näitab MVC komponentide ja kasutaja omavahelist seotust. Kui brauser saadab päringu *(request)* serverile, siis võtab päringu vastu kontroller. Mudelis tehakse päringule vastavad muudatused ja seejärel genereeritakse kasutajale muudetud kujul vaade. Vaade on sisuliselt see, mida brauser kuvab.



Joonis 5 MVC muster

3. Süsteemi realisatsioon

Selles peatükis tutvustatakse süsteemi tehnilist realisatsiooni: kasutatud tehnoloogiaid ja andmebaasisüsteemi ning räägitakse süsteemi turvalisusest. Lisaks näidatakse toidukorra lisamise tehnilist lahendust ja tutvustatakse kasutajaliidest. Süsteem programmeeriti tervikuna, kasutades IntelliJ IDEA 14 arenduskeskkonda. Projektist varukoopiate tegemiseks kasutati GIT koodihoidlat.

3.1. Kasutatud tehnoloogiad

3.1.1. Java SE 8

Java on objektorienteeritud programmeerimiskeel, mida arendab Oracle Corporation [7]. Java arendamisel lähtutakse põhimõttest *write once, run anywhere* (WORA), mis tähendab, et kompileeritud Java kood töötab kõigil platvormidel, kus on Java käivitamiseks vajalik *Java Virtual Machine* [8].

Antud töös on kasutatud programmiversiooni *Java Standard Edition* (SE) 8. *Java Standard Edition* (varasem J2SE) võimaldab luua klient-server rakendusi, pakkudes arendajale suure hulga valmis liideseid, mille abil on võimalik realiseerida keerukaid süsteeme[9]. Java kasuks otsustati selle suure populaarsuse tõttu. Programmeerimiskeele populaarsust kuu lõikes näitab TIOBE indeks, mis arvutatakse otsingumootorites tehtud otsingute põhjal – mida suurem on otsingutulemuste arv mingi programmeerimiskeele kohta, seda populaarsem see on [10]. Lisaks on Javale saadaval suur hulk lisateeke, mida käsitletakse peatüki järgnevates alapunktides.

3.1.2. Apache Maven

Apache Maven on Java juurde kuuluv rakenduse seadistus– ja haldusvahend, mis võimaldab ühe konfiguratsioonifailiga projekti kasutamiseks valmis seada [11]. Konfiguratsioonifailiks on pom.xml *(Project object model)*, mis on kindla struktuuriga. Kohustuslikud elemendid on `<groupId>` ja `<artifactId>`, kus esimene määrab ära paketi, kuhu antud projekt kuuluma hakkab, ja teine annab .jar failile *(java archive)* nime [12]. Antud projektis on *groupId* „ee.ttu.fooddiary“ ja *artifactId* „FoodDiary“.

3.1.3. Spring rakendusraamistik

Spring Framework on üks populaarsemaid Java rakendusraamistikke. Spring Framework koosneb hulgast eraldiseisvatest komponentidest, mis kõik kuuluvad paketti *org.springframework*. Rakendusraamistiku kasutamine teeb veebilehe programmeerimise arendaja jaoks lihtsamaks, sest suur osa vajalikust loogikast on rakendusraamistikus juba olemas: autentimine, sessioonihaldus, HTTP-pöördumiste vastuvõtmine ning edasisuunamine, andmebaasisuhtlus ja palju muud. Springi kasutamiseks tuleb vajalikud sõltuvused lisada Maveni pom.xml konfiguratsioonifaili. [13, 14]

Antud projektis Springi kasutamiseks vajaminevad osad on toodud tabelis 2. Need komponendid lisatakse sõltuvustena Maveni konfiguratsioonifaili.

Tabel 3 Spring rakendusraamistiku komponendid

Nimi	**Versioon**	**Vajadus**
Spring-core	4.0.6	Sisaldab baasfuntksionaalsust, mis on vajalik Springi toimimiseks.
Spring-webmvc	4.0.6	Sisaldab veebiarenduseks vajalikke komponente, sh võimaldab realiseerida MVC-arhitektuuri.
Spring-orm	4.0.6	Komponent, mis koos Hibernate'ga tagab andmebaasisuhtluse.
Spring-security	3.2.3	Rakenduse turvalisuse tagamiseks.

Rakenduse turvalisust (sh Spring Security) käsitletakse täpsemalt punktis 3.3.

3.1.4. JSP, JSPF, JSTL

JSP *(Java Server Page)* on Java veebirakenduste osa, millest genereeritakse vaade. JSP-fail võib sisaldada Java koodi, seega kuulub osa rakenduse äriloogikast .jsp failidesse [15].

JSPF *(Java Server Page Fragment)* on eraldiseisev objekt, mis hoiab endas osa .jsp faili koodist. See on korduvkasutatav ja võimaldab JSP faile paremini struktureerida, eraldades eri funktsionaalsuse eri fragmentidesse [16].

JSPF objekte on antud töös kasutatud toidukorra sisestamise etappides. Toidukorra sisestamine koosneb ühest JSP failist, mis omakorda koosneb kolmest JSPF failist – iga toidukorra sisestamise samm on eraldi JSPF fail.

- newMealFlow.jsp
    - mealType.jspf
    - mealEntry.jspf
    - mealConfirmation.jspf

JSTL *(JavaServer Pages Standard Tag Library)* võimaldab JSP ja JSPF failides kasutada teatud Java baasfunktsionaalsust nagu tingimuslause, itereerimine üle kollektsiooni, tühja objekti kontroll jm. JSTL jaotub mitmeks alamkategooriaks. Antud töös on kasutatud lisaks *core* teegile ka *fmt (Formatting)* teeki, mis sisaldab vajalikku funktsionaalsust, et viia numbriväli nõutud formaati.

Näiteks järgnev süntaks võimaldab ümardada numbri sajandike täpsuseni: `<fmt:formatNumber value="${dailyEnergy}" maxFractionDigits= "2"/>`.

3.1.5. Lombok

Lombok on vabavaraline Java entusiastide poolt arendatud teek Java programmeerimiskeele juurde. Lombokit kasutades on võimalik genereerida teatavad meetodid nagu konstruktorid, *getter'd, setter'd, toString* taustal. Selles töös on Java klassidel kasutatud annotatsioone `@Getter` ja `@Setter`, mis genereerivad taustal klassi muutujatele *getter* ja *setter* meetodid. Lisaks neile on vajadusel kasutatud `@Builder` annotatsiooni, mis vabastab klassi konstruktori tegemisest. `@Builder` on kasulik juhul, kui ühte ja sama objekti on eri juhtudel vaja initsialiseerida erineva arvu parameetritega. Lombokit kasutades on programmikoodis implementeeritud ainult sisulist tähtsust omavad meetodid, koodiridade arv on väiksem ja koodi on lihtsam lugeda. Lisas 2 (tabelis 4) on kirjeldatud näide Lomboki kasutamisest. [17]

3.1.6. Joda-time

Joda-time on Java teek, mis lihtsustab Java aja- ja kuupäevatöötlust. Enne versiooni Java SE 8 on aja- ja kuupäevatöötlus Javas tülikas, sest puudusid lihtsasti kasutatavad meetodid aja ja kuupäeva tüüpi muutujatega tehete tegemiseks. Sageli oli vaja kasutada *Calendar* objekti. Alates Java SE 8 versioonist on kuupäevatöötlust lihtsustatud, lisades uut funktsionaalsust paketti *java.time*. [18]

Antud töö on programmeeritud kasutades Java SE 8 versiooni. Siiski on siin kasutatud kuupäevade teisendamiseks Joda abiteeki, et kood oleks kompileeritav ka vanemate versioonidega kui Java SE 8. Programmikood kompileerub ka versioonis Java SE 7, kuid mitte Java versioonis Java SE 6, sest on kasutatud *diamond* operaatorit, mis on Javas kasutusel alates versioonist Java SE 7. Joda vajab tööks vähemalt Java SE 5 versiooni.

3.1.7. Hibernate

Hibernate on Java teek, mis võimaldab mudeli (Java *entity* klassi) ühendada objektrelatsioonilise andmebaasi tabeliga, võttes enda kanda andmebaasitabeli ja andmeobjekti vastastikuse initsialiseerimise. Hibernate ORM *(Object-relational mapping)* on Hibernate põhikomponent, kus on kirjeldatud kogu andmebaasisuhtluseks vajalik loogika. [19]

3.1.8. AJAX, HTML5, jQuery, Bootstrap

Need tehnoloogiad kirjeldavad süsteemi arhitektuuri *front-end* osa (joonis 4). HTML5 on käesoleval hetkel viimane HTML*(Hyper text markup language)* versioon [20]. HTML süntaksis kirjeldatakse veebilehe struktuur, mida brauser peab interpreteerima.

Dünaamilise veebilehe loomiseks kasutatakse brauseris töötavat programmeerimiskeelt. Javascript on kõige tuntum ECMAScripti osa [21]. Lisaks kasutatakse javascripti teeki jQuery, mis pakub suurt hulka valmis kirjutatud funktsionaalsust, võimaldades dünaamilist veebilehte arendaja jaoks lihtsamalt luua. Antud töös on jQuery abil loodud tabelid, kuhu kuvatakse otsingutulemus ja valitud toidud. Andmed saadakse tabelisse AJAX*(Asynchronous JavaScript and XML)* GET-päringuga. AJAX võimaldab suhelda brauseril taustal serveriga ilma, et kasutaja sellest aru saaks, sest lehte ei laeta tervikuna uuesti. [22]

Veebilehe kujunduse loomisel on kasutatud CSS lisateeki Bootstrap. Bootstrap pakub võimalust teha tänapäevast veebilehte, mis võtab mõõtmed vastavalt kasutaja ekraani pikslite arvule *(responsive design)*. Kasutajaliidest käsitletakse täpsemalt punktis 3.4. [23]

3.1.9. jQuery-autocomplete

jQuery-autocomplete (või AJAX-Autocomplete) on Tomas Kirda arendatud jQuery lisateek. See võimaldab kuvada tekstiotsingu tulemusi jooksvalt, kui kasutaja alles otsingusõna kirjutab. Brauser teeb taustal AJAX päringuid, mis sisaldavad kasutaja kirjutatud (poolikut) otsingusõne. Päringule vastab server JSON-formaadis otsingutulemusega olles eelnevalt teostanud andmebaasist otsingu. Kuna AJAX päring käivitub iga klahvivajutuse peale *(keyUp)*. Teeki on konfigureeritud nii, et päringuid tehtaks alates otsingusõnest, mis on minimaalselt 3 märki pikk. Illustreeriv näide teegi kasutamisest on peatükis 3.4 joonisel 7. [24]

3.1.10. Google Charts

Google Charts võimaldab luua erinevaid interaktiivseid graafikuid [25]. Antud töös on selle abil loodud sektordiagrammid ja joondiagrammid. Graafiku joonistamiseks tuleb ette anda JSON-formaadis väärtused, millest API genereerib HTML kujul tabeli. Seejärel moodustatakse soovitud kujul graafik kasutades Google Visualization API liidest.

3.2. Andmebaasisüsteem

Rakenduse andmebaasisüsteemiks on PostgreSQL 9.3 objektrelatsiooniline andmebaasisüsteem [26]. Kokku on rakenduse jaoks realiseeritud 8 tabelit, mis on omavahelistes suhetes (vt joonis 3). Andmete serveerimine rakendusele käib läbi vaadete, et tagada paremat andmete turvalisust. Vaadete loomisel on kasutatud lisaklauslit `WITH (security_barrier)`, mis tagab teatava reataseme turvalisuse [27].

Järgnev lause loob vaate tabelile `menuu_koostatud`. See tabel sisaldab salvestatud toidukorra nimetust, salvestatud toidukorra ID-d, kuupäeva ja välisvõtmeid isiku tabelisse ning toidukorra liigi tabelisse. Määrang `DISTINCT ON (menuu_nimetus, isik_kood)` lubab vaate tulemusse ühekordselt sama menuu_nimetus ja isik_kood

komibnatisooni. Ilma `DISTINCT ON` määranguta võib vaates olla sama isiku sama toidukorda mitu korda ja see ei ole selle vaate eesmärk. Kõik vaated on antud töös sama nimega, mis tabelid, kuid eristatud märkega _andmed. Järgnev näide ilmestab vaate loomist PostgreSQL andmebaasisüsteemis.

```
CREATE VIEW menuu_koostatud_andmed WITH (security_barrier) AS
  SELECT DISTINCT ON (menuu_nimetus, isik_kood) *
  FROM menuu_koostatud
  ORDER BY menuu_nimetus DESC;
```

Lisaks vaadetele on andmebaasis jõustatud indeksid olulistele tabeli veergudele, mis ei ole primaarvõtme veerud. Primaarvõtme veergudele loob andmebaasisüsteem indeksi automaatselt. Indekseeritud on tabelis Toit veerg „nimetus“, sest tabel Toit sisaldab üle 2700 rea ning tabelist sooritatakse otsinguid nimetuse alusel sageli.

Andmebaasis on jõustatud ka trigerid, millest lähemalt tutvustatakse ühte. Trigerid käivituvad `INSERT` või `UPDATE` lausete korral. Triger `trig_lisa_paeva_menuu_id` tabelis Toidukord lisab paeva_menuu_id tabelist Paeva_menuu juurde `INSERT` lausele enne, kui see käivitub tabelis Toidukord.

Enne `INSERT` lause käivitamist tabelis Toidukord lisatakse `INSERT` lausele parameeter paeva_menuu_id, mis on primaarvõti tabelis Paeva_menuu. Paeva_menuu_id määrab ära, et antud toidukorrad kuuluvad konkreetse isiku ja kuupäeva kombinatsiooni juurde. Selle põhjal on toidukordasid võimalik hiljem grupeerida.

```
CREATE TRIGGER trig_lisa_paeva_menuu_id BEFORE INSERT ON
toidukord
FOR EACH ROW WHEN (NEW.paeva_menuu_id IS NULL) EXECUTE PROCEDURE
f_lisa_paeva_menuu_id();

CREATE OR REPLACE FUNCTION f_lisa_paeva_menuu_id()
  RETURNS TRIGGER AS $$
BEGIN
  IF NEW.paeva_menuu_id IS NULL
  THEN
    NEW.paeva_menuu_id := (SELECT paeva_menuu_id
                           FROM paeva_menuu
                           WHERE     paeva_menuu.isik_kood     =
NEW.isik_kood AND paeva_menuu.kuupaev = CURRENT_DATE);
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';
```

Peale selle kontrollitakse kasutaja registreerimisel andmebaasi tasemel, kas sisestatud e-mail on rakenduse kontekstis vaba. Valideerimist käsitletakse täpsemalt punktis 3.3.3.

3.3. Rakenduse turvalisus

Selles peatükis tutvustatakse rakenduses kasutatavaid turvameetmeid, nende valikut, konfiguratsiooni ja põhjendatakse nende meetmete vajalikkust. Rakenduse turvalisuse konfigureerimiseks on loodud *security.xml* fail.

3.3.1. Spring Security

Spring Security on teek Java rakenduste turvalisuse tagamiseks. Seoses Spring Security kasutuselevõtuga tuleb teha vajalikud seadistused security.xml failis, mis on ühtlasi Spring Security konfiguratsioonifail. Seadistamisel on lähtutud vajadusest suhelda andmebaasiga. Seega on vajalik seadistada `AuthenticationManager` pöörduma andmebaasi poole, et süsteemi sisselogimisel kontrollida, kas sisestatud e-mail ja parool kuuluvad registreerunud kasutajale. [28]

Spring Security kasutuselevõtuga ei töötle rakendus enam vaikimisi POST-päringuid. Nende lubamiseks on võimalik seadistada konfiguratsioon vastavalt: `<intercept-url pattern="/register" access="hasRole('ROLE_ANONYMOUS')" method="POST"/>`. Sedasi saab rakenduse registreerimisel sisselogimata kasutaja saata POST-päringuga serverile täidetud registreerimisvormi.

Lisaks oli vaja seadistada `WebContentInterceptor` *bean* dispatcher-servlet konfiguratsioonis, sest vastasel juhul olnuks võimalik pärast süsteemist välja logimist saada *back* nuppu vajutades uuesti sisselogituks.

Security.xml konfiguratsioonis tuleb seadistada parameeter `disable-url-rewriting="true"`, et vältida JSESSIONID ilmumist brauseri aadressiribale. Selle asemel pannakse sessiooni ID küpsisesse. Rakenduse kasutajal peavad küpsised olema lubatud [29].

3.3.2. BCrypt

BCrypt on Spring Security osa mis võimaldab parooli räsimist koos soolaga. BCrypt'i kasutatakse uue kasutaja registreerimisel, kui kasutaja andmed on vaja salvestada andmebaasi, aga ka kasutaja sisselogimisel sisestatud parooli räsi arvutamisel. Andmebaasis on salvestatud parooli räsiväärtus. BCrypt'i kasutamisel on väiksem oht, et rünnakuga tekib võimalus kasutajate e-maili ja parooli kombinatsioonide lekkeks. BCrypt algoritmiga saab muuta sõne räsiks, aga mitte vastupidi. Seega ei ole võimalik andmebaasis salvestatud räsist tuletada kasutaja parooli. [30]

BCrypt'i kasutamiseks on vaja seadistada *security.xml* failis `BCryptPasswordEncoder` *bean*. BCrypt algoritmi tugevuseks *(strength)* on valitud 11. Tugevus näitab, kui palju tööd on vaja teha kirje räsimiseks koos juhuslikult genereeritud soolaga. *String'i* räsimiseks Javas tuleb välja kutsuda BCrypt klassi staatiline funktsioon *hashpw*. Järgnev näide ilmestab parooli räsimist koos soolaga.

```
BCrypt.hashpw(newUser.getPassword(), BCrypt.gensalt(11))
```

3.3.3. Rakendusepoolne valideerimine

Andmete valideerimine rakenduse *back-end* osas (vt joonis 4) on äärmiselt oluline tagamaks andmete turvalisust ja õigsust. Valideerimisel on kasutatud *Data Transfer Object* (DTO) mustrit. DTO on Java objekt, mis on mõeldud andmete valideerimiseks ja mudelile saatmiseks. DTO objekt initsialiseeritakse vormi saatmisel brauserist kontrollerisse. Objekt sisaldab samu väljasid (parameetreid), mida kasutaja sisestab vormil, kusjuures kõik väljad peavad olema Java objektid ega tohi olla lihttüüpi. DTO objekti kasutamiseks tuleb JSP lehel kasutada Springi *tag*s hulka kuuluvat vormielementi. Vormi implementatsioon on kujul: `<form:form modelAttribute="user" action= "registerUser" enctype="UTF-8" method='POST'>`, kus *modelAttribute* viitab kasutatud DTO objektile, *action* viitab kontrollerile, mis vormi teenindab, ning *enctype* määrab vormi kodeeringu.

DTO objekti valideerimiseks on kasutatud Java annotatsioone, sest see võimaldab rakendusraamistikul valideerimine ise läbi viia. Valideerimise väljakutsumiseks on DTO objektile kontrolleri meetodis lisatud annotatsioon `@Vaild`.

DTO objektis on igale väljale antud vajalikud annotatsioonid välja valideerimiseks. Näiteks `@NotEmpty` kontrollib, et väli ei oleks NULL ega tühi (pikkus peab olema vähemalt 1). E-maili valideerimiseks on kasutatud annotatsiooni `@UniqueEmail`, mis implementeeriti iseseisvalt. Selleks kirjeldati valideerimisloogikat sisaldav klass, mis implementeeris liidest `ConstraintValidator<UniqueEmail, String>`, kus `UniqueEmail` viitab annotatsiooni klassile ning `String` valideeritava objekti tüübile.

E-mail valideerimisel kontrollitakse selle vastavust regulaaravaldisele ning veendutakse, et antud e-mail ei ole juba süsteemis registreeritud. E-maili unikaalsuse kontrollimiseks on implementeeritud andmebaasis funktsioon, mis tagastab `TRUE`, kui e-mail on vaba (selle e-maili kohta on päringus vaatest Isik_autentimisandmed 0 rida). Enne andmebaasifunktsiooni poole pöördumist valideeritakse e-mail regulaaravaldise põhjal. Andmebaasi funktsiooni on rakenduses välja kutsutud otse JDBC-ühendust kasutades kujul:

```
String sql = "SELECT f_kas_email_on_vaba(?)";
preparedStatement = connection.prepareStatement(sql);
preparedStatement.setString(1, email);
resultSet = preparedStatement.executeQuery();
resultSet.next();
result = resultSet.getBoolean(1);
```

3.3.1. CSRF rünnak

CSRF *(Cross-site Request Forgery)* rünnak on veebirakenduste vastu korraldatav rünnak, kus ründaja saab varastada kasutaja sessiooni ja seeläbi kasutaja õigustes veebirakenduses toiminguid teha. Peamiselt võivad CSRF rünnaku ohvriks langeda pangad ja teised veebikeskkonnad, mis tegelevad raha ja maksetega. CSRF rünnaku abil on võimalik läbi viia *SQL Injection* rünnak. [31]

Antud süsteemile oleks CSRF rünnak eesmärgistatud, kui soovitakse, kasutades näiteks *SQL Injection* rünnakut, saada kasutaja e-maili ja parooli. CSRF rünnaku vastaseks kaitseks on Spring Security XML-konfiguratsioonis defineeritud CSRF kaitse kujul `<csrf/>`, sest programmis on kasutusel Spring Security versioon 3.2.3. Alates Spring Secuirty versioonist 4.0 on CSRF rünnaku vastane kaitse vaikimisi lubatud [32]. Kõik

vormid, kus kasutaja sisestab enda kohta käivaid andmeid, on varustatud CSRF-loaga *(CSRF-token)*, mis saadetakse kaasa POST päringuga. Päringuga saadetud CSRF-luba peab vastama serveris genereeritule. Vastasel juhul päringut ei täideta.

3.4. Kasutajaliides

Süsteemi kasutajaliides on lihtne. Kasutajale kuvatakse vaid seda, mis on rakenduse kasutamiseks hädavajalik. Lähtutud on põhimõttest, et rakendus peab olema kasutatav kõikides seadmetes. Seetõttu on rakendus programmeeritud kohalduva *(responsive)* veebilehena. Veebilehte saab vaadata igas seadmes, millel on brauser ja internetiühendus. Veebikuva ilusamaks näitamiseks on kasutatud Google RaleWay fonti [33]. Kujundusel on kasutatud Bootstrap raamistikku.

Lisaks on pööratud tähelepanu olulistele pisiasjadele. Päevase kehakaalu sisestamiseks on võimalus kasutada tekstisisestust, kuhu kasutaja saab sisestada ükskõik milliseid märke. Kui kasutaja sisestab kehakaalu, mis on komaarv ja eraldab arvu murdosa komaga, teisendab süsteem (javascripti funktsioon) koma punktiks. Funktsioonid, mis töötavad murdarvudega, nõuavad, et murdosa on eraldatud punktiga. Allolev koodinäide illustreerib kirjeldatud loogikat.

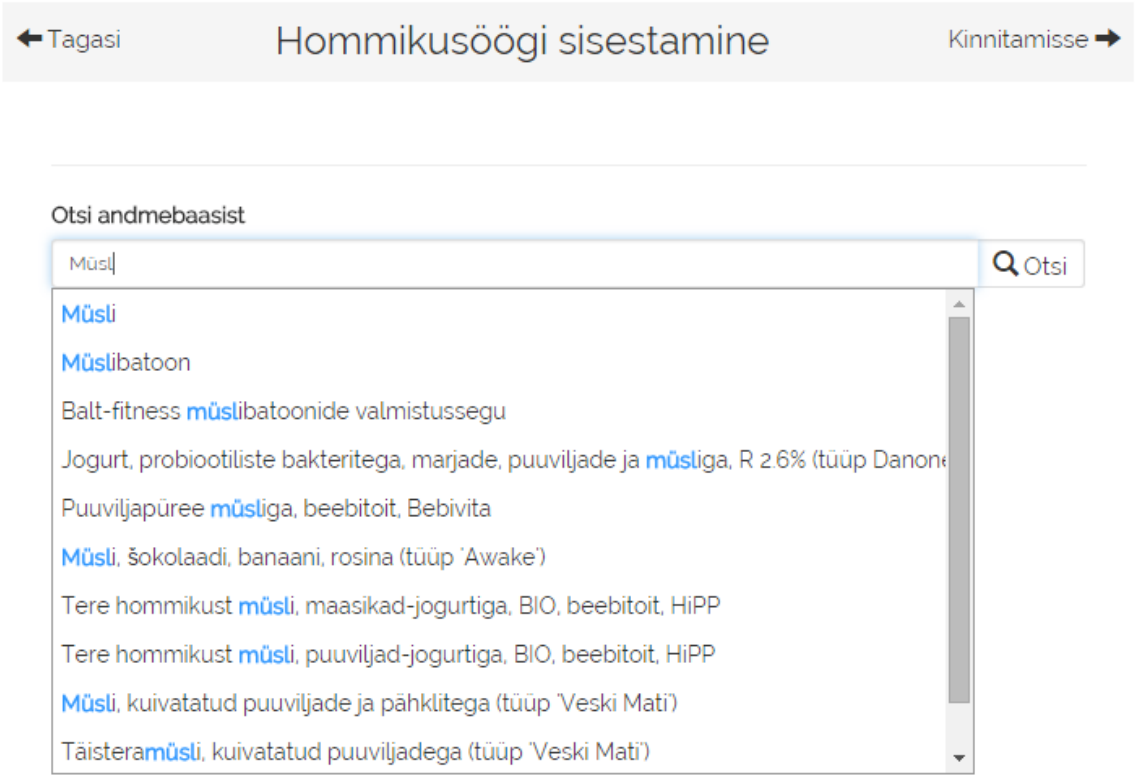
```
if (event.keyCode == 44) { // 44 tähendab koma (,)
  event.preventDefault();
  $(event.target).val($(event.target).val() + '.');
}
```

Järgnevatel joonistel 6, 7, 8 ja 9 on pildid prototüübi kasutajaliidesest.

Joonisel 6 olev kuva esineb päeva esimesel kasutaja sisselogimisel süsteemi. Siis tuleb määrata tänane kehakaal ja kõhu ümbermõõt. Aktiivsust iga päev üldjuhul ei muudeta.

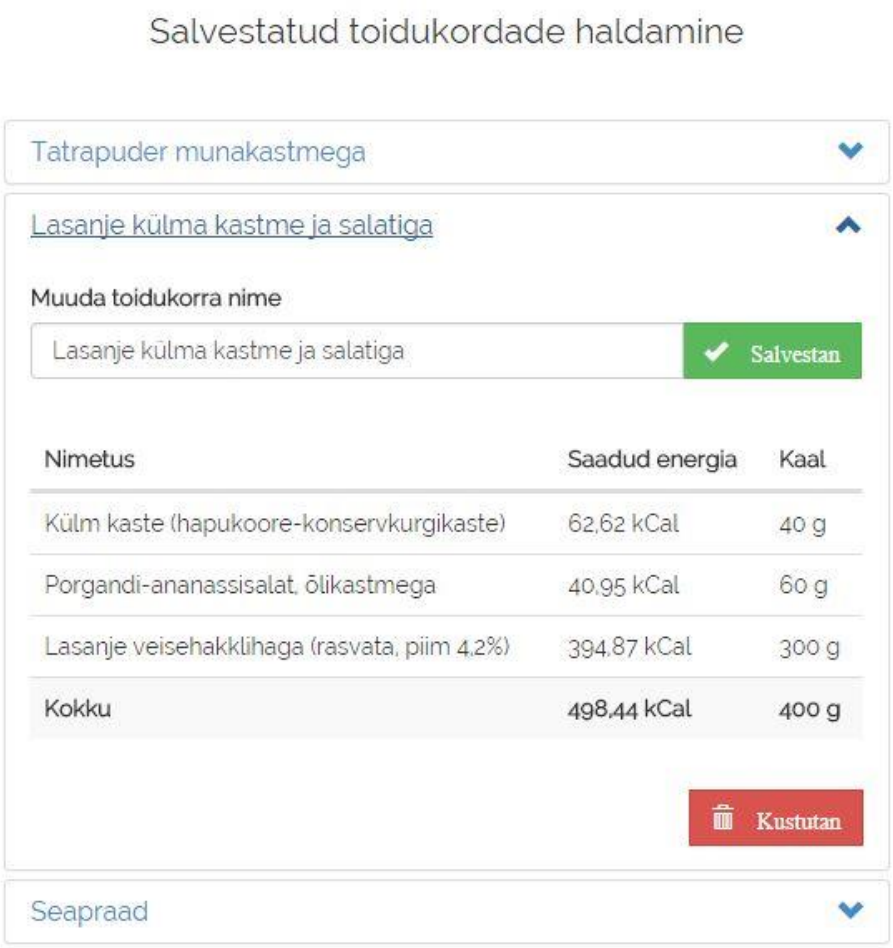
Joonis 6 Päeva alguses keha andmete sisestamine

Joonis 7 näitab andmebaasiotsingut. Kasutades jQuery-autocomplete teeki kuvatakse kasutajale jooksvalt otsingutulemusi kui kasutaja alles otsingusõne kirjutab.



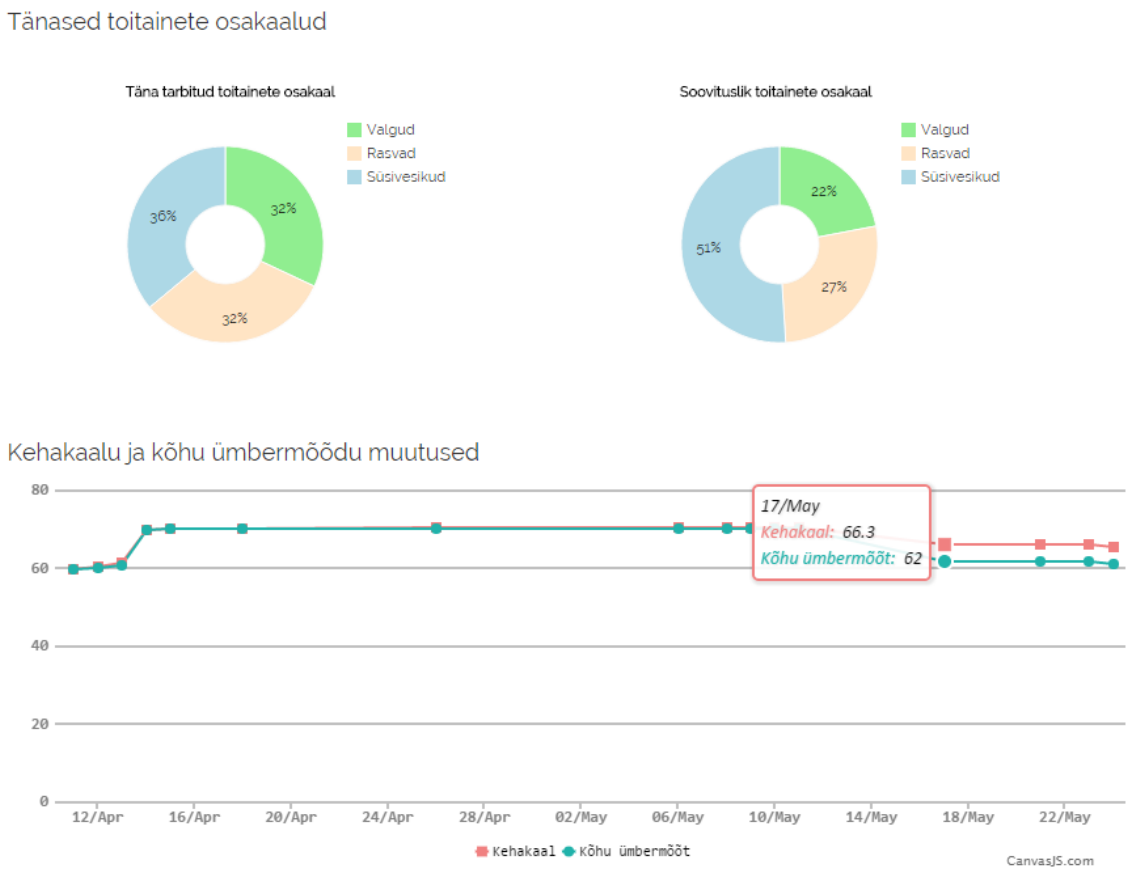
Joonis 7 jQuery-autocomplete näide

Joonis 8 näitab salvestatud toidukorra haldamist. Saab muuta salvestatud toidukorra nime ning salvestatud toidukorra süsteemist kustutada.



Joonis 8 Salvestatud toidukordade haldamine

Joonisel 9 on graafikud, mis näitavad kasutajale tarbitud toitainete suhet koos soovituslikuga ning kasutaja keha parameetrite ajalugu.



Joonis 9 Päevased toitainete osakaalud ja keha parameetrite muutuste ajalugu

4. Valminud süsteemi analüüs

Süsteemi kavandades püstitati hulk funktsionaalseid ja mittefunktsionaalseid nõudeid. Püstitatud nõuded said kõik realiseeritud. Programmeerimise käigus ilmnesid mitmed keerukused, mis said lahendatud. Üheks keerukamaks lahenduseks oli toidukorra lisamise sammude jada. Sammudes edasi-tagasi liikudes pidid kasutaja sisestatud, aga kinnitamata andmed püsima jääma. Andmebaasi kirjutatakse andmed pärast viimast sammu – kinnitamist. Teine keerukam protseduur oli kasutaja registreerimine, mille realiseerimiseks tuli tutvuda uute tehnoloogiatega ja need omavahel tööle panna.

Programmeerimiskeele valik osutus otstarbekaks. Java suurte laiendusvõimaluste tõttu oli programmeerimine lihtne, sest oli võimalus kasutada palju valmiskomponente. Lisaks on Javas mugav teha veaotsingut *(debug)*, sest arenduskeskkonnas on vastav võimalus sisse ehitatud. Alternatiivina olnuks võimalus rakendus programmeerida PHP-s või Pythonis. Lahenduse negatiivseks küljeks on vajadus internetiühenduse järele, sest rakendus asub serveris.

Süsteemi testiti kolme kasutajaga. Ilmnesid mitmed vead, mis kõrvaldati. Väikese ekraaniga seadmel jäi ülemine navigatsiooniriba ette otsingukastile, sest navigatsiooniriba oli määratud olema alati ekraani ülaosas nähtav. Toidu kaalu soovib süsteem grammides, kuid tekstilahtrisse saab lisada murdarve. Seetõttu ei läinud osa kaalusid süsteemi kirja, sest süsteemis on kaal täisarvu tüüpi. Lahenduseks kuvatakse kasutajale veateadet, kui kasutaja sisestab kaalu murdarvuna. Peale selle ei toimunud automaatset toidu otsingut andmebaasist, kui kasutaja vajutas soovitatud toidu nimele.

Testijatele meeldis süsteemi lihtsus. Toidukorra lisamiseks ei pea palju tööd tegema. Salvestatud toidukorda on võimalik kasutada põhjana ja sealt toitusid eemaldada ning juurde lisada. Peale selle meeldis testijatele rakenduse mobiiliversioon. Mobiilis keritakse ekraani vastavalt sellele, kuhu kasutaja pilku tahetakse suunata.

Lisaks leidis testijate poolt märkimist süsteemi ajamahukas kasutamine esimestel päevadel. Esmasel toidukorra sisestamisel ei paku süsteem kasutajale salvestatud toidukordi. Need tekivad kasutaja sisestatud toidukordadest. Seetõttu tuleb alguses kasutada palju andmebaasiotsingut. Seega oleks süsteemi üks võimalik edasiarendus luua keskne salvestatud toidukordade andmebaas, mis pakub valmiskujul toidukordasid kõigile kasutajatele. Toidukordade valik peaks olema lai, kust kasutaja saab endale meelepärased välja valida.

5. Kokkuvõte

Töös eesmärgiks oli realiseerida toitumispäeviku pidamist võimaldav süsteem, mis arvestab sportlaste vajadustega. Töö idee pärines sihtgrupilt. Põhirõhku pöörati võimalikult lihtsa kasutajaliidese tegemisele, kus toidukorra sisestamine ei võtaks palju aega.

Töö tulemusena valmis prototüüprakendus, mis realiseeriti Java veebirakendusena, kasutades Spring rakendusraamistiku. Veebirakendus on kasutatav kõigis brauseri ja internetiühendusega seadmetes sõltumata operatsioonisüsteemist. Rakendus võimaldab pidada toitmispäevikut ja näha kehakaalu ning kõhu ümbermõõdu muutuseid ajas.

Esimese järeldusena võib välja tuua vajaduse kasutajatesti järele, mille käigus avastati mitmed vead ja ebamugavused, millele ei osatud alguses tähelepanu pöörata. Suur osa ebamugavustest oli seotud mobiilivaatega, kuid leidus ka üksikuid funktsionaalseid pisivigu. Teiseks on kogu arendusprotsessi vältel oluline suhtlemine sihtgrupiga (tellijaga). Süsteemi nõuete kogumisel suheldi sihtgruppi kuuluvate sportlastega, et saada teada nende vajadusi. Kõige olulisemaks peeti võimalust näha tulemusi graafikutel ning rakenduse kasutatavust mobiilis. Lisaks osales üks sihtgrupi esindaja hilisemalt prototüübi kasutajatestis.

Süsteemi edasiste arenduste käigus tuleks kõigile kasutajatele pakkuda valmiskujul salvestatud toidukordasid. Peale selle võiks võimaldada süsteemi sisselogimist sotsiaalmeedia kontode abil. Süsteemi funktsionaalsuse ja mahu suurel kasvul oleks mõistlik automatiseerida osa kasutajaliidese testidest. Tervise arengu instituudi toitude andmebaasi uuenemise korral tuleks rakenduses kasutusele võtta andmebaasi uuem versioon.

6. Viited

[1] Pitsi, T., Kambek, L., Jõeleht, A. NutriData toidu koostise andmebaas, väljaanne 6. Tervise Arengu Instituut. 2014. Veebileht: http://www.nutridata.ee (02.05.2015)

[2] Pitsi, T., Kambek, L., Nutridata toitumise analüüsi programm. Tervise Arengu Instituut. 2010. Veebileht: http://www.nutridata.ee (02.05.2015)

[3] Kaloriarvuti. Fitness.ee. 2010. [WWW] http://www.fitness.ee/kaloriarvuti (02.05.2015)

[4] Myfitnesspal calorie counter app. Myfitnesspal Inc. 2015 [WWW] https://www.myfitnesspal.com/ (16.05.2015)

[5] Harris-Benedicti valem. Gottasport.com. 2014. [WWW] http://gottasport.com/weight-loss/71/harris-benedict-formula-for-women-and-men.html (02.05.2015)

[6] Happy Pregnancy uuring, Tanita kehakoostise monitor. Tartu Ülikool. 2012 – 2015. [WWW] http://www.happypregnancy.ut.ee/uuringus-osalejale/tanita-kehakoostise-monitor

[7] Java. Oracle Corporation. 2015. [WWW] http://www.oracle.com/technetwork/java/index.html (03.05.2015)

[8] Write once, run anywhere? *ComputerWeekly* 2002. [WWW] http://www.computerweekly.com/feature/Write-once-run-anywhere (03.05.2015)

[9] Java SE. Oracle Corporation 2015. [WWW] http://www.oracle.com/technetwork/java/javase/overview/index.html (03.05.2015)

[10] Tiobe index for April. *Tiobe Software BV* 2015. [WWW] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (03.05.2015)

[11] Apache Maven. 2015. [WWW] https://maven.apache.org/ (03.05.2015)

[12] Guide to naming conventions on groupId, artifactId and versioon. *Apache Maven*. 2015. [WWW] https://maven.apache.org/guides/mini/guide-naming-conventions.html (03.05.2015)

[13] Spring framework. 2015. [WWW] http://projects.spring.io/spring-framework/ (03.05.2015)

[14] Oliver White. Top 4 Java Web Frameworks Revealed: Real Life Usage Data of Spring MVC, Vaadin, GWT and JSF – *Zeroturnaround.* 2014. [WWW] http://zeroturnaround.com/rebellabs/top-4-java-web-frameworks-revealed-real-life-usage-data-of-spring-mvc-vaadin-gwt-and-jsf/ (03.05.2015)

[15] Java Server Pages technology. Oracle Corporation. [WWW] http://www.oracle.com/technetwork/java/javaee/jsp/index.html (03.05.2015)

[16] JspFragment. 2012. [WWW] https://tomcat.apache.org/tomcat-5.5-doc/jspapi/javax/servlet/jsp/tagext/JspFragment.html (03.05.2015)

[17] Reinier Zwitserloot. Project Lombok. 2015 [WWW] https://github.com/rzwitserloot/lombok

[18] Joda-time release 2.7. [WWW] http://www.joda.org/joda-time/ (03.05.2015)

[19] HIBERNATE - Relational Persistence for Idiomatic Java, release 4.2.19.Final. Hibernate Community. 2015. [WWW] http://docs.jboss.org/hibernate/orm/4.2/manual/en-US/html/ (03.05.2015)

[20] Ian Hickson, Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Edward O'Connor, Silvia Pfeiffer. HTML5. 2014 [WWW] http://www.w3.org/TR/html5/ (03.05.2015)

[21] Ecma-262 Standard. Introduction – *Ecma International.* 2011. [WWW] http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (03.05.2015)

[22] jQuery 2.1.3. The jQuery Foundation. 2015 [WWW] https://api.jquery.com/ (03.05.2015)

[23] Bootstrap 3. 2015. [WWW] http://getbootstrap.com/getting-started/ (03.05.2015)

[24] Thomas Kirda. Ajax Autocomplete for jQuery. 2015. [WWW] https://github.com/devbridge/jQuery-Autocomplete (04.05.2015)

[25] Google Charts. 2015 [WWW] https://developers.google.com/chart/ (04.05.2015)

[26] PostgreSQL 9.3.6 Documentation. The PostgreSQL Global Development Group. 2015. [WWW] http://www.postgresql.org/docs/9.3/static/index.html (04.05.2015)

[27] Rules and Privileges – *PostgreSQL 9.3.6 Documentation.* 2015. [WWW] http://www.postgresql.org/docs/9.3/static/rules-privileges.html (04.05.2015)

[28] Eugen Paraschiv. The Registration Process With Spring Security – *Baeldung*. 2014. [WWW] http://www.baeldung.com/registration-with-spring-mvc-and-spring-security (05.05.2015)

[29] Ben Alex, Luke Taylor The Security Namespace. Spring Security Reference. 2015. [WWW] http://docs.spring.io/spring-

security/site/docs/3.0.x/reference/appendix-namespace.html#nsa-http-attributes (05.05.2015)

[30] Johnny Shelley, Philip Stolarczyk bcrypt – *sourceforge*. 2002. [WWW] http://bcrypt.sourceforge.net/ (05.05.2015)

[31] Jeremiah Blatz. CSRF: Attack and Defense – *McAfee Inc*. 2011 [WWW] http://www.foundstone.com.au/uk/resources/white-papers/wp-csrf-attack-defense.pdf (05.05.2015)

[32] Ben Alex, Luke Taylor, Rob Winch. Cross Site Request Forgery (CSRF). 2015. [WWW] http://docs.spring.io/spring-security/site/docs/current/reference/html/csrf.html (05.05.2015)

[33] RaleWay. Google Fonts. 2015. [WWW] https://www.google.com/fonts#QuickUsePlace:quickUse (06.05.2015)

7. Lisad

7.1. BMR arvutamine

BMR arvutamine sõltub inimese soost, kaalust, pikkusest ja vanusest. Alljärgnevad valemid kirjeldavad BMR arvutamise mehe ja naise kohta.

Naised: $BMR = 655 + 9{,}6 * K + 1{,}8 * H - 4{,}7 * V$

Mehed: $BMR = 66 + 13{,}7 * K + 5 * H - 6{,}8 * V$

, kus K on vanus aastates, H pikkus sentimeetrites ja V vanus aastates.

7.2. Java klassi implementatsioon koos Lombok teegiga ja ilma

Järgnev tabel 4 võrdleb töös kasutatud klassi UserAuth, millel on kolm välja: kasutaja id, kasutaja e-mail ja kasutaja roll süsteemis. Mõlema implementatsiooni funktsionaalsus on sama, st kõigi kolme välja kohta on *getter* ja *setter* meetodid ja objekti kohta on implementeeritud *toString* meetod.

Tabel 4 Java klass Lombok teegiga ja ilma

Kasutades Lombok teeki ja annotatsioone `@Getter, @Setter, @ToString` ja `@AllArgsConstructor`	**Ilma Lombok teegita**

`@Getter`
`@Setter`
`@ToString`
`@AllArgsConstructor`
`public class UserAuth {`
 	
`private int userId;`
`private String email;`
`private String role;`
`}`
	`public class UserAuth{`
	
	`private int id;`
	`private String email;`
	`private String role;`
	
	`public UserAuth(int id, String email, String role) {`
	`this.id = id;`
	`this.email = email;`
	`this.role = role;`
	`}`
	
	`public int getId() {`
	`return id;`
	`}`

Kasutades Lombok teeki ja annotatsioone `@Getter, @Setter, @ToString` ja `@AllArgsConstructor`	**Ilma Lombok teegita**

	`public void setId(int id) {`
	`this.id = id;`
	`}`
	
	`public String getEmail() {`
	`return email;`
	`}`
	
	`public void setEmail (String email) {`
	`this.email = email;`
	`}`
	
	`public String getRole() {`
	`return role;`
	`}`
	
	`public void setRole(String role) {`
	`this.role = role;`
	`}`
	
	`@Override`
	`public String toString() {`
	`return "UserAuthRowMapper{" + "id=" + id + ", email='" + email + '\'' + ", role='" + role + '\'' + '}';`
	`}`
	`}`