

TALLINNA ÜLIKOOL

Haapsalu kolledž

Rakendusinformaatika õppekava

Ahti Irs

TUNNIPLAANI RAKENDUSE ARENDAMINE JA JUURUTAMINE  
TALLINNA ÜLIKOOLI HAAPSALU KOLLEDŽI NÄITEL

Diplomitöö

Juhendaja: Martti Raavel, MA

Haapsalu 2023

## SISUKORD

MÕISTED JA LÜHENDID .....	4
SISSEJUHATUS .....	7
1. TEOREETILINE ÜLEVAADE .....	9
1.1. Tunniplaani .....	9
1.2. Arenduses kasutatud tehnoloogiad .....	9
1.2.1. React .....	9
1.2.2. Node.js .....	11
1.3. Järjepidev integratsioon ja arendus (CI/CD) .....	11
1.4. Jätkusuutlik koodiarendus .....	12
2. METOODIKA .....	14
3. TUNNIPLAANI RAKENDUSE KIRJELDUS .....	15
3.1. Nõuded tunniplaani rakendusele .....	15
3.2. Valikpraktikas valminud rakenduse analüüs .....	15
3.3. Rakendusest puuduva funktsionaalsuse väljaselgitamine .....	17
4. ARENDUS .....	18
4.1. Funktsionaalsuse arendus .....	18
4.2. Arenduskeskkond .....	22
4.3. Projekti haldus .....	23
4.4. Koodi struktuur .....	24
4.5. Arenduse testkeskkond .....	25
5. JUURUTAMINE .....	27
5.1. Järjepideva integratsiooni ja arenduse mudel (CI/CD) .....	27
5.2. Arenduskeskkonna integratsioon .....	28
5.3. Testimine jätkuvas integratsioonis .....	29

5.4. Algandmete sisestus .....	30
5.5. Rakenduse edasine arendus .....	30
KOKKUVÕTE .....	32
SUMMARY .....	33
KASUTATUD KIRJANDUS .....	34
LISA 1. Küsimused intervjuueeritavatele .....	36

## MÕISTED JA LÜHENDID

**CSS** - (*Cascading Style Sheets*) on keel, mis võimaldab luua atraktiivse välimusega veebilehti. Selle abil saab määrata, kuidas erinevad elemendid veebilehel välja näevad. CSS kasutatakse stiilide ja kujunduste loomiseks, mis mõjutavad teksti suurust, värvi, tausta, asendit ja palju muud. See võimaldab veebilehtedel olla visuaalselt meeldivad ja kasutajasõbralikud. CSS annab veebiarendajatele võimaluse kohandada veebilehtede välimust vastavalt oma soovidele ning eristada neid teistest veebilehtedest. (Mozilla Corporation, s.a.)

**Docker** – on tööriist, mis võimaldab hallata ja käivitada Docker konteinereid (Docker Inc., s.a.).

**Docker konteiner** - on nagu virtuaalne kast tarkvarale. See "kast" sisaldab kõike, mida tarkvara vajab töötamiseks - koodi, süsteemitööriistu, raamatukogusid ja seadeid. Kui tarkvara on pakendatud Dockeri konteinerisse, siis see töötab ühtemoodi igas arvutis või serveris, kuhu Docker on installitud. Dockerit kasutades ei pea arendajad muretsema erinevate seadistuste pärast erinevates arvutites - kui see töötab ühes konteineris, peaks see töötama kõigis. (Docker Inc., s.a.)

**Docker image** - on nagu „juhiste“ kogu Dockeri konteineri loomiseks. Selles on kõik juhiseid, mida on vaja tarkvara töötamiseks vajaliku konteineri loomiseks. See hõlmab koodi, süsteemitööriistu, raamatukogusid ja seadeid. Kui Dockeri image on loodud, saab seda kasutada Dockeri konteinerite loomiseks. Iga konteiner, mis on loodud sama image põhjal, töötab samamoodi, sest see järgib samu "juhiseid". Dockeri pilti on kerge ja lihtne jagada, nii et arendajad saavad oma tarkvara hõlpsasti levitada ja teised saavad seda hõlpsasti kasutada. Piltide abil saavad arendajad tagada, et nende tarkvara töötab ühtemoodi igas arvutis või serveris, kus Docker on installitud. (Docker Inc., s.a.)

**Eesrakendus** – (*Frontend*) eesrakendus on kõik, mida kasutaja näeb ja millega suhtleb, kui nad klõpsavad lingil või sisestavad veebiaadressi. See on veebirakenduse kliendipoolne osa. Eesrakendus hõlmab kõike, mida kasutaja näeb ja millega suhtleb, sealhulgas nupud, tekst, värvid, pildid, videod ja palju muud. Eesrakenduse arendaja ülesanneteks on veebilehtede loomine, kasutajaliideste ehitamine, veebilehtede optimeerimine erinevatele seadmetele ja

brauseritele ning veebilehtede ligipääsetavuse ja jõudluse parandamine. Eesrakenduse arendamiseks kasutatakse tavaliselt HTML-i, CSS-i ja JavaScripti, samuti erinevaid raamistikke ja teekide, nagu React, Redux, Vue ja Angular. (Lemonaki, 2022).

**HTML** – (HyperText Markup Language) kasutatakse veebilehel esitatud sisu struktuuri määratlemiseks. Seda keelt iseloomustavad erinevad elemendid, mida kasutatakse sisu erinevate osade ümbritsemiseks, et neid esitada või käitada soovitud viisil. Ümbritsevad sildid võimaldavad muuta sõnu või pilte ümbersuunavaks lingiks mujale, määrata sõnade välimust kursiiviks ning muuta, suurendada või vähendada fondi suurust ja palju muud. (Mozilla Corporation, s.a.).

**JSON** – (JavaScript Object Notation) on andmevahetusformaad, mis on optimeeritud nii inim- kui ka masinaloetavuse jaoks. See formaat tugineb JavaScripti programmeerimiskeele alamhulgale, kuid on täielikult keelest sõltumatu, kasutades konventsioone, mis on laialdaselt tuntud C-keele perekonna programmeerijate seas. JSON andmevahetusformaati toetavad lisaks veel programmeerimiskeeled nagu C, C++, C#, Java, JavaScript, Perl, Python ja paljud teised. Nende omaduste tõttu on JSON ideaalne andmevahetuskeel, mis sobib erinevate rakenduste ja süsteemide vaheliseks suhtluseks. (json.org, s.a.)

**MySQL** - on avatud lähtekoodiga relatsiooniline andmebaasihaldussüsteem, mis on mõeldud andmete organiseerimiseks ja haldamiseks. Relatsioonilise andmebaasi struktuur võimaldab andmeid paigutada ühte või mitmesse tabelisse, kus andmeüksused on omavahel seotud. Need seosed aitavad kaasa andmete struktuursele korraldusele ja tõhusale haldamisele. (Oracle, s.a.)

**Rakendusliides** - API (*Application Programming Interface*) API-de abil saab luua suhtlust kahe tarkvarakomponendi vahel, kasutades selleks määratlusi ja protokolle. Tavaliselt toimub see suhtlus kliendi ja serveri vahel, kus päringu saatja on klient ja vastuse saatja on server. API-de roll tarkvaraarenduses on oluline, kuna need aitavad integreerida uusi rakendusi olemasolevate süsteemidega, kiirendada innovatsiooni, laiendada teenuseid erinevatele platvormidele ja hõlbustada hooldust. API-d võivad olla erinevat tüüpi, privaatset, avalikud, partneritele mõeldud või koosnevad mitmest eri tüüpi API-st. (Amazon Web Services, s.a.)

**Repositoorium** - on hoidla, mis sisaldab tarkvaraprojekti. Need võivad hõlmata erinevaid elemente, nagu koodifaile, dokumentatsiooni, andmebaase, pildifaile ja skripte, mis kõik on seotud konkreetse projekti või süsteemiga. Üks levinumaid platvorme, kus arendajad saavad luua ja hallata repositooriume, on GitHub. See platvorm võimaldab arendajatel jälgida oma koodi muudatusi, teha koostööd teiste arendajatega ja jälgida probleeme, muuhulgas paljusid muid funktsioone. Repositooriumid võivad olla kas avalikud või privaatsed. Avalikud repositooriumid on kõigile nähtavad ja neid saab kasutada igaüks, kes soovib uurida projekti koodi või panustada sellesse. Privaatsed repositooriumid nähtavad ainult neile, kellele on antud juurdepääsuõigused. (GitHub, Inc, s.a. c)

**SQL** - on struktureeritud päringukeel ehk SQL on programmeerimiskeel, mida kasutatakse andmete manipuleerimiseks relatsioonilises andmebaasis. SQL võimaldab programmeerijatel luua, muuta ja pärida andmeid, samuti reguleerida kasutajate juurdepääsuõigusi andmebaasile. See keel on MySQL-i ja teiste relatsiooniliste andmebaasihaldussüsteemide keskmes, võimaldades tõhusat andmehaldust. (Oracle, s.a.)

**Tagarakendus** - (*Backend*) tagarakendus tegeleb tehnoloogiatega, mis vastutavad kasutaja andmete salvestamise ja turvalise manipuleerimise eest. See on veebirakenduse serveripoolne osa. Tagarakendus hõlmab kõiki peidetud sisemisi töid ja protsesse, mis tagavad eesrakenduse korrektse ja sujuva toimimise. Tagarakenduse arendaja ülesanneteks on andmebaaside loomine, haldamine ja hooldamine, serverite ehitamine ja hooldamine, API-de (Application Programming Interface) loomine ja haldamine ning andmete valideerimine. Tagarakenduse arendamiseks kasutatakse tavaliselt serveripoolseid programmeerimiskeeli, nagu PHP, Ruby, Python, Java ja JavaScript (Node.js abil), samuti erinevaid raamistikke, nagu Ruby on Rails, Django, Flask ja Express. (Lemonaki, 2022).

**Winston** - on populaarne logimisraamistik Node.js rakendustele. See on paindlik, avatud lähtekoodiga ja sellel on suur toetav kogukond. Winston võimaldab luua logisid erinevatel tasemetel, nagu viga, hoiatus, info, http, verbose, debug ja silly. (Reflecting.io, 2022)

.

## SISSEJUHATUS

Haapsalu kolledžis kasutatav tunniplaanisüsteem on suuremate muudatusteta kasutusel olnud juba aastast 2015, millal tehti sellekohane diplomitöö (Ling, 2015). Kolmanda kursuse alguses läbitud valikpraktikas alustasime uue tunniplaani süsteemi loomist ja selle arendamiseks viisime tudengite ja õppejõudude seas läbi kasutusel oleva tunniplaaniga seotud küsitluse. Selgus, et hetkel kasutusel oleval tunniplaanil puudub osaliselt tudengite ja õppejõudude poolt oodatav funktsionaalsus. Sellest tulenevalt pakkus kool mulle võimalust arendada ja juurutada diplomitöö raames uus tunniplaani rakendus kasutades valikpraktikas tehtud tööd.

Valikpraktika uurimistööle tuginedes saab praegu kasutuses oleval tunniplaanil selgelt välja tuua järgnevad probleemid:

- aegunud disain;
- puudub integratsioonide võimalus teiste süsteemidega;
- ei võimalda kuvada koduseid töid ja tähtaegu;
- ei ole kasutatav mobiilsetel seadmetel.

Diplomitöö eesmärk on arendada valikpraktikas loodud tunniplaani kood tasemeni, mis võimaldaks rakenduse reaalselt kolledžis kasutusele võtta ja seejärel ka praktiliselt juurutada kooli IT taristus. Uuel tunniplaani süsteemil peab olema teostatud kõik küsitluste põhjal saadud kasutajate soovitud olulisimad omadused. Teise eesmärgina peaks arenduse käigus jälgima parimat praktikat koodi kirjutamisel, arenduskeskkonnas hoidmisel, dokumenteerimisel ja testimisel. Loodav kood peab võimaldama võimalikult vähese vaevaga arendust jätkata järgmiste aastate tudengitel ja looma omamoodi näidise või standardi kuidas, ja milliste vahenditega koodi ning dokumentatsiooni kirjutada ja vormindada.

Lähtuvalt diplomitöö eesmärgist sõnastasin järgmised küsimused:

- Mis on valikpraktika raames arendatud tunniplaani prototüübi funktsionaalsuses puudu, et seda saaks koolis kasutusele võtta?
- Kuidas juurutada koodi kooli infrastruktuuri?
- Kuidas kirjutada hästi hallatavat koodi?

Diplomitöö küsimuste põhjal püstitasin järgnevad ülesanded:

- Välja selgitada ja teostada vajalikud lahendused rakenduse kasutuselevõtuks.
- Uurida kuidas kasutatav struktuur ja valitud tehnoloogiad võimaldavad pidevat integratsiooni ja juurutust kooli IT taristuga.
- Välja selgitada hea koodikirjutamise parimad praktikad.

Töö koosneb sissejuhatusest, kuuest peatükist ja kokkuvõttest. Esimeses peatükis kirjeldan uurimistöö teoreetilist alust mis käsitleb tunniplaanisüsteemi vajalikkust ja tunniplaanisüsteemide ajas muutunud nõudeid, kaasaegset IT taristu ülesehitust ja arenduskeskkondade ja süsteemide hetkesesisu. Teises peatükis käsitlen diplomitöös püstitatud küsimustele vastuste leidmise meetodikat. Kolmandas käsitlen koodi ülesehitust, üldpõhimõtet ja andmestruktuuri ning analüüsin puuduvat funktsionaalsust. Neljandas peatükis vaatlen koodi arendust etappide kaupa. Viiendas peatükis tegelen koodi arenduskeskkonna integreerimist kooli IT taristuga järjepideva integratsiooni ja arenduse (edaspidi CI/CD) meetodil.



## **1. TEOREETILINE ÜLEVAADE**

### **1.1. Tunniplaan**

Tunniplaan on koolide jaoks õppetöö korraldamisel oluline tööriist. Õppeprotsessi sujuvaks läbiviimiseks on vajalik, et tunniplaan oleks hoolikalt läbimõeldud arvestamaks õppejõudude ning tudengite ajalisi võimalusi ning iseärasusi. Oskuslik tunniplaani koostamine aitab jaotada kooli ressursse parima võimaliku variatsiooni järgi ning vältida ajakonflikte, mis võivad tekkida ruumi või õppejõu broneerimisel korraga samasse ruumi või samale ajale. Seega pole tunniplaan lihtsalt tabel ruumi nimedega vaid kätkeb endas paljude detailide kogumit, mis on seatud parimasse võimalikku kombinatsiooni, seetõttu on selle koostamine üldjuhul üsnagi keerukas ning mahukas protsess. (Enenche, 2019)

Tunniplaani kasutamine on aja jooksul muutunud. 30 aastat tagasi piisas lihtsalt paberilehel seinale kleebitud tabelist. Tänapäeval on tunniplaan muutunud iseenesestmõistetavaks ja igaühele kättesaadavaks internetis vaadeldavaks leheks, mida oleks mugav jälgida erinevatelt internetti ühendatud seadmetelt. (Enenche, 2019)

### **1.2. Arenduses kasutatud tehnoloogiad**

Tehnoloogia valiku tingis põhiliselt juba valikpraktikas etteantud nõuded. Kuna õppekava kuraatori sõnul keskendutakse vähemalt lähimas tulevikus esirakenduste raamistike hulgast just Reacti õpetamisele, siis võtsime ka tunniplaani arenduse esirakenduses kasutusele Reacti raamistiku.

#### **1.2.1. React**

React on avatud lähtekoodiga JavaScripti raamistik, mis võimaldab arendajatel ehitada kasutajaliideseid veebirakendustele. Reacti peamine eesmärk on pakkuda kasutajaliidese loomiseks efektiivset ja deklaratiivset viisi, võimaldades arendajatel jagada oma koodi taaskasutatavate komponentide kujul. (Meta Open Source, s.a.)

Reacti kood koosneb komponentidest, millel on oma seisund (state) ja omadused (props). Komponentide seisund määrab nende käitumise ja väljundid, omadused aga võimaldavad

andmete edastamist komponentidele, mida saab muuta dünaamiliselt. Reacti põhimõte seisneb selles, et komponentide seisundit ei muudeta otse, vaid selle asemel genereeritakse uus seisund, mis asendab vana. (samas)

React võimaldab kasutajaliidese loomist kaasahaaravamaks muuta kasutades selliseid funktsioone nagu animatsioonid, staatiline analüüs, optimeerimine jne. Reacti kasutamiseks on vaja tunda JavaScripti ning mõningaid tööriistu, nagu näiteks Node.js ja npm. (samas)

React on üks populaarsemaid JavaScripti raamistikke, mis võimaldab arendajatel ehitada kasutajaliideseid, mis töötavad erinevates veebibrauserites. Reacti ökosüsteemis on palju erinevaid tööriistu ja biblioteeke, mis võimaldavad kasutajaliidese ehitamist lihtsamaks ja tõhusamaks muuta. (samas)

Reacti üks suurimaid eeliseid on virtuaalne DOM (Document Object Model ehk dokumentide objektimudel). See on JavaScripti objektide hierarhia, mis esindab veebilehe või rakenduse HTML-i ja CSS-i struktuuri. Virtuaalne DOM on Reacti poolt loodud tulemus, mille abil haldab React kasutajaliidese komponente ja nende seisundit. (Grider, s.a.).

Virtuaalne DOM töötab nii, et iga kord kui komponendi seisund muutub, genereerib React uue virtuaalse DOM-i. Seejärel võrdleb React uut ja vana virtuaalset DOM-i ning määrab, millised muudatused on vajalikud, et uuendada tegelikku DOM-i vastavalt uuele seisundile. Seejärel rakendab React muudatused tegelikus DOM-is. (samas)

Virtuaalse DOM-i kasutamine annab mitmeid eeliseid. Esiteks võimaldab see Reactil hoida rakenduse seisundit ainult JavaScriptis, ilma et oleks vaja käsitsi manipuleerida tegeliku DOM-iga. See võimaldab Reactil töötada kiiremini ja vähendada andmeliiklust võrreldes traditsioonilise lähenemisega, kus oleks vaja otse manipuleerida tegeliku DOM-iga. (samas)

Virtuaalse DOM-i kasutamine võimaldab ka optimeerimist, kuna React saab arvutada kõige efektiivsemad viisid muudatuste rakendamiseks tegelikus DOM-is. See vähendab rakenduse reageerimisaega ja parendab kasutajakogemust. (samas)

### **1.2.2. Node.js**

Node.js on avatud lähtekoodiga serveripoolse JavaScripti keskkond, mis võimaldab arendajatel ehitada võrgurakendusi. Node.js põhineb Google'i välja töötatud V8 JavaScripti mootoril, mis võimaldab JavaScripti töötada serveris. (OpenJS Foundation, s.a.)

Node.js võimaldab serveri koodi kirjutada JavaScriptis, mis võimaldab arendajatel kasutada üht keelt nii kliendipoolsel kui serveripoolsel arendamisel. Node.js võimaldab arendajatel kirjutada serveripoolseid rakendusi, et hõlbustada suure liiklusega võrgurakenduste arendamist ning skaleerimist. Node.js abil saab arendada ka suurema jõudlusega võrgurakendusi, kuna see kasutab asünkroonset programmeerimist, mis võimaldab serveril töödelda mitut taotlust üheaegselt. (samas)

Node.js üks häid omadusi on moodulipõhine lähenemine, võimaldades arendajatel kasutada olemasolevaid moduleid, et kiirendada arendustööd. Suureks eeliseks on Node.js omadus rakendust käivitada erinevates populaarsetes operatsioonisüsteemides nagu Linux, Windows või macOS. (samas)

### **1.3. Järjepidev integratsioon ja arendus (CI/CD)**

Pidev integratsioon ja arendus on tarkvaraarenduse praktika, mis sisaldab sagedast koodi muutust jagatud repositooriumis. Kuna koodi muudatused sagedaste uuenduste vahel on väiksemad siis vähendab see võimalik vigade leidmise aega. Kui arendajaid on mitu, siis lihtsustab oluliselt ka koodide liitmist, kuna võrreldavate ridade arv on väike. Peale koodi uuendust repositooriumis on võimalik koodi testida nii kodeerimise stiili, teha turvakontrollid ja teha funktsionaalsuse teste. Kui kontrollid on läbitud siis automatiseeritud töövoog juurutab muudatustega rakenduse kasutaja serveris automaatselt. Tunniplaani rakenduse arendamisel kasutan selleks Github'i poolt pakutavat Github Action pakutavaid võimalusi. (Github, s.a. a)

#### 1.4. Jätkusuutlik koodiarendus

Intervjuust rakendusinformaatika kuraatoriga tuli selgelt esile soov juurutatavat tunniplaani rakendust järgnevatel aastatel edasi arendada. Sellest tulenevalt on rakenduse ülesehitus ja koodi loetavus määrava tähtsusega. (Raavel, intervjuu 06.03.2023)

Robert C. Martini raamat "*Clean Code - A Handbook of Agile Software Craftsmanship*" toob esile, et puhas kood on kood, mis on kirjutatud nii, et teised arendajad saavad seda lihtsalt lugeda, mõista ja muuta. Selle saavutamiseks soovitab ta järgida mitmeid põhimõtteid ja parimaid tavaid, sealhulgas hoolikalt valitud nimede kasutamist, funktsioonide ja klasside lühidust, ühe vastutusalaga põhimõtte järgimist, vähemate argumentidega funktsioone. Selleks, et arendada jätkusuutlikku ja puhast koodi on üheks oluliseks tööriistaks koodi stiiljuhend. Hea koodi kirjutamisel tuleb tähelepanu pöörata järgmistele punktidele:

- **Loetavus:** Kood peab olema korralikult kirjutatud ja korraldatud, et teised arendajad saaksid seda kiiresti mõista ja muuta. See tähendab, et nimed peavad olema selged ja kirjeldavad, kood peab olema loogiliselt korraldatud, ja taustinfo või seletused peavad olema kommentaarides.
- **Hooldatavus:** Kood peab olema kergesti hooldatav. Kui kood on korralikult kirjutatud ja korraldatud, saab uusi funktsioone kiiremini ja vähemate vigadega lisada. Samuti on vigade parandamine lihtsam, kui kood on loetav ja hästi korraldatud.
- **Kvaliteet:** Hea stiiljuhend aitab hoida koodi kvaliteeti. Kui arendajad järgivad sama stiiljuhendit, on kood järjepidev ja kvaliteetsem. Kui kõik kasutavad sama koodi stiili, on tõenäoline, et vigu avastatakse ja parandatakse kiiremini.
- **Koostöö:** Kui kõik meeskonna liikmed kasutavad sama stiiljuhendit, on koostöö palju lihtsam. Koodi ülevaatamine ja mõistmine on kiirem, kui kõik kasutavad sama stiili. See tähendab, et meeskond saab kiiremini töötada ja vähem aega raisata koodi mõistmisele. (Martin, 2009)

Seega võiks öelda, et järjepidevalt sama stiiliga kergelt hallatavat koodi on võimalik erinevatel arendajatel eri aegadel kirjutada ainult siis kui nad järgivad sama stiili juhendit.

Kui koodi arendamises on aja jooksul tekkinud olulisi muudatusi tuleks juhend kaasajastada ja kood kaasajastatud juhendi kohaselt refaktoreerida. (samas)

Kuna Haapsalu kolledžil pole stiilijuhendit siis kirjutasin ja refaktoreerisin koodi lähtudes W3school ja Google soovitustest. Samas ka koostas Google<sup>1</sup>, W3School<sup>2</sup> ja raamatu „Clean Code“ poolt väljatoodud olulisimate nõuannete põhjal ka javascripti arenduse soovitusliku juhendi. Loodud juhendi järgimine peaks tagama tunniplaani arenduse koodi lihtsa loetavuse. Stiilijuhise koostamisel võtsin Google, W3School ja raamatu „Clean Code“ poolt väljatoodud olulisimad soovitusel. Koostatud juhend pole kindlasti veel täielik ja vajaks edaspidist täiendamist ning aegajalt ka kaasajastamist. Juhendi loomisel võtsin arvesse ka õppekava kuraatori soovitusi ning juhend on kuraatori poolt üle vaadatud ja testitud. Koostatud stiilijuhis asub repositooriumis <https://github.com/tluhk/HK-Tunniplaani>.

---

<sup>1</sup> <https://google.github.io/styleguide/jsguide.html#features-local-variable-declarations>

<sup>2</sup> [https://www.w3schools.com/js/js\\_conventions.asp](https://www.w3schools.com/js/js_conventions.asp)

## 2. METOODIKA

Diplomitöö eesmärgi saavutamiseks vajaliku info saamiseks kasutasin uurimisviisina struktureerimata intervjuerimist. Kokku viisin läbi kolm intervjuud:

- Tunniplaanisüsteemi funktsionaalsusnõuete ja tehnoloogia osas viisin läbi intervjuu Haapsalu kolledži rakendusinformaatika kuraatoriga (Raavel, intervjuu 06.03.2023).
- Kooli IT taristus tunniplaani juurutamiseks vajaliku informatsiooni hankimiseks viisin läbi intervjuu Haapsalu kolledži arvutivõrgu administraatoriga (Pulst, intervjuu 25.04.2023).
- Kolmanda intervjuu viisin läbi kooli haldustöötajaga kelle igapäevatööks on tunniplaani administreerimine. Intervjuu eesmärk oli saada kinnitust, et kõik igapäevatööks minimaalselt vajalik funktsionaalsus on rakenduse nõuetes olemas. (Kasesalu, intervjuu 28.04.2023)

### **3. TUNNIPLAANI RAKENDUSE KIRJELDUS**

Selles lõigus annan ülevaate varasemalt valikpraktika raames valminud tunniplaani üldisest struktuurist, funktsionaalsusest ja disainist. Selgitan ka andmebaasi andmestruktuuri ja tabelite omavahelisi seoseid. Analüüsin mis on puudu nõutud diplomitöö eesmärgiks seatud funktsionaalsusest.

#### **3.1. Nõuded tunniplaani rakendusele**

Nõuete väljaselgitamiseks viisin läbi intervjuu Haapsalu kolledži Rakendusinformaatika kuraatoriga. Intervjuu kokkuvõttena sain koostada nimekirja omadustest, mida peab uus rakendus kindlasti võimaldama, enne kui seda saab juurutada igapäevasesse kasutusse. Intervjuust selgus ka kasutatavate tehnoloogiate määratlus. Üks uue tunniplaani arenduse eesmärk on viia selle edasine arendamine läbivaks iga aastaseks protsessiks, siis tuleb kasutada tehnoloogiaid, mille õpetamisele lähiaastatel põhiliselt pühendutakse. (Raavel, intervjuu 06.03.2023)

Intervjuude alusel koostatud nõuete nimekiri:

- rakendus peab olema realiseeritud eesrakenduse ja rakendusliidesena;
- kuvama, tunniplaani elemente (aeg, koht, lektor, ruum);
- kuvama, kodutöid ja tähtaegu (autoriseeritud tlu.ee kasutajale);
- võimaldama autentimist gmaili aadressidega;
- autoriseeritud halduril võimaldama kõike muuta ja lisada;
- autoriseeritud lektoril võimaldama lisada ja redigeerida kodutöid;
- rakendus peab võimaldama logida ja salvestada rakendusliidesesse saabunud päringuid. (sammas)

#### **3.2. Valikpraktikas valminud rakenduse analüüs**

Valikpraktika raames alustatud tunniplaani rakendus sai oma baastehnoloogia nõuded kooli poolt ja funktsionaalsuse nõuded selgitasime välja tudengitele ja õppejõududele elektrooniliselt edastatud küsimustikuga. Koostasime kasutajalood ja töötasime välja

Tallinna Ülikooli Haapsalu kolledži stiiliraamatuga ühilduva veebirakenduse prototüübi. Küsitluse tulemusena ja tunniplaaniga administreerimisega tegelevate kolledži töötajatega saime kinnitatud olulisimad nõuded, mis olid järgnevad:

- iseseisvate tööde jälgimise lisamise võimalus;
- filtrite säilitamine ka ilma sisse logimata;
- õppeinfo kast peab toetama videoloengu lisamist ja vaatamist;
- õppeinfos peab nägema ka ainekaarti;
- tunniplaanis peaks kasutajale kuvama kui on lisatud iseseisvad tööd või videoloeng;
- värvitoonide ja kirjastiilide vastavus Tallinna Ülikooli stiiliraamatuga.

Valikpraktika raames sai sellel hetkel seatud enamik olulisematest nõuetest ka realiseeritud ja testitud. Minu roll valikpraktika meeskonnas oli projektijuht ja rakendusliidese arendaja.

Valikpraktika raames valminud tunniplaani veebirakenduses võimaldas kasutada erinevaid rolle kuid puudus võimalus kasutajat autentida ja autoriseerida (joonis 1).

The screenshot displays the TLÜ Haapsalu kolledž timetable system. On the left, there is a calendar for January 2023 with a date range of 29.01.2023 - 01.07.2023. Below the calendar are filters for 'Täna', 'Homme', 'Nädal', and 'Semester'. On the right, the timetable for 'Neljapäev 02. veebruar 2023' is shown. It includes a table with columns 'AEG', 'ÖPPEAINE', 'ÖPPEJÕUD', and 'RUUM'. The courses listed are 'Käsitöötehnoloogiad ja disain 1. kursus', 'Liiklusohutus 1. kursus', 'Rakendusinformaatika 1. kursus', and 'Tervisejuht 2. kursus'. The interface also features a search bar at the top and a 'Logi sisse' button.

Joonis 1. Valikpraktikas valminud tunniplaani veebikuva

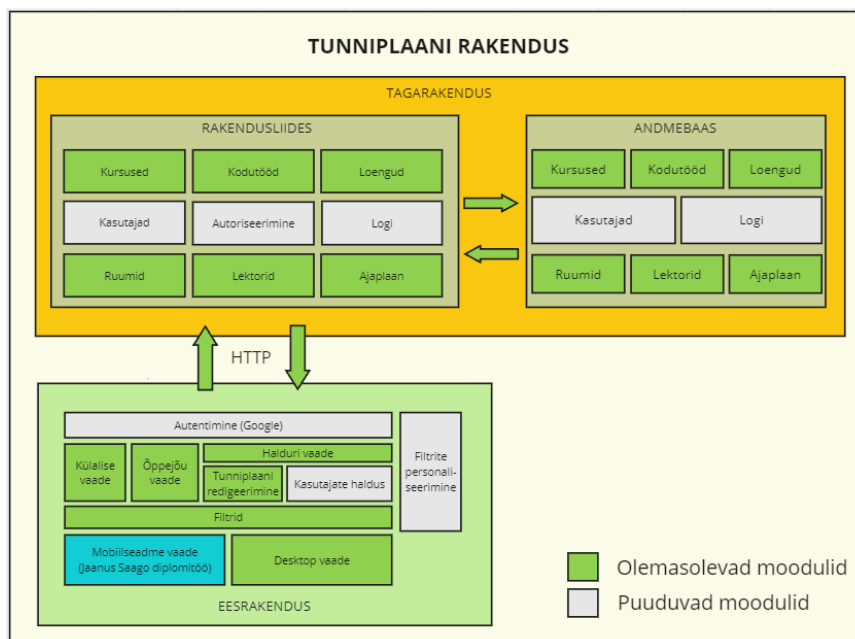


### 3.3. Rakendusest puuduva funktsionaalsuse väljaselgitamine

Eelnevalt loodud rakenduses on küll täidetud enamus valikpraktika ajal seatud eesmärgid, kuid rakenduse kasutusse võtmiseks seatud nõuetele see veel ei vasta. Tunniplaani kasutuselevõtuks tuleb arendada rakendusele lisaks olemasolevale järgnevad funktsionaalsused:

- autentimine Google kontodega;
- autoriseerimine Google mailiaadressiga;
- kasutajate haldusliides;
- filtrite seisu salvestumine;
- tegevuste logimine.

Analüüsi tulemusena saadud info põhjal koostas in joonise, mis iseloomustab valmis tunniplaani rakendust moodulite kaupa esitatuna. Rakendus koosneb kolmest suuremast osast, ees- ja tagaliidesest ning andmebaasi tabelitest. (joonis 2).



Joonis 2. Tunniplaani ülesehitus

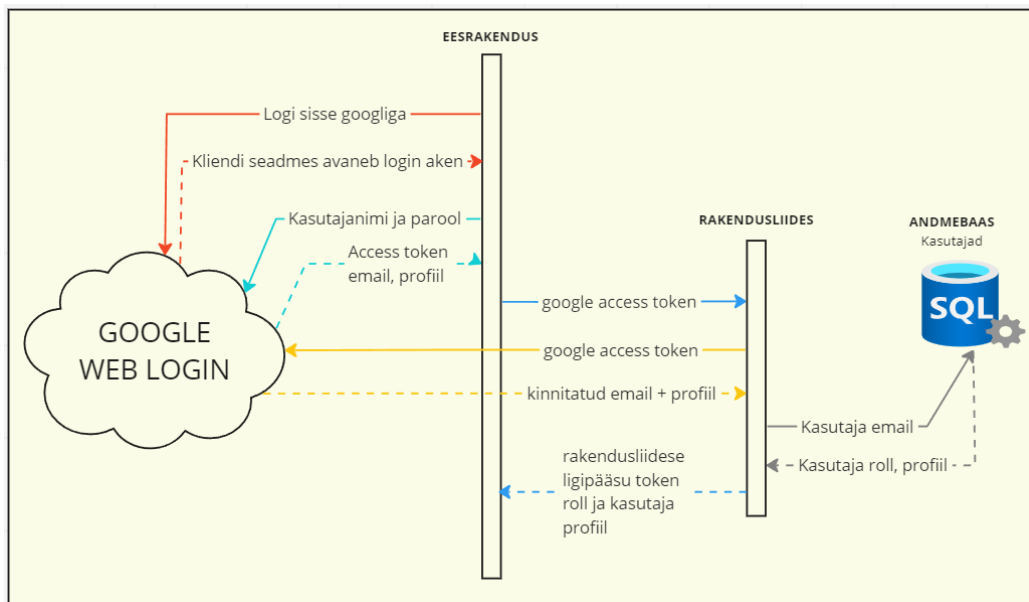
Puuduva funktsionaalsuse analüüsi tulemus on koodiarenduse ülesanneteks.

## 4. ARENDUS

### 4.1. Funktsionaalsuse arendus

Arenduse eesmärgid tulenesid puuduva funktsionaalsuse kaardistamisest. Alustasin nende teostamist kasutajate autentimisest ja autoriseerimisest. Tellija soovil tuli esmalt teostada kasutaja autentimine Google konto abil, kuna Haapsalu kolledž kasutab emailide haldamiseks Google teenust. Arenduse jaoks vajaliku info sain Google arendajatel mõeldud veebilehelt (Google, s.a. a). Probleeme tekitas asjaolu, et Google muutis oma pakutavat teenust, nii et varasemad teenusekasutajad toimivad aga uued peavad alates aprillist 2023 kasutama uut autentimis teenust (Google, s.a. b), aga viidatud juhendid kehtisid vana meetodi kohta ning paljud lingid Google enda juhendites lihtsalt ei toimi.

Lahenduse leidsin Udemy veebikoolitusest (Shetty, 2022), kus seletati teenuse olemus hästi lahti ja mille abil leidsin ka õiged pöörduspunktid. Toetudes saadud infole teostasin autentimise, kus eesrakendus küsib Googlelt kasutaja profiili ja saab vastuseks profiili koos ligipääsu võtmega (*Google Access token*). Seejärel edastab eesrakendus ligipääsu võtme rakendusliidesele, mis uuesti küsib Googlelt kasutades ligipääsu võtit, mis on kasutaja mailiaadress ja hangib saadud aadressi kasutades rakenduse kasutajate andmebaasist kasutaja profiili ja rolli. Rakendusliides koostab rakenduse ligipääsuvõtme, mis sisaldab kasutaja infot ja edastab selle eesrakendusele. Eesrakendus avab vastavalt rollile kasutaja õiguste kohase lisafunktsionaalsuse. Protsessi kirjeldus on ära toodud joonisel (joonis 3).



Joonis 3. Google „web login“ teenuse kasutamine tunniplaani rakenduses

Eesrakenduse edasine suhtlemine rakendusliidesega sisaldab alati ka rakenduse ligipääsuvõtit, mille järgi rakendusliides kas sooritab ja vastab päringutele või keeldub, kui soovitakse teostada kasutaja rollile lubamatut. Kasutaja rollid jagasin neljaks ja neil on järgnevad õigused:

- Haldur – kõik õigused;
- Lektor – näeb kogu õppeinfot ja saab sisestada oma aine õppeinfot;
- Tudeng – näeb kogu õppeinfot;
- Külaline – näeb ainult tunniplaani.


Autenditud kasutajate õigused tuvastatakse järgmiselt:

- Haldur – saab rolli info kasutajate andmebaasist;
- Lektor – tuvastatakse emaili olemasolu lektorite andmebaasis;
- Tudeng – mailiaadress lõpeb stringiga „@tlu.ee“;
- Külaline – kõik teised kasutajad kes ei vasta eelnevatele tingimustele.

Kasutajate autentimiseks ja neile kasutajaõiguste jagamiseks Google autentimise kaudu lisasin andmebaasi kasutajate tabeli, milles hoitakse kasutajate ees- ja perenime, emaili ja kasutaja rolli.

Selleks, et eesrakenduse kaudu saaks kasutajate nimekirja hallata lisasin rakendusliidesesse neli ligipääsupunkti: kasutajate nimekirja küsimine; kasutaja lisamine; kasutaja muutmine; kasutaja kustutamine.

Kasutajate haldamiseks eesrakenduse kaudu arendasime kasutajate halduse Reacti komponendi koostöös Jaanus Saagoga. Tema abil sai halduse moodul eelneva arendusega sama disaini ja mobiilivaate. Mina arendasin suhtluse rakendusliidesega. Kasutajate haldusliides on ligipääsetav ainult halduri õigustes kasutajale. Kasutajate halduse on realiseeritud samal põhimõttel, mis kogu eelnev rakendus kõik toimib samal lehel. Sisenedes kasutajate haldusesse asendatakse Reacti tunniplaani komponent kasutajate komponendiga. Lehe loomisel kasutasime juba eelnevalt loodud Reacti elemente mida on kasutatud lektorite administreerimise puhul. Kasutajate haldusliidese kasutusloogika on sarnane tunniplaani halduril loengute haldamisel (joonis 4 ja 5).

TALLINNA ÜLIKOOL  
Haapsalu kolledž

/ SAHTEL / RIIUL

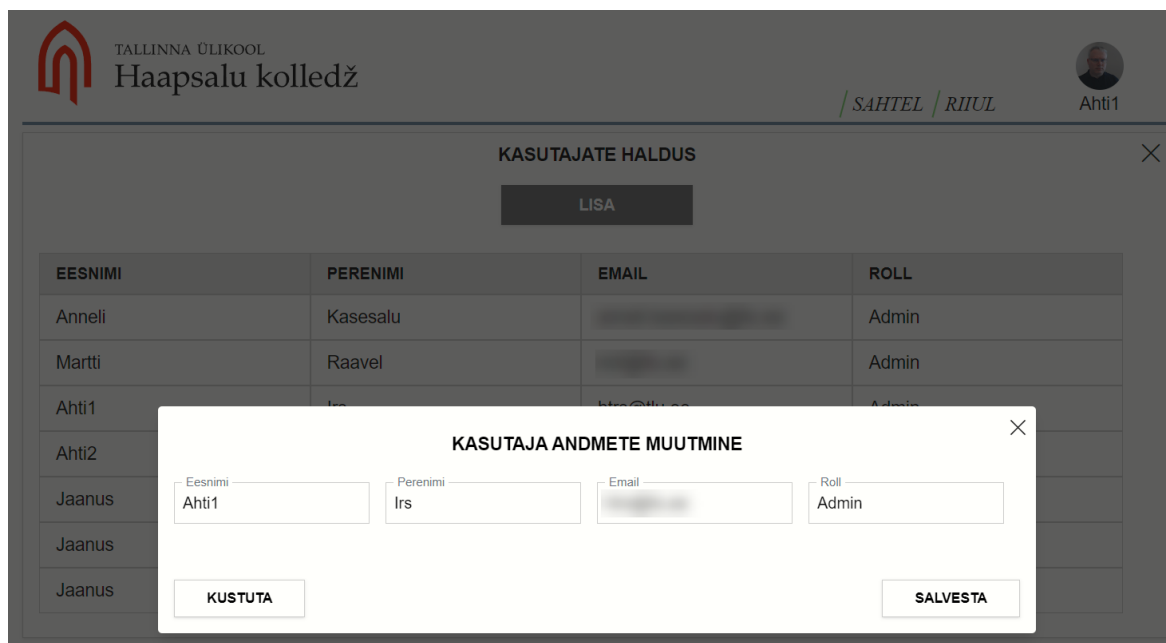
Ahti1

**KASUTAJATE HALDUS** X

LISA

EESNIMI	PERENIMI	EMAIL	ROLL
Anneli	Kasesalu		Admin
Martti	Raavel		Admin
Ahti1	Irs		Admin
Ahti2	Irs		Õppejõud
Jaanus	Saago		Admin
Jaanus	Saago		Õppejõud
Jaanus	Saago		Õpilane

Joonis 4. Kasutajate nimekiri



Joonis 5. Kasutaja muutmine ja kustutamine

Arenduse ülesanne, mille eesmärgiks oli filtrites valitud seisude meeldejätmine isegi ilma kasutajat autoriseerimata. Oleks mugav kui valik jääks meelde. Seda soovisid ka õppejõud, kuna põhiliselt vaatavad nad ainult enda loengute ajakava. Selle soovitud omaduse teostamine osutus veidi keerukamaks kui esmapilgul tundus.

Plaanisin kasutada lihtsalt filtri valiku salvestamist veebilehitseja lokaalsesse muutujasse ning uut sessiooni alustades kontrollida lokaalse muutuja olemasolu ja selle olemasolu puhul omistada see filtri algväärtuseks.

Eesrakenduse koodi ja Reactiga tutvudes selgus, et rakenduse filtrisüsteem on lahendatud nii, et põhilehele jõuab informatsioon filtrite seisust ühe objektina. Objektis on kirjeldatud kõikides filtrites seatud valikute väärtuste massiiv ilma kuvatavate nimedeta. Otsustasin soovitud võimaluse lahendada filtri komponenti arendades. Arendasin komponendile „SearchDropdown“ lisaomaduse „isRemembered“. Kui komponent kutsutakse välja nii, et on kaasa antud omadus „isRemembered“ väärtusega „true“ siis salvestatakse „selectedOption“ väärtus veebilehitseja lokaalsesse muutujasse. Muutuja nimeks pannakse komponendi väljakutsumisel komponendi nimi, mis on muutujas „name“.

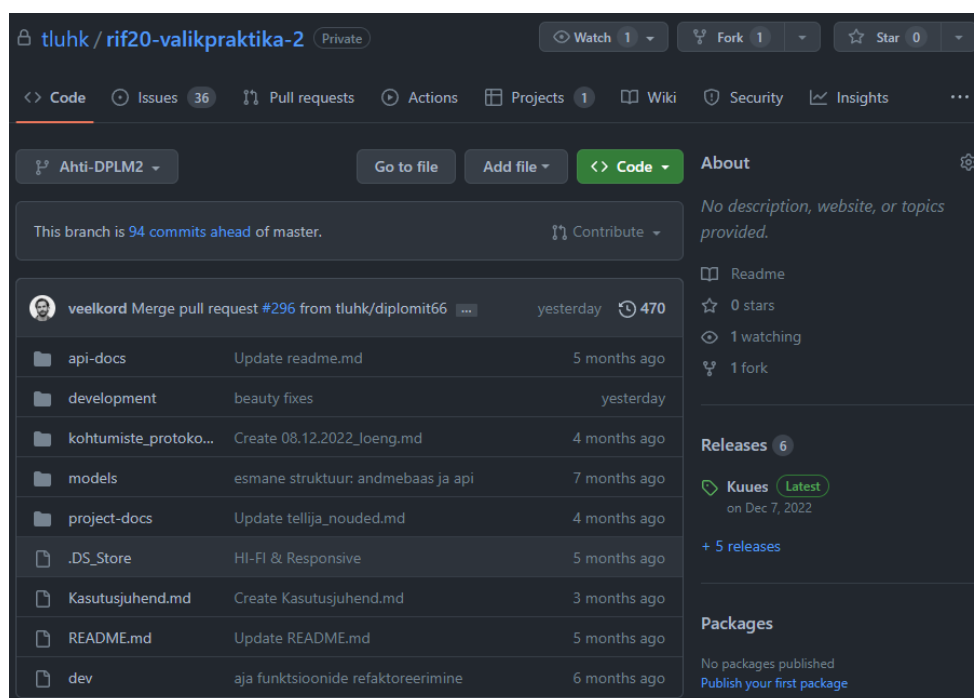
Kuna veebilehitseja lokaalsesse muutujasse saab panna ainult stringi siis salvestamisel pidin kasutama funktsiooni „JSON.stringify()“ ja filtri algseisu väljalugemisel „JSON.parse()“. Tekkis veel Reacti asünkroonsuse ja komponendi ülesehituse tõttu probleem, et kui filtri sisu muudeti, siis võeti vana seis kohe lokaalsest muutujast ja pandi tagasi filtrisse enne, kui uus seis jõudis lokaalsesse muutujasse. Selle probleemi lahendasin komponendi funktsioonide ülesehituse ja järjestuse muutmisega, muutes ajaliselt tegevuste järjekorda nii, et saavutasin soovitud tulemuse.

Viimase arendusülesande lahendamiseks valisin populaarse node npm paketi winston, mille integreerisin tagarakenduse koodi. Eelistuse winstoni paketi valikuks tingis selle paindlikkus logimise transporterite valikus. Võimalik on logisid suunata konsooli, faili, andmebaasi, http aadressile, logimisserveritele jne. Teine omadus, mis võimaldab logimist hästi seada, on Winstoni tasemete süsteem. Logimisteed on jaotatud seitsmesse erinevasse tasemesse. Moodulis logger.js on võimalik arendajal seada, mis taseme logid parasjagu kuhugi saadetakse. See võimaldab arenduse käigus seada taseme piir kõrgemale, mis laseb läbi kõik teated. Tootmises oleval koodil võib logimise tase olla madalamal, logitakse ainult olulised teated. (Better Stack, Inc, s.a.)

Tegevuslogi jäädvustamiseks lisasin andmebaasi logi nimelise tabeli. Mida ja kui palju sinna logida, tuleks järgnevate arenduste käigus otsustada vastavalt kasutusele ja vajadusele, sest andmebaasi kirjutamine on aeglane ja kõike logides kasvab andmebaas kiirelt väga suureks. Hetkel on rakendusliides seatud logima kõiki sissetulevaid päringuid konsooli ja salvestab need faili.

## **4.2. Arenduskeskkond**

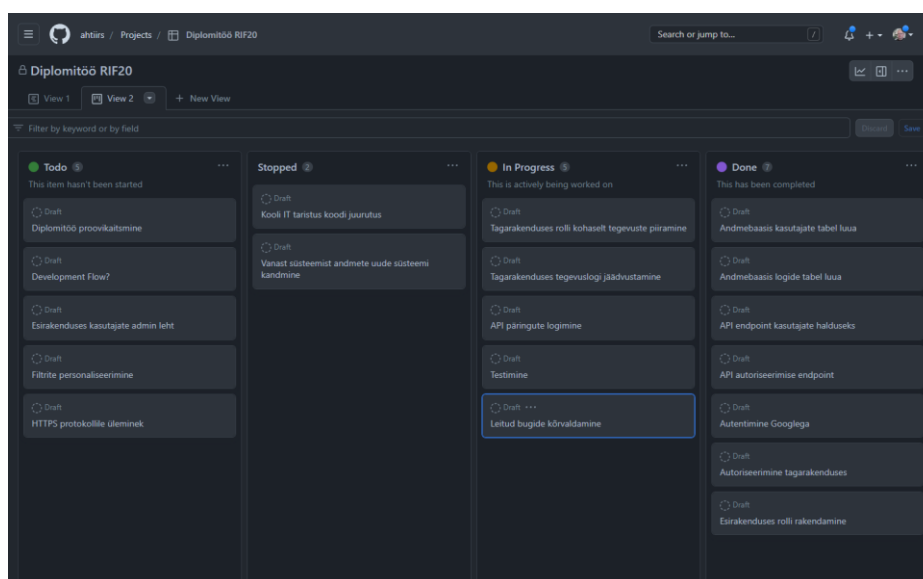
Koodi arenduses kasutasin veebipõhist versioonihaldus ja arenduskeskkonda Github'i, milles asus Haapsalu kolledži repositoorium <https://github.com/tluhk/HK-Tunniplaani> (joonis 6). Oma koodi arendasin eraldi harus nimega Ahti-DPLM2, mida aegajalt testimise eesmärgil liitsime Jaanus Saago haruga nimega diplomit66. Kuna kohati muutsime samu faile, siis failide automaatne ühendamine alati ei õnnestunud. Tuli teha koostööd ja failides tehtud muudatused üheskoos üle vaadata.



Joonis 6. Arendus ja versioonihalduskeskkond Github

### 4.3. Projekti haldus

Arenduses planeeritavaid, pooleli ja juba lõpetatud tegevuste haldamiseks kasutasin samuti Githubi keskkonnas pakutavat projekti juhtimise võimalust. Selle eeliseks on tihe seotus koodihaldusega, mis võimaldab väheste klikkidega teha igast ülesandest eraldi koodiharu või probleemi (*Issue*) (joonis 7). Lisaks projekti haldusele on see tehtud ka dokumenteerimise eesmärgil, et lihtsustada edaspidistel arendajatel töö jätkamist



Joonis 7. Arendus ja versioonihalduskeskkond Github

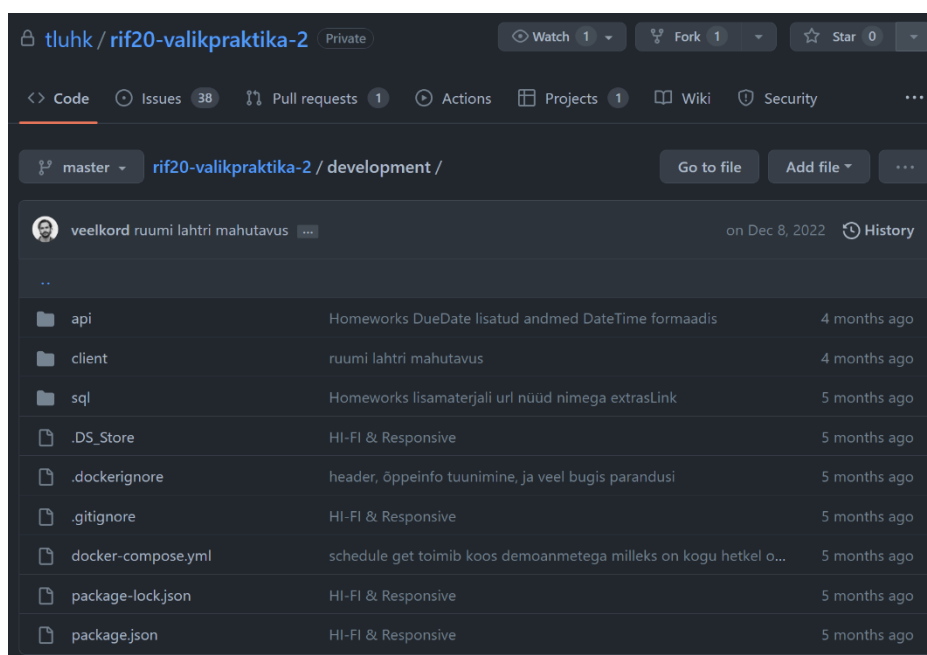
#### 4.4. Koodi struktuur

Kood asub Haapsalu kolledži repositooriumis <https://github.com/tluhk/HK-Tunniplaani> ja arendus käib alamkataloogis „development“, Teised kataloogid on arendusmeeskondade koosolekute protokollide, juhendmaterjalide ja projektidokumentatsioonide jaoks. Koodi jaguneb kolme alamkataloogi (joonis 8):

- kataloog „api“, sisaldab rakendusliidese lahendust;
- kataloog „client“, sisaldab rakenduse eesliidest;
- kataloog „sql“ sisaldab mysql andmebaasi struktuuri ja algandmeid.

Igas eelpool mainitud kataloogis asub fail Dockerfile, milles asuv skript käivitatakse Dockeri konteineri loomisel. Skript kirjeldab konteineri välist kataloogi, kus asub rakenduse kood ning haagib selle konteineri külge. Järgmisena kopeeritakse konteinerisse vajalike npm pakette kirjeldav faili package-lock.json ning käivitatakse vajalike npm pakettide installeerimine.





Joonis 8. Koodi struktuur versioonihalduskeskkonnas Github

Kirjeldatud kataloogidega samas asub „docker-compose.yml“ mis kirjeldab iga rakenduse komponendi käitamist Dockeris. Failis on kirjas iga komponendi Docker image nimi, konteineri nimi, sisemised ja välised IP pordid. Konteineritele määratud väliste IP portide sobivust juurutavasse keskkonda peaks kindlasti kontrollima ja vajadusel kasutatava võrguga sobivaks muutma.

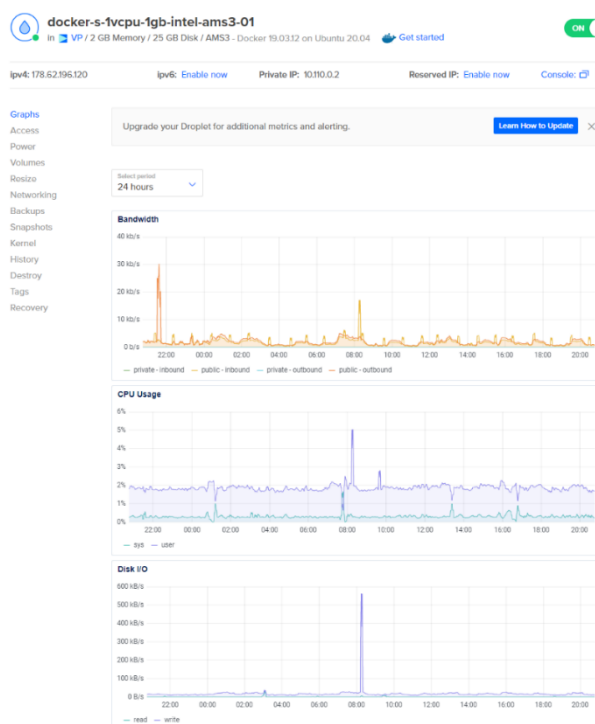
#### 4.5. Arenduse testkeskkond

Koodiarenduse käigus testide läbiviimiseks käivitasin koodi Digital Ocean keskkonnas renditud Linux masinal. Selleks, et oleks võimalik testida Google „Web login“ põhists autentimist, sidusin renditud virtuaalserveri avaliku IP aadressi ajutise nimega [tunniplaani.nutiasi.ee](http://tunniplaani.nutiasi.ee), millele pöördudes said kõik arendusse kaasatud isikud rakendust testida.

Testkeskkonna kasutamine andis lisaks koodi toimimise kontrollile ja kasutajate tagasisidele ka teadmise serveri ressursi vajaduste kohta. Praktikas sain toimima rakenduse, mille andmebaasis on 8 aasta jagu tunniplaani järgmiste parameetritega:

- operatsioonisüsteem Ubuntu 20.4;
- operatiivmälu 2GB;
- kõvaketta maht 25GB.

Rakenduse kasutamine selliste parameetritega serveril on mugav. Andmete kohalejõudmine ja filtreerimine toimib märgatava viivitusega. Serveri ressursside koormusgraafikud annavad kinnitust piisava ressursivaru olemasolus (joonis 9).



Joonis 9. Testserveri „Digital Ocean“ virtuaalserveri koormus

Kaasaja serverite võimsuste juures on kasutatav ressurss pigem tagasihoidlik. Sellest tulenevalt võiks valminud rakendust pidada vähenõudlikuks ning vähese ülalpidamiskuluga rakenduseks.

Poole arenduse pealt sain oma kasutusse kooli serveris asuva virtuaalserveri ja alates sellest hetkest toimus arendus ja testimine aadressil [tunniplaani.hk.tlu.ee](https://tunniplaani.hk.tlu.ee).

## 5. JUURUTAMINE

### 5.1. Järjepideva integratsiooni ja arenduse mudel (CI/CD)

CI/CD ehk pidev integratsioon ja pidev tarnimine või juurutamine on automatiseeritud protsessid, mis aitavad kiirendada tarkvaraarendust ja muuta see usaldusväärsemaks. CI/CD koosneb kahest komponendist (GitHub, Inc, s.a. b):

- Pidev integratsioon (*Continuous Integration*, CI): See protsess hõlmab koodimuudatuste automaatset ehitamist, testimist ja integreerimist ühisesse repositooriumisse;
- Pidev tarnimine või pidev juurutamine (*Continuous Delivery/Deployment*, CD): Pidev tarnimine viib koodimuudatused automaatselt tootmisvalmis keskkonda heakskiitmiseks, samas kui pidev juurutamine viib koodimuudatused otse klientideni.

CI/CD protsessi eelised on järgmised (GitHub, Inc, s.a. b):

- Arenduskiirus: Pidev tagasiside võimaldab arendajatel teha väiksemaid muudatusi sagedamini, mitte oodata ühte suurt väljalaset;
- Stabiilsus ja usaldusväärsus: Automaatne ja pidev testimine tagab, et koodibaasid püsivad stabiilsed ja on igal ajal valmis uueks tarneks;.
- Ärikasv: Kuna manuaalsed ülesanded on automatiseeritud, saavad organisatsioonid keskenduda innovatsioonile, kliendirahulolule ja tehnilise arendusele.

CI/CD protsessi kirjeldus (GitHub, Inc, s.a. b):

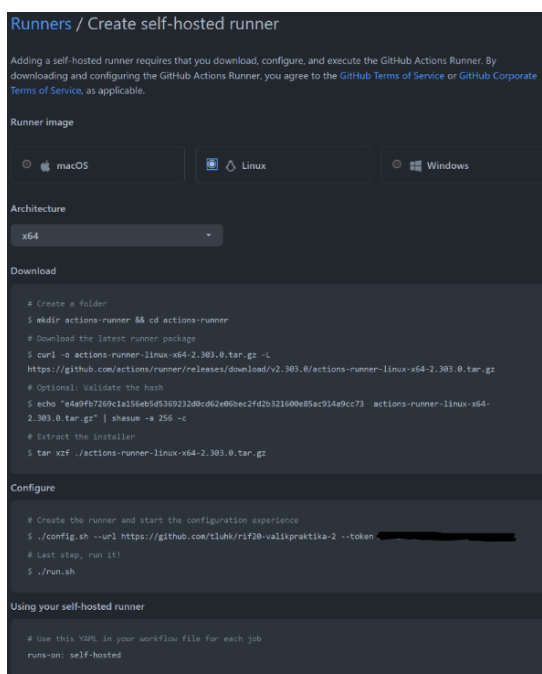
- Arendajad algatavad pull requestid, mis käivitavad esialgse koodi integreerimise ja üksustestid.
- Heakskiidetud commitid juurutatakse eelvaate keskkonda.
- GitHub Actions scriptid sooritavad vajaliku installatsiooni ja käivitavad testid.
- GitHub Apps teostab reaajas kontrolli pull requestide käigus.
- Heakskiidetud commitid ühendatakse põhiharuga täiendavate testide jaoks või juurutatakse kohe tootmisse.

Hea CI/CD töövoog automatiseerib koodi integreerimise, testimise ja juurutamise, et arendajatel oleks rohkem aega koodi jaoks. (samas)

## 5.2. Arenduskeskkonna integratsioon

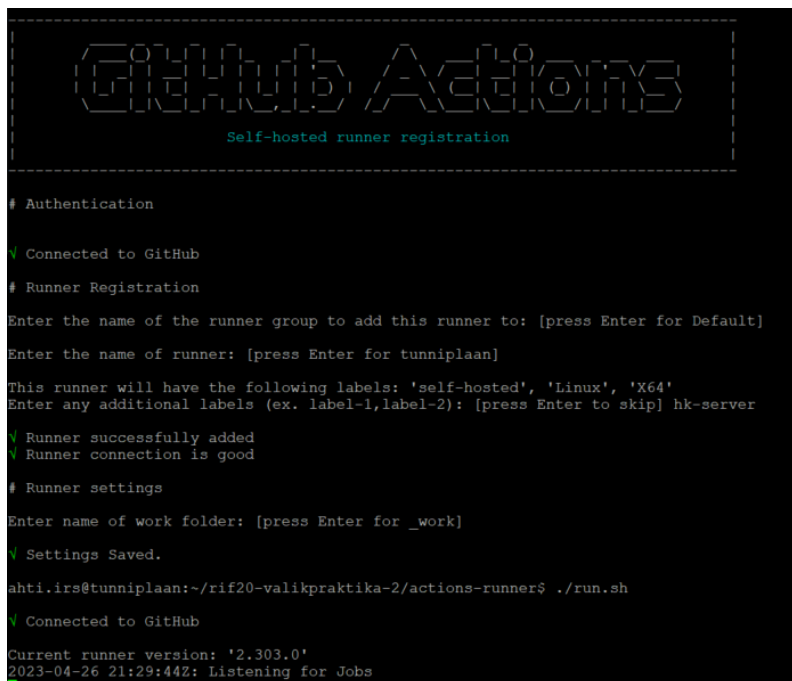
Käesoleva töö üks eesmärkidest on valminud rakendus juurutada kasutades järjepideva integratsiooni ja arenduse mudelit (CI/CD). Neid võimalusi on mitmeid, kuid valisin selleks Github keskkonna poolt pakutava Github Actions lahenduse. Sellise valiku peamiseks põhjuseks on rakendusinformaatika eriala kuraatorilt intervjuus saadud informatsioon, et kool pühendub lähitulevikus kõikides koodiarendusega seotud loengutes edaspidi versioonihaldustarkvara Github kasutamisele (Raavel, intervjuu 06.03.2023).

Järjepideva tarnimise koodihaldustarkvarast realiseerisin Github Actions vahenditega. Kuna loodav kood peab töötama kooli halduses oleval virtuaalsel serveril, siis pole võimalik kasutada Github Actions poolt pakutavaid juba ettevalmistatud töövooge. Sellise kasutusjuhtumi puhul tuleb defineerida koodihaldustarkvara keskkonnas oma valduses olev server (*self-hosted runner*) ja installeerida serveris teenused vastavalt etteantud juhendile (joonis 6).



Joonis 10. Enda halduses serveri sidumine teenusega Github Actions

Kui juhendis etteantud käsud on käivitatud enda hallatavas serveris, siis on server edukalt ühendatud kooli Github'i repositooriumiga ning Actions teenus aktiveeritud ja ootel (joonis 11).



```
-----
          GITHUB ACTIONS
          Self-hosted runner registration
-----

# Authentication

✓ Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]
Enter the name of runner: [press Enter for tunniplaan]

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip] hk-server

✓ Runner successfully added
✓ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]

✓ Settings Saved.

ahti.irs@tunniplaan:~/rif20-valikpraktika-2/actions-runner$ ./run.sh

✓ Connected to GitHub

Current runner version: '2.303.0'
2023-04-26 21:29:44Z: Listening for Jobs
```

Joonis 11. Serveri ekraanitõmmis õnnestunud seotud Githubi Actions teenusega

Enda serveris automaatse tegevusvoo realiseerimiseks tegin repositooriumisse faili „github/workflows/main.yml“ ja seal määrasin tegevused, mille puhul töövoog käivitub. Viimaseks tegevuseks panin enda virtuaalserveris asuva skripti nime, mis serveris käivitudes kloonib giti „production“ harust ja täidab rakenduse käivitamiseks vajalikud toimingud.

### 5.3. Testimine jätkuvas integratsioonis

Testimine on oluline osa CI/CD arendusprotsessist eesmärgiga tagada, et arendatav tarkvara käivitub edukalt tootmises. CI/CD protsessis on võimalik kasutada mitmeid erinevaid teste, näiteks koodi stiili test, Dockeri image eduka loomise test ning konteineri eduka käivitamise test jne. (Github, s.a. a)

CI/CD protsessis toimub testimine enamasti automaatselt, testid on integreeritud arendusprotsessi igasse etappi. Eesmärk on leida vigu ja defekte võimalikult varakult, et neid

saaks kiirelt parandada ning seeläbi tagada tarkvara kvaliteet. See vähendab vigade kuhjumist ning sellest põhjustatud tarkvara tõrkeid. Paljude väikeste vigadega tarkvara hilisem parandamine on oluliselt keerukam ja ajaliselt kulukam. Väikeste koodimuudatuste kiire integreerimine ja testimine, mis toob kohe esile ka võimaliku vea, on kokkuvõttes efektiivsem.

Tunniplaani rakenduse CI/CD protsessis vajab testimine kindlasti edasist arendust. Hetkel on loodud automaatne töövoog Github Actioni abil, mis viib arendaja poolt repositooriumi production harusse integreeritud koodi kooli serverisse. Server loob seal uued Dockeri konteinerid ja käivitab need.

#### **5.4. Algandmete sisestus**

Viisin läbi intervjuu kooli haldustöötajaga, kes on peamine tunniplaanisüsteemi haldur (Kasesalu, intervjuu 28.04.2023). Intervjuust selgus, et rakenduse mugavaks kasutuselevõtuks on vajalik, et seal oleks eelnevalt sisestatud õppejõud, loengud ja kursused. Vältimaks vanemate ja enam mittevajalike kirjade tekkimist, hindas ta piisavaks kui sisestan eelneva õppeaasta jooksul aset leidnud loengud ja neid lugenud õppejõudude andmed. Väljastasin vanast tunniplaanisüsteemist viimase õppeaasta kirjed. Sorteerisin, korrastasin ning sisestasin soovitud tingimustele vastavad kirjed andmebaasi algandmetesse.

#### **5.5. Rakenduse edasine arendus**

Tunniplaani rakendus on diplomitöö käigus arendatud tasemeni, mis võimaldab võtta selle igapäevasesse kasutusse, kuid see pole kindlasti lõplikult valmis. Rakendus on juurutatud meetodil, mis võimaldab sellele vähese vaevaga lisada funktsionaalsust või muuta disaini. Toon välja nimekirja funktsionaalsusest, mis võiksid olla järgmised arendusülesanded:

- koodi testide kirjutamine ja lisamine CI/CD töövoos protsessi nendega arvestamine (kui teste ei läbita, siis deployd ei toimu);
- autentimine Githubi kontoga;
- autentimine kasutajanime ja parooliga;
- eesrakenduse ja rakendusliidese omavahelise suhtluse viimine HTTPS protokollile;

- muudatuste teate saatmine automaatselt emailile;
- tunniplaani eksport exceli ja/või pdf faili;
- tunniplaani eksport Google kalendrisse;
- tume vaade.

## KOKKUVÕTE

Valikpraktika raames läbiviidud uuringutes selgus, et kasutatav tunniplaani rakendus ei vasta enam tudengite ega õppejõudude ootustele. Valikpraktikas sai alustatud uue tunniplaani rakenduse prototüübi loomist, kuid valminud funktsionaalsus ei võimaldanud seda veel kasutusele võtta.

Diplomitöö käigus uurisin välja rakenduse puuduva funktsionaalsuse, seadsin arenduse eesmärgid ning lõin lahenduse järjepidevaks integratsiooniks ja arenduseks.

Käesoleva diplomitööga sai edukalt lõpuni viidud rakenduse arendus tasemeni, mis võimaldab selle igapäevast kasutust. Lisaks arendusele sai loodud järjepideva arenduse ja integreerimise mudel, mida kasutades on lihtne rakendust väikeste sammudega edasi arendada ja seda kooli serveris automaatselt juurutada. Diplomitöö käigus sai ka loodud dokument, mis aitab järgmistel arendajatel hoida sama stiili.

Minule andis diplomitöö võimaluse tegeleda huvitava ja arendava projektiga. Õppisin palju uut kasutaja autentimise ning automaatse juurutamise kohta. Lisaks oli mul hea meel osaleda millegi nii praktilise asja loomisel. Sellest võiks olla kasu tulevaste aastate tudengitel nii kasutamise kui arendamise vaatenurgast.

Diplomitöö eesmärk sai edukalt täidetud. Valmis automaatselt juurutatav ja lihtsalt arendatav tunniplaani mis on valmis kasutuselevõtuks.



## SUMMARY

**Title in English:** Development and Implementation of a Timetable Application: A Case Study of Tallinn University's Haapsalu College.

In the studies conducted during the elective practice, it was revealed that the current timetable application no longer meets the expectations of students or lecturers. During the elective practice, the creation of a new timetable application prototype was initiated, but the completed functionality did not allow its implementation yet.

During the course of my diploma thesis, I investigated the missing functionality of the application, set the development goals, and created a solution for continuous integration and development.

With this diploma thesis, the application development was successfully completed to a level that allows its everyday use. In addition to the development, a model for continuous development and integration was created, making it easy to develop the application in small steps and automatically deploy it on the school server. During the course of the thesis, a document was also created to help future developers maintain the same style.

The diploma thesis gave me the opportunity to work on an interesting and developmental project. I learned a lot about user authentication and automatic deployment. Additionally, I was pleased to participate in creating something so practical. This could be of benefit to future students from both a usage and development perspective.

The goal of the diploma thesis was successfully fulfilled. An automatically deployable and easily developable timetable was completed, ready for implementation.

## KASUTATUD KIRJANDUS

- Amazon Web Services. (s.a.). What Is An API. [2023, veebruar 10].  
<https://aws.amazon.com/what-is/api/>
- Better Stack, Inc. (s.a.). A Complete Guide to Winston Logging in Node.js.  
[2023, märts 25]. <https://betterstack.com/community/guides/logging/how-to-install-setup-and-use-winston-and-morgan-to-log-node-js-applications/>
- Docker Inc. (s.a.). Use containers to Build, Share and Run your applications.  
[2023, aprill 22]. <https://www.docker.com/resources/what-container/>
- Enenche, U. L. (2019). The Importance and Usefulness of Timetable to Schools.  
[2023, veebruar 15]. [https://www.academia.edu/40472867/The\\_Importance\\_and\\_Usefulness\\_of\\_Timetable\\_to\\_Schools](https://www.academia.edu/40472867/The_Importance_and_Usefulness_of_Timetable_to_Schools)
- Github. (s.a. a). About continuous integration. [2023, mai 2].  
<https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration>
- GitHub, Inc. (s.a. b). CI/CD: The what, why, and how. [2023, märts 3].  
<https://resources.github.com/ci-cd/>
- GitHub, Inc. (s.a. c). GitHub Docs. [2023, veebruar 28].  
<https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories>
- Google. (s.a. a). Authorizing for Web. [2023, veebruar 7].  
<https://developers.google.com/identity/oauth2/web/guides/overview>
- Google. (s.a. b). Integrating Google Sign-In into your web app. [2023, veebruar 20].  
<https://developers.google.com/identity/sign-in/web/sign-in>
- Grider, S. (2022). Modern React with Redux. [2023, märts 10].  
<https://www.udemy.com/course/react-redux/>
- json.org. (s.a.). Introducing JSON. [2023, aprill 5].  
<https://www.json.org/json-en.html>
- Lemonaki, D. (18. märts 2022. a.). Frontend VS Backend – What's the Difference?  
[2023, mai 2]. <https://www.freecodecamp.org/news/frontend-vs-backend-whats-the-difference>

Martin, R. C. (2009). Clean Code - A Handbook of Agile Software Craftsmanship.  
Pearson Education, Inc.

Meta Open Source. (s.a.). React Project Details. [2023, aprill 15].

<https://opensource.fb.com/projects/react/>

Mozilla Corporation. (s.a.). HTML basics. [2023, mai 18].

[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)

Mozilla Corporation. (s.a.). What is CSS? [2023, mai 18].

[https://developer.mozilla.org/en-US/docs/Learn/CSS/First\\_steps/What\\_is\\_CSS](https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS)

OpenJS Foundation. (s.a.). About Node.js. [2023, märts 18].

<https://nodejs.org/en/about>

Oracle. (s.a.). What is MySQL? [2023, märts 25].

<https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>

Reflectoring.io. (2022). Node.js Logging with Winston. [2023, mai 10].

<https://reflectoring.io/node-logging-winston/>

Shetty, V. (2022). Enterprise OAuth 2.0 and OpenID Connect. [2023, märts 2].

<https://www.udemy.com/course/enterprise-oauth-for-developers/>

## **LISA 1. KÜSIMUSED INTERVJUEERITAVATELE**

### **Küsimused IT administraatorile:**

1. Kas IT-infrastruktuur toetab Dockeri konteinereid?
2. Milline on olemasolevate serverite ressursikasutus ja koormus?
3. Kas IT-infrastruktuuril on võimekus skaleeruda?
4. Millised on andmebaasi varundamise ja taastamise poliitikad?
5. Millised turvameetmed on IT-infrastruktuuris kasutusel?
6. Millist tuge pakub IT-infrastruktuuri meeskond? Küsi, millist tuge ja abi pakuvad IT-spetsialistid teie arenduse juurutamise ja haldamise osas.
7. Millised on andmekeskuste asukohad ja kuidas see mõjutab teie rakenduse jõudlust?

### **Küsimused Rakendusinformaatika eriala kuraatorile:**

8. Millise arenduskeele õpetamisele keskendub kool suurimas matus järgmistel aastatel?
9. Milliseid arenduskeskkondi ja vahendeid õpetamisel plaanite kasutada?
10. Milliseid raamistikke kool soovib süvenenult edaspidi õpetada?
11. Mida peab tunniplaani süsteem minimaalselt võimaldama õppejõu vaates, et see oleks igapäevaselt kasutusse võetav ja siis juba kasutuse käigus edasi arendatav?

### **Küsimused tunniplaanisüsteemi administraatorile:**

12. Mida peab tunniplaani süsteem minimaalselt võimaldama administraatori seisukohalt, et see oleks igapäevaselt kasutusse võetav ja siis juba kasutuse käigus edasi arendatav?