Tartu Ülikool
Arvutiteaduse instituut
Informaatika õppekava

**Mihkel Puusepp**

# Tarkvaralabor OÜle Korto kasutajaliidese loomine

**Bakalaurusetöö(6 EAP)**

Juhendaja: Vambola Leping

TARTU 2018

## Tarkvaralabor OÜle Korto kasutajaliidese loomine

**Lühikokkuvõte:**
Käesoleva bakalaurusetöö eesmärgiks on luua Tarkvaralabor OÜ Kortole kaasaegse väljanägemisega kasutajaliides.

Kasutajaliides annab kasutajale võimaluse:

1. Vaadata ühistuga seotud sõnumeid;
2. Sisestada kinnisvara elektri- ja veenäite;
3. Vaadata kinnisvaraga seotud arveid;
4. Vaadata ühistuga seotud andmeid;
5. Vaadata ja muuta kinnisvara, kinnisvaraga seotud isikute ja kasutaja andmeid.

**Võtmesõnad:**
Veebirakendus, Kasutajaliides, TypeScript, ReactJs, GraphQl

**CERCS:**
P175 Informaatika

## Creating Kortos user interface for Tarkvaralabor OÜ

**Abstract:**
The purpose of this Bachelor Thesis is to create a modern looking user interface for Tarkvaralabor OÜ Korto app.

The user interface provides the possibility to:

1. Check messages related to union;
2. Enter electricity and water usage readings;
3. Check bills related to property;
4. Check information related to union;
5. Look up and change data related to property and residents.

**Keywords:**
Web application, User interface, TypeScript, ReactJs, GraphQl

**CERCS:**
P175 Informatics

# Sisukord

## Sissejuhatus

OÜ Tarkvaralabori loodud Korteriyhistu pakub korteriühistutele tegevuste haldamiseks iseteeninduskeskonda. Teenus on kasutusel ligi 3000s majas üle eesti. Hetkel kasutusel olev Korteriyhistu veebileht hakkab ajale jalgu jääma ja vajab uut kasutajaliidest koos kaasaegse disainiga. Korteriyhistu kasutajaliidese suurimaks probleemiks on ebamugav kasutajakogemus nutitelefonidel.

Käesoleva bakalaurusetöö eesmärk on luua Tarkvaralabor OÜle kaasaegse ja uue välimusega Korto veebirakenduse eessüsteem, mille kaudu on korteriomanikel võimalus jälgida korteriga seotud andmeid ning sisestada näitusid. Lisaks aitab rakendus korteriühistutel hallata ühistuga seotud andmeid. Veebirakendus annab korteriühistule võimaluse:

1. Vaadata ühistuga seotud sõnumeid;
2. Sisestada kinnisvara elektri- ja veenäite;
3. Vaadata kinnisvaraga seotud arveid;
4. Vaadata ühistuga seotud andmeid;
5. Vaadata ja muuta kinnisvara, kinnisvaraga seotud isikute ja kasutaja andmeid.

Töö esimeses peatükis antakse ülevaade loodavale Korto veebirakendusele. Järgnevalt kirjutatakse rakenduse loomiseks kasutatud tehnoloogiatest. Töö viimases peatükis kirjeldatakse rakenduse süsteemi arhitektuurilist poolt ja kuidas on kasutatud eelnevalt kirjeldatud tehnoloogiaid.

# 1. Veebirakenduse kirjeldus

Käesolev veebirakendus on jagatud kaheks osaks: avalik ja privaatne. Avalikule rakendusele saavad ligi kõik veebrirakenduse kasutajad. Privaatse veebirakenduse ligipääsuks peab kasutaja end autentima. Esialgsel sisenemisel rakendusse kontrollitakse, kas kasutaja on ennast eelnevalt hetkel kasutatavas veebilehitsejas autentinud. Juhul kui kasutaja on ennast eelnevalt autentinud, suunatakse ta privaatsele rakendusele. Vastasel juhul suunatakse kasutaja avaliku osa esilehele.

## 1.1 Avalik veebirakendus

Avalik rakendus koosneb menüüst, vaadetest ja jalusest. Veebirakenduse avaliku osa põhivaateks on esileht. Kasutaja suunatakse esilehele veebirakenduse esialgsel avamisel või väljalogimisel.

### 1.1.1 Vaated

Veebirakenduse avaliku osa vaadete eesmärk on kasutajale tutvustada pakutavat teenust ja Tarkvaralabor OÜ-d. Avaliku rakenduse alla kuuluvad järgnevad vaated:

1. Esileht;
2. Tutvustus;
3. Hinnad;
4. Sisselogimine;
5. Meist/Kontakt;
6. Uudised;
7. Abi.

Kõik ülal välja toodud vaated on staatilised vaated.

#### Esileht

Esilehe eesmärk on anda kasutajale kiire ülevaade privaatse osa funktsioonidest. Uuel kasutajal on võimalus proovida tasuta privaatset rakendust või tutvuda näidisühistuga (Joonis 1).


Joonis 1. Viide näidisühistule

Näidisühistu annab võimaluse tutvuda privaatse rakenduse funktsioonidega kasutajat loomata.

### Tutvustus

Vaade "Tutvustus" selgitab kasutajale lähemalt, milliseid funktsioone sisaldab privaatne osa. Detailsemalt on kirjeldatud korteriomanikule, ühistu juhile, raamatupidajale ja haldusfirmale suunatud võimalused (vt. Lisa 1 Joonis 2).

### Hinnad

Vaade "Hinnad" kuvab tabelina, millised erinevad kasutajatüübid pakuvad erinevaid lisafunktsioone ja nendele vastavad hinnad (vt. Lisa 1 Joonis 3).

### Sisselogimine

Sisselogimise vaates saab kasutaja ennast autentida, et pääseda ligi privaatsele rakendusele (vt. Lisa 1 Joonis 4). Sisselogimiseks pakutakse kasutajale viit erinevat võimalust:

1. Parooliga;
2. Mobiili-ID;
3. ID-kaart;
4. Pangakonto;
5. Smart-ID.

Väiksema ekraanisuuruse puhul peidetakse menüüst ära valikud ID-kaart ja Pangakonto (vt. Lisa 1 Joonis 5). Põhjuseks on mitte näidata nutitelefoni kasutajatele ebapraktilisi sisenemise viise. Pangakonto kaudu sisse logimine on lisatud valikusse vajadusel, kui inimesed kasutavad koodikaardiga sisse logimist.

### Meist/Kontakt

Vaade "Meist/Kontakt" tutvustab kasutajale Tarkvaralabor OÜ meeskonda ja annab ülevaate ettevõtte ajaloost. Vaates on ära kirjeldatud, kuidas saab ettevõtte või erinevate liikmetega kontakteeruda.

### Uudised

Vaates “Uudised” on välja toodud Kortoga seotud sündmused. Kasutajale kuvatakse viis uudise lühivaadet korraga. Uudise lühivaatele vajutades avaneb täismahus uudis uuel vaatel. Soovi korral saab kasutaja vaadata ka vanemaid uudiseid, liikudes lehekülje valimise komponendil edasi.

### Abi

Vaates “Abi” on välja toodud rakenduse kohta korduma kippuvad küsimused ja vastused. Kasutaja saab küsimusi sorteerida erinevate gruppide kaupa. Lisaks on võimalus otsida küsimusi märksõnade abil.

## 1.1.2 Menüü

Menüü juhatab kasutaja teenusega seotud vaadetele.


Joonis 6. Avaliku osa menüü

Iga valik menüüs suunab valikule vastavale vaatele. Korto ikoonile vajutades suunatakse kasutaja esilehele.

## 1.1.3 Jalus

Jaluses on välja toodud kõik avaliku osa vaated, mis on grupeeritud vastavalt vaadete sisule.

| Teenusest | Meist | Lisaks |
|---|---|---|
| Esileht | Meist/Kontakt | Abi/KKK |
| Tutvustus | Uudised | Hanked |
| Hinnad | Korto Facebookis | Keel/язык |
| | | EST/RUS/ENG/LAT |

Joonis 7. Avaliku osa jalus

Jaluses on võimalik muuta rakenduse keelt sobivale valikule vajutades.

## 1.2 Privaatne veebirakendus

Privaatse rakenduse eesmärk on pakkuda kasutajale võimalust sisestada oma korteri kohta andmeid, näha ühistu andmeid, hallata korteri elanike ja näha ühistu ning korteriga seonduvad teateid. Privaatses rakenduses on järgnevad vaated:

1. Teated;
2. Näidud;
3. Arved;
4. Ühistu;
5. Seaded.

Privaatne osa koosneb navigeerimismenüüst ja vaadetest. Privaatse osa avamisel avaneb esimese vaatena teated.

### 1.2.1 Navigeerimismenüü

Navigeerimismenüü pakub kasutajale viisi liikuda erinevate vaadete ning nähtaval olevate kinnisvaraobjektide vahel (vt. Lisa 1 Joonis 8). Lisaks privaatse rakenduse vaadetele on lisatud link avaliku rakenduse osa vaatele "Abi". Avades avaliku vaate "Abi", avaneb valitud vaade veebilehitseja uues kaardis. Menüüsse on ka lisatud välja logimise nupp, millele vajutades logitakse veebilehitsejas kasutaja välja. Kasutajale nähtavad kinnisvara objektid on välja toodud rippmenüüs. Väiksel ekraanil asub menüü paremas servas. Menüüd saab vajadusel sulgeda, et kasutada maksimaalselt väikeste ekraanide laiust (vt. Lisa 1 Joonis 9). Muude ekraani suuruste korral asub menüü vasakul ääres ja on alati kasutajale nähtav.

### 1.2.2 Teated

"Teated" vaate eesmärk on kuvada kasutajale ühistuga seotud teateid ning arutelu (vt. Lisa 1 Joonis 10). "Teated" vaade koosneb kahest erinevast komponendist: nimekiri ja sisu. Vaade kuvab kasutajale korraga nii nimekirja kui ka teate sisu. Erandiks on väiksemad ekraanid, kus algselt kuvatakse kasutajale nimekiri teadetest ning teate valikul avaneb kasutajale teate sisu.

#### Teadete nimekiri

"Teadete nimekiri" kuvab kasutajale saabunud teateid. Teadete nimekirjas saab kuvada kolme erineva olekuga sõnumeid: märgistatud, lugemata ja loetud sõnum. Märgistatud teated on välja toodud nööpnõela ikooniga ja asuvad nimekirjas üleval. Lugemata teadete pealkiri on eristatud paksu kirjatüübiga. Lugemata teatele peale klikkides muutub teate olek loetuks. Loetud teate korral on pealkiri ilma eristava kirjatüübita.

### Teadete sisu

Teadete sisu komponent kuvab valitud teate sisu. Kasutajal on soovi korral võimalik jätta teatele kommentaar, mid on nähtav teistele ühistu elanikele. Teatele jäetud komentaarid on järjestatud kronoloogilises järjekorras vanemast uueni.

### 1.2.3 Näidud

Vaade "Näidud" pakub võimalust sisestada käesoleva kuu elektri- ja veenäitude väärtust. Sisestades vastava näidu, kuvatakse kasutajale kuu jooksul tarbitud kogus. Salvestamisel muutub näidu sisestusväli mitteaktiivseks. Kasutaja poolt salvestatud näitu on võimalik muuta vajutades nupule "Muuda". Sellisel juhul muutub näidu sisestamise element tagasi muudetavaks. Juhul kui kasutaja näidud on automaatloetud, kuvab kasutajaliides käesoleva kuu näidu ja tarbitud koguse (vt. Lisa 1 Joonis 11). Automaatlugemise korral ei lubata kasutajal näitu muuta.

### 1.2.4 Arved

Vaade "Arved" kuvab kasutajale valitud kinnisvaraga seotud arved valitud perioodil (vt. Lisa 1 Joonis 12). Kasutaja saab valida arvete perioodi vajutades aastanumbri kõrval asuvatele parem ja vasak nooltele. Kui kasutaja on jõudnud käesoleva aastani, siis muutub nool edasi mitteaktiivseks. Samamoodi muudetakse nool tagasi mitteaktiivseks, kui kasutaja on jõudnud kõige esimese aastani, kus on esinenud arveid. Valides aasta, kus ei ole esinenud arveid, kuvatakse kasutajale vastav teade arvete puudumisest. Vajutades "Laadi alla" nupule, laetakse alla vastava kuu arve faili. Tulpdiagramm näitab kasutajaliideses arve erinevate osade ja täisarve summa võrdlust. Liikudes hiirega tulba peale avaneb hüpikvihje, kus on kirjeldatud värvuste summa ja nende tähendus.

### 1.2.5 Ühistu

Vaate "Ühistu" eesmärk on edastada kasutajale ühistuga seotud informatsiooni. Vaate ülemises osas on välja toodud ühistuga seotud isikud koos kontaktandmetega (Joonis 13).


Joonis 13. Ühistu kontaktandmed

Vaade sisaldab kolme alamvaadet. Alamvaate raha kuvab kasutajale ühistu vabade vahendite, hoiuste ja võlgnevuse summat (Joonis 14).


Joonis 14. Ühistu finants andmed

Visuaalselt kuvatakse võrdlust viimase sissekandega. Väärtuse tõus kuvatakse rohelise ülesnoolega. Languse korral kuvatakse punast allanoolt.
Alamvaade "Päevik" kuvab kasutajale valitud kuu sündmusi kalendrivaates (vt. Lisa 1 Joonis 15). Väiksema ekraaniga seadmetel kuvatakse päevik loeteluna (vt. Lisa 1 Joonis 16). Kasutaja saab muuta kuud paremale või vasakule suunatud noolega. Sündmust sisaldav päev sisaldab sündmuse nime ja on eristatav helesinise tasuta järgi. Hetke kuupäeval on päevikus lisatud kuupäeva numbri ümber oranž märguande märgistus. Alamvaade "Failid" kuvab kasutajale kõik ühistuga seotud dokumendid. Dokumendid on grupeeritud järgnevalt:

1. Põhikiri ja kodukord;
2. Üldkoosolekud;
3. Juhatus;
4. Aastaaruanded;
5. Majanduskavad;
6. Blanketid;
7. Muud dokumendid.

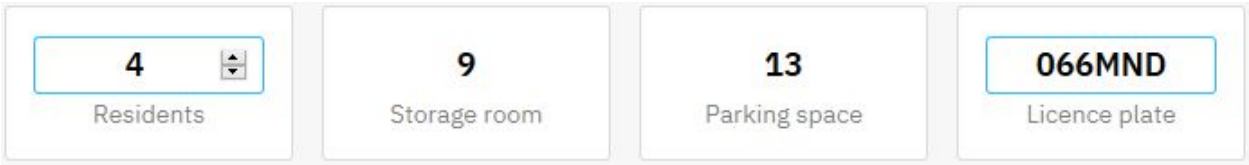
Dokumendid on järjestatud kronoloogilises järjestuses, uuemast vanemaks. Vajutades "Laadi alla" nupule, laeb kasutaja alla valitud dokumendi. Juhul kui grupis on rohkem kui kümme dokumenti, kuvatakse kasutajale grupis faile kümnekaupa (vt. Lisa 1 Joonis 17).

### 1.2.6 Seaded

Vaade "Seaded" annab kasutajale ülevaate valitud kinnisvara ja kasutaja andmetest. Vaade on jaotatud kaheks alamvaateks.

### Korter

Alamvaates "Korter" kuvatakse kasutajale andmeid korteri kohta. Esimeses infoblokis kuvatakse elanike arvu, hoiuruumi suurust, parkimiskoha numbrit ja korteri parkimiskohal parkiva auto numbrimärki (Joonis 18).




Joonis 18. Korteri andmed

Kasutajal on võimalik muuta korteri elanike arvu ning auto numbrimärki. Vaate teine infoblokk kuvab korteriga seotud isikuid. Korteriga seotud isikute andmed kuvatakse isikukaartidena(vt. Lisa 1 Joonis 19). Isikukaart sisaldab järgnevaid väljasid:

1. Nimi;
2. Elaniku tüüp;
3. Isikukood;
4. Telefoninumber;
5. E-posti aadress;
6. Näidu sisestamise meeldetuletus;
7. Korteriühistu teateid e-postile;
8. Arve keel;
9. Arve paberil;
10. Arve e-postiga;
11. E-arve kanal;
12. Pangakonto number.

Väljad “Näidu sisestamise meeldetuletus”, “Korteriühistu teated e-postile”, “Arve paberil” ja “Arve e-postiga” põhinevad kahendmuutuja loogikal. Vastavad väärtused saavad olla kas tõene või väär. Väljad “Arve keel” ja “E-arve kanal” on mitme valikuga väärtused, mis kuvatakse kasutajale rippmenüüna. Väli “Pangakonto number” kuvatakse kasutajale vaid juhul, kui on valitud e-arve kanal. Vaikeväärtuseks on rippmenüüs “Valik puudub”.

### Isiku muutmine

Isikukaardil sisendvälja väärtust muutes avaneb tööriistariba, mis annab kasutajale teada võimalusest salvestada muudetud andmed või katkestada muudatuste tegemine isikute andmetele. Vajutades nupule “Katkesta” tühistatakse isikute andmetele tehtud muudatused. Kahendmuutuja loogikal põhinevate väljade muutmisel sooritatakse muutus ilma kasutaja kinnitust küsimata. Juhul kui kasutaja on aktiveerinud tööriistariba, salvestatakse kahendmuutuja loogikal põhinevad väljad kasutaja kinnitusel. Mittekorrektse andmete salvestamise korral kuvatakse kasutajale vastava välja juures veateade (Joonis 20).


Joonis 20. Veateatega väli

Vajutades nupule “Eemalda kasutaja”, avaneb kinnitusaken, kus küsitakse kasutajalt kinnitust kasutaja eemaldamiseks. Tegevuse kinnitamisel eemaldatakse valitud isik kinnisvaraga seotud isikute hulgast.

### Isiku lisamine

Kasutaja saab lisada valitud kinnisvara elanike hulka uue isiku vajutades nupule “Lisa isik”. Nupule vajutades avaneb kasutajale tühi blankett isiku lisamiseks (vt. Lisa 1 Joonis 21). Nõutud väljadele on lisatud juurde visuaalne tähistus (Joonis 22).


Joonis 22. Nõutud välja tähistus

Juhul kui kasutaja teeb valiku, mille tagajärjel muutub blanketis väli nõutuks, lisatakse vastavale väljale visuaalne tähistus. Vajutades nupule “Salvesta” lisatakse isik vastava kinnisvaraga seotud isikute hulka. Mittekorrektsete andmete sisestamise korral kuvatakse kasutajale vastava välja juures veateade.

### Kasutaja

Alamvaates “Kasutaja” saab rakenduse kasutaja muuta parooli, kustutada kasutaja rakendusest ning muuta kasutaja rakenduse keelt (vt. Lisa 1 Joonis 23). Parooli muutmiseks peab kasutaja sisestama hetkel kasutusel oleva parooli, ning uus parool peab olema vähemalt 6 tähemärki pikk. Püsiparooli puudumisel kuvatakse kasutajale ainult välja uus salasõna. Kasutaja kustutamisel deaktiveeritakse kasutaja konto. Uue kasutaja loomisel sama isikukoodiga taastatakse kasutaja konto. Keele valimiseks kuvatakse kasutajale valikus olevad keeled. Valides uue keele salvestatakse kasutaja valitud keel ning uuendatakse rakenduse tõlked valitud keele tõlgetega.

## 2. Tehnoloogiad

Käesolev peatükk tutvustab lugejale veebirakenduse loomisel kasutatud tehnoloogiaid. Eesmärk on kirjeldada, mis on TypeScript ja kuidas see erineb tavalisest JavaScriptist. Peatükis seletatakse kasutajale millised on React teegi omadused ja töövõtted. Lisaks kirjeldatakse ära kuidas loodud veebirakendus suhtleb põhiprogrammiga ja millisel viisil lahendati rakenduses mitmekeelsus.

### 2.1 JavaScript

JavaScript on objektorjenteeritud programeerimis keel, mis kasutab interpretaatorit kui ka *first-class* funktsioone[1].

Programmeerimiskeeled, mis kasutavad interpretaatorit tõlgivad oma käsud programmi jooksutamise hetkel iga rea kaupa masinkeelde. Keeled, mis kasutavad interpretaatori asemel kompilaatorit tõlgivad oma koodi seevastu otse masinkeelde[2].
*First-class* funktsioone kasutavaks keeleks nimetatakse keelt, kus saab funktsioone anda edasi teistele funktsioonidele parameetritena, funktsioonid saavad tagastada teisi funktsioone ja igat funktsiooni saab salvestada muutujana[3].

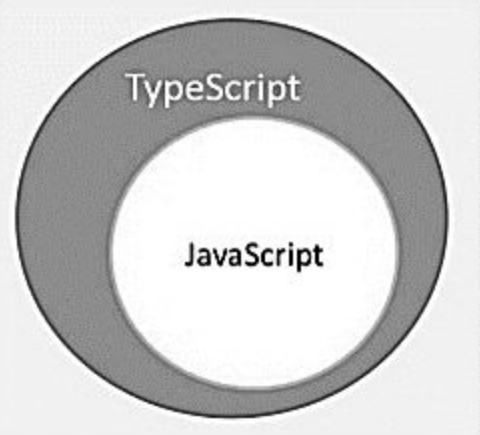
Javascript tuli esmakordselt välja aastal 1995 Netscape Navigator veebilehitsejas, ning pakkus võimalust lisada väikseid programme veebilehtedele. Nüüdseks toetavad kõik suuremad veebilehitsejate arendajad JavaScripti kasutamist oma veebilehitsejates[4].

## 2.2 ECMAScript

Ecma International on kirja pannud skriptimiskeelte standardi ECMA-262, mida kutsutakse ECMAScriptiks. ECMA-262 võib vaadelda kui ECMAScripti viite numbrit. ECMAScript määrab ära reeglid, töövõtted ja head tavad skriptimiskeeles, et vaadeldav keel oleks vastav ECMAScripti nõuetele. ECMAScripti standardit uuendatakse pidevalt ja kirjutamise hetkel viimane versioon on ECMAScript2017, lühendatult EC8. Lühidalt kokkuvõttes ECMAScript on standard ja JavaScript on skriptimiskeel mis lähtub sellest standardist[5].

## 2.3 TypeScript

TypeScript on *superset* JavaScriptile(Joonis 24), mis kompileerub JavaScriptiks[6].




Joonis 24. Illustratsioon Javascriptist ja Typescriptist

Javascriptiks kompileeritud kood vastab ECMAScript 3 või uuema standarditele. Lisaks toetab TypeScript ka ECMAScript 6 omadusi. TypeScript on keelelt sarnane JavaScriptile, kasutades sama süntaksi ja võtmesõnu, mis vastavad ECMAScript 2015 standarditele ja annab programmeerijale valiku määrata muutujatele tüüpi. TypeScriptist JavaScriptiks kompileerimise eest kannab hoolt

TypeScript kompilaator, lühidalt TSC[6]. Muuhulgas on TypeScriptil *Language service* API, mis pakub programmeerijale koodiviimistlusi, veakontrolli ja kiirparandusi. *Language service* on võimalik kasutada igas integreeritud programmeerimiskeskkonnas, kuna töötab TSC tasemel. Loojate eesmärgiks oli luua käesolev süsteem võimalikult kiirena, olenemata, kui suureks on käesolev koodibaas kasvanud. Lahenduses kasutati *on demand processing* põhimõtet, kus vaadeltakse ainult hetkel käsilolevat faili ja sõelutakse ühe veatüübi vastu korraga. Peale selle jaotatakse kompilaatori faasid erinevateks tsükliteks, mille kaudu saab jooksutada koodikontrolle kindlate failide peale[7].

### 2.3.1 Tüübi definitsioonid

TypeScripti tüübi defineerimise süsteem pakub kasutajale valikulist võimalust määrata muutujale tüüpe (Joonis 25). Tuleb meeles pidada, et see ei ole Typescripti kirjutamisel kohustuslik. Samas tuleb ka rõhutada, et TypeScript määrab muutujatele ise tüüpe kui kasutaja pole ise tüüpi määranud (Joonis 26). Süsteem aitab kiiremini tuvastada koodivigu ja anda kasutajale soovitusi koodi kirjutamisel.

TypeScript toetab järgnevaid sisseehitatud tüüpe:

1. Number, võtmesõnaks on *number*. TypeScript ja JavaScript ei arvesta eraldi täisarve ja ujuvkoma arve, seetõttu võib käsitleda mõlemat number tüübina;
2. Sõne, võtmesõnaks on *string*;
3. Kahendmuutuja, märksõnaks on *boolean*;
4. Void, märksõnaks *void*. Kasutatakse funktsioonide tagastustüübina, kui funktsioon ei tagasta väärtust;
5. Null, märksõnaks *null*;
6. Undefined, märksõnaks *undefined*.

```
const numberTüüpiMuutuja: number = 1
```

Joonis 25. Näide muutuja tüübi määramisest

```
let ilmaTähistuseta = 1
[ts] Type 'string' is not assignable to type 'number'.
let ilmaTähistuseta: number
ilmaTähistuseta += "1"
```

Joonis 26. Näide, kuidas Typescript määrab tüüpi

Lisaks saab kasutaja kirjeldada enda määratletud andmetüüpe nt. Masiive, liideseid enume jne (Joonis 27).

```
interface isik {
  eesNimi: string
  isikukood: number;
  keskmineNimi?: string
  pereNimi: string
  lapsed: isik[]
}
```

Joonis 27. Näide liidesest

Joonisel 27 on loodud liides nimega “isik”, kus on ära kirjeldatud muutujad ja nende tüübid, mis peavad “isik” tüüpi objektil olema. Väli “keskmineNimi” on tähistatud märgiga “?”, mis tähendab, et see ei ole kohustuslik väli “isik” tüüpi muutujal. Seetõttu on sellel väljal kaks võimaliku tüüpi, kas sõne või *undefined*.

Tüübiks *any* nimetatakse kõikide eelnevalt väljatoodud tüüpide ülemtüübiks (Joonis 28).

```
let ilmaTähistuseta : any = 1
ilmaTähistuseta += "1"
console.log(ilmaTähistuseta) // Oodatav vastus on "11"
```

Joonis 28. Näide *any* tüüpi tähistusest

Kuna muutuja “ilmaTähistuseta” tüüp on määratud *any*, siis ei anna TypeScript teada, et esineb koodiviga. Lisaks on JavaScriptis numbri ja sõne liitmine lubatud käsk, mille tõttu on see kood korrektne[8].

## 2.4 ReactJs

ReactJs (edaspidi React) on avatud lähtekoodiga Javascripti teek kirjutatud Jordan Walke poolt, kes on tarkvara Facebookis. React on mõeldud kasutajaliideste loomiseks, peamiselt üheleherakenduste loomiseks. Teek lubab luua veebirakendusi, kus saab kuvada erinevaid vaateid ja muuta vaadeldavaid andmeid ilma veebilehte uuesti laadimata. Eelnevalt mainitu saavutamiseks kasutab React komponendi põhist arendusviisi ning virtuaalset dokumendiobjektide mudelit[9].

### 2.4.1 JavaScript XML

Javascript XML (edaspidi JSX) on JavaScripti süntaksi laiendus, mille abil saab kirjutada hüpertekst-märgistuskeele (edaspidi HTML) märgendeid koos JavaScripti avaldistegab (Joonis 28)[10].

```
<div>{1+2+3+4+5}</div>
```

Joonis 28. Näide JavaScripti avaldisest koos HTML tähisega

Joonisel 28 välja toodud näitel kuvatakse HTML div märgise sees avaldise “1+2+3+4+5” väärtust. Kõik JavaScripti kood, mida JSX märgendite sees kasutatakse tuleb kirjutada loogeliste sulgude vahele. Vastasel juhul arvestab JSX seda kui tavalist teksti. JSX märgenditele omaduste määramine erineb HTML märgendite omaduste määramisest. Omaduste nimed on muudetud küürkirja, nt *Class* on JSX märgendi puhul *className* ja *tabindex* on *tabIndex*[10].

### 2.4.2 Virtual Document Object Model

Virtual Document Object Model (edaspidi VDOM) on programmeerimiskontseptsioon, kus hoitakse virtuaalselt uut DOM objekti. VDOM sünkroniseeritakse veebilehitsejas oleva DOM objektiga kasutades tehnoloogiale vastavat teeki, Reacti puhul ReactDOM[11]. Selle lähenemise puhul uuendatakse DOM objekti ainult nendes kohtades, kus muudatused on toimunud. Reacti võrdlemise algoritm teeb kindlaks, kas juurelemendi tüübid on samad. Erineva tüübi tuvastamise korral ehitatakse uuesti kogu võrreldav objekti osa. Samade tüüpide puhul kontrollitakse, kas objekti omadused on samad. Erinevuste tuvastamisel uuendatakse omadused. Peale kontrolle rakendatakse sama algoritmi alamobjektidele. Selle loogika teostamisel kasutati Heuristic algoritmi[12].

### 2.4.3 Create React App

Create React App (edaspidi CRA) on käsureainterpretaator, mis seab kasutaja jaoks valmis minimaalse sisuga React projekti. Vajadus CRA jaoks tekkis kuna Reactil põhinevate projektid kasutavad tüüpiliselt paljusid populaarseid tööriistu, mis oli vaja konfigureerida ennem kui arendajad saavad alustada koodi kirjutamisega. CRA loob minimaalse Reacti projekti, millel on juba valmis seatud vajaminevate lisatööristade konfiguratsioonid ja hoiab ka nende konfiguratsiooni failid kasutajale peidetuna, et vältida ebajalike muudatusi[13].

## 2.5 GraphQl

GraphQL on spetsiaalne päringukeel tagarakendite loomiseks, mille abil saavad rakendused andmeid pärida. GraphQL on loodud Facebooki poolt. Arendamist alustati aastal 2012 ja esimene algeline versioon tuli välja aastal 2015. 2016. aasta oktoobris tuli välja esimene stabiilne versioon GraphQL-ist. Facebooki idee oli luua uut tüüpi suhtlusviis andmebaasi ja kliendi vahel. GraphQL suurimaks eeliseks REST-i ees peetakse võimalust pärida serverilt kõik andmed ühe päringuga, REST-i puhul võib selleks vaja minna rohkem kui ühte päringut. Lühidalt öeldes kontrollib GraphQL-i kasutades päringu vastust klient, REST-il kontrollib päringu vastust server. REST-i puhul

vastab igale aadressile kindlat tüüpi info, GraphQL seevastu aga kasutab kõikide andmete edastamiseks vaid ühte kindlat aadressi[17].
GraphQL on mitmeplatvormiline päringukeel, mida on võimalik kasutada Javascriptis, Rubys, Scalas ja paljudes muudes programeerimiskeeltes[18].

### 2.5.1 Päringud

GraphQL toetab kahte tüüpi päringuid, tavalisi andme päringuid ja mutatsioone. Tavalised andmepäringud tagastavad kliendile ainult andmeid, samas kui mutatsioonid on mõeldud andmebaasis muudatuste tegemiseks. GraphQL päringud on sarnased JSON tüüpi andmestruktuurile. Päringus tuleb esialgselt ära kirjeldada, kas tegemist on päringu või mutatsiooniga. Päringu jaoks kasutatakse tähistust *query*(Joonis 29) ja mutatsiooni jaoks kasutame *mutation*. Mutatsioonil ja päringu puhul kasutatakse sama struktuuri[19].

```
query päringNimele {
  name
}
```

Joonis 29. Näidis päring

Kui andmebaasis on defineeritud väärtus muutujale “nimi”, siis joonisel 29 välja toodud päring tagastab sellele vastava väärtuse(Joonis 30).

```
{
  nimi: "Peeter"
}
```

Joonis 30. Näide päringu vastusest

Lisaks saab teha ka päringuid, kus saab kaasa anda muutujaid, mille põhjal tagastatakse sellele muutujale vastav väärtus (Joonis 31).

```
query päringKoosMuutujatega($isikukood: string) {
    isik(isikukood: $isikukood) {
        nimi
    }
}
```

Joonis 31. Näide muutujatega päringust

Joonisel 31 näidatud päring nimega “päringKoosMuutujatega” nõuab, et talle antaks kaasa muutuja nimega isikukood, mis on string tüüpi. Päring oskab selle muutuja põhjal tagastada isiku kelle isikukood on võrdne päringule kaasa antud isikukoodiga.

### 2.5.2 Skeem

GraphQL toetab lisaks ka andmete struktuuri kirjeldust. Selle abil saab server kontrollida, et andmed oleks oodatud tüüpi. Lisaks on võimalik ära kirjeldada projektile eriomaseid andmetüüpe. Baasandmetüüpideks on Int*, Float, String, Boolean* ja ID. Int on 32 bitine number. *Float* on kahe komakoha täpsusega ujuvkoma arv. String on tekstitüübi väärtus ja boolean on tõeväärtusega andmetüüp. ID väärtustab unikaalset väärtust, mis on string tüüpi. Andmetüüpe saab ka määrata *Non-Null*-iks, mis tähendab, et selle väärtus ei saa olla null-tüüpi ja peab alati tagastama enda tüüpi väärtuse. Selle tähiseks on skeemis hüüumärk. Masiive saab tähistada, kui lisada andmetüübile ümber kantsulud, nt. [Int] tähistab Int tüüpi masiivi. GraphQL toetab ka liideseid (inglise keeles *interface*), mille eesmärgiks on luua projektile eriomaseid andmetüüpe (Joonis 32)[20].

```
interface isik {
  isikukood: Int!
  eesNimi: String!
  kesmineNimi: String
  pereNimi: String!
  vanus: Int!
  lapsed: [isik]
}
```

Joonis 32. Näide GraphQl liidesest.

Näitest on näha, et liidesel isik on “eesNimi”, “pereNimi”, mis on teksti tüüpi *Non-Null* väljad. Lisaks isikukood ja vanus, mis on Int tüüpi *Non-Null* väljad. Väli “lapsed” on masiiv isik tüüpi objektidest, mis ei pea alati vastust tagastama. Samamoodi võib väli “keskmineNimi” tagastada tühja väärtuse.

## 2.6 Es2015-i18n-tag

Es2015-i18n-tag on JavaScripti teek, mis lisab projektile i18n ja i10n toetuse (tõlkimine ja rahvusvahelistumine). Teek kasutab tõlkevõtmete kirjeldamiseks sõneliteraale, mis lubab kasutajal luua dünaamilisi tõlkevõtmeid. Lisaks kasutatakse JSON skeemi, mille abil on kerge tõlkevõtmeid ja tõlkeväärtusi hallata. Igale JSON skeemi võtmele vastab kindel sõneliteraal tüüpi tõlkevõti. Lisaks pakub teek erinevaid funktsioone hallata projektis kasutatavaid tõlkevõtmeid. Es2015-i18n-tag teek järgib ECMA-402 standardit[14].

## 2.7 Prettier

Prettier on koodivormindi, mis hoiab koodi vormi etteantud reeglite põhjal ühtlasena[15].

# 3. Arhitektuur

Käesoleva peatüki eesmärk on ära seletada, kuidas on kavandatud projekti arhitektuur, kuidas erinevad tehnoloogiad selles projektis töötavad ja milliseid programmeerimise tavasid kasutatakse.

## 3.1 Koodi vorm

Projektis kirjutatud TypeScripti koodi muutujate ja liideste nimetamisel on kasutatud küürkirja, mis tagab koodi loetavuse pikemate muutuja nimede korral. Failinimede puhul on kasutatud kahte erinevat küürkirja stiili varianti. Vaadete, vaatepõhiste loogika ja komponent failide esimene täht on suure esimese tähega. Vastav eriomadus lubab projekti kaustas eristada vaadetega seotud faile. Projektis kasutatavad abi failid kasutavad väikse algustähega küürkirja stiili, et eristada neid vaadete failidest. Samuti kasutatakse liideste nimetamisel suurtähte. Lisaks on liidestel alati prefiks “I”, et eristada neid koodimuutujatest. Projektis kasutatavate muutujate puhul kasutatakse reeglina tavalist küürkirja stiili, kuid seoses erinevate kirjastiilidega eessüsteemis ja tagarakenduses, tuli kasutada tagarakendusega seotud muutujatel madukirja stiili (ingl *snake case*).

Projektis on kasutusel järgnevad Prettier reeglid koodi vormindamiseks:

```
1. arrowParens: avoid;
2. bracketSpacing: true;
3. disableLanguages: ["vue"];
4. eslintIntegration: false;
5. ignorePath: .prettierignore;
6. jsxBracketSameLine: false;
7. parser: babylon;
8. printWidth: 80;
9. proseWrap: preserve;
10.  requireConfig: false;
11.  semi: true;
12.  singleQuote: false;
13.  stylelintIntegration: false;
14.  tabWidth: 2;
15.  trailingComma: none;
16.  useTabs: false.
```

Projekti kaustas asuva faili salvestamisel kontrollib Prettier, kas faili sisu vastab eelnevalt välja toodud reeglitele. Kui esineb vastuolu, siis vormindab käesoleva faili ümber vastavalt reeglitele.

TypeScripti koodi korrektsust kontrollib projektis tslinter, mis jälgib ette antud reeglite hulka. Projektis on kasutusel järgnevad tslinteri reegli hulgad:

- tslint:recommended;

- tslint-react;
- tslint-config-prettier.

Koodis esinevad vead etteantud reeglite põhjal edastatakse kasutajale integreeritud programmeerimiskeskkonnas ja koodi kompileerimisel. Vajadusel on võimalik keelata reeglite jälgimist koodis. Selleks tuleb koodis kommentaarina lisada vastav lipp järgnevatest valikutest:

- `/* tslint:disable */`. Keelab reeglite kasutamise kogu failis alates lipu asukohast;
- `// tslint:disable-next-line.` Keelab reeglite kasutamise järgneval real;
- `// tslint:disable-next-line:rule1 rule2 rule3`. Keelab etteantud reeglite kasutamise järgneval real[16].

Välja toodud reeglid ja koodivormindid aitavad arendajatel vältida koodivigu ja hoiab koodibaasi ühtlaselt loetavana.

## 3.2 Projekti ülesehitus

Käesolevas alapeatükis kirjeldatakse ära projekti ülesehituse ja seletatakse, miks on tehtud vastavad valikud arhitektuuri osas.

### 3.2.1 Allikaskataloog

Allikaskataloog sisaldab järgnevaid kaustu:

1. *Components*;
2. *Constants*;
3. *Gfx*;
4. *Mutations*;
5. *Queries*;
6. *Services*;
7. *Typings*;
8. *Views*.

Lisaks sisaldab allikaskataloog veel projekti juurfaili `index.tsx`, mis on projekti struktuuris kõige kõrgemal asuv fail. Index failist asub dokumendiobjektide mudeli puu juur element.

### Components

Kaust *Components* sisaldab projektis kasutatavaid komponente. Projektis esineb kahte erinevat tüüpi komponente: projektis kasutatavad komponendid ja vaatepõhised komponendid. Kui komponenti kasutab rohkem kui ühte vaadet, hoitakse komponendi koodi allikaskataloogi *Components* kaustas. Vaatepõhise komponendi koodi hoitakse vastava vaate kaustas.

### Constants

Kaust *Constants* sisaldab faile, mis talletavad projektiga seotud globaalparameetreid. Heaks tavaks on hoida ühes kohas kõiki konstante, mida kasutavad erinevad komponendid ja vaated projektis. Juhul kui tekib vajadus muuta kindlat konstanti, siis tuleb teha muudatus ainult ühes failis.

### Gfx

Kaust *Gfx* sisaldab projekti fondi stiili, kasutatud pilte ja üldist projekti kujundust. Komponentide ja vaadete põhist kujundust hoitakse vastava elemendi kaustas.

### Mutations

Kaust *Mutations* sisaldab projektis kasutatud GraphQl mutatsiooni faile.

### Queries

Kaust *Queries* sisaldab projektis kasutatud GraphQl päringu faile. Projektis hoitakse päringud ja mutatsiooni failid eraldi kaustades, et hoida erinevate eesmärkidega failid eraldi kaustades.

### Services

Kaust *Services* sisaldab projektiga seotud abifunktsioonide jaoks loodud faile. Selle alla kuuluvad nt. brauseri lokaalsest mälust info küsimise funktsioonid.

### Typings

Kaust *Typings* sisaldab faile, milles kirjeldatakse projektis kasutatavate objektide tüübidefinitsioone.

### Views

Töö käigus loodav eessüsteem on jagatud avalikuks ja privaatseks osaks. Sellest tulenevalt on vaated jagatud avalikeks ja privaatseteks vaadeteks.
Avalike vaadete hulka kuuluvad järgnevad vaated:

1. *About*;
2. *Help*;
3. *Index*;
4. *Introduction*;
5. *Login*;
6. *News*;
7. *News-article*;
8. *Not-found*;
9. *Prices*;
10. *Privacy*;

11. *Request-for-proposal*.

Privaatsete vaadete hulka kuuluvad järgnevad vaated:

1. *Association*;
2. *Invoices*;
3. *Messages*;
4. *Overview*;
5. *Readings*;
6. *Settings*;
7. *User-interface-guidelines*.

Siinkohal tuleb ära märkida, et vaate *User-interface-guidelines* on ainult arenduskeskkonnas ligipääsetav. Vaate eesmärgiks on demonstreerida projektis kasutatud stiilide ja komponentide välimust.

### 3.2.2 Vaated ja komponendid

Vaadete ja komponentide kaustad sisaldavad loogika faili, Html malli, scss tüüpi stiili faili ja stiili faili põhjal genereeritud kaskaadlaadistik faili. Vaatepõhise komponendi korral sisaldab vaate kaust lisaks eelnevalt välja toodud failidele ka komponendi kausta. Vaatepõhiseid kompinente hoitakse vaatega samas kaustas, et hoida kõik vaatele kuuluvad elemendid ühes kohas.

## 3.3 Kujundus

Projektis kasutatav kujundus on kirjutatud Sass (ingl *Syntactically Awesome Style Sheets*) stiili keeles. Projekti kompileerimisel genereeritakse Sass failist kaskaadlaadistik fail, mida veebilehitsejad kasutada oskavad.

## 3.4 Tõlkesüsteem

Projekti juur failis on ära määratud, et iga juurfaili tütarelement on tütarelement komponendile *TranslationsProvider*. Komponendi *TranslationsProvideri* eesmärgiks on pakkuda igale tütarelemendile ligipääs hetkel valitud tõlgetele. Rakenduse käivitamisel küsib *TranslationProvider* serverilt baasseadistuses ära märgitud keele tõlked ja rakendab need eesrakenduses. Tõlgete väljakutsumiseks eesrakenduses tuleb sisestada HTML mallis i18n märgend, millele järgneb tõlkevõtme sõneliteraal. Sisestatud i18n märgend annab rakendusele teada, et järgnev sõneliteraal on tõlkevõti, mille järgi määrata sisestatud kohale vastav tõlge. Juhul kui tõlkevõtmele ei leitud vastavat tõlget, määratakse sisestatud kohale tõlke asemel ette antud tõlkevõti. Seoses tehnoloogia omadusega kasutada väärtusena tõlkevõtit olukorras, kus ei leidu tõlget, otsustasime kasutada inglise keelt tõlkevõtmete kirjutamisel. Iga tõlkevõti on inglisekeelne tõlge vastaval positsioonil asuvale tekstile.

Lahenduse idee on, et kui tõlkevõti puudub, siis ei ole rakenduse töö täielikult häiritud. Lisaks saavad ka arendajad kiiremas korras selgusele, milline tõlge andmebaasist puudu on.

### 3.4.1 Tõlgete haldamine

Kasutusel olev teek pakub arendajatele võimaluse jooksutada skripti, mille eesmärgiks on üle koodibaasi kokku koguda kõik kasutusel olevad tõlkevõtmed. Vastavalt skripti tulemusele on võimalik kiirelt ja tõhusalt selgeks teha, millistel tõlkevõtmetel puuduvad andmebaasis vastavad tõlked.

## 3.5 Suhtlus tagarakendusega

Projekti juurfailis on ära määratud, et iga juurfaili tütarelement on tütarelement komponendile *ApolloProvider*. Komponent *ApolloProvider* haldab rakenduses päringute ja mutatsioonide saatmist serverile ning tagarakenduselt tagasi saadud vastuseid. Tagarakendusest andmete küsimiseks kasutatakse GraphQl päringuid ja andmete muutmiseks mutatsioone.

### 3.5.1 Päringud

Rakenduses kasutatavad päringud on kirjeldatud kaustas *queries*, mis asub allikaskataloogis. Vaade impordib vajaliku päringu, mis edastatakse tagarakendusele. Tagarakendus saadab vastavalt päringule tagasi eesrakenduse küsitud andmed. Andmete pärimise ajal näidatakse kasutajale laadimise vaadet. Kui päringu vaste on jõudnud esirakendusse, vahetatakse laadimise vaade ära kasutaja poolt avatud vaatega.

### 3.5.2 Mutatsioonid

Rakenduses kasutatavad mutatsioonid on kirjeldatud kaustas *mutations*, mis asub allikaskataloogis. Vaade impordib vajaliku mutatsiooni. Mutatsiooni saatmisel tagarakendusele lisatakse kaasa mutatsioonis nõutud informatsioon. Sellele järgnevalt saadab tagarakendus tagasi mutatsiooni tulemuse. Mutatsiooni õnnestumise korral kuvatakse kasutajale õnnestumise teadaanne. Ebaõnnestumise korral saadakse tagarakenduselt kirjeldus vea kohta, mis kuvatakse kasutajale.

# 4. Praktilise töö tulemus

Käesolev peatükk võtab kokku praktilise töö tulemuse. Bakalaurusetöö esitamise hetkeks on arendus kestnud neli kuud. Esitamise ajaks on valmis saadud suurem osa avaliku rakenduse vaadetest ja komponentidest. Puudu on lehekülje valija vaates “Uudised” ning otsingu komponent vaates “Abi”.

Privaatses osas on valmis saadud vaade “Seaded”. Vaadetes “Ühistu” ja “Arved” on puudu graafikud. Vaates “Näidud” on valmis kasutajale andmete kuvamine. Juurde on vaja lisada näitude sisestamine. Vaates “Sõnumid” on teostatud kasutajale sõnumite kuvamine ning erinevate sõnumite vahel

valimine. Lisaks on vaja juurde lisada sõnumitele ja teadetele vastamine ja sõnumite staatuse muutmine. Privaatse osa menüüs on teostamata sõnumite valiku juures olev märguanne, mis tähistab lugemata sõnumite arvu. Privaatsele osale tuleb juurde lisada kasutaja põhised õigused rakenduses.

## Kokkuvõte

Käesoleva bakalaurusetöö eesmärgiks oli luua kaasaegse välimusega Korto veebirakenduse eessüsteem.

Töö tulemusena on valmis loodud rakendus, mis võimaldab kasutaja näha ühistuga seotud sõnumeid, kinnisvaraga seotud arveid ja ühistu andmeid. Lisaks saab kasutaja näha ja muuta kinnisvara, kinnisvaraga seotud isikute ja kasutaja andmeid.

Veebirakenduse arendamist jätkatakse peale töö esitamist edasi, sest rakenduse kõik vajalikud funktsioonid ei ole lõpuni teostatud. Valmis on vaja teostada järgnevad funktsioonid:

1. Sõnumitele ja teadetele vastamine;
2. Sõnumite staatuse muutumine;
3. Privaatse osa menüü lugemata sõnumite märguanne;
4. Näitude sisestamine;
5. Kasutaja õigused rakenduses;
6. Graafikud.

Tarkvaralabor OÜ poolt on paika pandud, et rakenduse esimeses versioonis peavad valmis saama tegemata töödest esimesed viis punkti. Graafikute lisamine esimeses versioonis ei ole suure tähtsusega ning lükati edasi järgnevasse versiooni.

Teises peatükis välja toodud tehnoloogiad on valitud selletõttu, et töö kirjutaja töövõtja firmas kasutatakse igapäevatöös veebilahenduste loomiseks just neid tehnoloogiaid.

## Viited


[1] https://developer.mozilla.org/bm/docs/Web/JavaScript (viidatud 10.03.2018)
[2] https://www.upwork.com/hiring/development/the-basics-of-compiled-languages-interpreted-languages-and-just-in-time-compilers/ (vaadatud 10.03.2018)
[3] https://developer.mozilla.org/en-US/docs/Glossary/First-class_Function (vaadatud 10.03.2018)
[4] http://eloquentjavascript.net/00_intro.html#p_DYAUkRfhwQ (vaadatud 10.03.2018)
[5] https://medium.freecodecamp.org/whats-the-difference-between-javascript-and-ecmascript-cba48c73a2b5 (vaadatud 10.03.2018)
[6] https://www.tutorialspoint.com/typescript/typescript_overview.htm (vaadatud 11.03.2018)
[7] https://github.com/Microsoft/TypeScript/wiki/Using-the-Language-Service-API (vaadatud 12.03.2018)
[8] https://www.tutorialspoint.com/typescript/typescript_types.htm(vaadatud 21.03.2018)
[9] https://www.c-sharpcorner.com/article/what-and-why-reactjs/ (vaadatud 22.03.2018)
[10] https://reactjs.org/docs/introducing-jsx.html (vaadatud 24.03.2018)
[11] https://reactjs.org/docs/faq-internals.html(vaadatud 02.04.2018)
[12] https://reactjs.org/docs/reconciliation.html(vaadatud 02.04.2018)
[13] https://reactjs.org/blog/2016/07/22/create-apps-with-no-configuration.html(vaadatud 02.04.2018)
[14] https://github.com/skolmer/es2015-i18n-tag (vaadatud 07.04.2018)
[15] https://github.com/prettier/prettier (vaadatud 10.04.2018)
[16] https://palantir.github.io/tslint/usage/rule-flags/ (vaadatud 12.05.2018)
[17] https://graphql.org/ (vaadatud 20.03.2018)
[18] https://graphql.org/code/ (vaadatud 20.03.2018)
[19] https://graphql.org/learn/queries/ (vaadatud 20.03.2018)
[20] https://graphql.org/learn/schema/ (vaadatud 20.03.2018)

# Lisad

## 1. Joonised

## Litsents

**Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks**

Mina, **Mihkel Puusepp**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   **Tarkvaralabor OÜle Korto kasutajaliidese loomine**,
(*lõputöö pealkiri*)

   mille juhendaja on Vambola Leping,
   (*juhendaja nimi*)

   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **12.05.2018**