Tallinna Ülikool

Digitehnoloogiate instituut

Meteor.js raamistiku eestikeelne õppematerjal

Seminaritöö

Autor: Marten Rebane

Juhendaja: Jaagup Kippar

Autor: ..... „.....„2016

Juhendaja: ..... „.....„2016

Instituudi juhataja: ..... „.....„2016

Tallinn 2016

Sõnastik

API (Application Programming Interface)	rakendusliides
back-end	tagaosa
Bootstrap	raamistik, mis võimaldab teha dünaamilisi veebilehti
checked	linnuke märgitud kasti sisse
child	laps
dependencies	sõltumised, välised paketid mille peale projekt toetub
events	sündmused
form	vorm
front-end	esiosa
helpers	abilised
id	identifikaator
imports	importimised
npm	paketihaldur
package	pakett
parent	vanem
tag	märgis
template	šabloon, mall
todo	ülesanne

Sisukord

Sissejuhatus

Käesoleva seminaritöö eesmärgiks on luua eestikeelne õppematerjal Meteor.js raamistiku kohta, mis annab ülevaate raamistiku enda, olemasolevate õppematerjalide kohta, õpetada installeerima raamistikku arvutisse, luua näiteprogramm näitamaks raamistiku võimalusi ja lisada disain välise raamistikuga.

Seminaritöö autor on valinud just selle raamistiku, sest eestikeelsed õppematerjalid puuduvad ning see on kiirelt arenev tehnoloogia.

Õppematerjali võivad kasutada nii algajad, kes pole Meteor raamistikuga kokku puutunud, kui ka need, kes on sellega kokku puutunud ning soovivad algajate asju üle korrata.

Käesolev seminaritöö on jaotatud kolmeks osaks: esimeses osas antakse ülevaade raamistikust, teises osas arvutisse installeerimine ning kolmandas osas näiteprogrammi koostamine.

1 Meteor.js veebiraamistiku tutvustus

Meteor.js on avatud lähtekoodiga reaktiivne JavaScripti raamistik, mis võimaldab teha reaalajas töötavaid rakendusi. Reaktiivne tähendab, et kõik raamistiku osad töötavad koos, et reageerida muudatustele reaalajas (Hochhaus & Schoebel, 2015). Brauser ei tegele vaid andmete näitamisega, vaid jälgib ka muudatusi andmetega, et "reageerida" nendele muudatusetele (Strack, 2012).

Raamistik on kirjutatud täielikult JavaScriptis ning sellega on võimalik kirjutada *front-end* ja *back-end* koodi vaid ühes programmeerimiskeeles. Kõik rakendused töötavad reaalajas, mis tähendab, et kõik muudatused lehel tehakse koheselt ilma veebilehe värskenduseta. Arendamise ajal salvestades muudatused, tehakse veebilehele automaatselt värskendus ning tulemused on kiirelt näha.

Kasutada saab pakette, mida on võimalik projektidesse lisada. Saadaval on nii Meteor.js-i enda, kui ka NPM süsteemi paketid. Meteor.js pakette on võimalik leida Atmosphere lehelt[1] ning NPM pakette NPM lehelt[2].

Meteor.js-is on kasutusel *Spacebars*, mis on *handlebars*-i sarnane šabloonkeel ning võimaldab muuta reaktiivselt andmete konteksti. *Spacebars*-i šabloonid näevad välja nagu lihtne HTML koos spetsiaalsete "vuntside" märgistega, milleks on looksulud {{ }} (Spacebars templates).

Andmebaasiks on kasutusel MongoDB, mis on NoSQL andmebaas, kus on kasutusel JSON kujul dokumendid.

Selle raamistikuga on võimalik teha ka hübriidrakendusi mobiilidele, kuid hetkel (oktoober 2016) on see võimalus vaid macOS süsteemidel. Windowsi tugi puudub.

Meteori lehel on välja toodud erinevad projektid, kus raamistikku on kasutatud. Populaarsemad neist on Mazda auto konfiguraator, millega saab valida Mazda autosid, Classcraft, mis aitab õpetajatel tundi läbi viia tehes tunnist rollimängu, Rocket.Chat, millega saab sõnumineerida grupis ja teha videokõnesid (Meteor). Need ja paljud teised veebilahendused on välja toodud Meteori lehel, mis asub Meteor Showcase lehelt[3].

[1] https://atmospherejs.com/
[2] https://www.npmjs.com/
[3] https://www.meteor.com/showcase

Meteori rakendust on võimalik majutada ka veebis. Kõige lihtsam lahendus sellele oleks kasutada Meteor Galaxy[4] veebimajutust. Teenus on tasuline, kuid pakub kõige lihtsamat ja kiiremat lahendust rakendust kiirelt veebis tööle saada. Pärast teenuse ostmist on võimalik tehtud rakendus veebi saata vaid "meteor deploy" käsuga.

Tasub tähele panna, et kõik Meteor.js-ga tehtud rakendused on alguses ebaturvalised. See on selletõttu, et prototüüpimine oleks kiirem ning ei ole kohe mõeldud produktsiooni saatmisele. Meteor on ebaturvaline kahe paketi tõttu – autopublish ja insecure. Autopublish näitab kõiki andmebaasis olevaid asju brauseris ning insecure lubab igaühel muuta andmeid andmebaasis, täpselt nagu neil oleks serveri ligipääs (Greif, 2014). Enne produktsiooni saatmist, tuleks need paketid eemaldada. Meteori turvalisuse kohta on rohkem võimalik lugeda Discover Meteor lehelt[5].

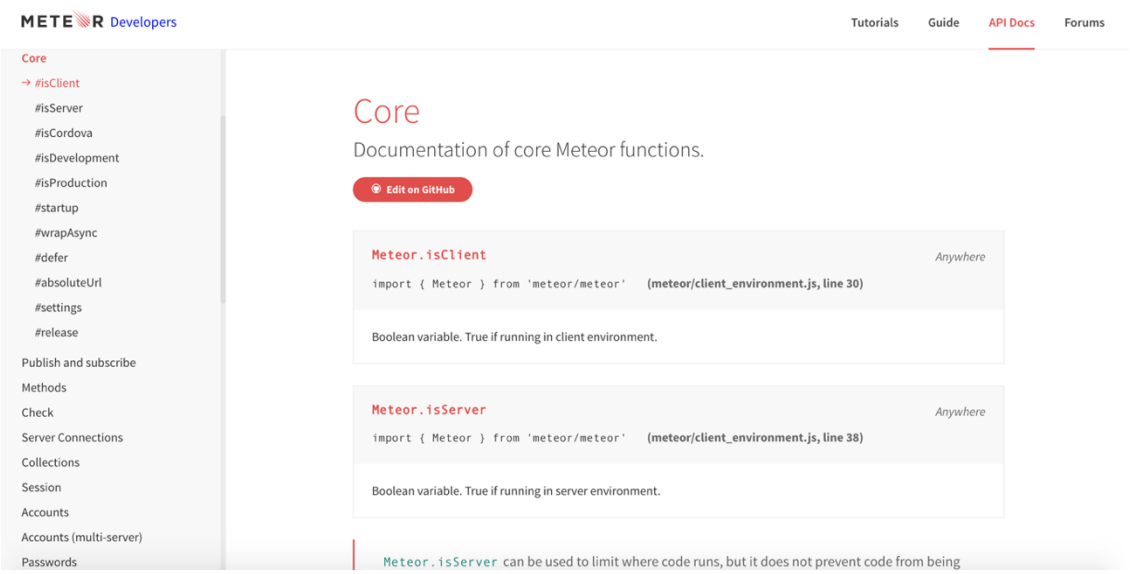
1.1 Meteor.js õppematerjalid

Meteor.js raamistiku kohta on õppematerjale enamasti vaid inglise keeles. Raamistik uueneb kiirelt ning paljud raamatud kui ka internetist leitavad õppevahendid on jäämas vanaks. Välja olen toonud dokumentatsiooni ja YouTube videod, sest neis on kõige rohkem infot Meteor.js õppimise kohta. YouTube videotes on nii algajatele kui ka edasijõudnumatele kasutajatele õpetusi.

1.2 Meteor.js dokumentatsioon

Meteor.js dokumentatsioon on koostatud nii, et vasakul küljel on kirjas raamistiku API (*Application Programming Interface*), paketid ja käsuviiba võimalused. Vaadates teatud võimalusi lähemalt, on näha kuidas meetodit kasutatakse, mida see teeb ning paremal pool on kirjas kus seda kasutada võib (vt Joonis 1).

[4] https://www.meteor.com/hosting
[5] https://www.discovermeteor.com/blog/meteor-and-security/



Joonis 1. Meteor.js dokumentatsioon

Dokumentatsiooni võib leida Meteor dokumentatsiooni lehelt[6].

1.3 YouTube õpetusvideod

YouTube kanal LevelUpTuts on teinud ingliskeelsed õpetusvideod nii algajatele kui ka edasijõudnutele. Algajate osas õpetatakse erinevaid Meteor.js aspekte ning edasijõudnute osas minnakse keerulisemate kontseptide peale.

Algajate osa leiab YouTube lehelt[7].

Edasijõudnud osa leiab samuti YouTube lehelt[8].

[6] https://docs.meteor.com/

[7] https://www.youtube.com/watch?v=hgjyr6BPAtA&list=PLLnpHn493BHECNl9I8gwos-hEfFrer7TV

[8] https://www.youtube.com/watch?v=BI8IslJHSag&list=PLLnpHn493BHFYZUSK62aVycgcAouqBt7V

2 Meteor.js installeerimine

2.1 Installeerimine macOS ja Linuxile

Meteor.js tuleb installeerida enne kasutamist süsteemi ning selleks tuleb minna Meteor lehele[9] ning vajutada sealt *Install now* (vt Joonis 2).

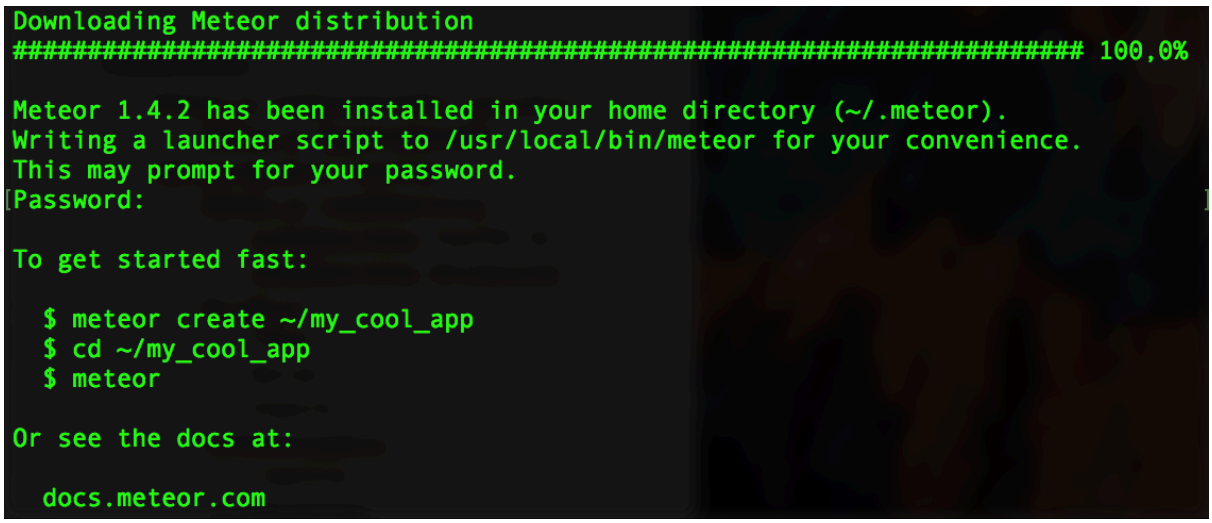
Joonis 2. Meteor.js koduleht

Lehelt on näha, et Meteor.js installeerimine macOS ja Linuxi operatsioonisüsteemidele on tehtav vaid ühe käsurea käsklusega, mille käivitamisel installeeritakse raamistik automaatselt:

```
curl https://install.meteor.com/ | sh
```

Installeerimise ajal võidakse küsida administraatori parooli. Pärast parooli sisestamist on Meteor.js kasutamisvalmis (vt Joonis 3).

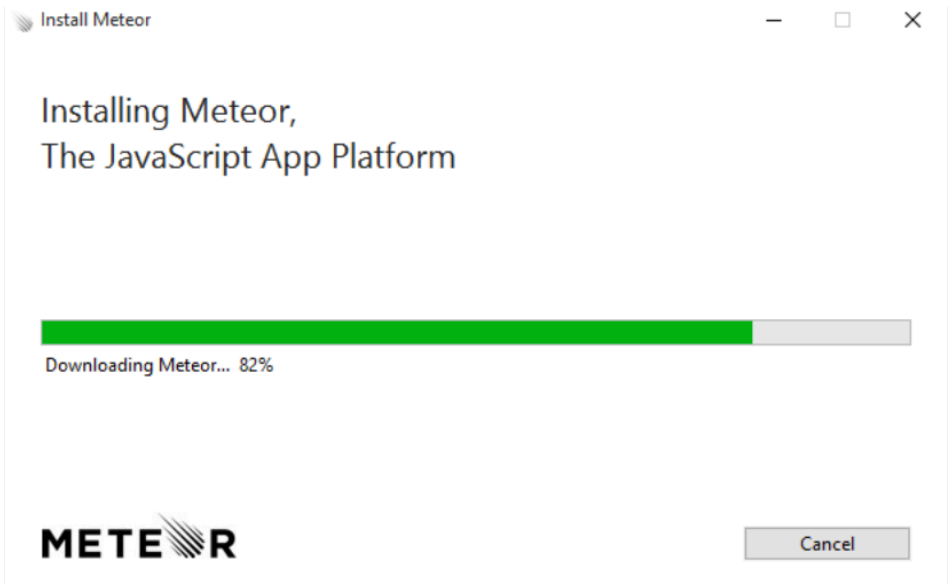
[9] https://www.meteor.com/



Joonis 3. Meteor.js installeerimine macOS-ile

2.2 Installeerimine Windowsile

Meteor.js installeerimiseks Windowsile tuleb minna Meteor lehele[9] ja seal vajutada *Install Now* nupule (vt Joonis 2). Järgnevalt vajutada *Download installer* nupule. Kui installer on allalaetud, tuleb installeerimiseks vaid fail avada ning Meteor.js installeeritakse automaatselt (vt Joonis 4). Installeerimise lõpus küsitakse, et kas soovite teha kasutajat, siis seal võib vajutada nupule "Skip this step".



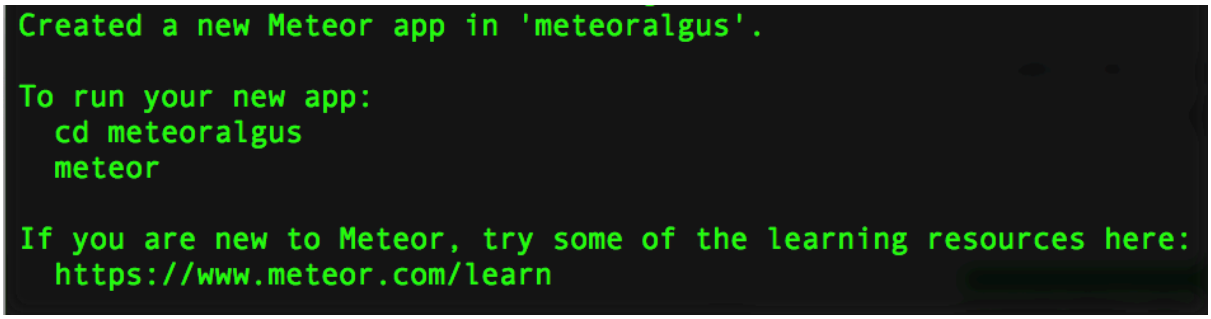
Joonis 4. Meteor.js installeerimine Windowsile

2.3 Projekti loomine ning käivitamine

Kui Meteor.js on installeeritud, tuleb teha uus projekt. Projekti asukoht on vabalt valitav ning selleks peab avama käsuviiba ning navigeerima kohta, kus projekti arendada. Projektinime võib igaüks ise valida. Tehakse näiterakendus valmis ning näidatakse õpetust, kuidas seda tööle saada (vt Joonis 5).

Selle õppematerjali näites on projektinimi "meteoralgus" ning käsureale tuleb kirjutada:

```
meteor create meteoralgus
```



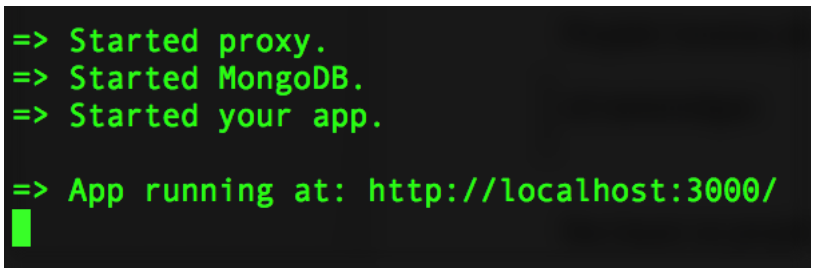
Joonis 5. Projekti loomine

Projekti loomise järel tuleb navigeerida projektikausta käsklusega

```
cd meteoralgus
```

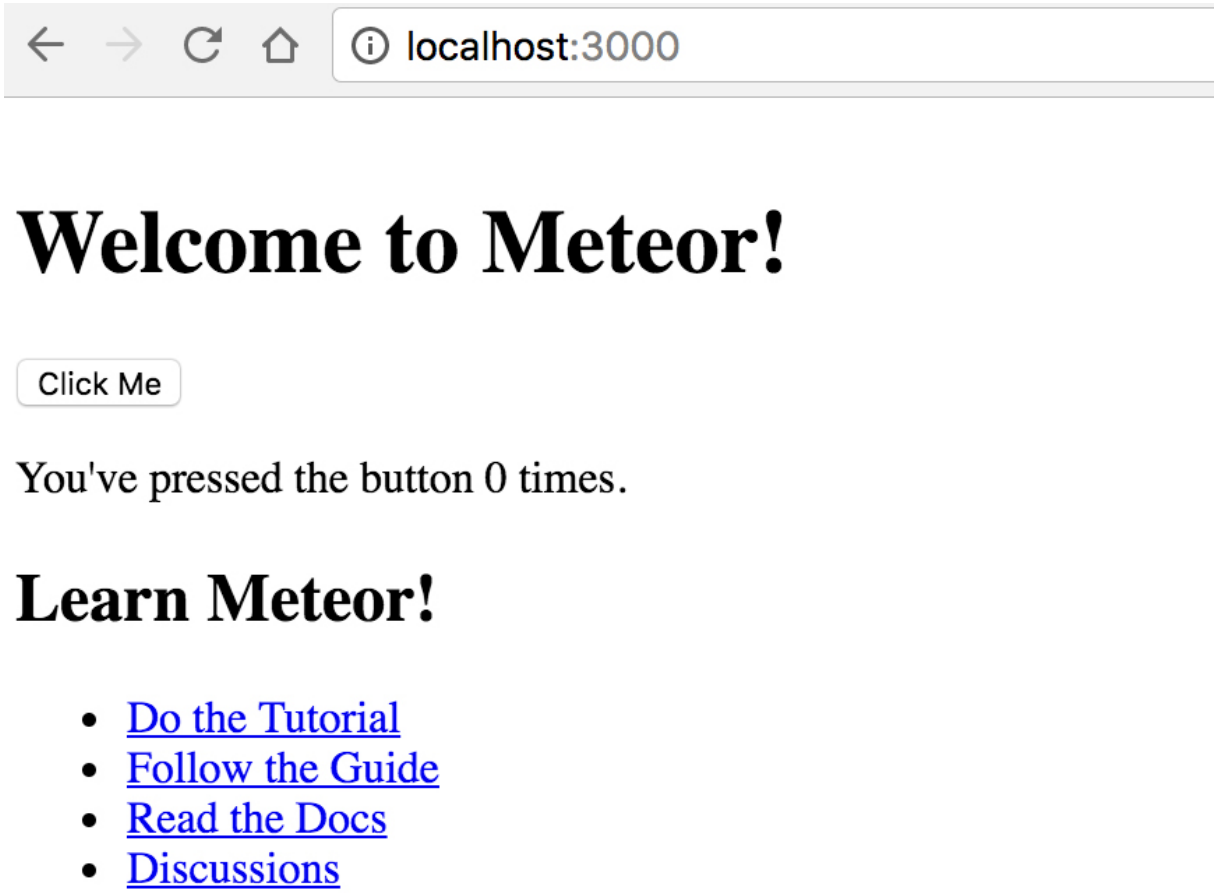
Projekti käivitamiseks tuleb käsuviibaga olla projekti juurkaustas ning ühe käsklusega tehakse kohalik server, loodakse kohaliku MongoDB andmebaasiga ühendus ning käivitatakse rakendus kohalikus serveris vaikimisi pordil 3000 (vt Joonis 6). Rakenduse käivitamiseks on käsklus

```
meteor
```



Joonis 6. Projekti käivitamine

Projekti nägemiseks veebis tuleb avada brauser ning navigeerida lehele http://localhost:3000/. Veebilehel on näha algset projekti, kus saab vajutada nupule, mis loeb, mitu korda on nupule vajutatud ning samuti on lehel erinevaid linke Meteor.js-ga tutvumiseks (vt Joonis 7).

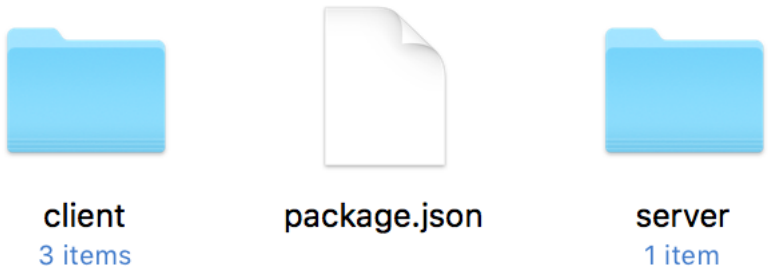


Joonis 7. Projekti algne veebileht

3 Näiteprogrammi loomine

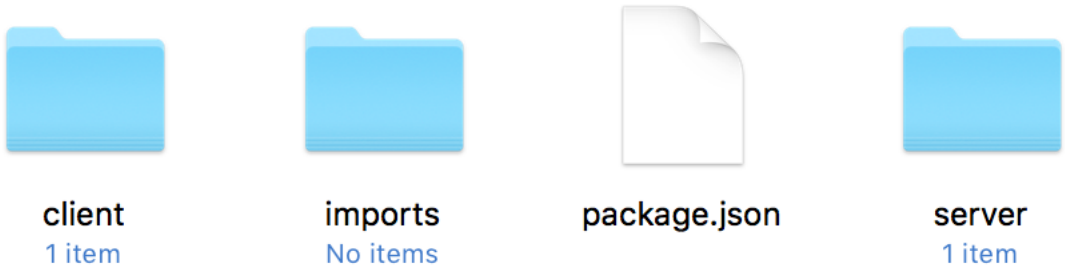
3.1 Projekti struktuur

Meteor.js projekti juurkaustas on algselt näha kahte kausta – *Client* ja *Server* ning ühte faili – package.json. Client kaust on mõeldud failidele, mis on nähtaval brauseris kasutajale. Hetkel on seal kolm faili – main.css, main.html ja main.js. Server kaust on mõeldud failidele, mis töötavad serveri peal ning ei ole brauseris kliendile nähtavad nagu näiteks andmebaasi ühendus. Package.json failis on kirjas projektiga seotud andmed nagu projekti nimi, skriptid ja *dependencies*, kus on kirjas erinevad paketid, mis on projektis kasutuses (vt Joonis 8).



Joonis 8. Algne kaustastruktuur

Meteor.js uuematel versioonidel on kaustastruktuur muutunud. Lisaks eelnevatele failidele on juurde lisatud kaust nimega *imports*. Imports kausta pannakse kõik failid, mis pärast imporditakse vastavalt kliendipoolel ja serveripoolel. Selleks tuleb teha juurkausta juurde uus kaust nimega *imports* (vt Joonis 9).

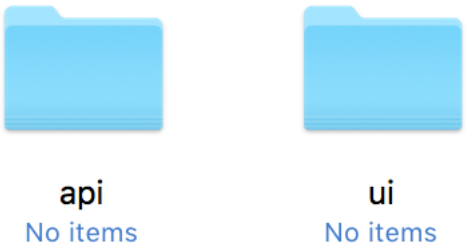


Joonis 9. Lisatud *imports* kaust

3.2 Näiteprogrammi funktsionaalsus

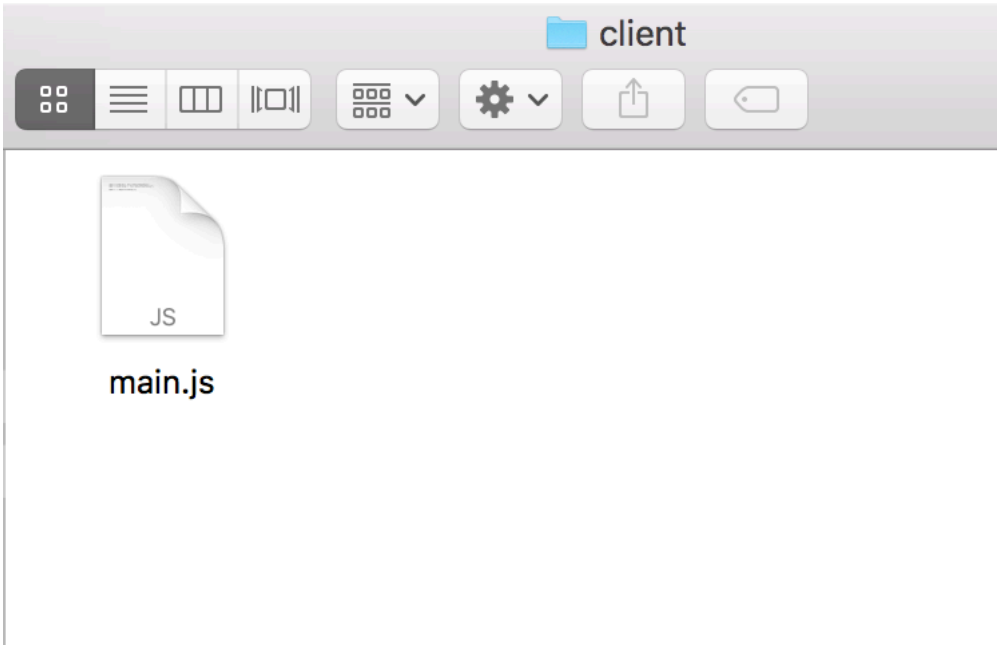
3.2.1 Projekti ettevalmistamine

Enne näiteprogrammi tegemist, tuleb eemaldada Meteori enda projektikood ning lisada teatud kaustad juurde. Kausta *imports* tuleks teha kaustad api ja ui. Api kaustas käivad serveripoolsed failid ning ui kaustas kliendipoolsed failid (vt Joonis 10).



Joonis 10. Api ja ui kaustad

Kaustast *client* eemaldada failid main.html ja main.css. Neid faile ei ole vaja, kuna failid laetakse *imports* kaustast. *Client* kausta jääb vaid main.js (vt Joonis 11).



Joonis 11. Eemaldatud mittevajalikud failid

Main.js failis tuleb eemaldada mittevajalik koodiosa. Kõik read peale esimese, peaks kustutama. Imporditakse *Template*, sest sellega on võimalik kirjutada malle ning sündmuseid (vt Koodinäide 1).

```
import { Template } from 'meteor/templating';
```

Koodinäide 1. Vajalik koodiosa

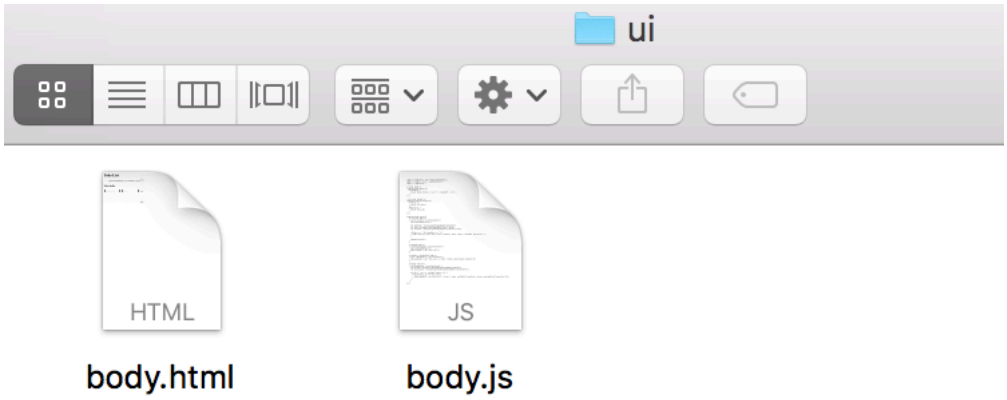
Kausta imports/api tuleb teha uus fail nimega todos.js. Todos.js faili lisatakse andmebaasi kollektsiooni nimi, mis on kasutusel MongoDB andmebaasis (vt Koodinäide 2).

Alguses imporditakse Mongo kasutusvõimalus meteor/mongo paketist ning teisel real tehakse uus kollektsioon MongoDB andmebaasi ning eksporditakse, et seda saaks kasutada projekti teistes failides andmebaasi päringute jaoks. Const võtmesõna tähendab konstanti, et seda väärtust muuta võimalik ei ole.

```
import { Mongo } from 'meteor/mongo';
export const Todos = new Mongo.Collection('todos');
```

Koodinäide 2. Andmebaasi kollektsioon

Imports/ui kausta tuleb teha kaks faili, body.html ja body.js (vt Joonis 12). Hiljem need kaks faili imporditakse ning näidatakse kasutajale leheküljel.



Joonis 12. Kliendipoolsed failid

Body.js failis tuleb importida Template, Todos, ja eelnevalt tehtud fail body.html. Todos imporditakse eelnevalt tehtud todos.js failist, kus eksporditi MongoDB kollektsioon ning selles failis tehakse andmebaasipäringuid. Kui tavaliselt kasutatakse JavaScripti faile HTML failides <script></script> märgiste vahel, siis Meteoris tuleb importida HTML fail JavaScripti failis.

Template.body on *parent* mall, mis on seotud <body> *tag*-iga, mille sees saab olla teisigi malle ning see tagastab massiivi. HTML-is body *tag*-i sees saab kasutada {{#each leiaTodod}}, mis käib tsükli läbi ja sisestab iga väärtuse malli. "leiaTodod" funktsioon väljastab kõik ülesanded, mis andmebaasi kollektsioonis on ning järjestab uuemad ülesanded ettepoole (vt Koodinäide 3).

```
import { Template } from 'meteor/templating';
import { Todos } from '../api/todos.js';
import './body.html';

Template.body.helpers({
leiaTodod() {
    return Todos.find({}, { sort: { createdAt: -1 }});
  }
});
```

Koodinäide 3. Ülesannete leidmine andmebaasist

Body.html-is tehakse tavaline lehekülje struktuur body *tag*-iga ning nüüd saab kasutada body.js failis kirjutatud "leiaTodod" funktsiooni ning leitakse kõik väärtused *each* tsükliga ja listina väljastatakse ülesanded. Mallid tehakse peale body *tag*-i ning {{ todo }} väljastab spetsiifilised dokumendid massiivist (vt Koodinäide 4).

Maillid kirjutatakse <template> tag-ide vahele ning neid saab veebilehel näidata kirjutades looksulgude vahele ja > märgi taha (vt Koodinäide 5).

```
<body>
  <div class="container">
    <header>
      <h1>Ülesannete loetelu</h1>
    </header>

    <ul>
      {{#each leiaTodod}}
        {{> todoList}}
      {{/each}}
    </ul>
  </div>
</body>

<template name="todoList">
  <li>
    {{ todo }}
  </li>
</template>
```

Koodinäide 4. Ülesannete väljastamine veebilehel

```
{{> todoList }}
```

Koodinäide 5. Mallide kasutamine

Kuna *imports* kaustast ei impordita faile automaatselt, tuleb body.js importida ka *client* kaustas olevasse main.js faili (vt Koodinäide 6).

```
import '../imports/ui/body.js';
```

Koodinäide 6. Body.js importimine

Andmebaas on seotud serveripoolega, seetõttu tuleb importida imports/api kaustast todos.js (vt Koodinäide 7).

```
import '../imports/api/todos.js';
```

Koodinäide 7. Andmebaasi kollektsiooni importimine

Pärast neid tegevusi on veebilehel näha vaid pealkirja. Ülesandeid hetkel ei näidata, sest neid pole veel lisatud (vt Joonis 13). Kui brauser peaks näitama konsoolis veateadet, tuleb serverile

teha taaskäivitus minnes käsuviibale ning peatada server vajutas Ctrl + C klahvikombinatsiooni ja kirjutada uuesti käsklus

```
meteor
```

Joonis 13. Lehe pealkiri

3.2.2 Lisamine

Uute ülesannete lisamiseks tuleb lisada body.html faili vorm, kuhu saab kirjutada ülesande nime ja detailid (vt Koodinäide 8). Tekstikastidele on lisatud *id*-d, et body.js failis oleks võimalik nende väärtused kätte saada.

```
<h2>Lisa ülesanne</h2>
  <form id="todoVorm">
    <input type="text" placeholder="Ülesanne" id="todo" />
    <input type="text" placeholder="Detailid" id="lisaInfo" />

    <input type="submit" value="Lisa" id="lisaTodo" />
  </form>
```

Koodinäide 8. Ülesande lisamise vorm

todoList malli peab juurde lisama võimaluse näha detaile ülesannete kohta ning nende lisamise aeg (vt Koodinäide 9).

```
<template name="todoList">
     <li>
          <span>{{ todo }}</span>
          <span>{{ notes }}</span>
          <span>{{ createdAt }}</span>
     </li>
</template>
```

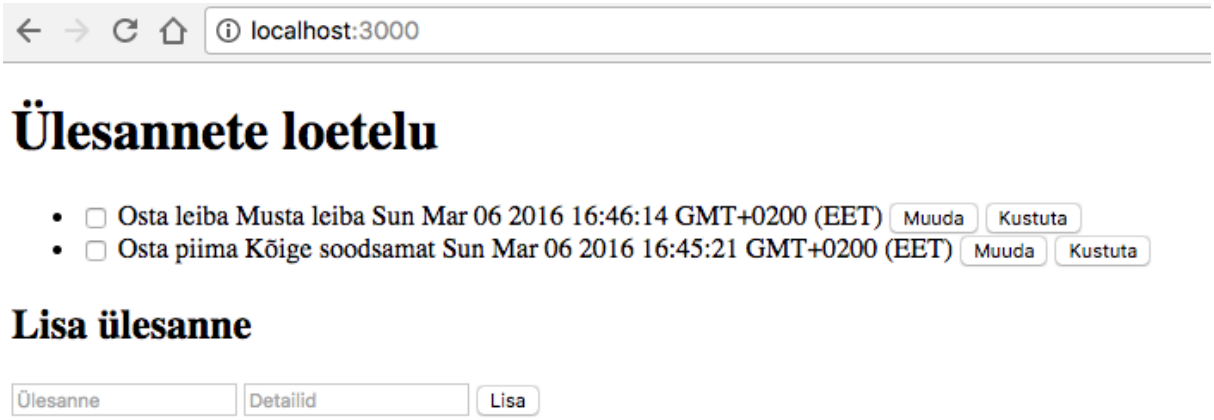
Koodinäide 9. Ülesannete detailide ja loomise aja väljastamine

Body.js failis lisatakse sündmus, mis tunneb ära, kui on vajutatud "lisaTodo" nupu peale. Selle funktsiooniga on kaasas "*event*" parameeter, millega saab ära hoida brauseri vaikimisi sündmust (submit-i puhul lehevärskendamist). Saadakse kätte sisestatud andmed ning kontrollitakse ega need tühjad ei ole ja andmed sisestatakse andmebaasi koos ülesande koostamise ajaga. Pärast seda vorm lehel tühjendatakse (vt Koodinäide 10).

```
Template.body.events({

  'click #lisaTodo': function(event) {
    event.preventDefault();

    var todoVorm = document.getElementById("todoVorm");
    var todo = document.getElementById("todo").value;
    var lisaInfo = document.getElementById("lisaInfo").value;

    if(todo !== '' && lisaInfo !== '') {
      Todos.insert({ todo: todo, notes: lisaInfo, done: false,
createdAt: new Date() });
    }

    todoVorm.reset();
  },
});
```

Koodinäide 10. Ülesannete lisamine andmebaasi

Veebilehel on nüüd võimalik lisada andmebaasi ülesandeid. Ülesanded ilmuvad reaalajas, ei pea tegema lehevärskendusi (vt Joonis 14) .



Joonis 14. Ülesannete loetelu

3.2.3 Muutmine

Alates HTML5-st on võimalik muuta elementide väärtuseid neile vaid peale klikkides. Muuta tuleb malli "todoList", kus olemasolevatele elementidele, mida tahetakse muuta tuleb juurde lisada atribuut `contenteditable = "true"`.

Contenteditable atribuut täpsustab kas elemendi sisu on muudetav või mitte (W3Schools).

Samuti tuleb juurde lisada *class* atribuut, millega on võimalik need väärtused kätte saada ning "Muuda" nupp, mis teostab andmebaasis muudatused. Atribuutides lisatakse juurde ka *id* väärtus, mis sisaldab selle ülesande *id* numbrit (vt Koodinäide 11).

```
<template name="todoList">
  <li>
    <span contenteditable="true" id="{{idVaartus}}"
class="uusTodo">{{ todo }}</span>
    <span contenteditable="true" class="uusLisaInfo">{{ notes
}}</span>
    <span class="lisamiseAeg">{{ createdAt }}</span>
    <button type="button" id="muudaTodo">Muuda</button>
  </li>
</template>
```

Koodinäide 11. Ülesannete muutmise võimalus

Body.js failis peab alguses tegema uue abilise, sest ei kasutata enam "body" mallinime. Uus abiline saab olema "body" abilise *child* abiline, sest "todoList" mall on paigutatud "body" malli

sisse. Sinna abilisse tuleb kirjutada funktsioon väärtuse leidmiseks andmebaasist, mis antakse kaasa HTML failis *id* atribuudis (vt Koodinäide 12).

```
Template.todoList.helpers({
  idVaartus() {
    return this._id;
  }
});
```

Koodinäide 12. Abiline leidmaks *id* väärtuse andmebaasist

Nii nagu lisamise võimaluses tuleb sarnaselt teha ka muutmises. Järgnev kood tuleb kirjutada lisamise järele peale "," (koma) (vt Koodinäide 13).

```
'click #muudaTodo': function() {
    let uusTodo = document.getElementsByClassName("uusTodo");
    let uusLisaInfo =
document.getElementsByClassName("uusLisaInfo");

    for(let i = 0; i < uusTodo.length; i++) {
      if(uusTodo[i].id === this._id) {
        Todos.update({ _id: this._id }, { $set: { todo:
uusTodo[i].innerText, notes: uusLisaInfo[i].innerText }});
      }
    }
  },
```

Koodinäide 13. Ülesande muutmise funktsioon

Veebilehel on võimalik ülesande ja detailide peale vajuta ning neid muuta. Pärast muutmist vajutada "Muuda" nuppu ning need muudatused teostatakse andmebaasis (vt Joonis 15).

Joonis 15. Muudetud ülesanded ja detailid

3.2.4 Kustutamine

Ülesannete kustutamiseks tuleb lisada "Kustuta" nupp juurde body.html faili "Muuda" nupu alla (vt Koodinäide 14).

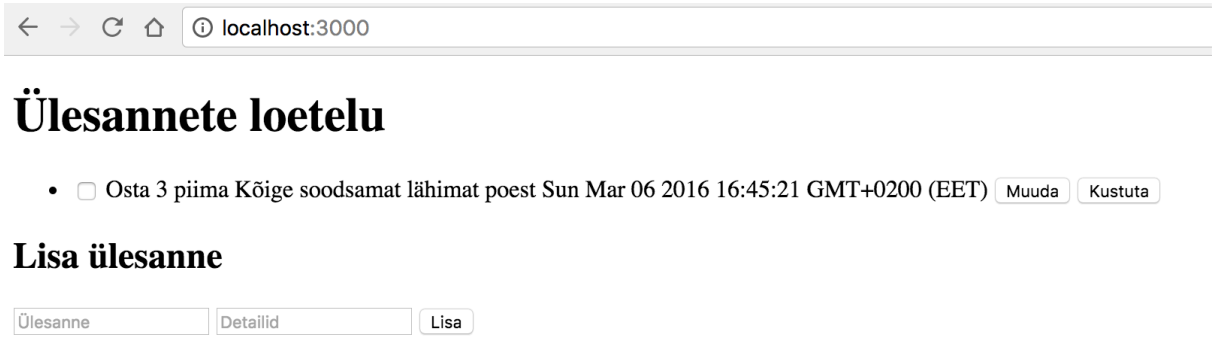
```
<button type="button" id="kustutaTodo">Kustuta</button>
```

Koodinäide 14. Kustuta nupp

Body.js failis peab lisama kustutamise funktsiooni muutmise koodi alla pärast "," (koma). (vt Koodinäide 15). Seotud *id*-ga kustutatakse ülesanne andmebaasist (vt Joonis 16).

```
'click #kustutaTodo': function(event) {
  event.preventDefault();
  Todos.remove({ _id: this._id });
},
```

Koodinäide 15. Kustuta funktsioon



Joonis 16. Kustutatud üks ülesanne

3.2.5 Maha kriipsutamine

Ülesande valmis märkimiseks lisatakse juurde märkeruut body.html faili. *Checked* atribuuti lisatakse abilise väärtus, mis näitab, kas ülesanne on tehtud või mitte. Märkeruut lisatakse ülesande ette (vt Koodinäide 16).

```
<input type="checkbox" checked={{onTehtud}}
class="todoStaatus" />
```

Koodinäide 16. Märkeruut

Body.js failis peab lisama uue abilise funktsiooni todoList malli juurde ning see väljastab andmebaasist ülesande staatuse (tehtud või mitte) (vt Koodinäide 17).

```
onTehtud() {
  return this.done;
}
```

Koodinäide 17. Ülesande staatuse väljastamine

Andmeid andmebaasis uuendatakse kui ülesanne on tehtud, lisatakse juurde "tehtud" klass. Kui ülesanne muudetakse mittetehtuks, eemaldatakse "tehtud" klass (vt Koodinäide 18).

```
'click .todoStaatus': function() {
  Todos.update({ _id: this._id }, { $set: { done: event.target.checked }});

  var uusTodo = document.getElementsByClassName("uusTodo");
  var uusLisaInfo = document.getElementsByClassName("uusLisaInfo");
  var lisamiseAeg = document.getElementsByClassName("lisamiseAeg");

  for(var i = 0; i < uusTodo.length; i++) {
    if(uusTodo[i].id === this._id && event.target.checked === true) {
      uusTodo[i].classList.add("tehtud");
      uusLisaInfo[i].classList.add("tehtud");
      lisamiseAeg[i].classList.add("tehtud");
    }
    if(uusTodo[i].id === this._id && event.target.checked === false) {
      uusTodo[i].classList.remove("tehtud");
      uusLisaInfo[i].classList.remove("tehtud");
      lisamiseAeg[i].classList.remove("tehtud");
    }
  }
}
```

Koodinäide 18. "Tehtud" klassi lisamine ja eemaldamine

Ülesande tehtud märkimisel tõmmatakse sellele ülesandele kriips peale ning selleks on vaja teha uus fail kausta imports/ui nimega body.css ja sinna lisada stiil klassile “tehtud” (vt Koodinäide 19).

```
.tehtud {
  text-decoration: line-through;
}
```

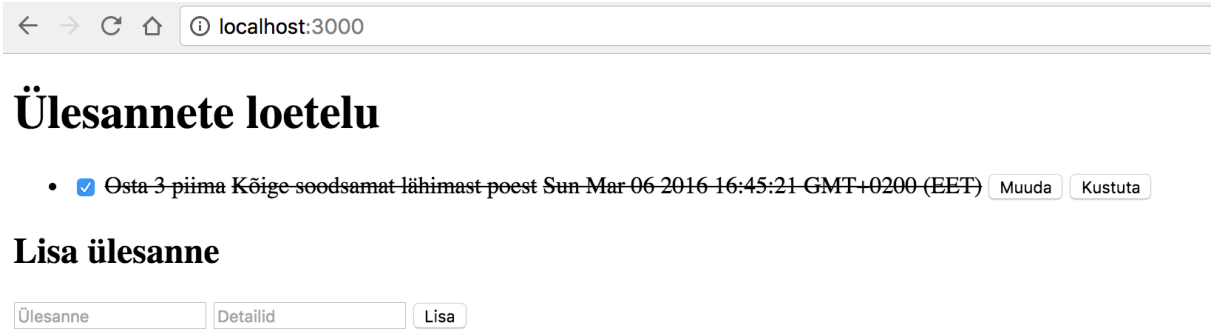
Koodinäide 19. "Tehtud" klassi stiil

See fail tuleb importida body.js faili samamoodi nagu HTML fail (vt Koodinäide 20).

```
import './body.css';
```

Koodinäide 20. CSS faili importimine

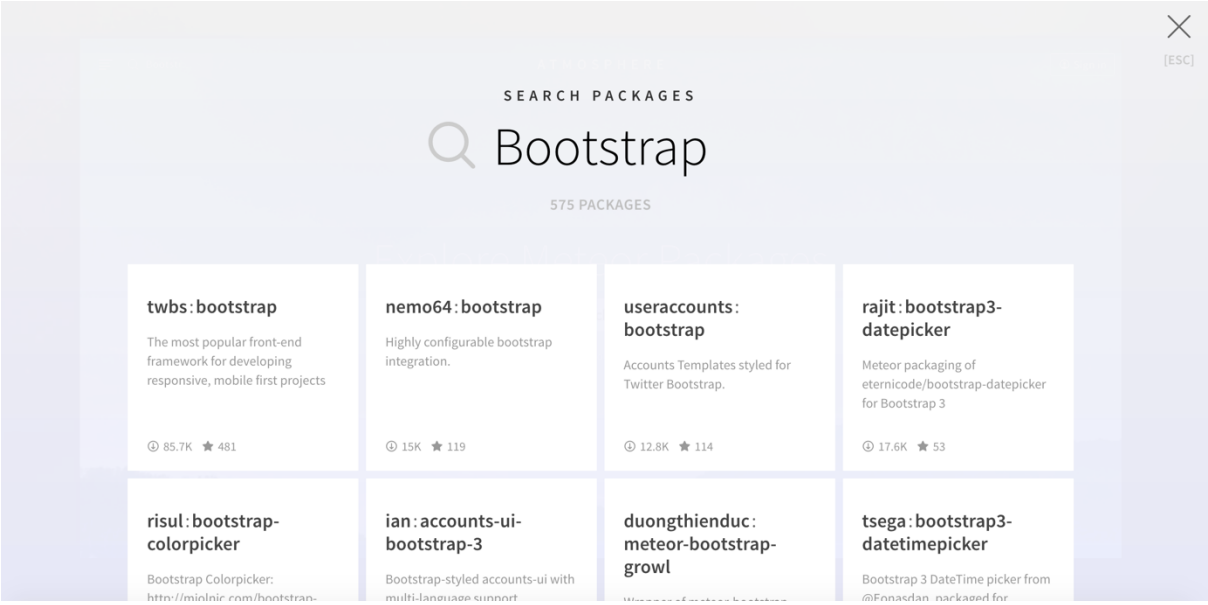
Veebilehel on näha, et valmis märkimisel tõmmatakse ülesandele kriips peale (vt Joonis 17).



Joonis 17. Tehtud ülesanne

3.3 Bootstrap disaini lisamine

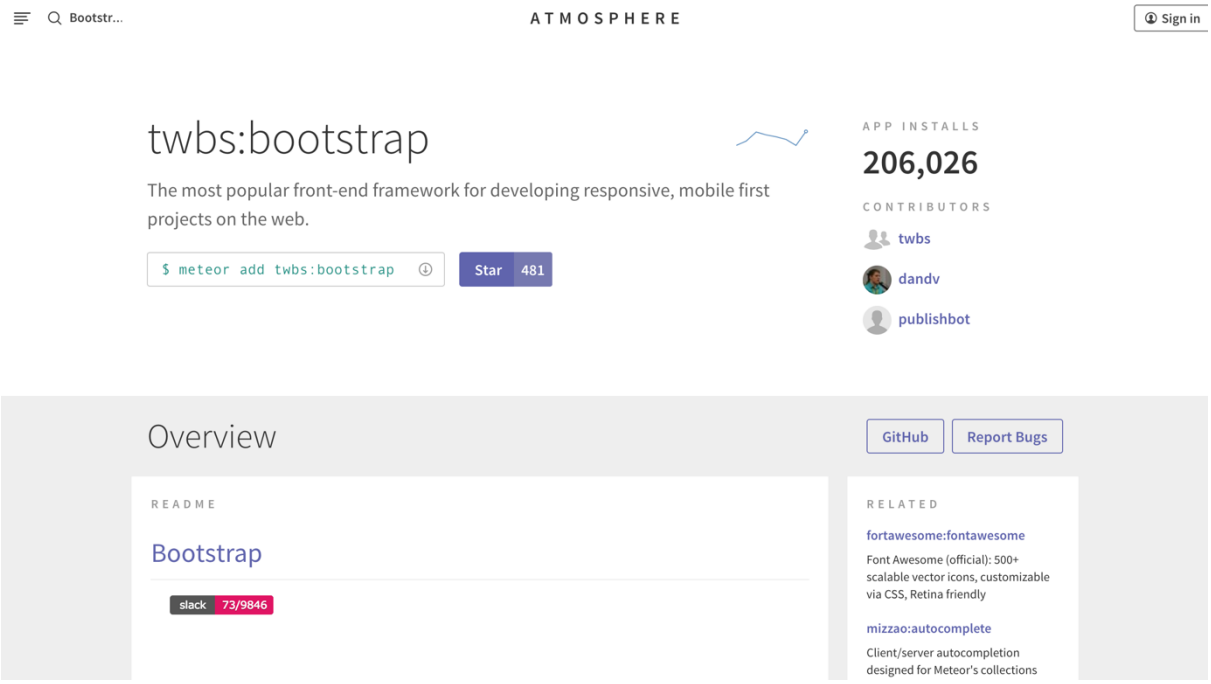
Meteor.js-i projekti on võimalik lisada erinevaid pakette. Käesolevas õppematerjalis kasutatakse Meteor raamistikule mõeldud Bootstrapi paketti. Pakette saab lisada käsurealt. Bootstrapi lisamiseks tuleb minna Atmosphere lehele[1] ja kirjutada otsingusse Bootstrap (vt Joonis 18).



Joonis 18. Bootstrapi raamistiku otsing

Otsingulehel tuleb valikuid mitmeid, kuid käesolevas õppematerjalis valitakse esimene, sest see on ametlik Bootstrapi pakett.

Lähemalt vaadates on näha paketinimi, käsurea käsklus installeerimiseks, mitu korda on installeeritud, ülevaade ja palju muud (vt Joonis 19).



Joonis 19. Paketileht

Kopeerida tuleb käsurea käsklus installeerimiseks

```
meteor add twbs:bootstrap
```

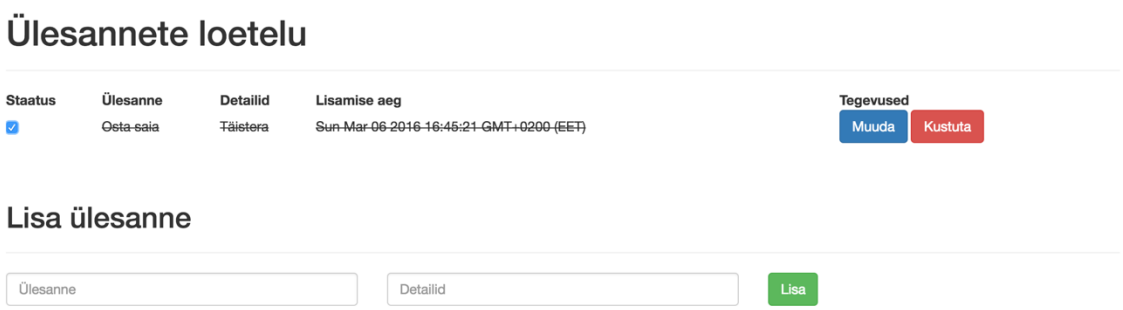
Seejärel lisatakse projekti juurde Bootstrapi pakett (vt Joonis 20).

```
Changes to your project's package version selections:

twbs:bootstrap  added, version 3.3.6


twbs:bootstrap: The most popular front-end framework for developing responsive,
mobile first projects on the web.
```

Joonis 20. Bootstrap paketi installeerimine

Kui Bootstrapi pakett on lisatud, on võimalik muuta body.html sisu ning kasutada Bootstrap disaini võimalusi (vt Lisa 1).

Kui Bootstrapi komponendid on lisatud, on lehekülg dünaamiline ning näeb hea välja arvutist vaadates (vt Joonis 21).



Joonis 21. Bootstrapi disain

4 Tagasiside

Õppematerjali testisid kaks õpilast, kes polnud varem Meteor.js raamistikuga kokku puutunud.

Õppematerjali testijatele meeldis, et pole palju koodi vaja kirjutada, et saavutada reaalajaline rakendus. Samuti meeldis, et pakette saab lisada vaid ühe käsuviiba käsklusega ning ei pea selleks kirjutama koodi failidesse, selline lahendus hoiab koodiread lühemad.

Probleem tekkis, kui jõuti "Projekti ettevalmistamine" peatüki lõppu ning üritati rakendust käima saada, kuid tuli veateade, et ei leita ühte moodulit. Selle sai lahendatud, kui tehti serverile taaskäivitus. Samuti tekkis küsimus *template* kohta, et miks "body" jaoks ei tehtud eraldi malli, kuid "todoList" jaoks tehti.

Pärast tagasiside saamist, lisas seminaritöö autor juurde õpetuse kuidas teha serverile taaskäivitus, kui see veateade peaks tekkima. Samuti lisati selgitus "body" ja "todoList" mallide kohta.

Kokkuvõte

Käesoleva seminaritöö eesmärgiks oli teha eestikeelne õppematerjal Meteor.js raamistiku kohta. Õppematerjali on lisatud koodinäited ning lehekülje näited, milles on näha teostatud muudatusi.

Töö käigus õpetati kasutama installeerima Meteor.js-i arvutisse, looma ning käivitama projekti, tegema esialgseid muudatusi kausta struktuuris, näiteprogrammi lisama funktsionaalsust ning disainima tehtud näiteprogrammi Bootstrap raamistikuga.

Edasiarenduseks võiks õpetada, kuidas navigeerida teistele lehtedele ilma lehe värskenduseta, mille jaoks tuleks lisada navigeerimise pakett, koostada vajalikud leheküljed ning defineerida need leheküljed navigeerimise paketis.

Kasutatud kirjandus

Blaze. (kuupäev puudub). *Spacebars templates*, allikas http://blazejs.org/guide/spacebars.html (07.03.2016)

Greif, S. (3. juuli 2014. a.). *Meteor & Security: Setting the Record Straight*, allikas https://www.discovermeteor.com/blog/meteor-and-security/ (07.03.2016)

Hochhaus, S., & Schoebel, M. C. (2015). *Meteor in Action.* New York: Manning Publications.

Meteor. (kuupäev puudub). *API Docs*, allikas https://docs.meteor.com/ (07.03.2016)

Meteor. (kuupäev puudub). *Meteor Showcase*, allikas https://www.meteor.com/showcase (07.03.2016)

Otto, M., & Thornton, J. (kuupäev puudub). *Bootstrap dokumentatsioon*, allikas http://getbootstrap.com/ (07.03.2016)

Strack, I. (2012). *Getting Started with Meteor.js JavaScript Framework.* Birmingham: Packt Publishing.

W3Schools. (kuupäev puudub). *HTML contenteditable Attribute*, allikas http://www.w3schools.com/tags/att_global_contenteditable.asp (07.03.2016)

LISAD

Lisa 1. Bootstrapi koodinäide

```
<head>
  <meta name="viewport" content="width=device-width, initial-
scale=1">
</head>
<body>
  <div class="container">
    <div class="page-header">
      <h1>Ülesannete loetelu</h1>
    </div>
    <div class="table-responsive">
      <table class="table">
        <tr>
          <th>Staatus</th>
          <th>Ülesanne</th>
          <th>Detailid</th>
          <th>Lisamise aeg</th>
          <th>Tegevused</th>
        </tr>
        {{#each leiaTodod}}
          {{> todoList}}
        {{/each}}
      </table>
    </div>

    <div class="page-header">
      <h2>Lisa ülesanne</h2>
    </div>
    <form id="todoVorm" class="form-horizontal">
      <div class="form-group">
        <div class="col-md-4">
          <input type="text" placeholder="Ülesanne" id="todo"
class="form-control" />
        </div>
        <div class="col-md-4">
          <input type="text" placeholder="Detailid" id="lisaInfo"
class="form-control" />
```

```
        </div>

        <div class="col-md-4">
          <input type="submit" value="Lisa" id="lisaTodo"
class="btn btn-success" />
        </div>
      </div>
    </form>

  </div>
</body>

<template name="todoList">
    <tr>
      <td>
        <input type="checkbox" checked={{onTehtud}}
class="todoStaatus" />
      </td>
      <td>
        <span contenteditable="true" id="{{idVaartus}}"
class="uusTodo">{{ todo }}</span>
      </td>
      <td>
        <span contenteditable="true" class="uusLisaInfo">{{ notes
}}</span>
      </td>
      <td>
        <span class="lisamiseAeg">{{ createdAt }}</span>
      </td>
      <td>
        <button type="button" class="btn btn-primary"
id="muudaTodo">Muuda</button>
        <button type="button" class="btn btn-danger"
id="kustutaTodo">Kustuta</button>
      </td>
    </tr>
</template>
```

Lisa 1. Bootstrapi koodinäide