TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut

Madis Peetersoo 123863IAPB

SPORTLIO - DJANGO RAAMISTIKUL SOTSIAALVÕRK SPORTLASTELE

Bakalaureusetöö

Jaagup Irve
Tehnikateaduste magister
Tarkvarainsener

Tallinn 2015

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: Madis Peetersoo

25.05.2015

Annotatsioon

Töö eesmärgiks oli luua veebirakendus, millega sportlastel oleks võimalik enda tulemuste ülevaatlikku kontrolli omada ning fänne enda saavutuste ja tegemistega kursis hoida, luues selleks iga sportlase jaoks personaalse lehekülje, mida saaks kasutada ka enda tutvustamiseks potentsiaalsetele sponsoritele.

Veebirakendus on loodud professionaalse IT-ettevõtte Thorgate [24] poolt, kus ka töö autor töötab tarkvara arendajana. Antud töös välja toodud funktsionaalsustest on töö autori poolt realiseeritud sportlase ajajoone vaade ning sportlase profiili loomine – st projekti tuumikosad.

Töö tulemusena valmis veebiraamistiku Django [25] peal loodud võrgurakendus Sportlio, kus 20.05.2015 seisuga on 663 kasutajat ja 439 sportlase profiili.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, kuute peatükki, seitset joonist, 11 koodijuppi ja ühte tabelit.

Abstract

Sportlio - social network for athletes on Django framework

The purpose of this thesis was to solve a fundamental problem in the daily lives of athletes – they do not have an easy and efficient solution to keep track of their competition results. A popular tool for that was Microsoft Excel.

The project's idea originates from an Estonian athlete Mikk Meerents who turned with his idea to a professional IT team Thorgate [24], where the author of this thesis works as a software developer. The result of this collaboration was a working web application– Sportlio.

The author of this thesis is responsible for developing an athlete's profile and creating a personalized athlete page – the core functionalities of Sportlio.

The application is developed on a Python framework Django [25]. It uses many of the built in features of Django – models, class based views *etc*, as well as third party applications (such as South migrations). In the development of front-end, jQuery [16] and SCSS [15] are widely used.

Sportlio has two main user groups – athletes and regular users. Athletes can create their own profiles, add their competition results and make posts based on their trainings and/or their daily life or whatever subject they prefer. Everything is drawn together to a complete and personalized Sportlio page, for regular users to browse. Regular users can also become fans of their favorite athletes, which guarantees a daily feed of their favorites athletes' posts right to their e-mail account.

The thesis is written in Estonian, it contains 33 pages, six chapters, seven figures, 11 code snippets and one table.

Lühendite ja mõistete sõnastik

MVT	*Model-View-Template*, mudel-vaade-mall.
URL	*Uniform Resource Locator*, internetiaadress.
DRY	*Don't repeat yourself*, tarkvaraarenduse põhimõte, mille eesmärgiks on vältida igasugust informatsiooni kordamist [7].
Ajax	*Asynchronous Javascript and XML*, veebiprogrammeerimise tehnika kliendi poolelt serverisse asünkroonsete päringute tegemiseks [20].
DOM	*Document Object Model*, liides *(interface)*, mis lubab programmidel ja koodijuppidel (*script)* dünaamiliselt dokumendi sisule, struktuurile ja stiilidele ligi pääseda ning neid muuta [17].
API	*Application Programming Interface*, arvuti operatsioonisüsteemiga või rakendusprogrammiga määratud reeglistik, mille alusel rakendusprogramm kasutab operatsioonisüsteemi või teise rakendusprogrammi teenuseid [20].
HTTP	*Hypertext Transfer Protocol*, hüperteksti edasusprotokoll – klient-server protokoll HTML dokumentide vahetamiseks [20].
HTTPS	*HTTP Secure*, hüperteksti edastusprotokoll üle turvasoklite kihi [20].
SSL	*Secure Sockets layer* - turvasoklite kiht, informatsiooni krüpteerimissüsteem [20].

WSGI	*Web Server Gateway Interface* - veebiserverite ja võrgurakenduste või –raamistike vaheline liides Pythonis [26].
UNIX	Operatsioonisüsteem [20].
CSS	*Cascading Style Sheets* - kesklaadistik, mis kirjeldab, kuidas HTML dokumente esitada [20].
Sass	*Syntatically Awesome Style Sheets*, laadilehtede keel.
SCSS	*Sassy CSS*, laadilehtede keel, Sassi uuem süntaksiline vorm [15].
Modal	Aken, mis avaneb enda vanema ette [27].
Webhook	HTTP päring, mis teostatakse, kui mingi kindel sündmus toimub – lihtne sündmusest teadaandmise viis üle HTTP protokolli [28].
Mixin	Klass, kuhu koondatakse mingi kindel funktsionaalsus ning hiljem kasutatakse teiste klasside loomiseks [29].

Sisukord

Jooniste nimekiri

Tabelite nimekiri

Koodide nimekiri

1. Sissejuhatus

Töö eesmärgiks on lahendada professionaalsete sportlaste seas levinud probleem – sportlastel pole lihtsat ja mugavat süsteemi enda võistlustulemuste ülevaatlikuks visualiseerimiseks. Tihtipeale kasutatakse praegu lihtsaid Microsoft Exceli tabeleid, mida vajadusel ka potentsiaalsetele sponsoritele näidatakse.

Valminud veebirakendus Sportlio on loodud professionaalse IT-firma Thorgate [24] poolt, kus ka töö autor tarkvaraarendajana töötab. Antud dokumendis välja toodud funktsionaalsustest on töö autori poolt realiseeritud sportlase ajajoone vaade ning sportlase profiili loomine – st projekti tuumikosad.

Sportlio mõte pärineb kergejõustiklaselt Mikk Meerentsilt, kes pöördus enda ideega Thorgate'i poole. Koostöö tulemusena arendati lihtsast mõttest reaalselt töötav ja sportlaste maailma parandav ja efektiivseks muutev veebirakendus.

Töö esimese osas antakse täpsem ülevaade töö ärilisest ja tehnilisest taustast ning töö teises pooles demonstreeritakse rakenduse kirjelduses valminud veebirakenduse tähtsamaid funktsionaalsuseid.

2. Äriline taust

2.1. Turuanalüüs

Hetkel ei eksisteeri sellist veebikeskkonda (peale Sportlio), kuhu sportlane saaks ise enda sporditulemusi sisestada. On rahvusvahelised veebilehed (näiteks www.iaaf.org), kuhu kuvatakse automaatselt informatsiooni võistlustulemuste arhiivi päringute kaudu. Tihiti on nende probleemiks aga aegunud/ebakorrektsed andmed. Samuti ei saa sellistest keskkondadest ülevaadet kesk- ja madalama klassi sportlastest, kuna informatsiooni kogutakse vaid tippvõistluste kohta.

Alternatiivina leidub veel selliseid veebilehti, kes pakuvad sportlastele võimaluse luua enda profiil (näiteks www.makeachamp.com), kuid millel puudub funktsionaalsus enda tulemuste kohta arve pidamiseks. Antud valdkonda kuulub ka sportlaste seas suhteliselt populaarne variant – luua endale Facebookis fännileht. Seetõttu on tulemuste hoidmiseks parim lahendus lihtne Exceli tabel. [22]

2.2. Äriplaani kokkuvõte

Sportlio juures on tulu teenimise saavutamine kõige keerulisem. Algne mõte ühisrahastusplatvormi ei olnud asjakohane, kuna sportlaste finantseering peaks pärinema ettevõtetelt, mitte üksikisikutelt. Kõige efektiivsem ja kiirem võimalus ettevõtete käest raha genereerida on pakkuda neile reklaamiteenust.

Sportlio pakub võimalust osta kasutajate profiilidesse reklaami, mille tuludest üks osa kuulub sportlasele ning teine osa Sportliole. Reklaamiteenust ostnud ettevõte saab näidata ennast sportlase profiilis tema toetajana, aga kiiremini ja väiksema summa eest kui sõlmides sportlasega eraldi sponsorluslepinguid. 2015. aasta märtsikuu kümne enim külastatud sportlaste profiili on juba reklaam ostetud [9]. [22]

2.3. Edasised arenguplaanid

Antud hetkel on kõige olulisemateks tegevusteks reklaami laialdane müümine ning andmebaasi suurendamine – nii kasutajate arvu kui ka toetatud spordialade poolest.

Tulevikus omab Sportlio arhiivi tegevuse lõpetanud sportlaste kohta. Funktsionaalsuse osas ei ole hetkel suuremahulist tootearendust ette nähtud, kuid seevastu on plaanis teostada väiksemaid parandustöid – näiteks lisada eraldi võimalus, mille kaudu saaks meedia alla laadida sportlase ajakohase foto. [8]

3. Tehniline taust

3.1. Kasutatud tehnoloogiad

3.1.1. Python / Django

Django on avatud lähtekoodiga programmeerimiskeeles Python kirjutatud veebiraamistik, mis järgib mudel-vaade-mall (MVT) arhitektuurimustrit [1][2].

Django peamiseks eesmärgiks on lihtsustada keerukatel andmebaasidel baseeruvate veebirakenduste loomist ning rõhub koodi taaskasutatavusele, kiirele arendusprotsessile ja põhimõttele „ära korda ennast“ (DRY).

Django raamistiku peale loodud enimtuntud veebirakenduste hulka kuuluvad Pinterest [3], Instagram[4] ja Mozilla[5].

3.1.1.1. Mudelid

Mudelid on MVT disainimustri „M“ osa. Django mudel on andmebaasis hoitavate andmete kirjeldus, mis üldisemalt tähendab, et iga mudel vastab andmebaasis ühele tabelile.

- Iga mudel on `django.db.models.Model` klassi Pythoni alamklass;
- Mudeli iga atribuut vastab andmebaasis ühele väljale;
- Tänu mudelitele on Djangos võimalik kasutada automaatselt genereeritud andmebaasiga suhtlemise APIt.

Koodis 1 olevast Django mudelist `Athlete` genereeritakse ja käivitatakse koodis 2 välja toodud SQL lause.

```
from django.db import models
class Athlete(models.Model):
    name = models.CharField(max_length=100)
    birthday = models.DateField()
```

Kood 1. Django mudeli näide

```
CREATE TABLE „athlete_athlete“ (
    „id“ integer NOT NULL PRIMARY KEY,
    „name“ varchar(100) NOT NULL,
    „birthday“ date NOT NULL
)
```

Kood 2. Django mudelist genereeritud SQL lause

Märkused:

- Tabeli nimi „athlete_athlete“ genereeritakse mudeli andmetest, kuid vajadusel on võimalik seda muuta. Antud juhul võetakse viimane „athlete“ mudeli nimest ja esimene „athlete“ rakenduse nimest ning need ühendatakse alakriipsuga.
- Automaatselt lisatakse väli „id“, kuid ka seda on võimalik muuta.
- Antud näites on SQL kohandatud PostgreSQLi jaoks, kuid on märkimisväärt, et Django kohandab genereeritud SQLi projekti seadistustes määratud andmebaasisüsteemi jaoks.

3.1.1.2. Klassipõhised vaated

Vaade (*view*) on väljakutsutav koodijupp (*callable*), mis võtab sisendiks päringu (*request*) ja tagastab vastuse (*response*). Vaade võib olla ka keerukam kui funktsioon ning Djangol on sisse ehitatud mõned klassid, mida saab kasutada vaadetena. See võimaldab lihtsalt struktureerida vaateid ning taaskasutada kirjutatud koodi tänu pärimistele (*inheritance*) ning *mixin*’idele. [7]

Koodis 3 on seadistatud lihtne URLide seadistus, mis aadressile ’localhost:8000/athletes/’ tehtud päringule määrab vastavusse vaate `AthleteListView`.

```
# athletes/urls.py
from django.conf.urls import url
from athletes.views import AthleteListView

urlpatterns = [
    url(r’^athletes/$’, AthleteListView.as_view(), name=’athlete-list’),
]
```

Kood 3. Django URLide seadistus

Koodis 4 kasutakse `django.views.generic.ListView`’d mis on mõeldud mingi mudeli objektide loetelu esitamiseks. Selles on määratud, et Django kasutaks mudelit `Athlete`,

filtreeritakse välja ainult need sportlased, kelle `is_featured` atribuut on määratud tõeseks ning saadetakse need loendi tüüpi `athletes` muutujas HTML malli `list.html`, mis asub seadistustes määratud mallide kausta `athletes` alamkaustas.

```
# athletes/views.py
from django.views.generic import ListView
from athletes.models import Athlete

class AthleteListView(ListView):
    model = Athlete
    template_name = „atheletes/list.html“
    context_object_name = „athletes“
    queryset = Athlete.objects.filter(is_featured=True)
```

Kood 4. Klassipõhine vaade

Kuna `AthleteListView` väärtustas muutuja `athletes` meid huvitavate sportlastega, siis koodis 5 olevas HTML mallis saabki neid kuvada, itereerides üle muutuja `athletes`.

```
{# athletes/list.html #}
{% extends ’base.html’ %}
{% block content %}
    <ul>
        {% for athlete in athletes %}
            <li>{{ athlete }} </li>
        {% endfor %}
    </ul>
```

Kood 5. HTML mall

3.1.2. South migratsioonid

Kuni Django versioonini 1.8 puudus Djangos funktsionaalsus andmebaasi skeemides muudatuste tegemiseks. Kuna Sportlio arendamiseks on kasutusel Django versiooni 1.6.5 (mis oli projekti algushetkel ajakohane), on andmebaasi skeemide muudatuste realiseerimiseks kasutusel South. South on tööriist, mis pakub Django rakendustele terviklikke ja kergelt kasutatavaid andmebaasi migratsioone [10] ning mille eesmärgiks on elimineerida tülikat ja ajakulukat tööd, mida põhjustab Django rakenduse andmebaasi skeemi muutumine.

Olles eelnevalt loonud `Athlete` rakendusse samanimelise mudeli, genereerib South selle põhjal esimese migratsiooni (kood 6), mille alusel oskab luua andmebaasi tabeli.

```
class Migration(SchemaMigration):
   def forwards(self, orm):
       # Adding model 'Athlete'
       db.create_table('athletes_athlete', (
           ('id', self.gf('django.db.models.fields.AutoField')(primary_key\
                          =True)),
           ('name', self.gf('django.db.models.fields.CharField')\
                            (max_length=100)),
           ('birthday', self.gf('django.db.models.fields.DateField')\
           ()),))
       db.send_create_signal('athletes', ['Athlete'])
```

Kood 6. South'i esmase migratsiooni forwards meetod

Koodis 7 on nüüd mudelile `Athlete` lisatud uus väli `country` ning Django tuvastab, et andmebaasis vastavat tabeliveergu ei ole ning kuvab järgneva veateate: `ProgrammingError – column athletes_athlete.country does not exist.`

```
from django.db import models

class Athlete(models.Model):
    name = models.CharField(max_length=100)
    birthday = models.DateField()
    country = models.CharField(max_length=40)
```

Kood 7. Django mudeli Athlete täiendus

Siinkohal ongi vajalik South, mis loob uue migratsiooni (kood 8), mille käivitamisel modifitseeritakse andmebaasi vastavalt vajadusele – antud juhul luuakse `athletes_athlete` tabelisse uus veerg `country`.

```
class Migration(SchemaMigration):
   def forwards(self, orm):
       # Adding field 'Athlete.country'
       db.add_column('athletes_athlete','country',\
                     self.gf('django.db.models.fields.CharField')\
                             (max_length=40, default='Estonia'),\
                             keep_default=False)
```

Kood 8. South'i migratsiooni, mis lisab andmebaasi tabelisse ühe veeru, forwards meetod

3.1.3. PostgreSQL

PostgreSQL on vabavaraline objekt-relatsiooniline andmebaasisüsteem. Andmebaasi serverina on selle peamine eesmärk turvaliselt hoiustada andmeid ning võimaldada teistel tarkvararakendustel vajadusel andmete poole pöörduda. PostgreSQL suudab hallata nii väikseid, ühe seadme rakendusi, kui ka suuri veebirakendusi, mida kasutavad samaaegselt mitmed kasutajad. [11]

3.1.4. Mercurial

Erinevate programmeerijate poolt tehtud koodimuudatuste jälgimine ning ühtseks ja töötavaks koodiks ühendamine on vähegi suurema projekti õnnestumiseks hädavajalik. Seda aitavad teha erinevad versioonihaldustarkvarad, milledest antud projektis on kasutusel Mercurial.

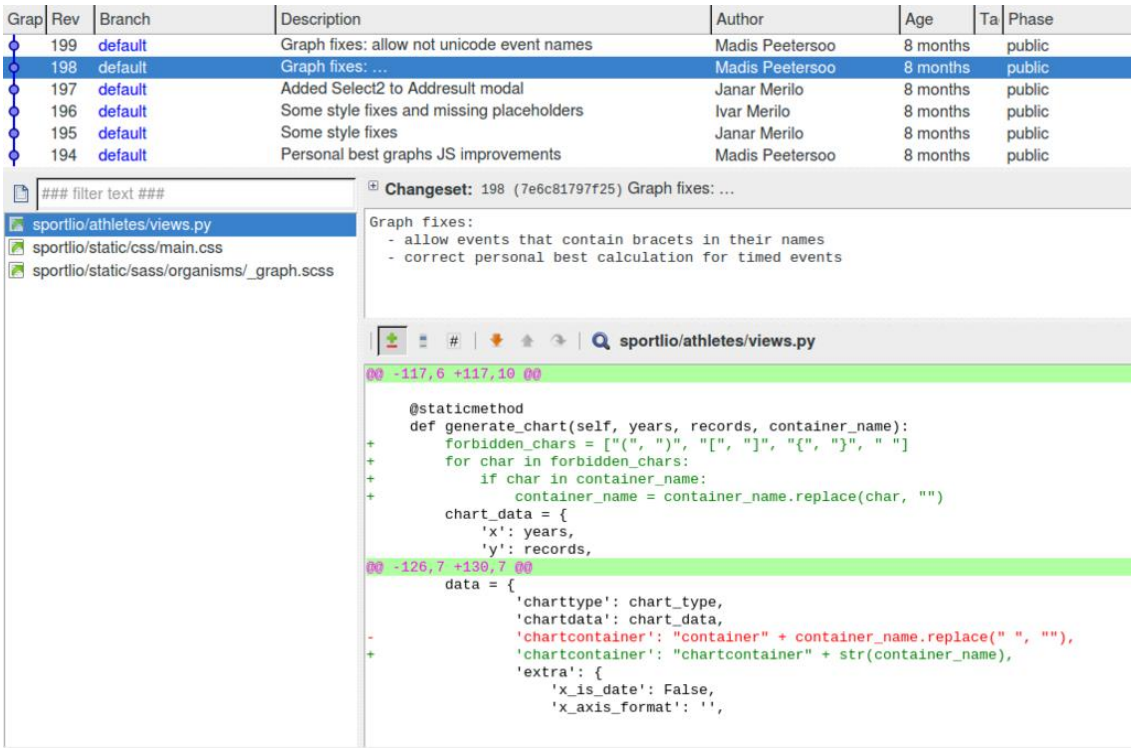
Mercurial on mahult väike, kuid võimekas hajutatud (*distributed)* vabavaraline versioonihaldustarkvara. Teiste hajutatud versioonihaldustarkvarade hulgast sarnaneb Mercurialiga enim Git, kuid nende kahe peamised erinevused on järgnevad:

- Mitmed sisse ehitatud tagasivõtmise käsud – Mercuriali `revert`, `backout` ja `rollback` käsklused võimaldavad mugavalt naasta failide mingisse kindlasse versiooni. Gitil on võrdväärsena vaid `revert` käsk, mille kasutamine pole triviaalne.

- Sisseehitatud veebiserver – Mercuriali lihtsas integreeritud veebiserveris saab hoida (*host)* koodi repositooriumit, mis võimaldab teistel kasutajatel kiirelt ja kergelt koodile ligi pääseda.

- Ajaloo säilitamine `copy`/`move` operatsioonidel – Mercurial säilitab nii `copy` kui ka `move` operatsioonide korral andmete täielikku ajalugu, Git seda ei tee. [23]

Joonisel 1 on näha ühte Mercuriali graafilist kasutajaliidest – TortoiseHG'd.



Joonis 1. Eraanitõmmis Mercuriali kliendi TortoiseHG's Sportlio repositooriumist

3.1.5. Nginx ja Gunicorn

Nginx on vabavaraline pöördproksi server HTTP, HTTPS jpt protokollide jaoks. Nginxi põhieesmärkideks on tagada suur jõudlus, vähene mälukasutus ning väga paljude klienti samaaegne teenindamine (*concurrency)*.

Käesolevas projektis on nginx kasutusel, kuna see katab väga hästi projekti järgnevat kolme peamist vajadust:

- SSL *termination* – sissetulevate SSL ühenduste haldamine, SSLi dekrüpteerimine ja dekrüpteeritud päringu edasi suunamine;
- staatiliste failide serveerimine;
- päringute Gunicorn'ile edasi suunamine [12].

Kõik päringud, mis ei hõlma endast staatiliste failide serveerimist, suunab nginx edasi Gunicornile, mis on Pythoni WSGI HTTP server UNIX'le. Gunicorn töötleb päringut, suunab vastuse nginxile, kes omakorda suunab saadud vastuse tagasi kliendile. [13][14]

3.1.6. Sass ja SCSS

Sass on kujunduskeel (*stylesheet language)*, mis lihtsustab CSSi kirjutamist ning võimaldab programmerijatel rakendada DRY printsiipi. SCSS on omakorda Sassi uuem süntaksiline vorm. Sassi kood kompileeritakse CSSiks, mida HTML oskab rakendada. [15]

Käesolevas projektis on enimkasutatavad Sassi funktsionaalsused CSSi *selector'*ite sügavuti kirjutamine (*nesting)* ning muutujate kasutamine. Tabelis 1 on näha mõlemad need funktsionaalsused ning Sassist kompileeritud CSS.

SCSS kood	SCSS koodist kompileeritud CSS

`$gray-dark: #344152;`
`$title-color: $gray-dark;`
`body.register {`
`  .container.auth {`
`  h1.title {`
`  margin: 60px 0 50px;`
`  color: $title-color;`
`  }`
`  a {`
`  text-decoration: underline;`
`  &:hover {`
`  text-decoration: none;`
`  }`
`  }`
`  }`
`  }`
`}`
	`body.register .container.auth h1.title {`
	`  margin: 60px 0 50px;`
	`  color: #344152;`
	`}`
	`body.register .container.auth a {`
	`  text-decoration: underline;`
	`}`
	`body.register .container.auth a:hover {`
	`  text-decoration: none;`
	`}`

Tabel 1. SCSSi sügavuti kirjutamise näide

Sassis on võimalik ka *mixin'*e luua, mis meenutavad programmeerimiskeeltes kasutatavaid funktsioone – nende eesmärgiks on grupeerida hulga CSSi stiile ning võimaldada neid vajadusel lihtsasti kasutada, minimaliseerides nii koodi kordamist. Koodis 9 on näha *mixin*'i loomine ning selle kasutamine.

```
@import "compass/css3/transform";
@import "compass/css3/transition";

@mixin scale-on-hover($scale, $duration, $style:"ease-in") {
   @include transition-timing-function($style);
   @include transition-duration($duration);
   @include backface-visibility(hidden);
   &:hover {
       @include transform(scale($scale));
   }
}

.timeline__event__gallery__item {
  @include scale-on-hover(1.2, 0.2s);
}
```

Kood 9. SCSS mixin ja selle kasutamine.

3.1.7. jQuery

jQuery on kiire, mahult väike ning paljude funktsionaalsustega JavaScripti teek (*library).* jQuery muudab HTML dokumendi manipuleerimise, sündmuste käsitlemise (*event handling)*, animatsioonide ning Ajax päringute tegemise kergemaks, kasutades selleks APIt [16].

Koodis 10 on näha jQuery elementaarset funktsionaalsust – HTML dokumendi sündmuste kuulamine (*event listening)* ning elementide manipuleerimine.

```
<head>
// Laeme sisse jQuery'i faili
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
        // "ready" sündmus (event) käivitatakse, kui DOM'i laadimine on lõpetatud
        $(document).ready(function() {
                // See funtsioon pannakse käima, kui vajutatakse HTML elemendile,
                // millel on küljes klass 'js-slide-down'. Kuigi antud juhul on
                // selektorina kasutatud klassi, võiks selle asemel kasutada ka
                // keerulisemaid selektoreid
                $('.js-slide-down').click(function() {
                        // Toome jQuery'sse sisseehitatud animatsiooniga soovitud
                        // elemendi nähtavale
                        $('#js-slide-me-down').slideDown();
                });
        });
</script>
</head>
```

Kood 10. Javascript, mis kasutab jQuery't, koodinäide

3.2. Kasutatud teenused

3.2.1. Amazon Web Services

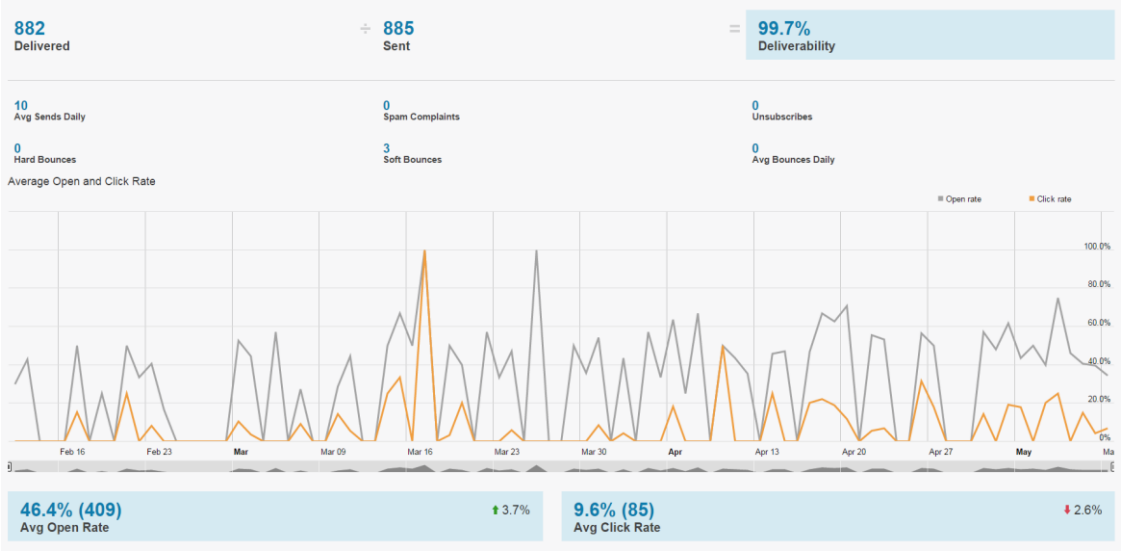
Amazon Web Services pakub palju erinevaid pilvepõhiseid teenuseid – automaatselt skaleeruvaid servereid, andmebaase, analüütikat, failide hoiustamist jpm [18].

Sportlio rakendus töötab Amazoni serveris ning seetõttu on antud projekti raames kasutusel Amazon Elastic Compute Cloud (Amazon EC2), mis võimaldab mugavalt servereid rentida ning vastavalt vajadusele seadistada.

3.2.2. Mandrill

Mandrill on emaili infrastruktuuri teenus, mis on hetkel omasugustest suurim – sellel on üle 375 000 aktiivse kasutaja. Mandrilli eesmärgiks on ühelt-ühele emailide saatmine.

Mandrilli tugevusteks on kiire ülesseadistamine, kõrge emailide kohale jõudmise määr, *webhook'*id, hästi dokumenteeritud API ning analüütiline veebiliides (joonis 2).[19] Mandrill lubab saata kuus kuni 12 000 emaili tasuta, mis on antud projekti jaoks täiesti piisav – emaile saadetakse kasutajate parooli taastamiseks ning fännidele maksimaalselt kord päevas sportlase viimaste seinapostituste kohta. Kolmekuulises ajavahemikus 11.02.2015 – 12.05.2015 on välja saadetud 885 emaili.



Joonis 2. Ekraanitõmmis Mandrilli veebiliidese analüütikast

Koodis 11 on näha Djangosse sisseehitatud `django.core.mail.EmailMessage`'i kasutamist Mandrilliga emaili saatmiseks.

```
from django.conf import settings
from django.core.mail import EmailMessage
from django.template.loader import render_to_string

def send_email(rcpt_email, email_subject, template_name, template_vars,\
               from_email=None):
   template_vars.update({
       'SITE_URL': settings.SITE_URL,
   })
   email_content = render_to_string(template_name, template_vars)
   msg = EmailMessage(email_subject, email_content, from_email, [rcpt_email])
   msg.content_subtype = "html"
   msg.send()
```

Kood 11. Emaili saatmise funktsioon

4. Rakenduse kirjeldus

Rakenduse kasutajagrupid järgi jagada kaheks – sportlased ja tavakasutajad, erinedes suuresti kasutatava funktsionaalsuse poolest.

4.1. Sportlane

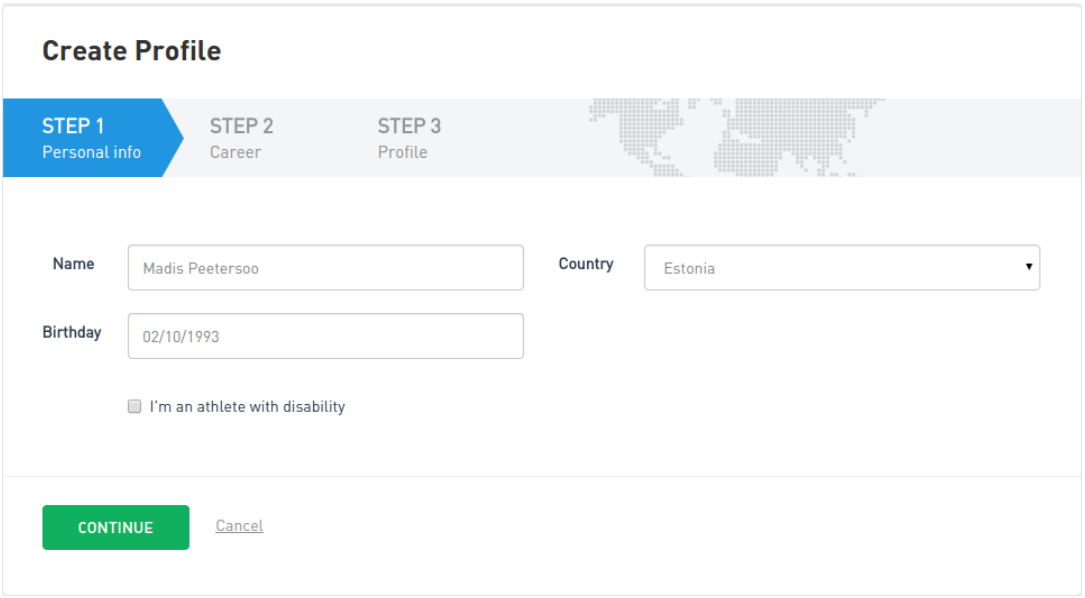
Sportlase Sportlio kasutamise peamiseks eesmärgiks on luua enda kohta terviklik ja ülevaatlik personaalne lehekülg, mida saaks kasutada enda presenteerimiseks fännidele ja potentsiaalsetele sponsoritele.

4.1.1. Profiili loomine

Sportlase profiili loomise eelduseks on Sportlio portaalis konto omamine, mille loomine on kõigile tasuta. Profiili loomine koosneb kolmest eraldiseisvast sammust – st vormis sisestatud andmed salvestatakse andmebaasi iga sammu lõpus, mis võimaldab vajadusel profiili loomise pooleli jätta ning hiljem selle juurde naasta ilma, et eelmistes sammudes sisestatud andmed kustuksid.

Esimeses sammus (joonis 3) tuleb sisestada sportlase järgnev personaalne informatsioon:

- nimi;
- sünnikuupäev, mida kasutatakse sportlase vanuse arvutamiseks;
- riik, mille põhjal kuvatakse sportlase ajajoonel vastava riigi lippu;
- vajadusel puude olemasolu, mis tuuakse olemasolu korral ajajoonel välja.

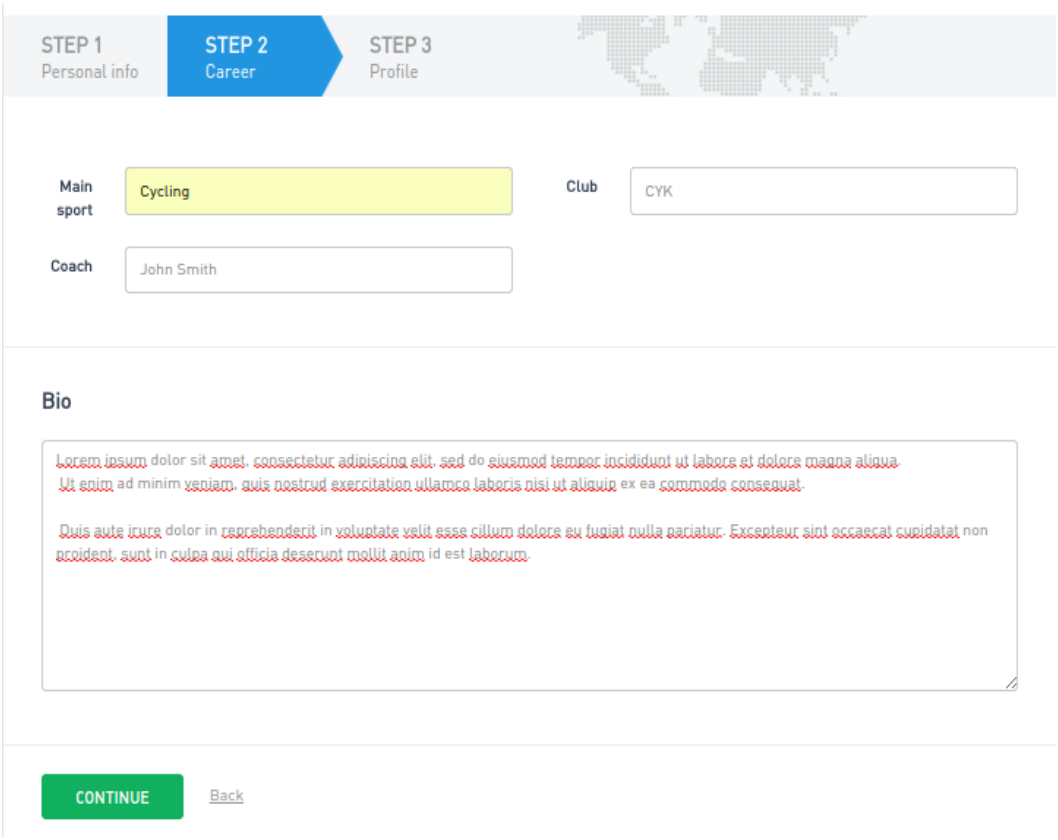


Joonis 3. Ekraanitõmmis profiili loomise esimesest sammust

Teises sammus (joonis 4) tuleb sisestada järgnev sportlaskarjääriga seonduv:

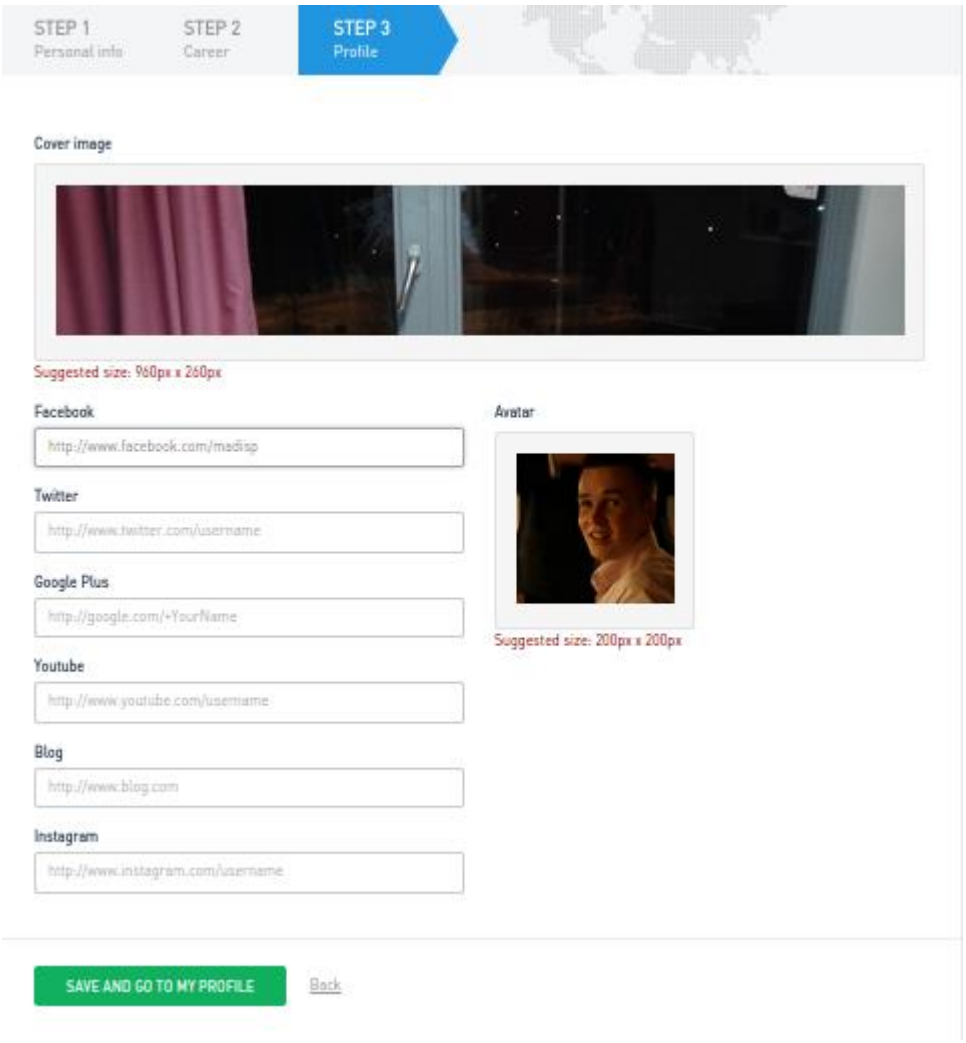
- peamine spordiala, mis tuuakse välja ajajoonel sportlase nime all;
- treener (valikuline);
- spordiklubi (valikuline);
- biograafia.

Eeltoodud loetelu viimase kolme välja põhjal konstrueeritakse sportlase ajajoone biograafia sektsioon.



Joonis 4. Ekraanitõmmis profiili loomise teisest sammust

Profiili loomise viimases sammus (joonis 5) tuleb sportlasel üles laadida ajajoone pilt (*cover image)* ja profiili pilt (*avatar)*. Pildid laetakse serverisse asünkroonselt, kasutades selleks Pythoni pakki django-upthor [21]. Lisaks on sportlasel võimalus välja tuua enda erinevate sotsiaalvõrgustike lingid (kõik valikulised).



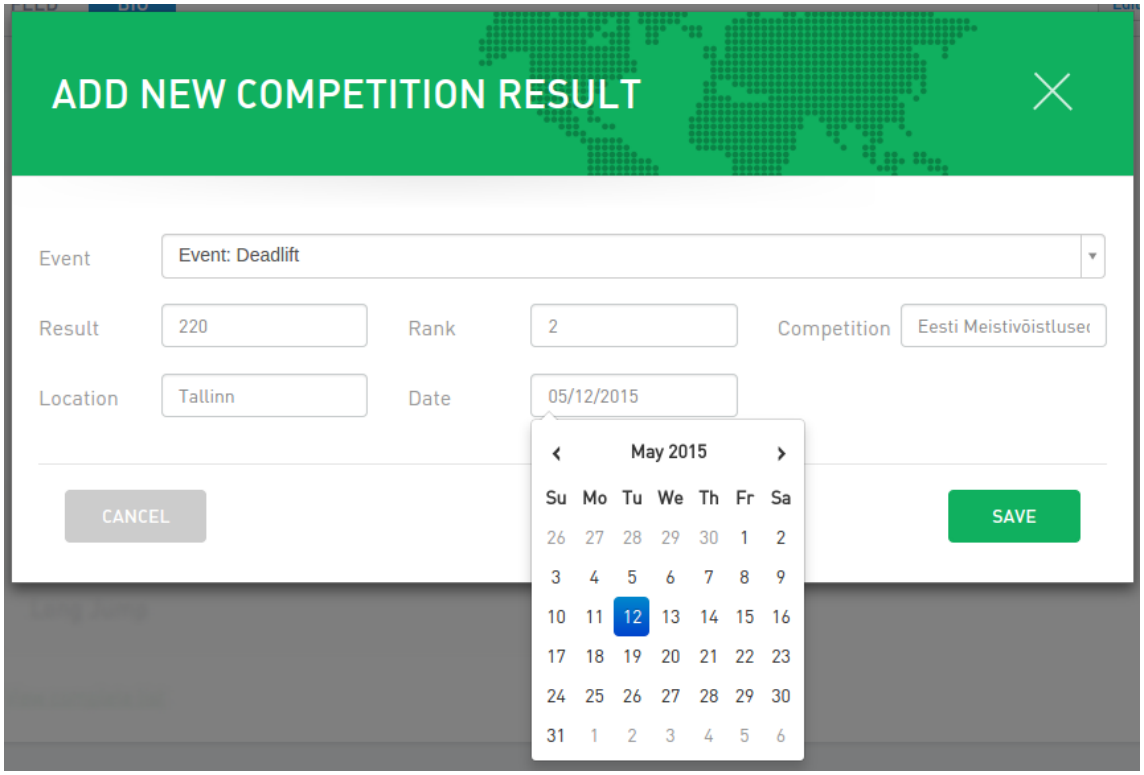
Joonis 5. Ekraanitõmmis profiili loomise kolmandast sammust

4.1.2. Võistlustulemuste lisamine

Võistlustulemuste lisamiseks tuleb sportlasel enda ajajoonel klõpsata *„Add new competition results“* nuppu, mille tulemusena avaneb vastava vormiga *modal* (joonis 6), kus tuleb sisestada järgnevad andmed:

- spordiala (*event)*, mis tuleb valida eeldefineeritud valikute hulgast;
- tulemus (*result)*;

- saavutatud koht (*rank)* (valikuline)*;*
- võistluse nimetus *(competition)* (valikuline);
- võistluse toimumispaik *(location)* (valikuline);
- kuupäev *(date)*.



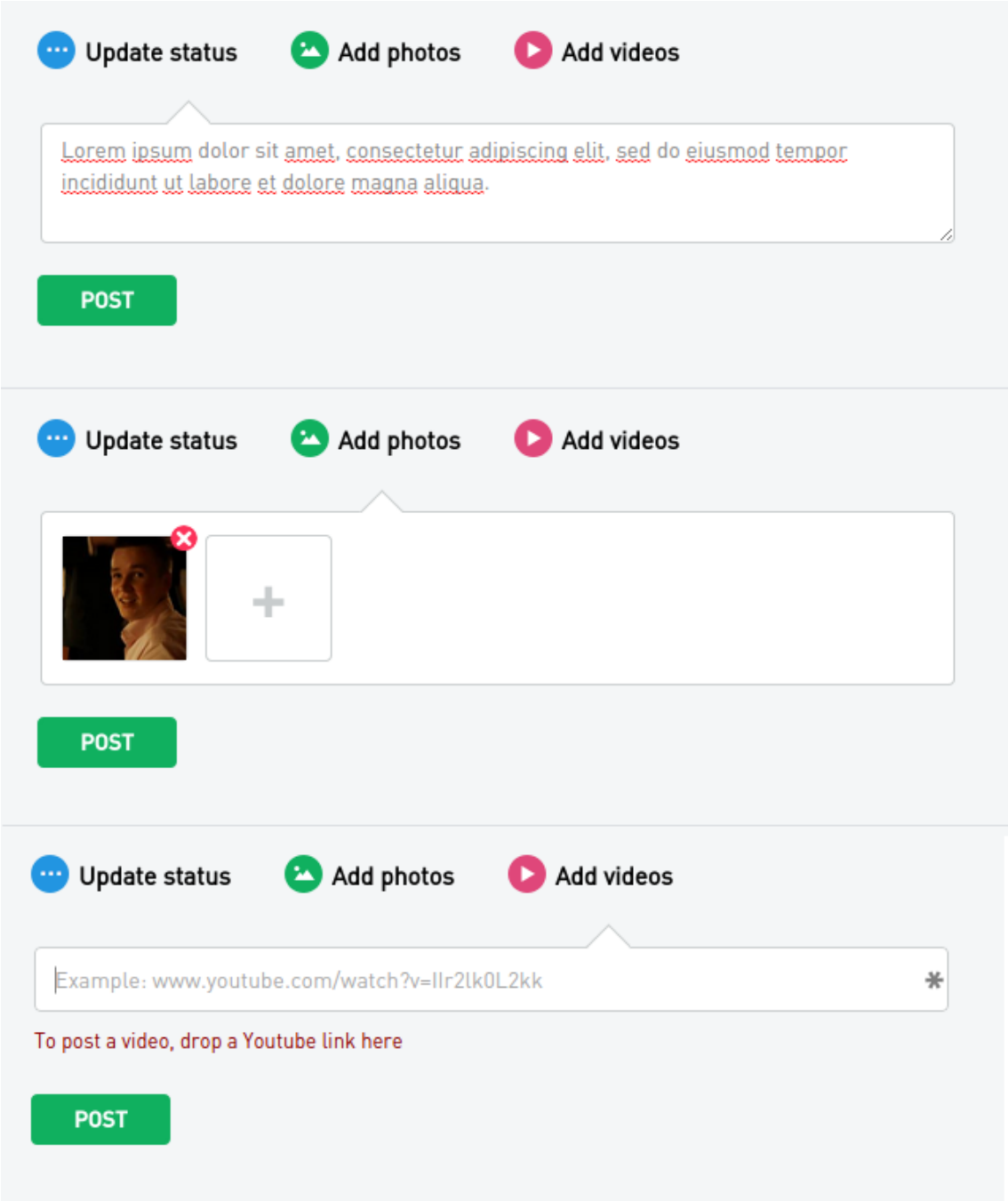
Joonis 6. Ekraanitõmmis tulemuse lisamisest

4.1.3. Ajajoone postituse lisamine

Sportlasel on võimalik enda ajajoonele lisada kolme sorti postitusi: teksti, fotosid ja videoid, klõpsates selleks ajajoonel vastavalt „*Update status*“, „*Add photos*“ või „*Add videos*“ nuppudele.

Tekstipostituse lisamiseks avaneb sportlasele harilik tekstikast (*textbox)*. Pilte on võimalik mugavalt mitme kaupa lisada ning video lisamiseks tuleb vormi kopeerida YouTube’i video URL.

Kõikide erinevate postituste lisamise vormid on kokkupandult näha joonisel 7.



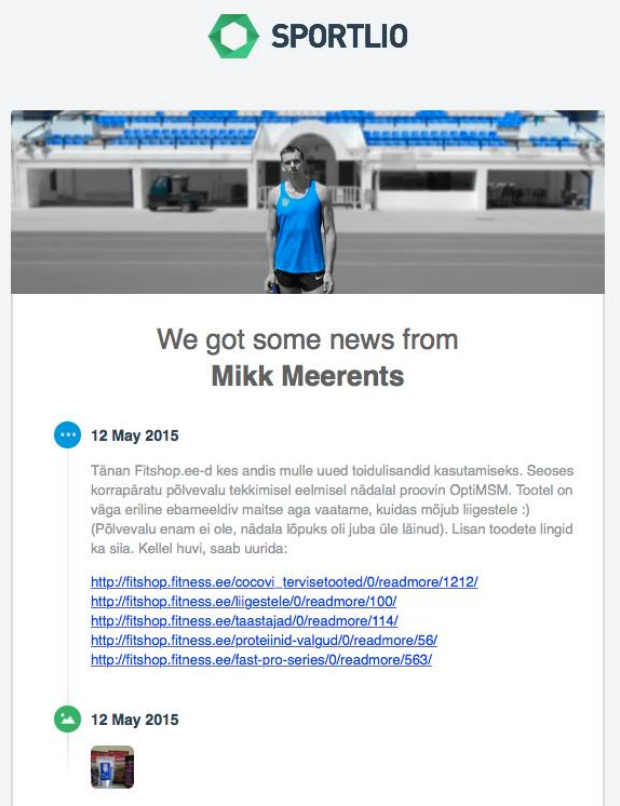
Joonis 7. Ekraanitõmmis erinevate ajajoonte postituste lisamisest

4.2. Tavakasutaja

Tavakasutaja peamiseks Sportlio kasutamise eesmärgiks on erinevate sportlaste ja nende tegemistega kursis olemine – seda kõike näeb sportlase leheküljelt.

4.2.1. Fänniks hakkamine

Selleks, et saada oma lemmiksportlaste ajajoonepostituste kohta meeldetuletusi, on võimalik hakata tema fänniks. Selleks tuleb klõpsata sportlase lehe päises olevale südamekesele. Fännidele saadetakse iga päev sportlase ajajoonepostituste korral email, kus need ka välja toodud on (joonis 8).



Joonis 8. Ekraanitõmmis fännile saadetud emailist

5. Kokkuvõte

Töö eesmärgiks oli luua veebirakendus, millega sportlastel oleks võimalik enda tulemuste üle järge pidada ning fänne enda saavutuste ja tegemistega kursis hoida, luues selleks iga sportlase kohta personaalne lehekülg, mida saaks kasutada ka enda tutvustamiseks potentsiaalsetele sponsoritele.

Töö tulemusena valmis veebirakendus Sportlio, kus 20.05.2015 seisuga on 663 kasutajat ja 439 sportlase profiili. Rakenduses on kahte tüüpi kasutajaid – sportlased ja tavakasutajad. Rakenduses tõsisemaid vigu, mis põhjustaks serveri töö katkemise, ei ole olnud – küll aga on ilmnenud viga, kus kasutaja on välisvõtme väljale sisestanud ebakorrektse väärtuse, mida väljale vastavas tabelis tegelikult ei eksisteeri.

Rakendus on arendatud Pythoni raamistikul Django ning kasutab paljusid selle sisseehitatud funktsionaalsuseid – mudeleid, klassipõhiseid vaateid jne. Samuti on kasutusel erinevad kolmandate osapoolte rakendused (nagu Southi migratsioonid). Visuaalse poole loomiseks on laialdaselt kasutatud jQueryt ja SCSSi.

Sportlaste peamisteks kasutatavateks funktsionaalsusteks on enda profiili loomine, võistlustulemuste sisestamine ning informeerivate postituste tegemine, millest valmib igale sportlasele personaalne ning terviklik lehekülg. Tavakasutaja peamiseks tegevuseks loodud veebirakenduses ongi eelmainitud personaalsete sportlaste lehekülgede vaatamine ning soovi korral lemmiksportlaste fänniks hakkamine, mis tagab nende seinapostituste uudistevoo otse kasutaja meilikontole.

Tuleviku väljavaadetena on antud projekti jaoks tähtsaim andmebaasi suurendamine - nii kasutajate arvu kui ka toetatud spordialade poolest. Funktsionaalsuse osas ei ole hetkel suuremahulist tootearendust ette nähtud, kuid seevastu on plaanis teostada väiksemaid parandustöid – näiteks lisada eraldi võimalus, mille kaudu saaks meedia alla laadida sportlase ajakohase pildi.

6. Kasutatud kirjandus

1. Django FAQ about MVC in Django [WWW]

https://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names (20.05.2015)

2. Holovaty, A., Kaplan-Moss, J. The Definitive Guide to Django: Web Development Done Right. Aspress. 12.2007

3. Pinterest [WWW] https://www.pinterest.com/ (20.05.2015)

4. Instagram[WWW] https://instagram.com/ (20.05.2015)

5. Mozilla [WWW] https://www.mozilla.org/ (20.05.2015)

7. Django [WWW] https://docs.djangoproject.com/en/1.8/topics/class-based-views/ (20.05.2015)

8. Hunt, A., Thomas, D. The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley Professional. 20.10.1999

9. Sportlio blog [WWW] http://blog.sportlio.com/?p=53 (20.05.2015)

10. South [WWW] https://south.readthedocs.org/en/latest/ (20.05.2015)

11. PostgreSQL [WWW] http://www.postgresql.org/about/ (20.05.2015)

12. Brown, A., Wilson, G. The Architecture Of Open Source Applications Volume II: Structure, Scale and a Few More Fearless Hacks. 07.05.2008

13. Gunicorn [WWW] http://gunicorn.org/#community (20.05.2015)

14. [WWW] http://serverfault.com/questions/331256/why-do-i-need-nginx-and-something-like-gunicorn (20.05.2015)

15. Sass ja Scss [WWW]

http://sass-lang.com/documentation/file.SASS_CHANGELOG.html#scss_sassy_css (20.05.2015)

16. jQuery [WWW] https://jquery.com/ (20.05.2015)

17. HTML DOM [WWW] http://www.w3schools.com/js/js_htmldom.asp (20.05.2015)

18. Amazon WebServices [WWW] http://aws.amazon.com/products/?nc2=h_ql_reinvent (20.05.2015)

19. Mandrill [WWW] https://mandrill.com/about/ (20.05.2015)

20. Vallaste e-teatmik [WWW] http://vallaste.ee/ (20.05.2015)

21. django-upthor [WWW] https://github.com/Jyrno42/django-upthor (20.05.2015)

22. Kirjavahetus Sportlio OÜ juhatuse liikme ja reaalse eestvedaja Mikk Meerentsiga 26.05.2015 (ekiri)

23. Mercurial [WWW] http://www.ibm.com/developerworks/aix/library/au-mercurial/ (20.05.2015)

24. Thorgate [WWW] http://www.thorgate.eu/ (20.05.2015)

25. Django [WWW] https://www.djangoproject.com/ (20.05.2015)

26. WSGI [WWW] https://www.python.org/dev/peps/pep-0333/ (25.05.2015)

27. Webopedia [WWW] http://www.webopedia.com/TERM/M/modal_window.html (25.05.2015)

28. Webhook [WWW] https://webhooks.pbworks.com/w/page/13385124/FrontPage (25.05.2015)

29. Mixin [WWW] http://www.linuxjournal.com/article/4540 (25.05.2015)