TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

VEEBITEENUST KASUTAV KASUTAJA TEGEVUSTE JÄLGIMISE JA SALVESTAMISE RAKENDUS

Bakalaureusetöö

Üliõpilane: Mart Madisson
Üliõpilaskood: 096043IAPB
Juhendaja: Raul Liivrand

Tallinn
2015

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

……. ……

(kuupäev) *(allkiri)*

Annotatsioon

Käesoleva töö eesmärgiks on luua rakendus, mis kogub andmeid kasutajate tegevuse kohta veebilehel ning võimaldab pärida andmeid nende kohta. Antud rakendus on suunatud veebilehtedele, kelle omanikel puuduvad IT-alased teadmised. Seetõttu on väga tähtis, et antud rakendust oleks võimalikult lihtne installeerida veebilehele. Tekkinud andmekogust on võimalik pärida erinevaid tulemusi kasutaja tegevuse andmetest, mille põhjal saab veebilehe autor otsustada, mida lehel paremaks teha.

Lõputöö on kirjutatud eesti keeles ning sisaldab 38 lehekülge, 5 peatükki ja 5 joonist.

Annotation

The aim of this work is to create an application that collects data on the users' activities on the website and allows to query information about them. This application is directed to websites whose owners lack the required IT knowledge. Therefore it is very important that this application is as easily installable on a website as possible. The resulting database renders it possible to get various results about the data of the user's activity, on the basis of which the author of the website can decide how to improve the page.

The thesis is written in Estonian, and contains 38 pages, 5 chapters and 5 figures.

Jooniste nimekiri

Sisukord

1. Sissejuhatus

Selles peatükis kirjeldatakse töö tausta, probleemi, eesmärki ja metoodikat ning antakse ülevaade töö sisust. Rakenduse kood on avalik aadressil https://github.com/wiz6/loputoo.

1.1. Taust ja probleem

Tänapäeval on IT-teenused küllaltki kallid ning paljud väikeettevõtjad maksavad väga palju tavalise veebianalüüsi eest. Eesmärgiks on luua rakendus, mis aitab välja tuua konkreetse veebilehe kasutamise miinused ka plussid.

1.2. Ülesande püstitus

Lõputöö eesmärgid:

- püüda kinni kasutaja tegevused veebilehel
- luua veebilehe kasutajate tegevuste andmekogum
- luua funktsioonid, mis kuvavad andmekogumi põhjal veebilehe kasutajate kasutamise harjumusi(API)

1.3. Metoodika

Ülesandes püstitatud eesmärkide saavutamiseks on töö jagatud osadeks. Esmalt valiti tehnoloogiad, mida kasutada. Seejärel joonistati rakenduse disain ja erinevate komponentide vahelised seosed. Järgmisena kirjutati rakendusele andmebaas ning struktueeriti failide kataloog. Sellele järgnes algse disaini koodi kirjutamine. Esmalt kirjutati kliendi poolne osa, kus toimus kasutajate tegevuse kinnipüüdmine ja salvestamine andmebaasi. Programmeerimise käigus muutus andmebaasi struktuur. Viimasena kirjutati rakenduse see osa, mis küsib andmebaasist andmeid.

1.4. Ülevaade tööst

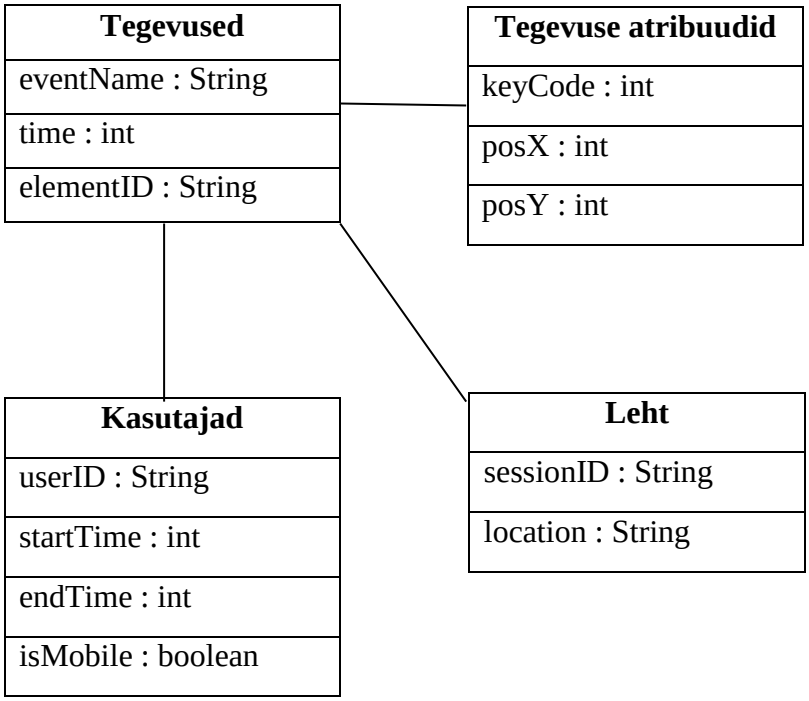
Antud töö koosneb kahest põhiosast. Esimeses osas tuuakse välja rakenduse kontseptuaalne andmemudel ja andmebaasi skeem, kirjeldatakse andmekogumi teenuseidning tuuakse välja nende poolt pakutav andmemudel. Teises osas kirjeldatakse täpsemalt rakenduse arhitektuuri ja tehnoloogiat.

2. Rakenduse funktsionaalsus

Selles peatükis tuuakse välja rakenduse kontseptuaalne andmemudel ja andmebaasi skeem. Samuti kirjeldatakse andmekogumi teenuseid ning tuuakse välja nende andmemudel.

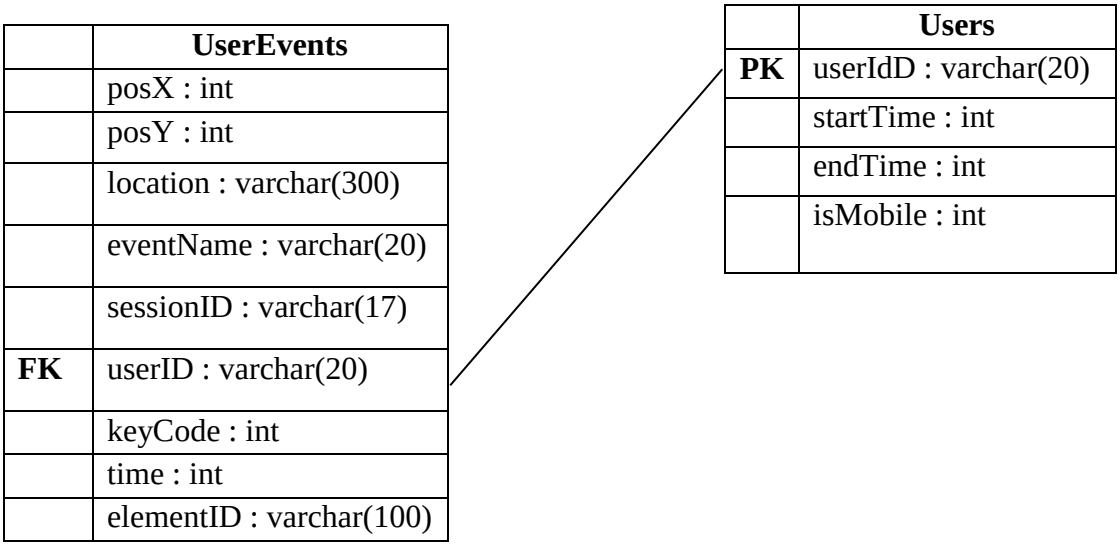
2.1. Kontseptuaalne andmemudel

Rakenduse põhiobjektiks on tegevused. Igat tegevust iseloomustavad atribuudid. Tegevustega on seotud kasutajad. Väljad „sessionID" ja „location" pole otseselt seotud tegevusega, vaid pigem veebilehega. Lehe atribuudid iseloomustavad pigem lehte kui tegevust.



Joonis 1: Kontseptuaalne andmemudel

2.2. Andmebaasi skeem



Joonis 2: Andmebaasi skeem

Tabelid on omavahel seotud userID-ga. Tabelis „Users" on userID primaarvõti(*primay key*) tabelis „usersEvents" välisvõti(*foreign key*). Järgnevalt tuuakse välja tabelite kirjeldus:

UserEvents

Veeru nimetus	Tüüp	Pikkus	Võimalikud väärtused	Kohustuslik
posX	tisarv		Positiivsed arvud	Ei
posY	täisarv		Positiivsed arvud	Ei
location	tekst			Jah
sessionID	tekst	17	Numbritest ja tekstist koosnev	Jah
eventName	tekst		*click, mousemove, keyup*	Jah
userID	tekst	8	Numbritest ja tekstist	Jah

			koosnev	
keyCode	täisarv		56, 75, 45	Ei
elementID	tekst			Ei

Users

Veeru nimetus	**Tüüp**	**Pikkus**	**Võimalikud väärtused**	**Koustuslik**
startTime	täisarv		UNIX timestamp	Jah
endTime	täisarv		UNIX timestamp	Jah
isMobile	täisarv		1 – mobiil, 2 – muu seade	Ei
userID	tekst		Numbritest ja tekstist koosnev	Jah

2.3. Tegevuste salvestamise kirjeldus

Kõik tegevused püütakse kinni Javascriptiga kasutaja brauseris. Kui kasutaja siseneb veebilehele, siis laetakse veebilehele Javascripti objekt „Tracker", mis hakkab seda tööd tegema. Kinni püürtakse kolme erineva kursoritegevus:

- klikk (*click*) –ühekordne klikk kursori poolt
- nupuvajutus (*keyup*) – nupuvajutus klaviatuuril
- kursori liikumine (*mousemove*)

Lehe laadimisel registreeritakse lehele(dokumendi külge) Javascripti meetodid: „*registerClickEvent*", „*registerKeyUpEvent*", „*registerMouseMoveEvent*", mis sisaldavad jQuery meetodit:

```
$(document).bind('tegevuse nimi', function(e) {
   //tegevuse kinni püüdmine
});
```

Kui nupuvajutus(*keyup*) ja klikk(*click*) salvestatakse koguaeg, olenemata tegevuse intervallist, siis tegevusele kursori liikumine(*mousemove*) on määratud piirang, tegevust ei salvestata tihedamini kui 300 millisekundit. Samuti tuleb silmas pidada, et „*keyup*" tegevuse puhul ei anna brauser kaasa kursori koordinaate. Aga selle probleemi lahendab teadmine, et kui elemendil teha nupuvajutus, siis see eeldab, et enne seda on toimunud „*mousemove*" (sellel tegevusel on X-ja Y-koordinaadid saadaval).

Nagu näha on „bind" meetodi *callback*'il parameeter „e", mis sisaldab erinevaid tegevuste omadusi, mida brauser kaasa annab. Siit objektist saadakse tegevuse jaoks vajalikud andmed. Kui lehel registreeritakse mõni tegevus, siis objekti „e" küljest võetud vajalikud andmed pannakse Javascripti „UserEvent" objekti. „UserEvent" koosneb järgmistest väljadest:

- posX – tegevuse X-koordinaat. See saadakse brauserilt.
- posY – tegevuse Y-koordinaat. See saadakse brauserilt.
- eventName – tegevuse nimi. See saadakse brauserilt.
- keyCode – nupuvajutuse ASCII kood. See saadakse brauserit.
- time – tegevuse aeg millisekundites alates 01.01.1970. See saadakse brauserilt.
- location – veebilehe URL(*pathname*). See saadakse „windowsi" objektilt ehk brauserilt.
- isMobile – tuvastab, kas tegemist on mobiili või millegi muuga. Kui mobiiliga, siis väärtus 1, muuljuhul 0. Need andmed saadakse brauselit.
- elementID – elemendi ID, kus tegevus toimus. See saadakse brauserilt.
- startTime – kasutaja esmase lehele sisenemise aeg. See saadakse Javascripti „Date" objektist.
- sessionID – lehe URLi unikaalne väärtus. Selle väärtuse unikaalsust arvutatakse välja järgmiselt:

  ```
  Math.floor(Math.random()*1e15)+newDate().getMilliseconds().toString(36).toU
  pperCase()
  ```

Javascript genereerib juhusliku arvu 1 ja 0 vahel. See korrutatakse astmega 15 ja ümardatakse üles ning saadakse 15-kohaline arv. Selle arvule liidetakse praeguse ajahetke millisekundid, mis on tehtud 36-ndik süsteemi. Lõpuks tehakse kogu tulemus suurteks tähtedeks. Kokku saadakse 17-kohaline väärtus. Seda väärtust hoitakse brauseri objektis „localStorage". See tähendab seda, et see väärtus püsib seal seni, kuni lehte laetakse uuesti.

- userID – kasutaja unikaalne väärtus. See väärtus genereeritakse täpselt samamoodi nagu sessionID. Ainus erinevus on see, et juhuslik arv korrutatakse läbi 6-astmega. Saadakse 8-kohaline väärtus. Seda väärtust hoitakse brauseri „sessionStorage" objektis. See tähendab seda, et see väärtus on seal seni, kuni kasutaja sulgeb kõik veebilehe aknad.

Kõik ühel URL-il toimunud tegevused salvestatakse massiivi. Veebilehele on registreeritud ka Javascripti meetod(*unbeforeunload*), mis käivitatakse hetk enne lehelt lahkumist. Selle koha peal salvestatakse AJAX-iga kõik lehel toimunud tegevused andmebaasi.

2.4. Andmekogumi teenuste kirjeldus

Selles peatükis kirjeldatakse andmekogumite teenuseid. Näidatakse kuidas teenus võtab andmed andmebaasist, kuhu andmed pannakse ning kuidas arvutab lõpptulemuse. Samuti tuuakse välja ka teenuse *endpoint*'(URL millega saab teenuse kätte) aadress.Andmekogumil on järgmised teenused:

1. otsi kasutajat ID järgi(*getUserById)* – leiab andmebaasist kasutaja ette antud ID järgi.

   Andmebaasist pärimine:

```
SELECT * FROM users where userID=?
```

   Tagastatav andmestruktuur andmebaasist: User objekt

   Tagastatavad väljad andmebaasist: userID, startTime, endTime, isMobile

Teenuse endpoinst aadress: service.php?action=getUserById

Endpoint'i parameetrid:

- id – kasutaja unikaalne väärtus (kohustuslik)

Tagastatav andmestruktuur *endpoint*'is: JSON

Tagastatavad väljad *endpoint*'is: userID, startTime, endTime, isMobile

Tagastatava andmestruktuuri näide endpoint'is:

```
{
     "userID":"515863D6",
     "startTime":"1431538381535",
     "endTime":"1431538435000",
     "isMobile":"0"
}
```

2. otsi kõik kasutajad(*getAllUsers)*

   Andmebaasist pärimine:

   ```
   SELECT * FROM users ORDER BY {$orderBy} {$order}
   ```

   Tagastatav andmestruktuur andmebaasist: massiiv User objektidega

   Tagastatavad väljad andmebaasist: userID, startTime, endTime, isMobile

   Teenuse endpoinst aadress: service.php?action=getAllUsers

   Endpoint'i parameetrid:

   - orderBy – väli mille järgi sorteeritakse (valikuline)
   - order – sorteerimisvälja järjekord(valikuline). Vaikimisi: ASC

   Tagastatav andmestruktuur endpoint'is: JSON massiiv

   Tagastatavad väljad *endpoint*'is: userID, startTime, endTime, isMobile

   Tagastatava andmestruktuuri näide *endpoint*'is:

   ```
   [
      {
         "userID":"515863D6",
         "startTime":"1431538381535",
         "endTime":"1431538435000",
         "isMobile":"0"
      },
      {
   ```

```
      "userID":"771609OK",
      "startTime":"1431547009553",
      "endTime":"1431547248000",
      "isMobile":"0"
   }
]
```


3. otsi kõik kasutaja tegevused kasutaja ID järgi(*getEventsByUserId)*

   Andmebaasist pärimine:

```
SELECT * FROM userEvents WHERE userID=?
```

   Tagastatav andmestruktuur andmebaasist: massiiv UserEvent objektidega

   Tagastatavad väljad andmebaasist: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

   Teenuse endpoinst aadress: service.php?action=getAllUsers

   Endpoint'i parameetrid:

   - userID – kasutaja unikaalne väärtus(kohustuslik)

   Tagastatav andmestruktuur endpoint'is: JSON massiiv

   Tagastatavad väljad *endpoint*'is: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

   Tagastatava andmestruktuuri näide *endpoint*'is:


```
[
   {
      "posX":"276",
      "posY":"1",
      "time":"1431538383145",
      "location":"\/loputoo\/test.html",
      "eventName":"mousemove",
      "sessionID":"2084859025198969R",
      "userID":"515863D6",
      "keyCode":null,
      "elementID":"name",
   },
{
      "posX":"694",
```

```
        "posY":"666",
        "time":"1431538384401",
        "location":"\/loputoo\/test.html",
        "eventName":"mousemove",
        "sessionID":"2084859025198969R",
        "userID":"515863D6",
        "keyCode":null,
        "elementID":"name"
    }
]
```


4. otsi kõik tegevused sessiooni ID järgi(*getEventsBySessionId)*

   Andmebaasist pärimine:

   ```
   SELECT * FROM userEvents WHERE userID=?
   ```

   Tagastatav andmestruktuur andmebaasist: massiiv UserEvent objektidega

   Tagastatavad väljad andmebaasist: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

   Teenuse endpoinst aadress: service.php?action=getEventsBySessionId

   Endpoint'i parameetrid:

   - sessionID – sessiooni ID(kohustuslik)

   Tagastatav andmestruktuur *endpoint*'is: JSON massiiv

   Tagastatavad väljad *endpoint*'is: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

   Tagastatava andmestruktuuri näide *endpoint*'is: vt eelmist punkti

5. otsi kõik unikaalsed sessioonid kasutaja ID järgi(*getSessionsByUserID)*

   Andmebaasist pärimine:

   ```
   SELECT distinct sessionID, locationfrom userEvents
   where userID=?
   ```

   Tagastatav andmestruktuur andmebaasist: massiiv

   Tagastatavad väljad andmebaasist: sessionID, location

   Teenuse endpoinst aadress: service.php?action=getSessionsByUserID

Endpoint'i parameetrid:

- userID – kasutaja unikaalne väärtus (kohustuslik)

Tagastatav andmestruktuur *endpoint*'is: JSON massiiv

Tagastatavad väljad *endpoint*'is: sessionID, location

Tagastatava andmestruktuuri näide endpoint'is:


```
[
   {
      "location":"\/loputoo\/test.html",
      "sessionID":"2084859025198969R",
   },
   {
      "location":"\/loputoo\/test.html",
      "sessionID":"2084859025198969R",
   }
]
```


6. otsi kõik tegevused kasutaja ID ja sessiooni ID järgi(*getEventsByUserAndSession)*

   Andmebaasist pärimine:

```
SELECT * from userEvents where userID=? AND
sessionID=?
```

   Tagastatav andmestruktuur andmebaasist: massiiv UserEvent objektidega

   Tagastatavad väljad andmebaasist: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

   Teenuse endpoinst aadress: service.php?action=getEventsByUserAndSession

   Endpoint'i parameetrid:

   - userID – kasutaja unikaalne väärtus(kohustuslik)
   - sessionID – sessiooni ID(kohustuslik)

   Tagastatav andmestruktuur *endpoint*'is: JSON massiiv

   Tagastatavad väljad endpoint'is: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

   Tagastatava andmestruktuuri näide *endpoint*'is: vt punkt 3

7. leia kõige populaarsemad HTML elemendi ID(*getMostPopularElemnentID)*

   Andmebaasist pärimine:

```
SELECT elementID, count(elementID) as count FROM
userEvents where eventName=? group by elementID ORDER
BY count {$order}
```

   Tagastatav andmestruktuur andmebaasist: massiiv objektidega

   Tagastatavad väljad andmebaasist: elementID, count

   Teenuse endpoinst aadress: service.php?action=getMostPopularElementID

   Endpoint'i parameetrid:

   - eventName – tegevuse nimi (valikuline)
   - order – sorteerimis järjekord (valikuline). Vaikimisi: suurim

   Tagastatav andmestruktuur *endpoint*'is: JSON massiiv

   Tagastatavad väljad *endpoint*'is: elementID, count

   Tagastatava andmestruktuuri näide *endpoint*'is:

```
[
   {"elementID":"kast2","count":"3"},
   {"elementID":"kast1","count":"2"},
   {"elementID":"kast3","count":"1"},
   {"elementID":null,"count":"0"}
]
```

8. leia kõige külastatum leht(*getMostVisitedPage)*

   Andmebaasist pärimine:

```
SELECT count(location) as count, location from
userEvents group by location ORDER BY count {$order}
```

   Tagastatav andmestruktuur andmebaasist: objekt

   Tagastatavad väljad andmebaasist: elementID, count

   Teenuse endpoinst aadress: service.php?action=getMostVisitedPage

   Endpoint'i parameetrid:

   - eventName – tegevuse nimi (valikuline)

- order – sorteerimis järjekord(valikuline). Vaikimisi: suurim

Tagastatav andmestruktuur *endpoint*'is: JSON massiiv

Tagastatavad väljad *endpoint*'is: elementID, count

Tagastatava andmestruktuuri näide *endpoint*'is:

```
[
      {"count":"151","location":"\/loputoo\/test.html"},
      {"count":"38","location":"\/loputoo\/test2.html"}
]
```

9. leia kursori tegevuse suurim ajavahe(*getAllSessionEventTimeDiffByEvent*) – leiab ette antud kursori tegevuse(*click, mousemove, keyup*) suurimad ajavahemikud.
   Selle teenuse teostamisel päriti kõigepealt algandmed andmebaasist.
   Andmebaasist pärimine:

```
SELECT * from userEvents where eventName=? order by
time asc
```

   Tagastatav andmestruktuur andmebaasist: userEvent massiiv
   Seejärel hakati neid andmeid töötlema järgneva algoritmiga:

```
var Result = array(); //uus tulemuste massiiv
for (UserEvents as Event) {
kui (eksisteerib Result[sessionID] &&
Result[sessionID][parimAeg]<ABS(Result[sessionID][eelmineAeg]-Event[time])
    {
        Result[SessionID][parimAeg] = (Result[sessionID][eelmineAeg]-Event[time];
        Result[SessionID][eelmineAeg] = Event[time];
    }
 kui ei eksisteeri Result[sessionID]
    {
        Result[sessionID][parimAeg] = 0;
        Result[sessionID][eelmineAeg] = Event[time];

    }
}
```

Pärast seda kustutatakse üleliigne väli „eelmineAeg” massiivist.

Teenuse *endpoint*’i aadress:

service.php?action=getAllSessionEventTimeDiffByEvent

Endpoint’i parameetrid:

- eventName – tegevuse nimi (valikuline)

Tagastatav andmestruktuur *endpoint*’is: JSONobjekt

Tagastatavad väljad *endpoint*’is: best

Tagastatava andmestruktuuri näide *endpoint*’is:


```
{
	"686354255303740L":1399,
	"3788269434589890Q9":0,
	"659580547362563Q":0,
	"72807194059714620":0,
	"609844459686428MR":319,
	"724890261422842E5":145
}
```


10. leia kõige kauem lehel olnud kasutaja andmed(*getMostUserPageTime)*

    Andmebaasist pärimine:

    ```
    SELECT endTime-startTime as diff, userID from users
    order by diff {$order}limit 1
    ```

    Tagastatav andmestruktuur andmebaasist: objekt

    Tagastatavad väljad andmebaasist: diff(ajavahemik millisekundites), userID

    Teenuse *endpoint*’i aadress: service.php?action=getMostUserPageTime

    Endpoint’i parameetrid:

    - order – sorteerimis järjekord(valikuline). Vaikimisi: suurim eespool

    Tagastatav andmestruktuur *endpoint*’is: JSON object

    Tagastatavad väljad *endpoint*’is: diff(ajavahemik millisekundites), userID

    Tagastatava andmestruktuuri näide *endpoint*’is:


    ```
    {"diff":"53465","userID":"515863D6"}
    ```


11. kuva kõige pikema trajektoriga tegevuste andmed (*getLongestPath)* – kuvab kõige pikema „*mousemove*” tegevust teinud andmed sessiooni piires.

Selle teenuse teostamisel päriti kõigepealt andmebaasist algandmed.

Andmebaasist pärimine:

```
SELECT * from userEvents where eventName="mousemove"
order by sessionID, time asc
```

Tagastatav andmestruktuur andmebaasist: userEvent massiiv

Antud tulemuse väljaarvutamiseks kasutati kahe punkti vahelise pikkuse arvutamise valemit: d = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Teenuse endpoinst aadress: service.php?action=getLongestPath

Tagastatav andmestruktuur *endpoint*'is: JSON objekt

Tagastatavad väljad *endpoint*'is: sessionID, length(pikslites)

Tagastatava andmestruktuuri näide *endpoint*'is:


```
{
      "15624541323632MM":{"length":1430.5172533471},
      "2084859025198969R":{"length":1493.8542634942},
      "222457223106175IA":{"length":1505.6452005676},
      "255613659508526IL":{"length":2725.1345878886},
      "292369467206291O":{"length":0}
}
```


12. kuva kasutajad kursori tegevuse nime järgi(*getUsersByEvent)*

Andmebaasist pärimine:

```
SELECT count(userID) as count, userID from userEvents
where eventName=? GROUP BY userID order by count
{$order} LIMIT 1
```

Tagastatav andmestruktuur andmebaasist: massiiv

Tagastatavad väljad andmebaasist: count, userID

Teenuse *endpoint*'i aadress: service.php?action=getUsersByEvent

Endpoint'i parameetrid:

- order - sorteerimis järjestus. Vaikimisi: suurim enne

- eventName – tegevuse nimi (kohustuslik)

Tagastatav andmestruktuur *endpoint*'is: JSON objekt

Tagastatavad väljad *endpoint*'is: count(kokku kasutaja tegevusi), userID

Tagastatava andmestruktuuri näide *endpoint*'is:

```
{"count":"13","userID":"493764MP"}
```

13. kuva ühe kasutaja külastuste arv kokku ühel URL-il(*getPageVisitorsCount)*

    Andmebaasist pärimine:

```
SELECT count(userID) as count, userID from userEvents
where location=? group by userID
```

    Tagastatav andmestruktuur andmebaasist: massiiv

    Tagastatavad väljad andmebaasist: count(külastuste arv kokku), userID

    Teenuse *endpoint*'i aadress: service.php?action=getPageVisitorsCount

    Endpoint'i parameetrid:

    - url - lehekülje URL

    Tagastatav andmestruktuur *endpoint*'is: JSON massiiv objektidega

    Tagastatavad väljad *endpoint*'is: count(külastuste arv kokku), userID

    Tagastatava andmestruktuuri näide *endpoint*'is:

```
[
      {"count":"24",userID":"493764MP"},
      {"count":"14",userID":"9530503D"}
]
```

14. kuva elemendil veedetud aeg(*getElementVisitedTime)* – kuvab elemendil veedetud aeg kokku ette antud elemndi ID järgi sessiooni piires.

    Selle teenuse teostamisel päriti kõigepealt algandmed andmebaasist.

    Andmebaasist pärimine:

```
SELECT * from userEvents order by sessionID, time asc
```

    Tagastatav andmestruktuur andmebaasist: userEvent massiiv

Tulemuse saamine oli küllaltki lihtne. Kuna SQL tagastas elemendi ID-d grupeerituna(sessiooni piires), siis veedetud aeg elemendil on võrdne grupi esimese ja viimase elemendi aja vahega. Kuna elemendi ID gruppe võis olla mitu, siis igat tulemust võrreldi eelmise tulemusega. Tagastati ainult suurim väärtus.

Teenuse *endpoint*'i aadress: service.php?action=getElementVisitedTime

Tagastatav andmestruktuur *endpoint*'is: JSON objekt

Tagastatavad väljad *endpoint*'is: elementID, total(millisekundites), sessionID

Tagastatava andmestruktuuri näide *endpoint*'is:


```
{
kast1":{"total":2168,"sessionID":"881948210299015JO"},
"kast3":{"total":936,"sessionID":"292619297513738QS"},
"userID":{"total":0,"sessionID":"96865047165192690"},
"kast2":{"total":303,"sessionID":"96865047165192690"}
}
```


15. kuva element, mida on kõige rohkem taaskülastatud(*getMostRepeatedElement)* – element, mille pealt on ära mindud ja siis kohe tagasi tuldud.

    Selle teenuse teostamisel päriti kõigepealt andmebaasist algandmed.

    Andmebaasist pärimine:

    ```
    SELECT * from userEvents order by sessionID, time asc
    ```

    Tagastatav andmestruktuur andmebaasist: userEvent massiiv

    Tagastatavad väljad andmebaasist: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

    Kuna SQL tagastab elemendi ID-d grupeerituna sessiooni piires, siis lõpptulemuse saamiseks, tuleb kontrollida igat elementi tema järgmise ning ülejärgmise elemendiga. Kui järgmine element pole võrdne kontrollitava elemendiga ja ülejärgmine on, siis on tegemist taaskasutatava elemendiga.

    Teenuse *endpoint*'i aadress: service.php?action=getMostRepeatedElement

    Tagastatav andmestruktuur *endpoint*'is: JSON objekt

    Tagastatavad väljad *endpoint*'is: elementID, total(elementide arv kokku)

    Tagastatava andmestruktuuri näide *endpoint*'is:

{"kast3":{"total":3},"":{"total":3}}

16. kuva kõik trükitud sõnad(*getTypedKeywords)* – kuvab kõik ühes sessioonis trükitud sõnad.

    Selle teenuse teostamisel päriti kõigepealt algandmed andmebaasist.

    Andmebaasist pärimine:

```
SELECT * from userEvents order by sessionID, time asc
```

    Tagastatav andmestruktuur andmebaasist: userEvent massiiv

    Tagastatavad väljad andmebaasist: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

    Kuna SQL tagastas elemendi ID-d grupeerituna sessiooni piires, siis lõpptulemuseks saadi järjestikused read, kus tegevuse nimi oli „keyup”.

    Teenuse endpoinst aadress: service.php?action=getTypedKeyword

    Tagastatav andmestruktuur *endpoint*’is: JSON objekt

    Tagastatavad väljad *endpoint*’is: sessionID(massiiv)

    Tagastatava andmestruktuuri näide *endpoint*’is:

{
"427321959286928NZ":["SECOND SESSION"],
"699292946141213IN":["TERE","ABCDEFG"]
}

17. kuva elemendid millelt on leheküljelt lahkutud(*getLastUsedElement)*

    Andmebaasist pärimine:

```
select * from userEvents group by sessionID HAVING
max(time);
```

    Tagastatav andmestruktuur andmebaasist: massiiv userEvent objektidega

    Tagastatavad väljad andmebaasist: posX, posY, time, location, eventName, sessionID, userID, keyCode, elementID

    Teenuse *endpoint*’ist aadress: service.php?action=getLastUsedElement

Tagastatav andmestruktuur *endpoint*'is: JSON massiiv objektidega

Tagastatavad väljad *endpoint*'is: count(kui palju kasutajaid kasutab seadet), device(seade)

Tagastatava andmestruktuuri näide *endpoint*'is:


```
[
{
"posX":"97","posY":"112","time":"1431768885522","location":"\/lo
putoo\/test.html","eventName":"click","sessionID":"2224572231061
75IA","userID":"9530503D","keyCode":null,"elementID":"kast1"
},
{
"posX":"105","posY":"71","time":"1431768884561","location":"\/lo
putoo\/test.html","eventName":"click","sessionID":"2224572231061
75IA","userID":"9530503D","keyCode":null,"elementID":"kast2"
},
{
"posX":"111","posY":"197","time":"1431768884297","location":"\/l
oputoo\/test.html","eventName":"click","sessionID":"222457223106
175IA","userID":"9530503D","keyCode":null,"elementID":"kast3"
}
]
```


18. kuva kasutaja seade(*getUserDevice)* – kuvab kõik kasutajad seadmetüübi järgi

    Andmebaasist pärimine:

    ```
    SELECT count(isMobile) as count from users group by
    isMobile order by isMobile {$order}
    ```

    Tagastatav andmestruktuur andmebaasist: massiiv

    Tagastatavad väljad andmebaasist: count(kui palju kasutajaid kasutab seadet), device(seade)

    Teenuse *endpoint*'i aadress: service.php?action=getUserDevice

    Endpoint'i parameetrid:

    - order- seadme sorteerimis järjekord(1 – mobiili järgi, 0 – muu)

Tagastatav andmestruktuur *endpoint*'is: JSON massiiv objektidega

Tagastatavad väljad *endpoint*'is: count(kui palju kasutajaid kasutab seadet), device(seade)

Tagastatava andmestruktuuri näide *endpoint*'is:

```
[
{"count":"1",device":"1"},
{"count":"1","device":"0"}
```

3. Rakenduse arhitektuur ja tehnoloogia

Selles peatükis kirjeldatakse täpsemalt rakenduse arhitektuuri ja disaini ning seda, milliseid tehnoloogiaid kasutati programmeerimisel.

Antud rakendus kasutab järgnevaid tehnoloogiaid:

Javascript – Netscape Communication'i poolt loodud kliendi-poolne(*Client-Side*) interpreteeritav objektorienteeritud programmeerimiskeel, mida kasutatakse koos HTML-iga veebilehtede koostamisel. Veebilehe laadimisel kuvab brauser selle vastavalt HTML-dokumendi tekstile ja täidab ka selles paikneva Javascript'i programmi.

Antud töös on Javascript'i kasutamine asendamatu, sest suur osa tööst tehtakse ära brauseris.

JQuery – Javascript'i raamistik, mis lihtsustab Javascript'i koodi kirjutamist. See tähendab, et sama tulemuse võib saada mõne rea koodiga Jquery-s kui poole rohkemate ridadega Javascriptis. Antud töös võeti see kasutusele just AJAX-päringute puhul, sest neid on JQuerys lihtsam koostada.

HTML5 – markeerimiskeel, mille eesmärgiks on luua korralik veebileht brauserist ja platvormist sõltumata. See on HTML4 edasiarendus.

CSS – programmeerimiskeel, veebilehtede kujundamiseks.

PHP(*Hypertext preprocessor*) – serveripoolneskriptimiskeel veebiarenduseks, mida saab kasutada näiteks koos HTML-iga.

MySQL – avatud lähtekoodiga relatsiooniline andmebaas, mis kasutab struktureeritud päringukeelt SQL. SQL on enamlevinud päringukeel andmebaasiga suhtlemiseks.

JSON(*Javascript Object Notation*)– kergekaaluline andmevahetusformaat, mis põhineb Javascript'i programmeerimiskeele alamhulgal. JSON on tekstiformaat ning sõltumatu programmeerimiskeelest.

Näiteks, kui kasutada rakenduse objekti User, millel on suvalised väärtused, siis JSON kujul näeks see välja nii:

```
{
  "userID": 1,
  "startTime": 123456789,
  "endTime": 1234567899,
  "isMobile": 0
}
```

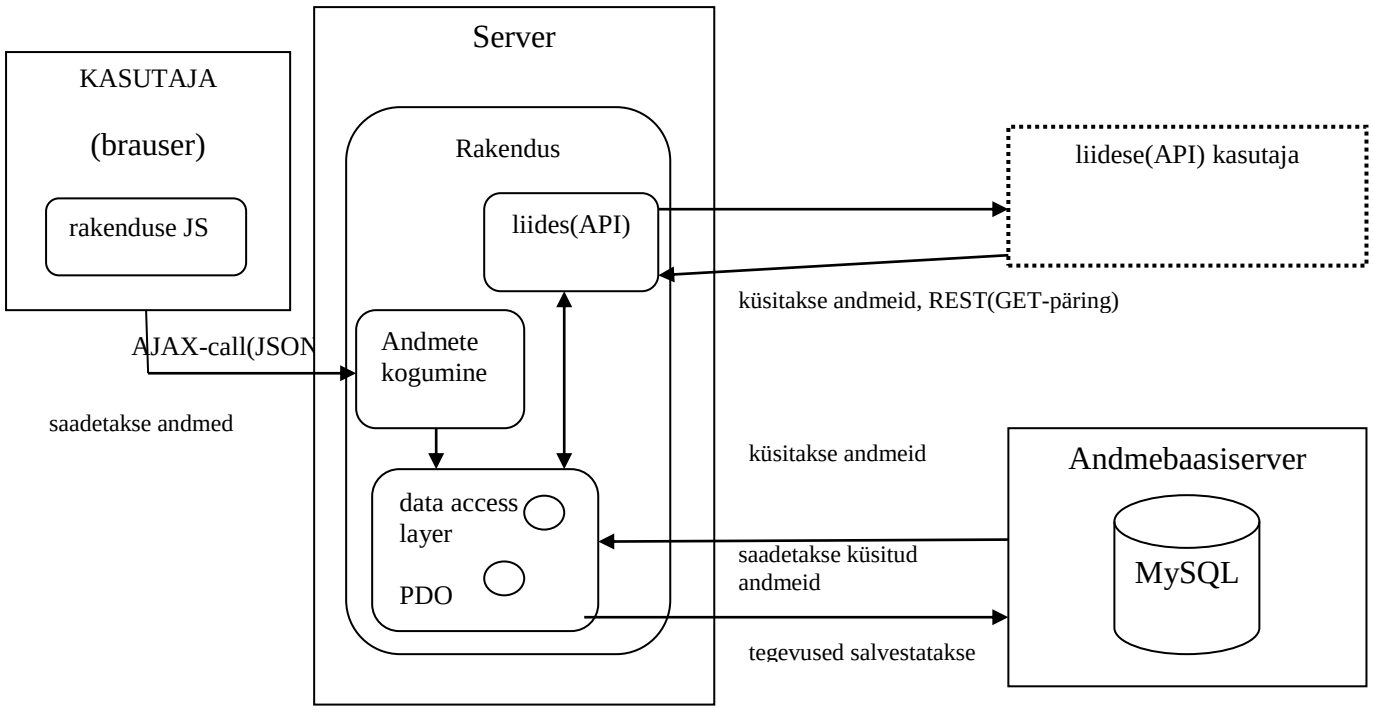
AJAX(*Asynchronous Javascript And XML*) – kogum omavahel seotud veebiarenduse tehnikaid, mis on kasutuses rakenduse kliendi poolel, et luua interaktiivseid veebirakendusi. AJAX võimaldab veebilehtedel Javascript'iga serverisse tagaplaanil päringuid teha(ilma et peaks lehte uuendama) ja andmeid vastu võtta, segamata avatud lehekülje kuvamist ja olekut. Andmete saamiseks kasutatakse XMLHttpRequest objekti.

REST(*Representational State Transfer*) – arhitektuuristiil veebiteenuste loomiseks, milles kasutatakse ühenduse loomiselHTTP-d.

Antud rakenduse tarkvarasüsteem koosneb 3osast:

1) klient(brauser)
2) serveriosa, kus asub serveripoolne loogika
3) andmebaasi server

Brauseris püütakse kinni erinevad kasutajate tegevused, mis saadetakse JSON kujul serverisse. Serveris töödeldakse andmed vastavale kujule ning saadetakse andmebaasi.Andmeid on võimalik küsida läbi REST-veebiteenuse.



Joonis 3: Rakenduse arhitektuur

3.1. Disain

Rakenduse komponentide kirjeldused:

EventController –võtab vastu brauseri pöördumisi ja otsustab, mida nendega teha. See klass pakub järgmisi funktsioone:

- kasutaja tegevuste andmete salvestamine andmebaasi
- kõikide kasutajate küsimine
- kasutaja veebilehe andmebaasi andmete salvestamine

DAO – andmebaasi klasside ülemklass. Siin teostatakse andmebaasiühenduse loomine.

EventDAO – suhtleb andmebaasiga. Teostab CRU funktsioone andmebaasis. Siin klassis luuakse andmebaasi andmete põhjal ka vajalikud tulemused.

PageDAO – suhtleb andmebaasiga. Siin on kirjeldatud kõik veebilehega seonduvad andmebaasi funktsioonid.

User–klass kasutaja objekti jaoks.Järgmisteomadustega:

- userId – kasutaja unikaalne väärtus.
- startTime – kasutaja esmane veebilehe külastuse aeg.
- endTime – kasutaja lahkumise aeg veebilehelt.
- isMobile – ütleb, kas kasutaja kasutas veebilehte mobiilist või mujalt.

UserEvent – klass kasutaja tegevuse objekti jaoks.Järgmisteomadustega:

- posX – kursori x-positsioon lehel.
- posY – kursori y-positsioon lehel.
- time – tegevuse salvestamise aeg.
- location – veebilehe url, kus tegevus salvestati
- eventName – tegevuse nimi
- sessionID – *location*'i unikaalne väärtus
- userID – kasutaja unikaalne väärtus
- keyCode –nupuvajutuse kood

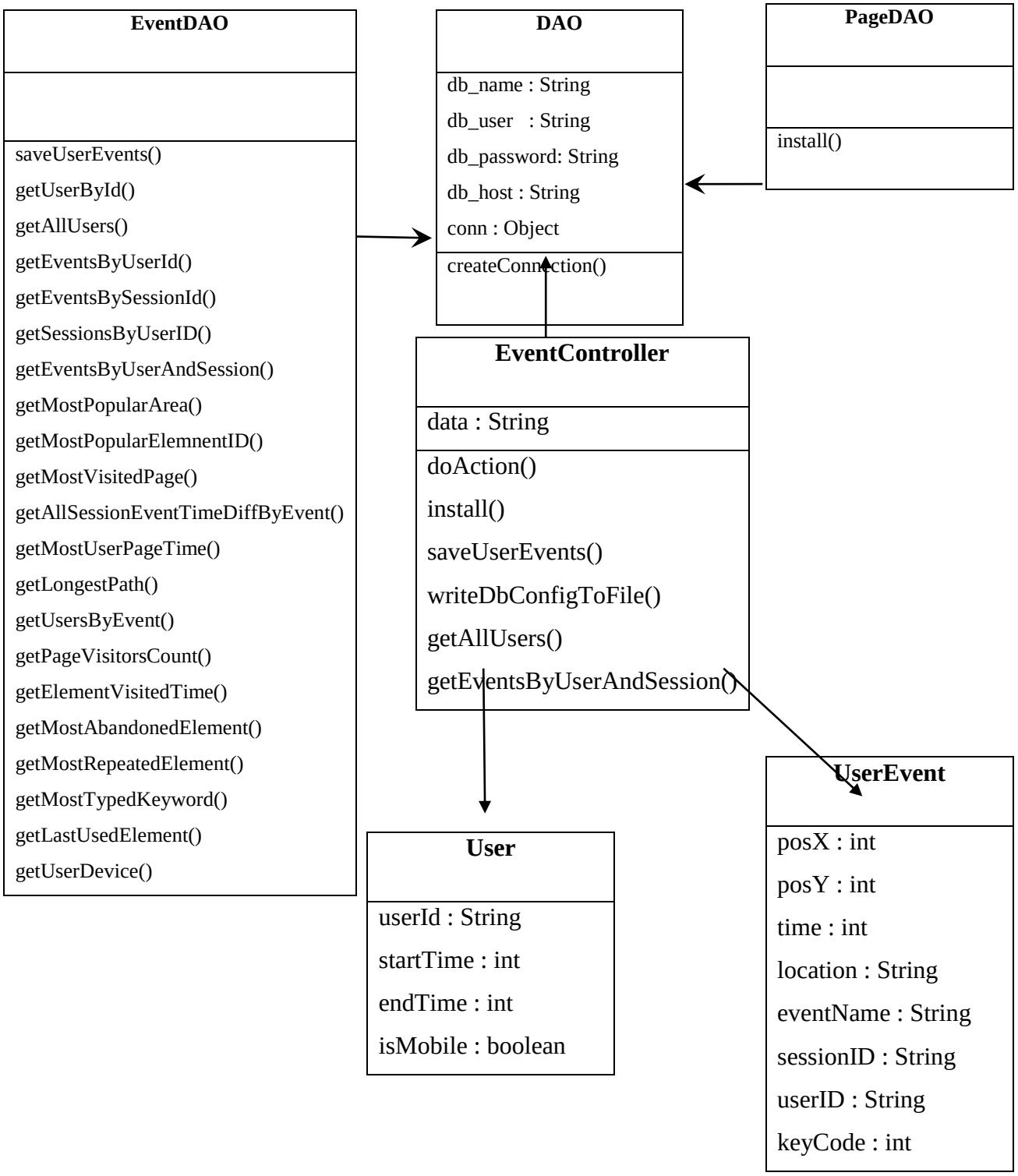
Autoloader – siin laetakse sisse kõik rakenduse klassid.

Konfiguratsioon – siin defineeritakse kogurakendusele vajalik konfiguratsioon. Rakenduselonjärgmisedkonfiguratsiooni väljad:

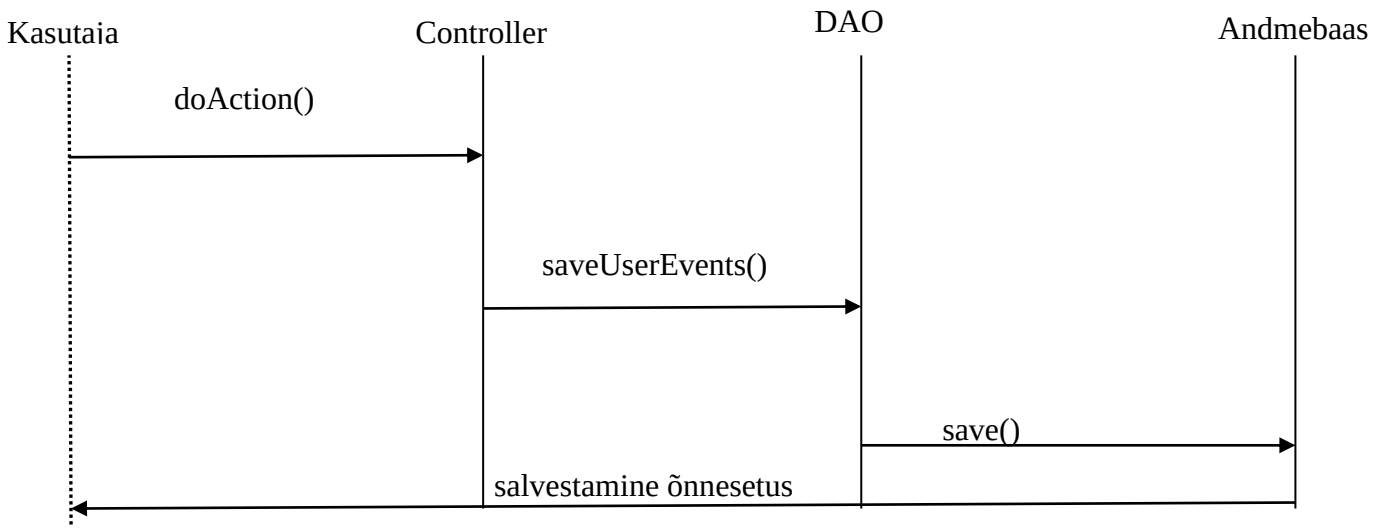
- cdn_uri – Jquery aadress, mis imporditakse rakendusse

Installer – selle jooksutamisel installeeritakse vajalikud andebaasitabelid.

Service – klass, kus on ära kirjeldatud kõik veebiteenuse meetodid. Seda klassi kasutab API.



Joonis 4: Rakenduse klassidiagramm



Joonis 5. Tegevuse salvestamise jadadiagramm

3.2. Disainimustrid

Antud rakendus kasutab järgmisi disainimustreid:

- **Fassaadi muster**(Session Facade) – erinevad tegevuste meetodid on kasutaja eestpeidetud.
- **Singel muster**(*Singleton pattern*)–ressursside kokkuhoiu mõttes kasutatakse ainult ühte viidet andmebaasile.
- **DAO muster(***Data Access Object pattern*)**–** selle mustri põhimõte on olla vahelüliks rakenduse ja andmebaasi vahel. See jagab andmebaasi ning ülejäänud rakenduse kaheks osaks, mille plussiks on: kui ühte osa muuta, ei mõjuta see teist osa.

3.3. Ülevaade rakenduse loogikast

Kui kasutaja tuleb veebilehele, genereeritakse talle unikaalne kasutaja ID ja sessiooni ID, mida hoitakse brauseri mälus. Selleks kasutatakseHTML5 objekte sessionStorage ja localStorage. Kohe hakatakse püüdma kasutaja kursori tegevusi. Need tegevused salvestatakse Javascript'i objekti. Püütakse järgmisi tegevusi: klikk(*click*), nupuvajutus(*keyup*), kursori liikumine(*mousemove*). Kursori liikumist ei püüta tihedamini kui 300ms. Kõik tegevuse objektid salvestatakse ühtsesse Javascript'i massiivi. Kui kasutaja lahkub lehelt(lahkub URL-ilt või uuendab seda, aga domeen jääb samaks), käivitatakse hetk enne lahkumist Javascript'i funktsioon(*onbeforeunload*), mis saadab selle massiivi serverile AJAX-päringuga JSON kujul.Kõik päringud serverile käivad läbi *tracker.php* faili, mis kutsub välja kontrolleri, kes otsustab etteantud GET-parameetrite abil, mida edasi teha. Kontroller kutsub välja vastava DAO-klassi, mis salvestab kasutaja tegevuste massiivi koos kasutaja andmetega andmebaasi.Selline tegevus käib koguaeg hetk enne lehelt lahkumist. Kui kasutaja lahkub nüüd veebilehelt(domeenist) lõplikult, siis salvestatakse ka kasutaja lahkumise aeg.

3.4. Rakendust kasutatav veebiteenus

Andmekogumit kasutab REST veebiteenus, mis kasutab HTTP GET-meetodit, et saada ligi rakenduse andmetele. Selleks tuleb pöörduda faili **service.php** poole ning kaasa anda URL-parameeter nimega „action", mis ütleb, millist meetodit küsitakse. Igas meetodis tehakse PHP-ga andmebaasi päring. Rakenduse andmebaasiühenduse loomisel ja erinevate andmebaasi operatsioonide tegemiseks kasutatakse PHP-moodulit PDO(*PHP Data Object*). PDO-l on väga hea liides, mis muudab koodi puhtamaks ja loetavamaks ning samuti toetab see tegevusi, mida vana MySQL ei toeta. Kõik päringute vastused saadab server tagasi JSON kujul.

3.5. Andmekasutuse API

Antud töö raamesse ei jäänud see osa, mis hakkaks kasutama API-st saadud andmeid. Rakenduses kogutud andmetele on reaalselt olemas nii mõnigi kasutusjuht. Kuna API on avalik, siis saavad seda kasutada kõik väljaspoolt serverit. Kuna rakendus on mõeldud siiski veebilehele, kus ta paikneb, siis võiks järgmistes arendustes luua

liidese, kus tuua välja kõikide API-meetodite tulemused järjestatuna mingite kindlate parameetrite järgi. Samuti saab antud meetodeid kasutada selleks, et luua graafiline pilt kasutaja tegevustest,näiteks näidata kasutaja kursori liikumist mingil kindlal lehel või märgistada täpiga kõik lehel tehtud hiireklikid. Eesmärk oleks, et veebilehe autor saaksülevaate tema veebilehe kasutajate harjumustest ning selletõttu otsustada, mida on lehel tarvis muuta. Variante, mida nende andmetega peale hakata, on küllaltki palju.

4. Kokkuvõte

Töö eesmärgiks oli luua rakendus, mis hakkab koguma kasutajate tegevusi veebilehel, ning luua funktsionid nende andmetega tegelemiseks. Tulemusena loodi rakendus, mida saab lihtsalt installeerida veebilehele ning mis hakkab koguma andmeid kasutaja tegevuste kohta veebilehel. Samuti valmisid ka erinevad funktsioonid, mis erinevate arvutuste järgi annavadkasutajate tegevuste statistikat veebilehe autorile. Antud töö skoopi ei mahtunud see osa, mis hakkaks andmekogumi põhjal hindama veebilehe kasutusharjumusi. See osa võiks jääda edasisse arendusse. Seega on töökäigus esitatud eesmärk täidetud.

Summary

The aim of this work was to create an application that will collect data on the users' activities on the website and to create functions to deal with this data. As a result an application was created which can be easily installed on a website and which will collect data on users' activities on the page. Also, various functions were made that on the basis of a variety of calculations provide the author of the website the statistics on the users' activities. The part that would evaluate the data of the users' web habits did not fit the scope of this work. This part could be left for further development. Thus, the aim of this work is achieved.

5. Kasutatud kirjandus

1. AJAX [WWW]

http://et.wikipedia.org/wiki/Ajax (05.05.2015)

2. Mis on MYSQL või SQL andmebaas? [WWW]

http://www.ithooldus.ee/Kusimused-ja-vastused/mis-on-mysql-voi-sql-andmebaas(05.05.2015)

3. JSON [WWW]

http://et.wikipedia.org/wiki/JSON (05.05.2015)

4. HTML5 – Mis see on? [WWW]

http://metshein.com/index.php/veeb/html5/506-03-html5-mis-see-on(05.05.2015)

5. Mis on jQuery? [WWW]

http://metshein.com/index.php/veeb/jquery/589-01-mis-on-jquery(05.05.2015)

6.http://maurus.ttu.ee/ained/IDU0200_2014/doc/31/SissejuhatusRakArh_I.pdf[WWW] (09.05.2015)

7. http://maurus.ttu.ee/ained/IDU0200_2014/doc/82/MVC.pdf [WWW] (09.05.2015)

8. Mis on Javascript? [WWW]

http://metshein.com/index.php/veeb/javascript/565-01-mis-on-javascript (05.05.2015)

9. http://www.php.net [WWW] (05.05.2015)

10. http://jquery.com/[WWW] (05.05.2015)