

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dejan Mesar

**Sistem za sledenje in analizo uporabe
računalniških aplikacij**

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

MENTOR: izr. prof. dr. Viljan Mahnič

Ljubljana, 2014

Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljane ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil $\text{\LaTeX}$.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Proučite obstoječe servise za pridobivanje podatkov o rabi računalniških aplikacij, kot sta n.pr. Google Analize in Flurry Analytics, ter analizirajte njihove prednosti in slabosti. Na podlagi tega razvijte lasten sistem za sledenje in analizo ter prikažite njegove prednosti v primerjavi z ostalimi rešitvami. Podrobno opišite postopek razvoja in uporabljene tehnologije ter preizkusite delovanje vašega sistema na primerni testni aplikaciji.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Dejan Mesar, z vpisno številko **63080101**, sem avtor diplomskega dela z naslovom:

Sistem za sledenje in analizo uporabe računalniških aplikacij

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Viljana Mahničā,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 12. junij 2014

Podpis avtorja:

Hvala.

Always code as if the guy who ends up
maintaining your code will be a violent psy-
chopath who knows where you live.

– – Martin Golding

Kazalo

Povzetek

Abstract

1	Uvod	1
2	Zbiranje podatkov o uporabi računalniških aplikacij	5
2.1	Zbiranje podatkov obiska spletnih strani ali spletnih aplikacij . .	6
2.1.1	Lokalno zbiranje podatkov	6
2.1.2	Oddaljeno zbiranje podatkov	7
2.2	Zbiranje podatkov v mobilnih aplikacijah	7
2.3	Zbiranje podatkov v namiznih aplikacijah	8
2.4	Delovanje in uporaba Google Analiz	9
2.5	Delovanje in uporaba Flurry Analytics	12
2.6	Slabosti in prednosti trenutnih rešitev	16
3	Razvoj lastne rešitve	19
3.1	Načrtovane funkcionalnosti sistema	19
3.1.1	Aplikacije	20
3.1.2	Web Api	20
3.1.3	Pregledovanje sej in uporabnikov aplikacije	20
3.1.4	Izris pridobljenih podatkov	21
3.2	Uporabljene tehnologije	21

3.2.1	ASP.NET MVC 5	21
3.2.2	Entity Framework	23
3.2.3	Javascript	25
3.2.4	jQuery	27
3.2.5	Ajax	27
3.2.6	Web API 2	28
3.2.7	Chart.js	29
3.3	Struktura podatkovne baze	30
3.3.1	Application	31
3.3.2	Code	31
3.3.3	ApplicationCustomField	32
3.3.4	ApplicationUserField	32
3.3.5	Session	33
3.3.6	SessionCustomField	33
3.3.7	SessionData	34
3.3.8	User	35
3.3.9	UserFieldData	35
3.4	Expression Manager	36
3.4.1	Primer skripte	36
3.5	Kontrolerji	37
3.5.1	ApplicationController	37
3.5.2	SessionsController	37
3.5.3	UsersController	37
3.5.4	ChartsController	38
3.6	Web API	39
3.6.1	Začetek seje	39
3.6.2	Dodajanje dogodkov seji	40
3.7	Opisi pripomočkov	40
3.7.1	Struktura aplikacije	41
3.7.2	Pregledovalnik sej in uporabnikov	43

KAZALO

3.7.3	Izdelava grafov	44
3.7.4	Izdelava lijakastih grafov	45
4	Preizkus sistema in rezultati	47
4.1	Opis aplikacije	47
4.2	Sestava in seznam dogodkov	49
4.3	Rezultati	51
5	Zaključek	57
	Seznam slik	58
	Seznam tabel	60
	Seznam izsekov izvirne kode	62

Povzetek

Problem pridobivanja podatkov o uporabnikih aplikacij je prisoten že od začetka obstoja osebnih računalnikov. Čeprav so v preteklosti pridobivali podatke s pomočjo anket in analiz, so se razvile nove metode. Predstavili bomo, zakaj so uspešnejše. Prikaz delovanja bomo prikazali z analizo uporabe in delovanja dveh trenutno najpopularnejših servisov – Google Analize ter Flurry Analytics. Med analizo trenutnih rešitev smo odkrili nekaj njihovih slabosti. S pomočjo razvoja našega lastnega sistema smo te slabosti odpravili. Za razvoj našega sistema smo uporabili več naprednih tehnologij, ki so omogočile enostavnejši nadaljnji razvoj sistema ter njegovo skalabilnost. Naš sistem smo preizkusili tudi na testni aplikaciji. S pomočjo zbranih podatkov smo podali predloge za njeno izboljšavo. Končni rezultat dela je celovit sistem za zbiranje in obdelavo podatkov o uporabnikih katerekoli aplikacije.

Ključne besede:

analitika, ASP.NET MVC 5, zbiranje podatkov, Google Analize, Flurry Analytics, Web API

Abstract

The problem of obtaining data on the users of the applications has been present from the beginning of the existence of personal computers.

In the past, the information was gained mainly by means of surveys and analyses, but now new methods have been developed. We will present why new methods perform better. Demonstration of the new methods will be presented with analysis of the use and operation of the current most popular services – Google Analytics and Flurry Analytics.

During the analysis of the current solution, we found some weaknesses of the current solutions. However, with developing our own system, we can eliminate these weaknesses.

To develop our own system, we used more advanced technologies which enabled us to develop a simpler system and its scalability.

Our system was tested on a test application. With the data collected, we made proposals for its improvement. Then final result of the work is a comprehensive system for collecting and processing of the data on users of any application.

Key words:

analytics, ASP.NET MVC 5, collecting data, Google Analytics, Flurry Analytics, Web API

Seznam uporabljenih kratic in simbolov

MVC Model View Controller je programski model za uporabniški vmesnik, ki razdeli programsko opremo v tri povezane dele, ki ločijo logiko pridobivanja podatkov, obdelavo podatkov in prikaz podatkov.

ASP.NET Active Server Pages v .NET je tehnologija za razvoj spletnih aplikacij v ogrodju .NET.

EF Entity Framework je ogrodje, ki omogoča povezavo s podatkovno bazo v ogrodju .NET.

IIS Internet Information Services je programska oprema podjetja Microsoft, ki gosti spletne aplikacije.

AJAX Asynchronous JavaScript and XML je tehnologija, ki omogoča dinamično pošiljanje zahtev na strežnik in prikaz rezultatov brez osveževanja spletne strani.

JSON JavaScript Object Notation je format za prenos podatkov. V modernem spletu se uporablja zaradi svoje preproste sintakse in majhne velikosti.

XML Extensible Markup Language je označevalni jezik, ki nam pomaga opisati strukturo nekega objekta in njegovih podatkov. Podobno kot JSON, le da je sintaksa kompleksnejša in proizvede večjo velikost podatkov.

URL Uniform Resource Locator je niz znakov, ki predstavlja unikatno lokacijo na spletu. Sestavljen je iz štirih delov:

1. protokola (`http://`),
2. imena domene (`www.fri.uni-lj.si`),
3. poti datoteke (`/moj_direktorij/dokument.cshtml`) ter
4. podatkovnega niza (`ime=Dejan&priimek=Mesar`).

HTTP Hypertext Transfer Protocol je aplikacijski protokol, ki omogoča komunikacijo za svetovni splet. V grobem omogoča pošiljanje zahtevkov spletnim strežnikom in branje njihovega odziva.

Poglavje 1

Uvod

Živimo v svetu interneta, hitrega razvoja in hitrega mišljenja. Še pred kratkim smo programsko opremo kupovali v za to posebej specializiranih trgovinah ter zanjo bržkone zapravili ogromno denarja. Razvoj računalništva ter računalniške tehnologije, ki je v zadnjih letih potekal zelo pospešeno, je ljudem omogočil lažji dostop do računalnikov ter interneta. Samo v Evropi se je v zadnjih 12. letih število uporabnikov interneta povečalo za skoraj 400% [8]. Hkrati se je močno povečalo tudi število zaposlenih v računalniški industriji. Hitrost razvoja programske opreme se je povečala in dnevno nastajajo nove programske rešitve, s tem pa je konkurenca na tem področju vedno hujša.

Vsakodnevno nastajajo nova podjetja, ki jih poznamo pod imenom start-up podjetja. Gre za zagona podjetja, ki so v stalnem iskanju svojega trga. Zaradi hitrega razvoja ter nenehnega spreminjanja trga so podjetja začela zbirati podatke o načinu uporabe njihovih aplikacij, kar jim je v veliko pomoč pri nadaljnjem razvoju programske opreme, saj lahko tako aplikacije vedno izboljšujejo.

Velik delež teh podjetij uporablja storitve, ki omogočajo enostavno in dokaj splošno zbiranje podatkov o uporabnikih ter njihovo analizo. To so storitve, kot so Google Analize, Flurry Analytics, MixPanel in drugi. Vse storitve so v principu preproste za uporabo in enostavne za vgradnjo v njihovo rešitev,

vendar imajo tudi slabosti, kot so:

1. Slaba prilagoditev ciljni aplikaciji.
2. Namenjene so le mobilnim in spletnim aplikacijam.
3. Ne omogočajo vpogleda v podatke enega uporabnika, ampak podatke združujejo v skupine uporabnikov.
4. Podatki, ki jih pridobimo, niso v lasti našega podjetja.

Omenjene storitve s pomočjo agenta v brskalniku ali mobilni aplikaciji ugotovijo osnovne informacije o napravi (operacijski sistem, brskalnik, država, velikost zaslona, itd.), informacije o zadrževanju na strani oz. mobilni aplikaciji ter informacije o zgodovini obiskanih strani (od kje je uporabnik prišel, kam je šel, itd.). Nato jih pošljejo v obdelavo v oblak storitve, kjer se analizirajo in zgenerirajo poročila. Ta se lahko nato pregleduje v brskalniku.

Razlogov za razvoj lastne rešitve je veliko. Vse trenutne rešitve si v pogojih uporabe omogočijo shranjevanje in pregledovanje vseh podatkov, ki so jim poslana v obdelavo. Podatke si sicer lahko tudi prenesemo na naš strežnik, vendar dobimo takšne, ki so agregirani - združeni v skupine uporabnikov, ki so si med seboj podobni. Če bi lahko podatke pregledovali na osnovi enega uporabnika, bi lahko odkrili veliko več o tem, kako uporabniki uporabljajo našo aplikacijo. Poleg tega imamo tudi možnost odkrivanja napak v aplikaciji in hitro pomoč uporabnikom, saj lahko z analitiko in sledenjem odkrijemo, kako točno določen uporabnik uporablja aplikacijo in mu na podlagi tega takoj svetujemo.

Glavna prednost naše rešitve, ki smo jo razvili, je zagotavljanje zasebnosti podatkov. Podatki so v lasti podjetja in se ne posredujejo tretjim osebam in podjetjem. Druga prednost naše rešitve je prilagodljivost. Z ustrezno konfiguracijo lahko dosežemo, da se zbirajo točno tisti podatki, ki jih želimo zbirati in s tem zmanjšamo količino nepotrebnih. Pomembna prednost predlagane

rešitve je tudi splošnost (generičnost), saj lahko podatke, ki jih zbiramo, popolnoma posplošimo. Lahko dodajamo dodatne attribute in dodatne vrednosti ter analitiko gradimo natančno.

V nadaljevanju si bomo najprej ogledali, kako delujejo storitve za zbiranje podatkov o uporabnikih. Ogledali si bomo, kako te storitve pridobivajo podatke iz namiznih, mobilnih in spletnih aplikacij, ter kako jih obdelujejo in prikažejo rezultate. Sledil bo podrobnejši pregled njihovih prednosti in slabosti. V naslednjem poglavju bo sledil predlog naše rešitve, ki bo obsegal strukturo sistema, uporabljene tehnologije in opis pripomočkov, ki jih sistem omogoča. V zadnjem poglavju bomo preverili učinkovitost naše rešitve na realnem primeru aplikacije, ter kako je podjetje s pomočjo naše rešitve uspelo aplikacijo prilagoditi uporabnikom ter si s tem povečali uporabnost aplikacije in s tem tudi pridobili nove uporabnike.

Poglavje 2

Zbiranje podatkov o uporabi računalniških aplikacij

Ko kupci programske opreme še niso imeli interneta in so se morala podjetja odločiti, kako nadaljevati z razvojem njihove rešitve, so vsako leto ali četrtoletje naredili analizo njihovih uporabnikov z uporabo telefonskih ali papirnatih anket. Izdelali so jih z namenom pridobivanja odziva na njihovo rešitev. Stranke so take ankete rešile ter jih nato poslale podjetju. Pogosto so bile tako stranke deležne tudi ugodnosti ob nakupu nove verzije programske opreme. Podjetje pa je tako pridobilo konkretne podatke o tem, kaj stranke potrebujejo in kako izboljšati programsko opremo. Slabost tega načina je bila v tem, da so podjetja na te rezultate lahko čakala zelo dolgo, posledično je razvoj take programske opreme potekal počasi.

Široka uporaba internetnih priključkov je povzročila razvoj novih načinov pridobivanja teh podatkov. Podjetja so začela v svojo programsko opremo vgrajevati zbiranje podatkov o uporabnikih. Tudi mobilne aplikacije so v veliki meri hotele imeti čimveč podatkov o uporabnikih, zato se je ta trend uveljavil tudi tukaj.

Možnost enostavnega zbiranja podatkov o uporabnikih je imela posledice tudi na razvoj programske opreme. Podjetja so hitreje pridobivala podatke

o uporabnosti aplikacije in jo hitreje prilagajala ciljnim uporabnikom. To je koristno tako za podjetje, kot za uporabnike. Podjetja niso zapravljala časa in denarja z anketami ter nadaljnjim razvojem, uporabniki pa so dobili kvalitetnejše programske rešitve.

V nadaljevanju si bomo podrobneje pogledali, kako delujejo modernejši načini zbiranja podatkov o uporabnikih. Začeli bomo z analizo zbiranja podatkov za spletne aplikacije, mobilne aplikacije ter nato še namizne aplikacije. Sledila bo podrobnejša analiza delovanja Google Analize ter Flurry Analytics.

2.1 Zbiranje podatkov obiska spletnih strani ali spletnih aplikacij

Za zbiranje podatkov obiska spletne strani ali aplikacije uporabljamo dve metodi. Najstarejša je metoda lokalnega zbiranja podatkov. Druga metoda pa je oddaljeno zbiranje podatkov. Oddaljeno zbiranje podatkov je trenutno najbolj uporabljena metoda.

2.1.1 Lokalno zbiranje podatkov

Metoda lokalnega zbiranja podatkov je bila popularna, ko spletni brskalniki še niso imeli vgrajenega JavaScripta, ter niso še bili tako napredni, da bi omogočali dinamično nalaganje spletnih strani. S to metodo analiziramo dnevnik spletnega strežnika, ki vsebuje podatke o dostopih spletnih strani. Dnevnik spletnega strežnika je shranjen na vseh spletnih strežnikih in omogoča upravljalcem spletnih strežnikov poganjanje specializiranih programov, ki opravijo analizo nad podatki. Iz teh informacij lahko pridobimo osnovne podatke o naših obiskovalcih:

- Njihov IP naslov.
- Opis njihovega sistema in brskalnika.

- Čas obiska.
- Katero spletno stran je zahteval.

S pomočjo IP naslova lahko odkrijemo tudi okvirno lokacijo uporabnika. Z nekoliko več truda lahko tudi ugotovimo, katere spletne strani ter v kakšnem zaporedju si jih je uporabnik ogledal.

2.1.2 Oddaljeno zbiranje podatkov

Za razliko od prejšnje metode, ki zbira podatke lokalno, jih ta metoda zbira s pomočjo agenta, ki se zažene v uporabnikovem brskalniku. Ta agent nato pošlje podatke strežniku, ki jih obdela. Agent na uporabnikov računalnik naloži tudi nekaj piškotkov, ki ponavadi vsebujejo:

- uporabnikov identifikator,
- čas zadnjega obiska strani,
- od kje je uporabnik prišel.

S to metodo lahko podjetje pridobi več podatkov o uporabnikih, vendar so ti podatki manj zanesljivi, zato jim ne smemo popolnoma zaupati. Pokažejo nam lahko trend, ne pa točnih vrednosti. Točnost teh podatkov se izboljšuje z višanjem odstotka brskalnikov, ki imajo vključene piškotke in JavaScript.

Najbolj aktiven servis na tem področju je trenutno Google Analize, ki na preprost način omogoča zbiranje podatkov obiskovalcev ter uporabnikov mobilne aplikacije. Delovanje Google Analiz si bomo podrobneje pogledali v poglavju "Delovanje in uporaba Google Analiz".

2.2 Zbiranje podatkov v mobilnih aplikacijah

Čeprav Google Analize omogočajo tudi zbiranje podatkov uporabnikov mobilnih aplikacij, bi tukaj bilo bolje omeniti njegovega tekmeca - Flurry Analytics

(v nadaljevanju Flurry) [11]. Flurry je podjetje, ki ponuja servis za zbiranje podatkov o uporabnikih mobilnih aplikacij. Podpira platforme Android, iOS, Windows Phone ter Blackberry in ima v svojem sistemu že 1,2 milijarde uporabnikov aplikacij. V tem poglavju si bomo ogledali, kako deluje Flurry in podobne storitve. Implementacijo Flurry rešitve bomo pokazali v naslednjih poglavjih.

Flurry deluje na podoben način kot Google Analitika za spletne aplikacije. V aplikacijo razvijalci mobilnih aplikacij vgradijo agenta (knjižnico), ki ob zagonu aplikacije zbere osnovne podatke o uporabniku in jih pošlje na strežnik Flurry-ja v obdelavo. Obdelani podatki se nato prikažejo v nadzorni plošči.

Razvijalci lahko dodajajo tudi cilje. To so dogodki, ki se zgodijo takrat, ko je uporabnik opravil določeno nalogo. Pogost cilj je dogodek, ko uporabnik kupi produkt podjetja. Razvijalci morajo v svoji izvorni kodi točno določiti kdaj se je ta cilj zgodil in s pomočjo Flurry knjižnice ta podatek poslati Flurry-ju v obdelavo.

Flurry ponuja tudi sledenje in zbiranje podatkov o sesutju mobilnih aplikacij. Ko se zgodi nepričakovana napaka, jo Flurry knjižnica zajame in jo pošlje na strežnik, kjer se obdelava. Tako razvijalci lahko pregledujejo, koliko uporabnikom se je aplikacija sesula ter katera napaka je najbolj pogosta. Flurry razvijalcem omogoča pregled nad tem, kje v programski kodi se je napaka zgodila.

Tako Flurry kot Google ne dovoljujeta sledenja in točne identifikacije uporabnikov.

2.3 Zbiranje podatkov v namiznih aplikacijah

Sledenje in zbiranje podatkov o uporabnikih namiznih aplikacij deluje na enak način kot pri mobilnih aplikacijah. Potrebno je omeniti, da so storitve, ki omogočajo zbiranje podatkov iz namiznih aplikacij, manj popularne kot storitve, ki omogočajo zbiranje podatkov o uporabnikih iz mobilnih aplikacij ali

spletnih aplikacij.

Zanimivo je, da se namizne aplikacije še vedno uporabljajo in se najverjetneje še nekaj časa bodo. Zakaj torej ti servisi nočejo razširiti svoje ponudbe še na namizne aplikacije? Eden izmed razlogov je gotovo ta, da iz namiznih računalnikov ne morejo izluščiti toliko osebnih podatkov, kot jih lahko iz mobilnih aplikacij. Vendar pa to ne drži tudi za spletne aplikacije. Glavni razlog je verjetno v tem, da namizne aplikacije za uporabo ne potrebujejo internetne povezave, kar pomeni, da se lahko zbrani podatki izgubijo in jih ne moremo obdelati.

Naslednji velik problem je tudi količina programskih jezikov, ki se uporabljajo za razvoj namiznih aplikacij. Ko govorimo o mobilnih aplikacijah, govorimo o razvoju v naslednjih programskih jezikih: Java, Objective-C, C#, C++. Ko razmišljamo o razvoju namiznih aplikacij, pa lahko govorimo še o mnogih drugih programskih jezikih: C, Python, Ruby in mnogi drugi). Veliko teh programskih jezikov lahko uporabimo tudi za razvoj spletnih aplikacij.

2.4 Delovanje in uporaba Google Analiz

Google je podjetje, ki je ustvarilo trenutno največji spletni iskalnik. Iz garažnega podjetja je zraslo v eno največjih podjetij na celem svetu. Novembra 2005 so odprli prve prijave v sistem imenovan Google Analize [1], vendar so ga teden dni zatem tudi zaprli, saj niso zmogli obdelati tolikšne količine podatkov kot so jih dobili. Kasneje so možnost uporabe sistema dobili le izbranci. Dokončno se je storitev odprla avgusta 2006 in je še danes na voljo popolnoma brezplačno.

Za beleženje obiska spletnih strani uporablja modernejšo metodo (z uporabo agenta, ki se zažene v brskalniku obiskovalca). Podatke, ki jih dobi od agenta, shranjuje na Googlovih strežnikih ter jih tam tudi obdeluje. Podatki se v povprečju obdelujejo od 4 do 24 ur in so nato na voljo v nadzorni plošči na spletni strani Googlove Analitike.

Implementacija Googlove Analize je zelo preprosta:

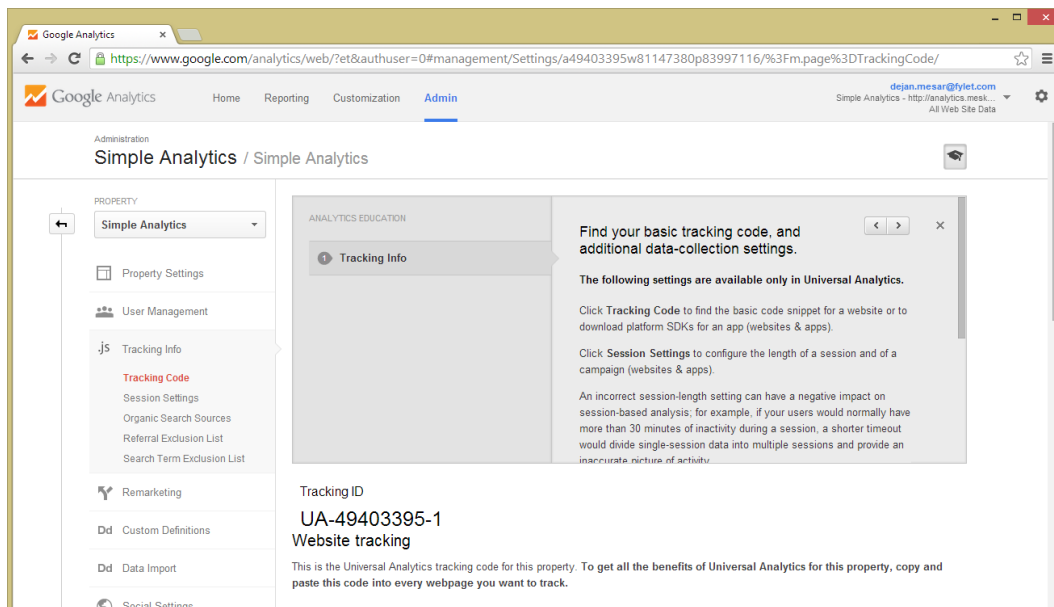
1. Ustvarimo si Google račun, (če ga še nimamo).
2. Prijavimo se v Google Analize na naslovu:
<http://www.google.com/analytics/>.
3. Izpolnimo podatke o naši prvi spletni strani ali mobilni aplikaciji (kasneje lahko dodamo še več spletnih ali mobilnih aplikacij).
4. Pojavi se nam stran (Slika 2.1), ki nam posreduje kodo, s katero lahko zbiramo podatke.
5. Kodo prilepimo na svojo spletno stran tako, da v HTML prilepimo JavaScript kodo (Izvorna koda 2.1).
6. Kodo moramo prilepiti v vsak HTML dokument.

```
<script>
  (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]
    ||function(){
      (i[r].q=i[r].q||[]).push(arguments)};i[r].l=1*new Date();a=s.
      createElement(o),
      m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.
      insertBefore(a,m)
    })(window,document,'script','//www.google-analytics.com/
      analytics.js','ga');

  ga('create','UA-3878562-6','sloserver.eu');
  ga('send','pageview');
</script>
```

Izvorna koda 2.1: JavaScript koda za implementacijo Google Analize v našo spletno aplikacijo

Google Analize niso le enostavno zbiranje podatkov z vgraditvijo JavaScript kode (Izvorna koda 2.1), temveč ponujajo tudi bolj napredne možnosti. Podjetje lahko določi pot, ki jo morajo opraviti obiskovalci, da podjetju prinesejo



Slika 2.1: Stran, ki se prikaže ob vnosu prve aplikacije za sledenje v Google Analizah

nek dobiček (na primer uporabnik se prijavi na novosti z uporabo e-poštnega naslova). Tako lahko podjetje ugotovi, kje se uporabniki izgubijo na začrtani poti. Definirajmo naslednje korake:

1. Uporabnik vstopi na vstopno stran.
2. Uporabnik vidi oglasno sporočilo na vstopni strani, klikne nanj in se premakne na stran, kjer lahko vpiše e-poštni naslov ter se tako naroči na novice.
3. Uporabnik vpiše e-poštni naslov in se s tem naroči na novice.

S primerno nastavitvijo Google Analize lahko podjetje ugotovi, kolikšen delež obiskovalcev spletne strani je prišel na spletno stran oglasnega sporočila, ter koliko izmed njih se je odločilo in se naročilo na novice. Podjetje si seveda želi, da bi se čimvečji delež obiskovalcev tudi naročil na novice. S pomočjo te

analitike lahko preizkušajo različne možnosti in opazujejo, kako se bo delež teh uporabnikov spreminjal. Tako lahko skozi čas izboljšujejo delež teh uporabnikov.

Čeprav so Google Analize brezplačno orodje, ima tudi nekaj omejitev. Ena izmed njih je količina shranjenih podatkov na strežniku (25.000 vrstic podatkov). Prav tako ne zagotavljajo, da bo servis deloval vedno, ampak imajo prednost tisti, ki za to storitev plačujejo. Brezplačna uporaba tako prinese tudi netočnost podatkov. Če torej naš promet preseže 25.000 podatkov, bi veljalo razmisliti o plačljivi verziji storitve.

Vedeti moramo tudi, da Google Analize izrecno prepovedujejo kakršnokoli poskušanje točne identifikacije uporabnika. To je prepovedano z zakonom in bi takšno sledenje potrebovalo dovoljenje uporabnikov.

2.5 Delovanje in uporaba Flurry Analytics

Flurry je podjetje, ki se ukvarja z zbiranjem in analizo zbranih podatkov iz mobilnih naprav. Na trg je prišel leta 2008 in od takrat se je podjetje razvilo v vodilno podjetje na svojem področju. Mesečno obvladujejo že preko 1,2 milijarde mobilnih naprav ter sodelujejo že z 125 000 podjetji. Število aplikacij, iz katerih zbirajo podatke, pa že presegla mejo 400 000.

Tako kot Google Analize tudi Flurry uporablja agenta za pridobivanje podatkov o napravah. Agent lahko deluje v vseh večjih mobilnih operacijskih sistemih:

- Android
- iOS
- Windows Phone
- Blackberry

Implementacija agentov v mobilno aplikacijo ni zahtevno opravilo. Kasneje si bomo ogledali, kako implementiramo Flurry v Android aplikacijo [12].

Agenti v mobilnih aplikacijah ob zagonu zberejo podatke o napravi ter jih čez celotno uporabo aplikacije pošiljajo na strežnik, kjer se nato obdelajo. S pomočjo tako zbranih podatkov lahko v najbolj osnovni konfiguraciji v administratorski plošči pregledujemo naslednje podatke:

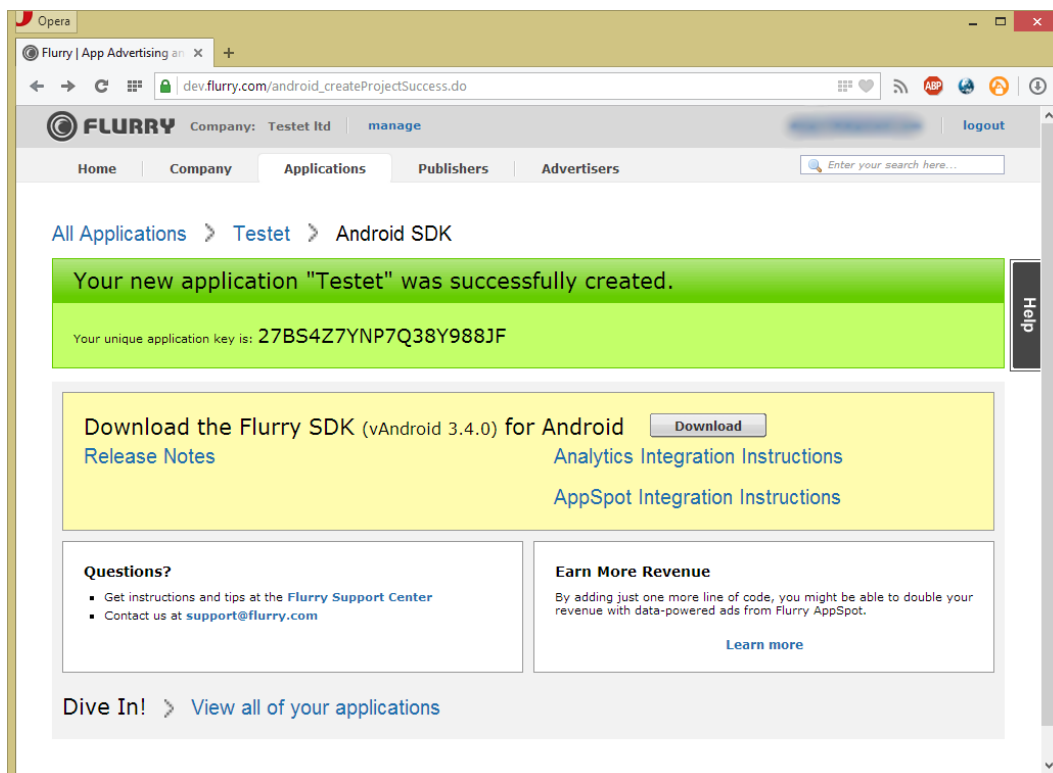
- število sej,
- aktivni uporabniki,
- novi uporabniki,
- dolžino sej (dolžina uporabe aplikacije),
- pogostost uporabljanja aplikacije,
- podatke o hitrosti naše aplikacije,
- kako dobro obdržimo uporabnike,
- podatke o napravi (verzija operacijskega sistema, proizvajalec),
- geografske podatki uporabnikov,
- in drugi.

Da pridobimo ključ, s katerim lahko zbiramo podatke, se moramo najprej registrirati na uradni strani Flurry-ja (<http://www.flurry.com>). Registracija od nas zahteva nekaj osebnih podatkov (ime, priimek, telefonsko številko, email, ime podjetja, e-poštni naslov). Nato izberemo platformo, na kateri je zgrajena naša aplikacija, ter ji dodelimo ime. Prikaže se nam spletna stran (Slika 2.2), ki nam poda prve informacije o implementaciji njihovega agenta v našo aplikacijo.

Za primer implementacije bomo opisali implementacijo agenta za Android aplikacijo. Najprej iz uradne spletne strani prenesemo Flurry SDK paket in

ga dodamo kot knjižnico v našo aplikacijo. Android aplikaciji moramo določiti nekaj dovoljenj, ki Flurry agentu dovoljujejo zbiranje podatkov o napravi ter njihovo pošiljanje na strežnik. Dodati moramo dovoljenja za dostop do interneta ter preverjanje prisotnosti interneta. Primer manifesta naše aplikacije predstavlja izvorna koda 2.2.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/  
    android"  
    package="com.flurry.mediated.admob"  
    android:versionCode="1"  
    android:versionName="1.0" >
```



Slika 2.2: Spletna stran ob prvi prijavi v Flurry

```
<uses-sdk
    android:minSdkVersion="10"
    android:targetSdkVersion="17" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.
    ACCESS_NETWORK_STATE"/>
</manifest>
```

Izvorna koda 2.2: Manifest Android aplikacije

Nazadnje moramo v vsak naš **Activity** razred dodati klice na **onStartSession** in **onEndSession** kot prikazuje izvorna koda 2.3.

```
import com.flurry.android.FlurryAgent;

@Override
protected void onStart()
{
    super.onStart();
    FlurryAgent.onStartSession(this, "YOUR_API_KEY");
}

@Override
protected void onStop()
{
    super.onStop();
    FlurryAgent.onEndSession(this);
}
```

Izvorna koda 2.3: Implementacija Flurry v Android aplikacijo

Zagotoviti moramo, da je ta koda v popolnoma vseh naših razredih, ki razširjajo razred **Activity**, saj le tako lahko zagotovimo točnost zbranih podatkov. Implementacija Flurry analitike v Android aplikacijo je tako končana.

2.6 Slabosti in prednosti trenutnih rešitev

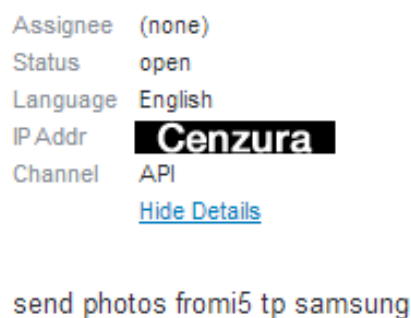
Velika prednost trenutnih rešitev je zagotovo njihova cena. Velik del njihovih storitev je popolnoma brezplačen ne glede na to, ali je storitev namenjena komercialni ali osebni uporabi. Na ta način lahko trenutne rešitve podjetja privadijo na njihov način zbiranja podatkov in prikaz le teh. Ko se podjetjem volumen zbranih podatkov močno poveča (zaradi uspešnosti poslovanja), so primorani svoj brezplačni račun zamenjati za plačljivega, saj le tako lahko še naprej točno zbirajo podatke o svojih uporabnikih.

Naslednja prednost je, da podatke shranjujejo na njihovem strežniku. Kar pomeni, da ima naročnik manjše stroške, saj mu ni potrebno poskrbeti za celotno arhitekturo strežnikov, ki bi prejeli, shranjevali in obdelovali prejete podatke. Toda ta prednost je lahko tudi slabost. Čeprav lahko vse obdelane podatke iz trenutnih rešitev izvozimo in jih nato pregledujemo tudi na službenih sistemih, so podatki dejansko v lasti podjetij, ki so razvile trenutne rešitve. Naročnik privoli v to, ko sprejme pogoje uporabe (ob registraciji v njihov sistem). Torej trenutne rešitve imajo kljub temu, da so brezplačne, dobiček. Vsi podatki so njihovi in jih lahko obdelujejo ter nato prikazujejo statistiko preko vseh naprav, od katerih zbirajo podatke.

Slabost je tudi v tem, da se ti podatki posredujejo naprej tretjim osebam. Novice o NSA (National Security Agency), ki so prišle na plan v letu 2013, so marsikoga pripravile do pomisleka o pošiljanju podatkov tretjim podjetjem. Uporabnikom se je začela vzbuhati skrb, da bodo njihove podatke podjetja posredovala tretjim podjetjem. Na spletu še nismo našli rešitve, ki bi omogočala zbiranje, hrambo in obdelavo podatkov, ki bi jo lahko podjetje enostavno namestilo na svoje strežnike. Tako podatkov o svojih uporabnikih ne bi posredovali tretji osebi. To bi po našem mnenju bila zelo dobra poslovna prednost.

Veliko število trenutnih rešitev omogoča zbiranje podatkov o številu uporabnikov, ki so opravili neko nalogo in dosegli pričakovani cilj (nakup, registra-

cija in podobno). Ne omogočajo pa povezave teh dogodkov na nivoju posameznega uporabnika, kar pomeni, da oddelek za pomoč uporabnikom ne zna ugotoviti, kako je uporabnik uporabljal aplikacijo. Uporabniki so velikokrat manj računalniško pismeni in težko opišejo problem, ki ga imajo z neko aplikacijo. Če ima podjetje vpogled v podatke o rabi programa določenega uporabnika, mu lahko s tem lažje in hitreje pomagajo. Tako uporabniku ostane pozitiven vtis o programu in podjetju, ki stoji za njim. Primer takega uporabnika lahko vidimo na sliki 2.3. Slika je vzeta iz sistema za pomoč uporabnikom. Nadaljna raziskava problema je pokazala, da uporabnik nima potrebnega operacijskega sistema za zagon našega programa. Pomankljivost je, da trenutne rešitve tega ne omogočajo.



Slika 2.3: Uporabniki so zelo skopi z informacijami o napakah

Čeprav za sledenje potrebujemo uporabnikovo dovoljenje, lahko takšno zbiranje podatkov pripomore k boljši pomoči uporabnikom. Podjetja potrebujejo manj časa za odgovor, uporabniki pa dobijo kvalitetnejši odgovor z veliko verjetnostjo rešitve njihovega problema že v prvem odgovoru. Podjetje lahko s takšnim zbiranjem podatkov podrobneje izlušči njihove prave uporabnike in ugotovi, kako je aplikacija dejansko uporabljena. To posledično pomeni, da se aplikacija bolj nagiba k potrebam uporabnika.

Poglavje 3

Razvoj lastne rešitve

Cilj diplomske naloge je sestaviti generičen sistem, ki lahko zbira podatke iz katerekoli platforme. Sistem mora omogočati tudi izdelavo grafov glede na podatke, ki si jih želimo ogledati. Omogočiti mora enostavno pošiljanje podatkov iz aplikacije na strežnik s pomočjo HTTP POST zahtevka. Implementacija pošiljanja preprostih HTTP POST zahtevkov ne bi smela biti težavna v nobenem programskem jeziku. Naša rešitev mora biti takšna, da jo lahko namestimo na lastni spletni strežnik. Podjetjem tako ni potrebno posredovati podatkov o uporabnikih tretjim osebam.

Ker hočemo lastno rešitev zgraditi na principu generičnosti, mora biti zelo prilagodljiva. To je še posebej pomembno za zagonska podjetja, ki svoje rešitve neprestano spreminjajo in izboljšujejo. S pomočjo lastne rešitve se lahko zbiranje podatkov hitro spreminja in prilagaja aplikaciji. Poleg tega lahko zaradi generičnosti uspemo na podlagi izsledkov diplomske naloge zgraditi tudi tako zahtevne sisteme kot, sta Flurry Analytics in Google Analize.

3.1 Načrtovane funkcionalnosti sistema

Ker gradimo generičen sistem, moramo poskrbeti za vse osnovne funkcionalnosti, ki bodo omogočale tudi nadgradnjo sistema za bolj napredne rešitve.

3.1.1 Aplikacije

Primarno želimo pridobivati podatke o aplikacijah (namiznih, mobilnih ali spletnih). Zato je smotrno, da omogočamo dodajanje aplikacij, njihov opis in identifikator. Aplikacija ima lahko več podatkovnih polj oziroma polj, ki shranjujejo podatke za sejo. K vsaki aplikaciji spadajo tudi uporabniki. Ti imajo lahko tudi določene lastnosti:

- spol,
- seznam lokacij,
- in druge (seznam naprav, seznam prijateljev, število prenesenih slik).

Poleg sej bi radi sledili tudi kdaj so uporabniki uporabili neko funkcionalnost programa. Zato mora aplikacija vzdrževati tudi seznam vseh dogodkov, ki bi jim radi sledili. Dogodki imajo določen tudi unikatni identifikator, ki se ga uporabi, ko se dogodek v programu zgodi. Vsakemu dogodku lahko ob pošiljanju na strežnik dodamo vrednosti, ki nas zanimajo. Recimo, možnemu dogodku “Start aplikacije” lahko dodamo vrednost, ki nam predstavlja unikatni identifikator uporabnika. Dogodkom je mogoče dodati tudi skripto, ki omogoča spreminjanje podatkov seje in uporabnika.

3.1.2 Web Api

Web Api mora delovati kot enostaven vmesnik, ki omogoča, da iz kateregakoli programskega jezika začnemo novo sejo ter ji dodajamo dogodke tekom izvajanja aplikacije. Med dodajanjem dogodka se mora izvesti tudi skripta dogodka, če tako določimo v nastavitvah le-tega.

3.1.3 Pregledovanje sej in uporabnikov aplikacije

Zaradi uporabnosti podatkov pri pomoči uporabnikom želimo imeti možnost pregledovanja sej in uporabnikov aplikacije.

3.1.4 Izris pridobljenih podatkov

Pri veliki količini podatkov je pomembno tudi to, kako so ti prikazani. Zaradi tega želimo, da ima stranka možnost izrisa grafov glede na njene filtre in lahko tako vidi točno tiste podatke, ki jo zanimajo. Želimo ji ponuditi možnost prikaza števila sej glede na čas ter prikaz podatkov glede na podatkovna polja aplikacije.

Stranke bi si želele tudi lijakasti graf, ki prikazuje, kako uporabniki napredujejo proti glavnemu cilju in nam prikazuje, koliko uporabnikov se izgubi v posameznem koraku do zastavljenega cilja.

3.2 Uporabljene tehnologije

Zaradi generičnosti in možnosti razširjanja storitve smo morali razmišljati tudi, katere tehnologije bomo uporabili. Odločili smo se za ogrodje ASP.NET MVC 5 [10], ki omogoča hiter razvoj storitve ter možnosti nadgrajevanja sistema. Vgrajenih ima že nekaj drugih tehnologij, ki si jih bomo v tem sklopu podrobneje ogledali. Ker ASP.NET MVC 5 ne vsebuje tehnologije, ki bi omogočala izrisovanje grafov, smo se odločili, da za to uporabimo zunanjo JavaScript knjižnico Chart.js, ki omogoča vektorsko izrisovanje grafov.

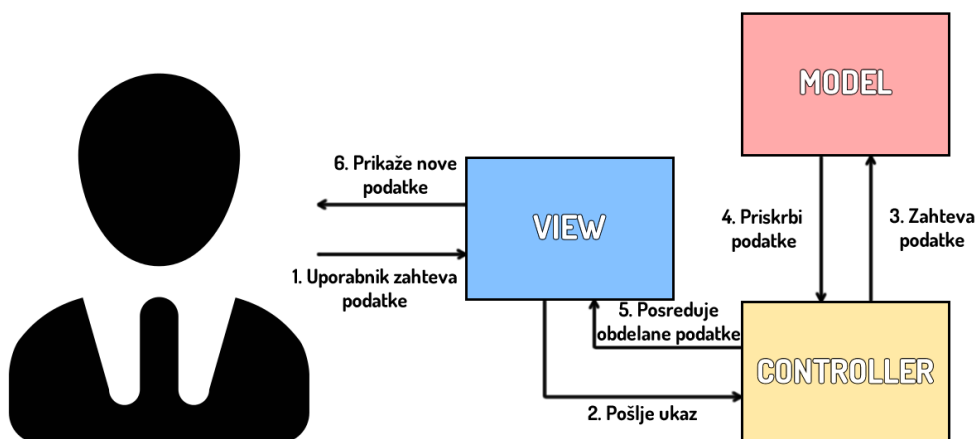
3.2.1 ASP.NET MVC 5

ASP.NET MVC je programsko ogrodje podjetja Microsoft, ki omogoča izdelavo spletnih aplikacij, ki vgrajujejo koncepte in ideje modela MVC (Model-View-Controller) v ogrodje ASP.NET.

Ogrodje ASP.NET je bilo prvič izdano leta 2002 in je delovalo po povsem drugačnem principu kot takratna ogrodja in programski jeziki za izdelavo spletnih aplikacij. ASP.NET je poskušalo spletne aplikacije čimbolj približati namiznim aplikacijam. To jim je takrat uspelo, saj so bile spletne aplikacije na las podobne namiznim aplikacijam, vendar je to prineslo drug problem.

ASP.NET je postal zelo zahteven in neprilagodljiv ter tako ni mogel več slediti smernicam modernega spleta. Prilagoditve so bile sicer možne, vendar so predstavljale zahtevno in mučno opravilo, zato so se razvijalci raje posluževali drugih spletnih ogrodij in programskih jezikov.

Microsoft je torej moral stopiti v korak s časom in idejo opustiti. Leta 2009 je izdal prvo verzijo ogrodja ASP.NET MVC 1.0, ki je bila zgrajena z namenom hitre, učinkovite in prilagodljive izdelave spletnih aplikacij po programskem modelu MVC. Začetki modela MVC segajo v leto 1979. Razvijalci so ta model dobro sprejeli, saj se razvoj spletnih aplikacij v njih najbolj obnese. Prav tako pa je model uporaben tudi v namiznih in mobilnih aplikacijah [2].



Slika 3.1: Prikaz obdelave uporabnikove zahteve po modelu MVC

Model MVC loči uporabniški vmesnik v tri glavne dele:

Model vsebuje seznam objektov, ki opisujejo podatke. Prav tako nakazuje, kako se podatki lahko dodajajo, spreminjajo.

View ali Pogled definira, kako se bo uporabniški vmesnik izrisal uporabniku.

Controller ali Kontroler je seznam razredov, ki procesirajo uporabnikove zahteve. Glede na zahtevo mu kontroler nato posreduje tudi zahtevan pogled z rezultati njegove zahteve.

3.2.2 Entity Framework

Entity Framework (EF) [5] je ogrodje, ki omogoča preslikavo objektov .NET v podatke, ki jih lahko shranjuje podatkovna baza. EF je privzeto takoj na voljo v ASP.NET in omogoča hitro vzpostavitev podatkovne baze. Omogoča več načinov vzpostavitve modela.

Prvi način je imenovan “Code First” ali “Najprej kodiranje”. Pri tem načinu začnemo s pisanjem razredov, ki vsebujejo opise podatkov, ki bi jih radi shranili v podatkovni bazi. EF nato s pomočjo teh razredov sam ustvari potrebne tabele in instance razredov tako, da je možno takojšnje pridobivanje in zapisovanje podatkov iz tega podatkovnega vira. Ta način omogoča hiter razvoj in enostavno prilagodljivost, saj nam podatkovne baze ni potrebno vnaprej izdelati v programu za upravljanje s podatkovnimi bazami. Ta način je učinkovit v začetku razvijanja spletne aplikacije, saj lahko tako na začetku razvoja enostavneje spreminjamo model.

Drugi način je imenovan “Schema First” ali “Najprej shema”. Uporaben je, ko že imamo postavljeno strukturo podatkovne baze. EF nam v tem primeru omogoča avtomatično izdelavo razredov modela po strukturi podatkovne baze.

Za našo rešitev smo izbrali pristop “Code First”. Oglejmo si še nekoliko podrobneje, kaj nam Entity Framework omogoča.

Recimo, da bi želeli v tabeli “Aplikacije” shranjevati vse objekte tipa **Aplikacija**. Za ta namen bi spisali naslednji razred:

```
public class Aplikacija
{
    public int Id { set; get; }
    public int Ime { set; get; }
    public int Opis { set; get; }
    public virtual Uporabnik Lastnik { set; get; }
}
```

Izvorna koda 3.1: Razred, ki opisuje vrstico tabele Aplikacije

Razred `Aplikacija` vsebuje za nas trenutno vse potrebne informacije o aplikaciji. Objekt razreda (Izvorna koda 3.1) predstavlja vrstico v tabeli “Aplikacije”. EF je zaradi `Id` atributa ugotovil, da razvijalec s tem misli na primarni ključ in ustrezno ustvari v podatkovni bazi tabelo “Aplikacije” s primarnim ključem `Id` in nastavi, da se za vsak nov dodan objekt tipa `Aplikacija` poveča `Id` atribut za ena.

EF skrbi tudi za tuje ključe. Ko EF iz tabele “Aplikacije” prebere nek objekt tipa `Aplikacija`, gre pogledat tudi v tabelo “Uporabniki”, kjer najde ustrezen objekt `Uporabnik` in ga dodeli ustreznemu atributu objekta.

Za dostop do podatkov posamezne tabele moramo napisati tudi razred, ki bo vseboval strukturo naše podatkovne baze. Ta razred bo razširil razred `DbContext` in bo vseboval vse tabele, do katerih želimo imeti dostop v naši podatkovni bazi. Za naš enostaven primer bi želeli imeti dostopni dve tabeli - “Aplikacije” in “Uporabniki”. Razred izgleda takole:

```
public class PodatkovnaBaza : DbContext
{
    public DbSet<Aplikacija> Aplikacije {set; get;}
    public DbSet<Uporabnik> Uporabniki {set; get;}
}
```

Izvorna koda 3.2: Primer razreda, ki predstavlja podatkovno bazo v Entity Framework

Razred ima dva atributa: `Aplikacije` in `Uporabniki`, ki predstavljata

tabele v podatkovni bazi. Objekt `DbSet` nam doda dodatne možnosti nad tabelama. Omogoča nam, da s tabelami ravnamo tako, kot ravnamo s seznamami objektov v .NET. V tem primeru atributa `Aplikacije` in `Uporabniki` vsebujeta sezname objektov `Aplikacija` in `Uporabnik`. Primer prikazuje naslednja izvorna koda (Izvorna koda 3.3):

```
PodatkovnaBaza baza = new PodatkovnaBaza();  
List<Aplikacije> AplikacijeUporabnika =  
    baza.Where(a => a.Lastnik.Ime == "Ivan").ToList();
```

Izvorna koda 3.3: Primer pridobivanja seznama `Aplikacij`, katerih lastnik je `Uporabnik` z imenom Ivan

3.2.3 Javascript

JavaScript je dinamični programski jezik, ki ga podpirajo skoraj vsi brskalniki. Je najbolj popularen programski jezik, ki omogoča dinamično spreminjanje spletne strani. JavaScript omogoča uporabniku, da kontrolira akcije brskalnika, asinhrono komunicira s spletnimi strežniki ter spreminja vsebino spletne strani.

Prva verzija programskega jezika JavaScript je bila izdana skupaj z brskalnikom Netscape Navigator 2.0 septembra 1995. Microsoft je leta 1996 vgradil JavaScript tudi v svoj spletni brskalnik Internet Explorer 3.0. Leta 1997 je bil JavaScript standardiziran v standardu imenovanem ECMAScript. Danes je to programski jezik, ki ga lahko poganjajo skoraj vsi brskalniki [3].

JavaScript je dinamični programski jezik, kar pomeni, da za določitev spremenljivk ni potrebno določiti točnega tipa spremenljivke. Izvorna koda 3.4 prikazuje, kako lahko spremenljivki `x` najprej priredimo numerično vrednost, nato pa vrednost drugega tipa (recimo niz znakov).

```
var x = 5; // Stevilka  
x = "pet"; // Niz znakov
```

Izvorna koda 3.4: JavaScript je dinamični jezik, zato se tip spremenljivke določi ob prirejanju vrednosti.

Ta dinamika nam pride zelo prav tudi pri JavaScript objektih, saj lahko objektom brez težav dinamično dodajamo attribute in metode. Objekte pa lahko kreiramo tudi tako, da samo naštejemo njihove attribute ter metode. Primer 3.5 prikazuje, kako enostavno lahko avtu določimo nov atribut in novo metodo.

```
var avto = {  
  "lastnik" : "Franc Mlinar",  
  "tip_vozila" : "Enoprostorec",  
  "st_sedezev" : 7,  
  "kdo_je_lastnik" : function() {  
    return this.lastnik;  
  },  
  "prevozeni_kilometri" : [10,0,20,30,12]  
}  
avto.velikost_koles = 16;  
avto.skupaj_kilometrov = function()  
{  
  var vsota = 0;  
  for(var i = 0; i < this.prevozeni_kilometri.length; i++)  
  {  
    vsota += this.prevozeni_kilometri[i];  
  }  
  return vsota;  
}
```

Izvorna koda 3.5: Primer dodajanja atributov in metod objektu

V modernem svetu se ne uporablja več veliko domorodnega (osnovnega) JavaScripta, vendar se uporabljajo različna ogrodja kot so jQuery, AngularJS,

Dojo in drugi. V naši rešitvi smo uporabili ogrodje jQuery.

3.2.4 jQuery

jQuery [4] je programsko ogrodje za JavaScript, ki ga je sestavil John Resig leta 2005. Njegova posebnost je v tem, da zelo olajša iskanje elementov v HTML dokumentu ter dodeljevanje obnašanja tem elementom. Od leta 2005 je ogrodje dobilo mnogo privržencev. Sčasoma so ga tudi optimizirali ter dodali nove funkcionalnosti.

Velika prednost jQuery ogrodja je tudi ta, da je kompatibilen z vsemi popularnimi brskalniki in tako omogoča razvijalcu, da se skoncentrira na dejansko rešitev in ne na reševanje problemov, ki nastanejo med različnimi brskalniki.

```
<input
  type="button"
  id="TestniGumb"
  data-message="Pozdravljen svet!" />

<script type="text/javascript">
$('#TestniGumb').on('click', function()
{
  alert($(this).attr('data-message'));
});
</script>
```

Izvorna koda 3.6: Primer uporabe ogrodja jQuery za dodelitev obnašanja ob kliku na gumb z id-jem `TestniGumb`

3.2.5 Ajax

AJAX (Asynchronous Javascript And XML) je tehnologija, ki omogoča pridobivanje podatkov iz spletnega strežnika in njihovo dinamično prikazovanje brez dodatnega nalaganja celotne spletne strani [7]. Ajax je nastal z namenom približati spletne aplikacije namiznim aplikacijam. Te lahko posodobijo le nek

del svoje aplikacije, ne da bi morali aplikacijo ponovno v celoti izrisati ali pa še huje - ponovno zagnati.

Ajax je strukturiran v JavaScriptu kot set funkcij in objektov, ki zmorejo poslati zahteve na spletni strežnik in prenesti njegov odgovor. Odgovor se nato lahko obdela in posodobi del spletne strani z rezultatom strežnika. Lahko je v večjih formatih (HTML, CSS, JS, JSON in podobni). Najpogosteje se trenutno uporabljata HTML in JSON.

Ajax tehnologija še posebej stremi k temu, da je enostavna za implementacijo. Z osveževanjem le dela spletne strani pripomore k zmanjšanju prometa med uporabnikom in spletnim strežnikom. Zaradi manjšega prometa se izboljša odzivnost spletne strani in s tem tudi uporabniška izkušnja. Spletna stran je tako bolj prijazna do uporabnika.

```
$.ajax({
  type: "POST",
  url: "/Users/Add",
  data: { ime: "Martin", priimek: "Krizman" }
})
.done(function( data ) {
  alert( "Uporabnik shranjen: " + msg.ime + " " + msg.priimek );
});
```

Izvorna koda 3.7: Primer Ajax klica s pomočjo jQuery ogrodja

3.2.6 Web API 2

Ogrodje ASP.NET MVC je bilo narejeno za prikaz spletnih strani v obliki HTML, vendar je sčasoma moderni splet pričel izdelovati spletne strani, ki so pridobivale podatke preko AJAX klicev, ki pa ne vračajo nujno HTML, ampak večinoma XML ali JSON. Zaradi tega je Microsoft v ASP.NET MVC 5 vgradil tehnologijo Web API 2 [6].

Web API omogoča enostavno in hitro odgovarjanje na RESTful zahteve. Za upravljanje teh zahtev se uporablja drugačen kontroler, imenovan

`ApiController`, ki vsebuje metode za odgovarjanje na zahteve. Web API ne podpira Pogledov (View-ov), saj ne vrača HTML-ja, ampak vrača XML, JSON ali gole bite. Zaradi tega je še posebej primeren za pridobivanje podatkov brez ponovnega nalaganja spletne strani ali pa za pridobivanje podatkov iz mobilne aplikacije.

3.2.7 Chart.js

Je knjižnica napisana v JavaScriptu, ki omogoča izdelavo grafov iz podanih podatkov. Avtor knjižnice je Nick Downie [9]. Knjižnica ustvari vektorske grafe, ki so preprosti za uporabo in modernega videza.

Knjižnica od nas zahteva, da ji posredujemo podatke o vodoravni osi (oznake) ter serije podatkov. Za vsako serijo podatkov lahko določimo njen izgled. Primer uporabe knjižnice prikazuje naslednja izvorna koda (Izvorna koda 3.2):

```
var ctx =
  document
    .getElementById("html_element_id")
    .getContext("2d");

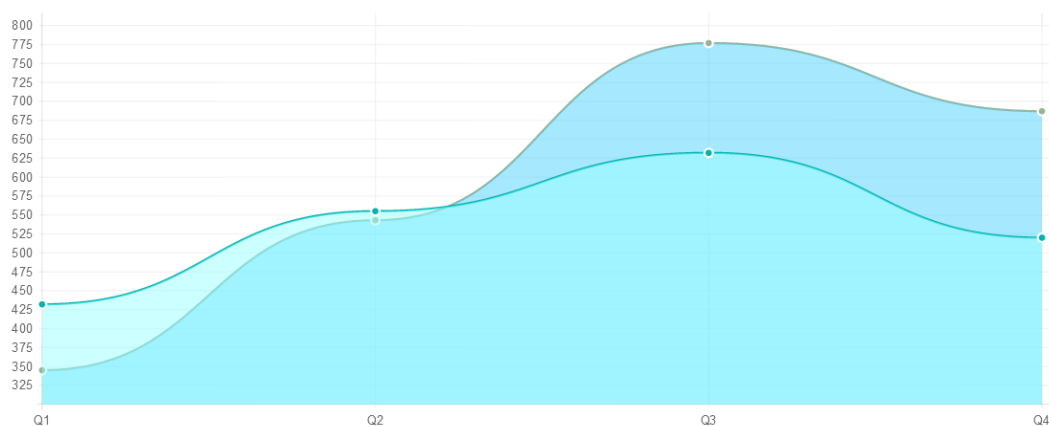
var data = {
  labels: ["Q1", "Q2", "Q3", "Q4"],
  datasets: [
    {
      fillColor : "rgba(77,210,255,0.5)",
      strokeColor : "rgba(153,179,153,1.0)",
      pointColor : "rgba(153,179,153,1.0)",
      pointStrokeColor: "#fff",
      data: [345, 543, 777, 687]
    },
    {
      fillColor : "rgba(153,255,255,0.5)",
      strokeColor : "rgba(0,179,179,1.0)",
      pointColor : "rgba(0,179,179,1.0)",
      pointStrokeColor: "#fff",
```

```
    data: [432, 555, 632, 520]
  }]
};

var options = {
  showTooltips: true,
  bezierCurve: true
};

var chart = new Chart(ctx, {}).Line(data, options);
```

Izvorna koda 3.8: Primer inicializacije preprostega grafa v Chart.js knjižnici, rezultat viden na sliki 3.2



Slika 3.2: Rezultat knjižnice Chart.js

3.3 Struktura podatkovne baze

Strukturo podatkovne baze predstavimo z modelom. Model je skupek razredov, ki predstavljajo tabele v podatkovni bazi. Objekti teh razredov pa so vrstice tabel. Ogledali si bomo razrede, ki sestavljajo podatkovno bazo, in jih opisali.

3.3.1 Application

Razred Application skupaj povezuje celotno aplikacijo. Ima osnovne podatke o dodatnih poljih sej ter uporabnikov, prav tako vsebuje identifikator, s katerim lahko ustvarimo novo sejo preko WebApi-ja. Strukturo in kodo opisuje naslednja tabela 3.1:

atribut	tip atributa	opis
Id	String	Unikaten identifikator
Name	String	Ime aplikacije
Description	String	Opis aplikacije
IsBanned	Bool	Dostopnost aplikacije
Codes	List<Code>	Seznam mogočih dogodkov
CustomFields	List<ApplicationCustomField>	Seznam dodatnih polj sej
UserFields	List<ApplicationUserField>	Dodatna polja uporabnikov

Tabela 3.1: Struktura razreda Application

3.3.2 Code

Razred Code (Tabela 3.2) vsebuje informacije o kodi dogodka, ki jo dodajamo aplikaciji. Koda dogodka vsebuje podatke o aplikaciji, ki vsebuje to kodo, ime, opis in barvo dogodka ter skripto, ki se izvede, če ima `UseExpression` vrednost `true`.

atribut	tip atributa	opis
Id	Integer	Unikaten identifikator
Key	String	Identifikator dogodka
Application	Application	Aplikacija, ki ji dogodek pripada
Name	String	Ime dogodka
Description	String	Opis dogodka
Color	String	Barva dogodka v formatu #FFFFFF
UseExpression	Bool	Določa ali naj dogodek izvede skripto
Expression	String	Skripta

Tabela 3.2: Struktura razreda Code

3.3.3 ApplicationCustomField

Razred `ApplicationCustomField` (Tabela 3.3) vsebuje podatke o dodatnih poljih, ki jih imajo seje neke aplikacije. Vsebuje ime ter tip polja.

atribut	tip atributa	opis
Id	Integer	Unikatni identifikator
Name	String	Ime polja (brez presledkov)
ValueType	FieldValueType	Tip podatka (lahko je String, Integer, Double, DateTime, Bool)
Application	Application	Aplikacija, ki ji pripada polje

Tabela 3.3: Struktura razreda ApplicationCustomField

3.3.4 ApplicationUserField

Razred `ApplicationUserField` oziroma dodatno polje uporabnika vsebuje podatke o dodatnih poljih, ki jih imajo uporabniki neke aplikacije. Vsebuje ime ter tip polja. Struktura tega razreda je identična strukturi razreda

ApplicationCustomFields (Tabela 3.3).

3.3.5 Session

Razred `Session` (Tabela 3.4) vsebuje podatke o posamezni seji neke aplikacije. Vsebuje čas začetka seje, seznam dodatnih polj z njihovimi vrednostmi (glej razred `SessionCustomField`) ter uporabnika, na katerega je seja vezana.

atribut	tip atributa	opis
Id	Integer	Unikatni identifikator
Sid	String	Unikatni identifikator v aplikaciji
Application	Application	Aplikacija, ki ji pripada seja
Time	DateTime	Čas vzpostavitve seje
CustomFields	List<SessionCustomField>	Podatki seje
UserId	String	Id Uporabnika aplikacije

Tabela 3.4: Struktura razreda `Session`

3.3.6 SessionCustomField

Razred `SessionCustomField` je zelo pomemben (Tabela 3.5), saj shranjuje podatek nekega dodatnega polja seje in je zato pomemben del našega sistema. Ima možnost shranjevanja večih tipov podatkov:

- naravno število,
- realno število,
- DA ali NE,
- niz znakov,

- datum in čas.

Zaradi enostavnosti smo razredu dodali dodaten atribut **Value**, ki poda podani podatek v pravo podatkovno polje znotraj vrstice. Tako smo lahko vedno prepričani, da bo podatek shranjen v točno takem formatu, kot smo si zamislili.

atribut	tip atributa	opis
Id	Integer	Unikaten identifikator
Session	Session	Seja, ki vsebuje ta podatek
ACF	ApplicationCustomField	Objekt ApplicationCustomField, ki vsebuje podatke o strukturi podatka
Value	Object	Atribut, ki skrbi za pretvorbo vseh vrst podatkov za shranjevanje v podatkovni bazi

Tabela 3.5: Struktura razreda SessionCustomField

3.3.7 SessionData

Razred **SessionData** (Tabela 3.6) vsebuje podatke o dogodku, ki smo ga dobili iz naše aplikacije preko Web API-ja. Vsebuje kodo dogodka in sejo, ki ji pripada, čas, ko smo dogodek prejeli ter vrednost, ki smo jo poslali preko zahtevka na WebApi.

atribut	tip atributa	opis
Id	Integer	Unikaten identifikator
Code	Code	Objekt informacij o dogodku
Session	Session	Seja, ki ji pripada ta dogodek
Time	DateTime	Čas dogodka, ko je bil dodan seji
Value	String	Podatek, ki je bil podan ob podanem dogodku

Tabela 3.6: Struktura razreda SessionData

3.3.8 User

Razred **User** (Tabela 3.7) predstavlja tabelo, ki vsebuje podatke o uporabnikih. Objekt tega razreda ima shranjeno aplikacijo, ki ji pripada, unikatni identifikator uporabnika ter seznam vseh vrednosti dodatnih polj uporabnika za določeno sejo.

atribut	tip atributa	opis
Id	Integer	Unikaten identifikator
UserId	String	Unikaten identifikator uporabnika glede na aplikacijo
Application	Application	Aplikacija, ki ji pripada uporabnik
UserData	List<UserFieldData>	Podatki o uporabniku

Tabela 3.7: Struktura razreda User

3.3.9 UserFieldData

Razred **UserFieldData** vsebuje podatke, ki jih shranjujejo dodatna uporabniška polja. Razred je identičen razredu **SessionCustomField** in je bil predstavljen v prejšnjih razdelkih.

3.4 Expression Manager

Ker želimo narediti našo rešitev čimbolj generično in prilagodljivo, smo dogodkom dodali atribut z vsebovano skripto, ki se zažene, ko se dogodek zgodi. `ExpressionManager` je razred, ki interpretira to skripto in izvrši njene ukaze.

V skripti želimo imeti na voljo naslednje podatke kot spremenljivke:

Objekt `Value` je niz znakov, ki smo ga skupaj z dogodkom poslali v storitev.

Objekt `Session` vsebuje podatke o trenutni seji.

Objekt `UserManager` nam pomaga pri pridobivanju uporabnika z določenim identifikatorjem trenutne aplikacije. Z njegovo pomočjo lahko dostopamo do objekta “`User`” ter mu tako spreminjamo podatke.

Objekt `Var` je enostaven razred, ki omogoča shranjevanje vrednosti v spremenljivke.

Objekt `CurrentTime` je objekt tipa `DateTime`, ki vsebuje trenuten čas.

Poleg omenjenih spremenljivk imamo dostop do dodatnih funkcij, ki nam pridejo prav pri pisanju te skripte. Za primer lahko podamo metode razreda `Math`, ki našim skriptam omogočajo uporabo matematičnih funkcij `log`, `sum`, `pow` in druge.

3.4.1 Primer skripte

Recimo, da želimo prešteti število komentarjev, ki jih je uporabnik objavil na portalu v naši aplikaciji. V aplikaciji smo definirali dogodek z imenom “`ObjavaKomentarja`”, ki kot `Value` pošlje id uporabnika, ki je objavil komentar. Aplikaciji smo dodali tudi dodatno polje z imenom “`skupaj_komentarji`”, ki vsebuje število komentarjev, ki jih je uporabnik objavil na portalu.

V nastavitvah dogodka lahko napišemo skripto 3.9, ki bo poskrbela, da se število v dodatnem polju “`skupaj_komentarji`” poveča vsakič, ko se v naši aplikaciji zgodi dogodek.

```
Session.SetCustomField("skupaj_komentarji", Session.  
GetCustomFieldInt("skupaj_komentarji") + 1)
```

Izvorna koda 3.9: Povečanje vrednosti v dodatnem polju z imenom "skupaj_komentarji"

Primer izvajanja skripte preko WebApi-ja si bomo ogledali v poglavju o Web API-ju.

3.5 Kontrolerji

Kot smo že nakazali, so kontrolerji dejansko v veliki večini le vezava med modelom ter izgledom aplikacije. V tem poglavju se bomo osredotočili na tiste kontrolerje, ki imajo pomembnejšo vlogo v delovanju aplikacije, ostale metode in akcije pa bomo le nakazali.

3.5.1 ApplicationController

`ApplicationController` je kontroler, ki skrbi za urejanje celotne aplikacije. Skrbi za dodajanje in urejanje aplikacij. Urejanje aplikacije pomeni tudi dodajanje dodatnih polj sejam ter dodajanje in urejanje dogodkov.

3.5.2 SessionsController

`SessionsController` uporabljamo za pregledovanje in filtriranje sej. Omogoča podrobni pregled določene seje in pregledovanje zaporedja dogodkov, ki so se zgodili med uporabo aplikacije, ki ji sledimo.

3.5.3 UsersController

`UsersController` uporabljamo za pregledovanje in filtriranje uporabnikov aplikacije.

3.5.4 ChartsController

`ChartsController` je kontroler, ki skrbi za prikaz zbranih podatkov v obliki vektorskih grafov. Ima možnost prikaza števila sej v določenem intervalu ali prikaz števila sej glede na čas s filtriranjem sej. Za prikaz teh podatkov uporablja črtasti graf. Ima tudi še možnost štetja po dodatnih poljih. Privzemimo, da imamo aplikacijo za komentarje na portalu. Kot smo že prej definirali, ima vsaka seja dodatno polje, ki pove, koliko komentarjev je bilo oddanih v tej seji. S pomočjo filtra za grafe lahko določimo, da namesto števila sej prikaže seštevek vrednosti tega dodatnega polja v vseh sejah, ki so se zgodile v tem določenem intervalu.

Na istem grafu lahko prikažemo večje število filtrov. Pri tem ima vsak filter določeno svojo barvo. Tako lahko primerjamo število oddanih komentarjev in število vseh sej v določenem času na istem grafu.

Kontroler omogoča tudi prikaz lijakastih grafov. To so grafi, ki prikazujejo, kako se neka skupina premika preko različnih ovir. Razložimo s primerom. Recimo, da želimo ugotoviti, koliko uporabnikov aplikacije je oddalo komentar. Za prvo skupino uporabimo prazen filter, ki nam prikaže vse uporabnike, ki so aplikacijo kadarkoli uporabljali v danem intervalu. Za drugo skupino izberemo uporabnike, ki so v danem intervalu objavili vsaj en komentar. Naredimo presek s prvo skupino in dobimo število uporabnikov iz prve skupine, ki so aplikacijo uporabili ter tudi komentirali. Lahko bi dodali še več skupin in tako prikazali, koliko uporabnikov dejansko oddaja komentarje glede na vse uporabnike, ki uporabljajo aplikacijo.

Lijakaste grafe se v večini uporablja za preverjanje deleža uporabnikov, ki od začetka uporabe aplikacije preidejo preko vseh ovir (dogodkov) do tega, da nekaj kupijo v aplikaciji (zadnji cilj).

3.6 Web API

Web API je sestavljen le iz dveh akcij (ustvarjanja seje ter dodajanja dogodkov že ustvarjeni seji). Zgrajen je na način, da omogoča kar najbolj enostavno uporabo, saj zahteva le, da se mu preko HTTP POST protokola podajo ustrezni podatki. Kot rezultat ustrezno vrne določene rezultate. Na primer, če pride do napake, da niso podani vsi potrebno podatki, se kot rezultat dobi napako `BadRequest (400)` z obrazložitvijo le te.

Vsebina zahteve mora vsebovati `www-form-encoded` podatke. To so podatki, ki imajo naštetna imena podatkovnih polj ter podane njihove vrednosti. Na primer, če želimo na strežnik poslati podatkovna polja z imeni `Ime` in `Priimek`, mora biti vsebina takega zahtevka `"Ime=Dejan&Priimek=Mesar"`.

V nadaljevanju si bomo ogledali zahteve ter delovanje vseh akcij, ki so potrebne za zbiranje podatkov o seji aplikacije.

3.6.1 Začetek seje

Akcija zahteva naslednje podatke:

AppId Je obvezen podatek in mora vsebovat veljaven `Id` aplikacije. `Id` aplikacije dobimo v podrobnostih aplikacije v našem sistemu.

Time Če podatek ni podan, se uporabi čas, ko je bil zahtevek prejet.

Akcija zahteva, da je **AppId** veljaven podatek, saj tako lahko dobi ustrezno aplikacijo iz podatkovne baze. Pri tem tudi preveri, da je aplikacija aktivna preko `IsBanned` atributa. Nato se ustvari nova seja v izbrani aplikaciji ter se shrani v podatkovno bazo. Če je bil postopek uspešno končan, dobi uporabnik v odgovor identifikator zgenerirane seje, ki ga nato lahko uporabi za dodajanje dogodkov.

3.6.2 Dodajanje dogodkov seji

Akcija zahteva naslednje podatke:

AppId Je obvezen podatek in mora vsebovat veljaven **Id** aplikacije. **Id** aplikacije dobimo v podrobnostih aplikacije v našem sistemu.

Sid Je obvezen podatek in mora vsebovat veljaven **Id** seje za podano aplikacijo.

Time Če podatek ni podan, se uporabi čas, ko je bil zahtevek prejet.

Key Je obvezen podatek in določa, kateri dogodek želimo dodati seji.

Value Ni obvezen podatek, uporablja se ga lahko za pošiljanje informacij iz aplikacije, skupaj z dogodkom, v našo storitev.

Če so vsi prejeti podatki veljavni, se izvede postopek dodajanja dogodka v podatkovno bazo. Ustvari se dogodek in nastavi se mu podane vrednosti. Skripta dogodka se izvede, če ima objekt dogodka (**Code**) nastavljen atribut **UseExpression** na vrednost **true**.

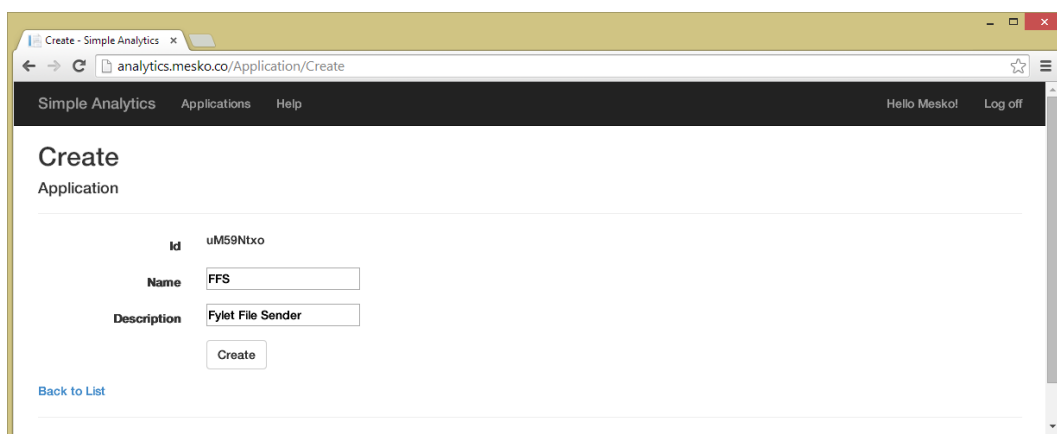
Hitrost dodajanja dogodka na strežnik je tukaj zelo odvisna tudi od skripte, ki je vezana na nek dogodek. Če se ob izvajanju skripte zgodi neka napaka, se sporoči aplikaciji, da je prišlo do napake pri izvajanju skripte, vendar se dogodek še vedno uspešno vpiše v podatkovno bazo. Ob uspešnem izvajanju skripte se vrne odgovor s prazno vsebino.

3.7 Opisi pripomočkov

Pri razvoju našega sistema smo razvili nekaj ključnih pripomočkov, ki si jih bomo podrobneje pregledali. Najprej se moramo spoznati z grajenjem strukture aplikacije za sledenje, saj brez te strukture ne moremo zbirati podatkov. Razvili smo tudi pripomočke za pregledovanje natančnih podatkov aplikacije

in njenih uporabnikov ter še pripomoček za prikazovanje linijskih grafov in lijakastih grafov.

Ob zagonu našega sistema v brskalniku se moramo najprej prijaviti. Po prijavi se v brskalniku prikažejo vse aplikacije za sledenje in ponudi se nam možnost, da ustvarimo tudi novo aplikacijo za sledenje (Slika 3.3).

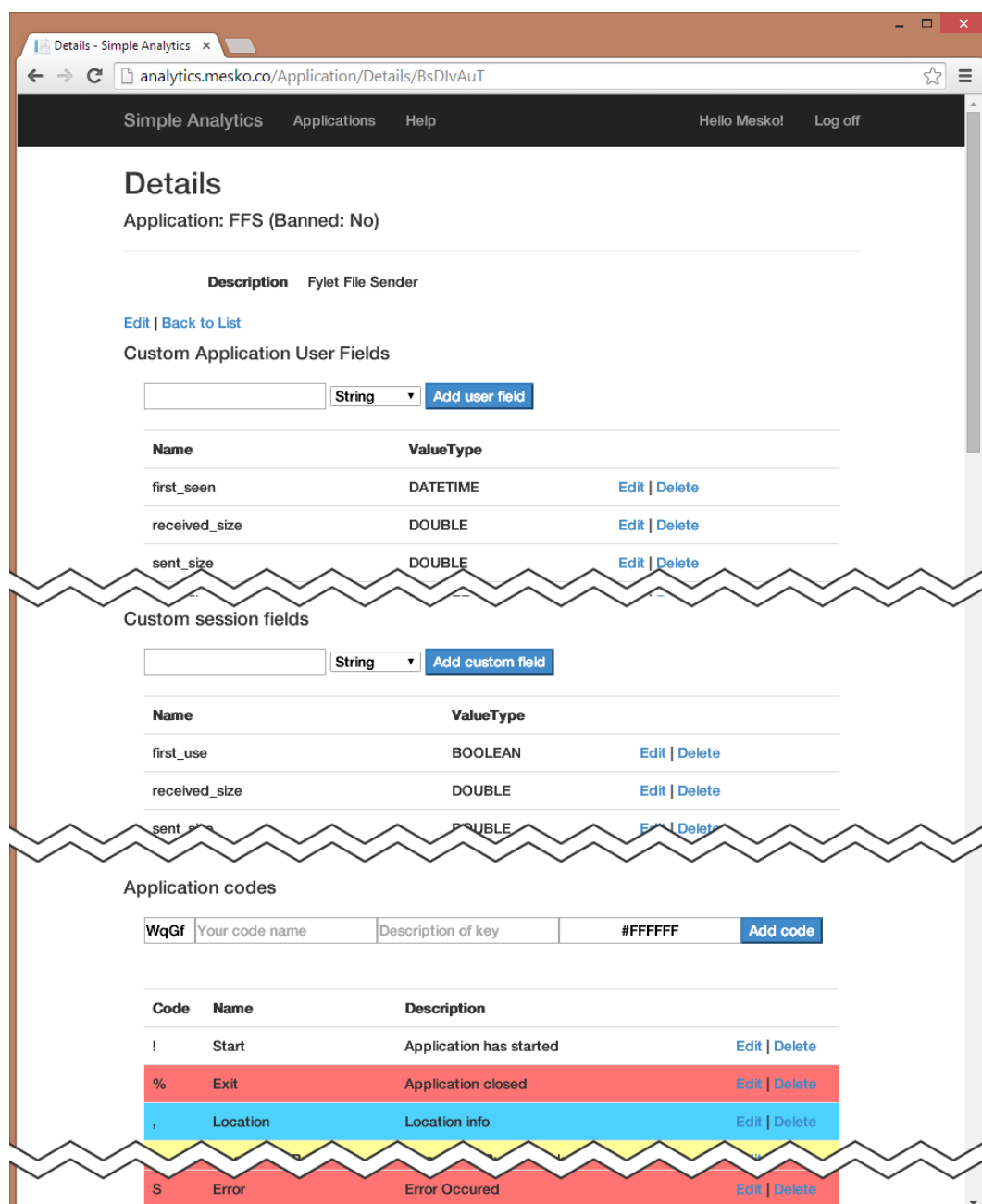


Slika 3.3: Dodajanje nove aplikacije

Pri dodajanju nove aplikacije nam že ponudi tudi vrednost **AppId**, ki je vidna v polju imenovanem **Id**. Po uspešnem dodajanju aplikacije lahko začnemo njeno strukturiranje z našim naslednjim pripomočkom.

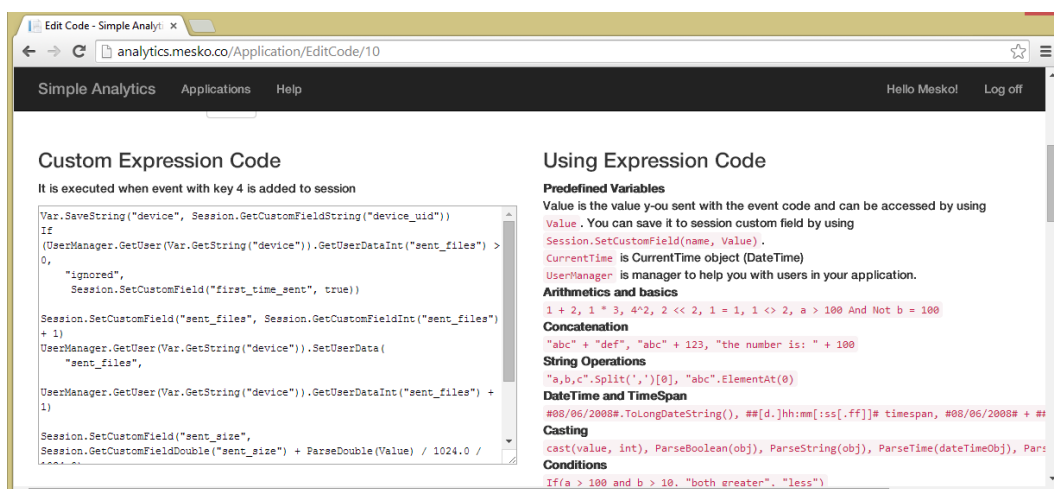
3.7.1 Struktura aplikacije

Ob kliku na povezavo do podrobnosti aplikacije se nam odpre nova stran (Slika 3.4), kjer lahko najdemo vse podrobnosti izbrane aplikacije. Na voljo nam je seznam dodatnih polj uporabnika, seznam dodatnih polj sej ter seznam vseh dogodkov, ki so obarvani tudi glede na njim določeno barvo. Na vrhu vsakega seznama imamo polja, s katerimi lahko dodajamo nove elemente v seznam. Če želimo nek element določenega seznama podrobno urejati, lahko kliknemo na povezavo za urejanje in odpre se nam stran, ki nam omogoča urejanje izbranega elementa.



Slika 3.4: Deli spletne strani o strukturi aplikacije.

Pri urejanju dogodka imamo na voljo tudi možnost vpisa skripte, ki se zažene ob dodajanju dogodka v določeno sejo. Slika 3.5 nam prikazuje okno, v katero vpišemo skripto. Ob tem oknu imamo tudi kratko pomoč, ki nam pokaže osnovne funkcije Expression Manager-ja. Tukaj lahko tudi testiramo našo skripto na testni seji, ki ji lahko spreminjamo vsa polja, ki jih trenutno ima aplikacija.



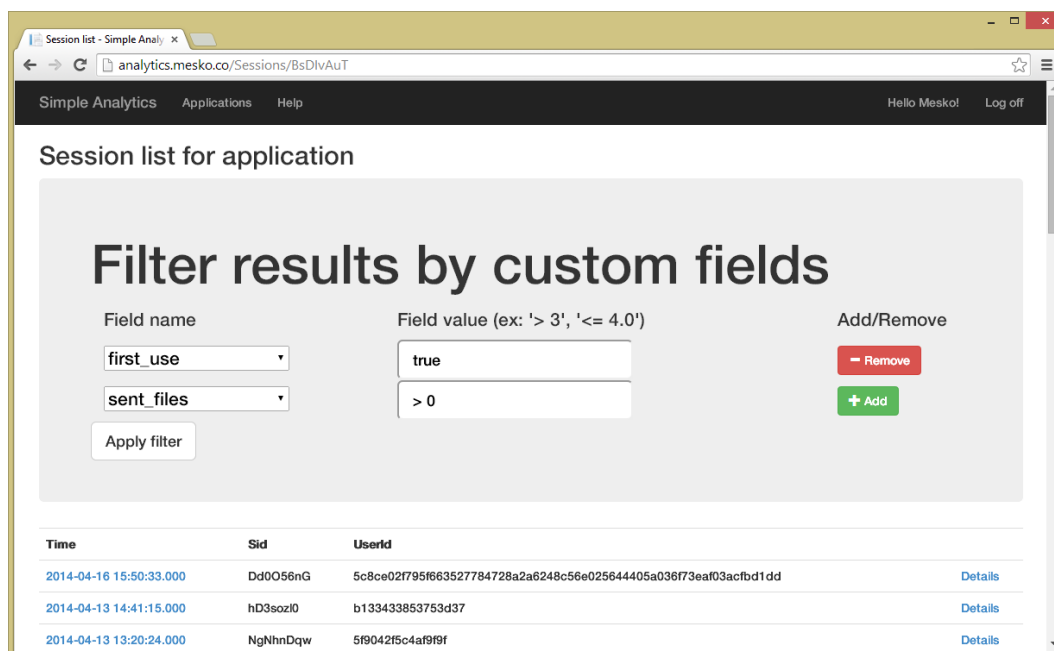
Slika 3.5: Okno za urejanje skripte dogodka.

3.7.2 Pregledovalnik sej in uporabnikov

Pregledovanje sej in uporabnikov je pomembno z več vidikov. Lahko nam omogoči podroben vpogled, kako neka skupina uporabnikov uporablja aplikacijo ali pa omogoča hitro pomoč za uporabnike v stiski (slednja potrebuje dovoljenje uporabnika, saj je zbiranje podatkov brez anonimizacije po zakonu prepovedano brez dovoljenja uporabnika).

Oba pregledovalnika vsebujeta enak modul za filtriranje sej ali uporabnikov. Modul nam omogoča filtriranje sej in uporabnikov glede na njihove podatke ali dodatna podatkovna polja, ki smo jih definirali v strukturi aplikacije. Glede na tip podatka, ki je shranjen v tem polju, lahko filtriramo glede na datum,

številk, niz znakov, ali je podatek DA ter ali niz vsebuje nek določen niz znakov. Rezultat je nato seznam sej oz. uporabnikov, ki so skladni s podanimi filtri. Primer filtriranja po večih poljih je viden na sliki 3.6.



Slika 3.6: Primer filtriranja sej.

3.7.3 Izdelava grafov

Pri izdelavi grafov je pomembno, da prikazujemo samo tiste podatke, ki jih želimo analizirati. To je po navadi na primer število sej, ki imajo neko določeno lastnost (recimo prenos datotek večji od 0). Tako je tudi tukaj na podoben način vgrajeno filtriranje sej. V pripomočku imamo prvotno na voljo le eno serijo podatkov, ki prikazuje število vseh sej preko določenega datuma. Grafu lahko dodajamo nove serije podatkov in za vsako od njih imamo lahko drugačne filtre. S tem lahko omogočimo primerjanje med filtri. Primerjamo lahko na primer število sej s prenesenimi datotekami s številom vseh sej.

Namesto števila sej lahko prikažemo seštevek vrednosti dodatnega polja

Create new graph - Simpli x

analytics.mesko.co/Charts/Create/BsDlvAuT

Simple Analytics Applications Help Hello Mesko! Log off

Create new graph for application FFS

Series #1

Field name

Select custom field

Field value (ex: '> 3', '<= 4.0')

Add/Remove

+ Add

Count or SUM fields

Count sessions

Series #2

Field name

first_use

Field value (ex: '> 3', '<= 4.0')

> 0

Add/Remove

+ Add

Count or SUM fields

Count sessions

Add new series

☐ Use logarithmic scale (Log10)

Daterange

03/21/2014

to

04/20/2014

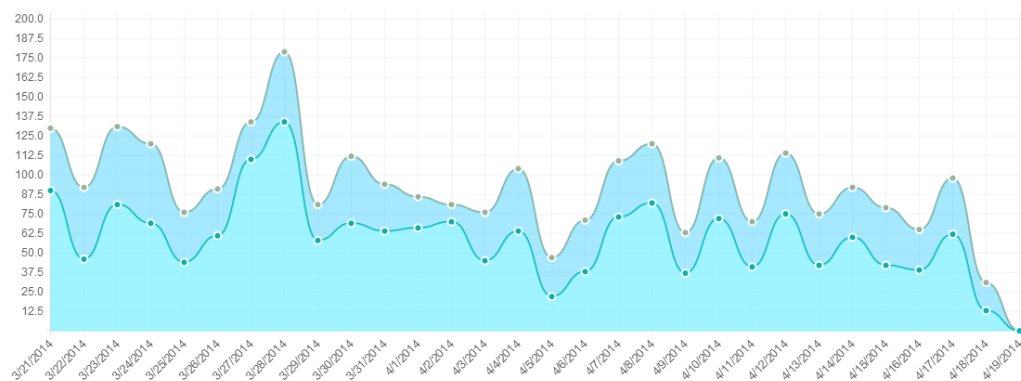
Apply filter

Slika 3.7: Primer izdelave linijskega grafikona.

seje. To nam tako omogoča, da prikazemo število prenesenih datotek namesto skupnega števila sej. S filtrom, ki ga prikazuje slika 3.7 dobimo graf, ki ga prikazuje slika 3.8.

3.7.4 Izdelava lijakastih grafov

Izdelava lijakastega grafa je v primerjavi z navadnim grafom različna v tem, da se za prvo serijo izbere neko množico uporabnikov, ki se jo nato filtrira skozi nadaljne filtre. Tako tukaj nimamo na voljo serij, ampak množico uporabnikov. V prvi skupini velikokrat uporabimo kar celotno množico uporabnikov določene aplikacije v nekem časovnem obdobju (2 tedna, 1 teden, 1 mesec itd.).



Slika 3.8: Rezultat filtriranja iz slike 3.7.

Vsaka nadaljna množica prvotno množico skrči na uporabnike, ki so skladni s podanimi filtri. Tako počasi pridemo do števila uporabnikov, ki so se skozi preprek e prebili do cilja (recimo nakupa artikla). Primer takega grafa je viden v poglavju “Preizkus sistema in rezultati” na sliki 4.4.

Poglavje 4

Preizkus sistema in rezultati

Za namen preizkusa sistema smo v sistem uvedli zbiranje podatkov za enostavno mobilno in namizno aplikacijo za deljenje datotek med njimi. S tem preizkusom smo se želeli prepričati, da je sistem dovolj zmogljiv, da lahko obdela večje količine podatkov. Ob koncu testiranja je velikost podatkovne baze obsegala preko 300 MB in število sej je preseglo 28 000 sej ter 5000 unikatnih uporabnikov. Podatki so bili zbrani v času od 1.1.2014 do 1.5.2014.

S pomočjo pripomočkov našega sistema želimo ugotoviti vedenje uporabnikov te aplikacije ter priti do zaključka, ki bi pripomogel k izboljšanju aplikacije. Z zbranimi podatki lahko tako spreminjamo nadaljni razvoj programa v smer, da ga bodo končni uporabniki bolje uporabljali.

4.1 Opis aplikacije

Namen aplikacije je deljenje datotek med napravami preko varnega in direktnega kanala. Ob zagonu aplikacije se trenutni seji zgenerira unikatna številka, ki deluje kot naslov naprave. S pomočjo te številke lahko druga naprava s to aplikacijo deli neko datoteko preko varnega in direktnega kanala.

Aplikacija je razvita za naslednje sisteme in tehnologije:

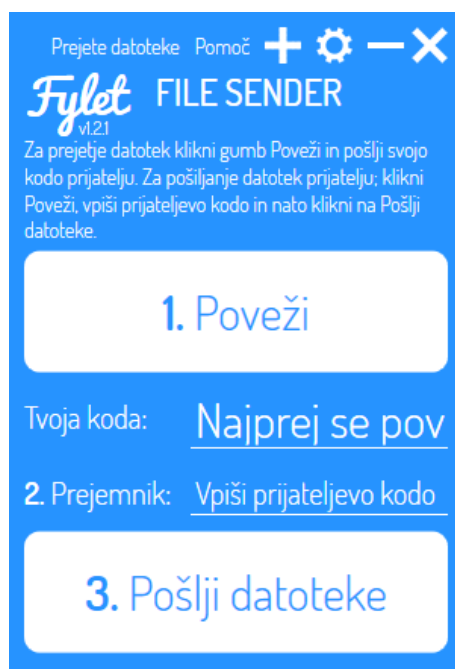
- Microsoft .NET 4.0 Framework (Windows XP ali novejši),

- Android 2.2 ali novejši,
- Java za Linux in Mac OSX okolje.

Izgled in uporaba aplikacije se od sistema do sistema bistveno ne razlikuje. Ko uporabnik zažene aplikacijo, se mu na zaslonu odpre novo okno, na katerem sta dva gumba in dve vnosni polji.

Prvi gumb (poveži) skrbi za povezavo s strežnikom, ki poskrbi, da se dve napravi lahko povežeta z direktnim kanalom. Ker je kanal direkten in povezava ne poteka preko tretjega strežnika, je povezava varna pred tretjimi osebami. Za dodatno varnost je povezava tudi kriptirana.

Ob uspešno vzpostavitvi povezave do strežnika, se zgenerira unikatna 5 do 10 mestna številčna šifra, ki skrbi za identifikacijo trenutne naprave.



Slika 4.1: Prvi zaslon testne aplikacije.

Če uporabnik želi drugi napravi poslati datoteke, mora vedeti šifro tuje naprave. Šifro tuje naprave vpiše v drugo vnosno polje ter nato ob kliku na

drugi gumb (pošlji datoteke) izbere datoteke, ki jih želi prenesti. Aplikacija nato poskuša vzpostaviti direkten kanal med napravama. Po uspešni vzpostavitvi kanala se izbrane datoteke prenesejo do ciljne naprave.

Uporabniku, ki prejema datoteke, se pred prenosom prikaže pogovorno okno, da mu želi oddaljena naprav poslati datoteke. V pogovornemu oknu ima možnost sprejeti ali odkloniti datoteke. Prav tako ima možnost samodejnega sprejetja zahteve za prejetje datotek v nastavitvah aplikacije.

Aplikacija ima na vrhu tudi gumb “Prejete datoteke”, ki nam odpre mapo kjer se shranjujejo prenešene datoteke, gumb “Možnosti” ter gumba za pomajšanje okna in zapiranje aplikacije.

4.2 Sestava in seznam dogodkov

V aplikaciji vse stremi k enemu cilju: pošiljanju datotek; Zato smo aplikaciji dodali določene dogodke, od katerih je odvisen ta cilj. Ti dogodki so:

- Uporabnik je zagnal aplikacijo,
- Uporabnik je povezan s strežnikom,
- Uporabnik je poslal datoteko,
- Uporabnik je prejel datoteko.

S temi dogodki lahko ugotovimo, koliko naših uporabnikov uporablja našo aplikacijo tako kot smo si zamislili. Za identifikacijo uporabnikov se ob prvem zagonu zgenerira identifikacijska številka, ki jo nato ob zagonu aplikacije pošljemo skupaj z dogodkom “start”.

Za začetek seje smo napisali metodo `CreateNewSession` (Izvorna koda 4.1), ki na strežnik pošlje potrebne podatke. Vsaka platforma je seveda zahtevala nekoliko drugačen način, vendar je uporaba na vseh platformah ista.

```
private static string CreateNewSession(string AppId)
{
    string sid = "";
    try
    {
        WebClient.Headers[HttpRequestHeader.ContentType] = "
            application/x-www-form-urlencoded";
        var data = string.Format("AppId={0}", AppId);
        var res = System.Web.Helpers.Json.Decode(WebClient.
            UploadString("http://analytics.mesko.co/api/Session/Create
            ", "POST", data));
        sid = res.Message as string;
    }
    catch (WebException)
    {
        throw;
    }
    return sid;
}
```

Izvorna koda 4.1: Implementacija začetka seje v C#

Za pošiljanje dogodkov smo spisali metodo **SendEvent** (Izvorna koda 4.2), ki pošlje zahtevo na Web API za dodajanje dogodka trenutni seji.

```
private static void SendEvent(string AppId, string Sid, string
    Code, string Value)
{
    string data = "";
    try
    {
        WebClient.Headers[HttpRequestHeader.ContentType] = "
            application/x-www-form-urlencoded";
        data = string.Format("AppId={0}&Sid={1}&Key={2}&Value
            ={3}", AppId, Sid, Code, System.Uri.EscapeDataString(
            Value));
    }
```

```
        var res = System.Web.Helpers.Json.Decode(WebClient.UploadString(
            "'http://analytics.mesko.co/api/Session/AddEvent'", "POST", data));
    }
    catch (WebException)
    {
        throw;
    }
}
```

Izvorna koda 4.2: Primer metode, ki doda dogodek seji v C#

Po preteku določenega obdobja smo naredili analizo zbranih podatkov. Cilj naše analize je ugotoviti, koliko uporabnikov je preneslo vsaj eno datoteko. Glede na cilj smo definirali naslednje analize, ki bi nas zanimajo:

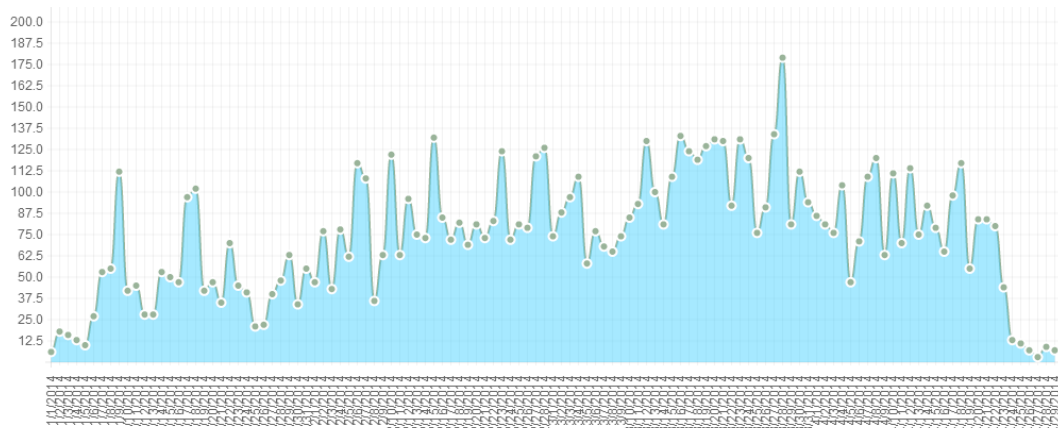
- Število aktivnih sej na dan,
- Število prenesenih datotek na dan,
- Lijakasti graf izgubljanja uporabnikov do prenosa datoteke.

4.3 Rezultati

Podatke o rabi aplikacije smo zbirali od 1.1.2014 do 1.5.2014. Pridobili smo dovolj podatkov, da preverimo, ali naša aplikacija služi svojemu namenu.

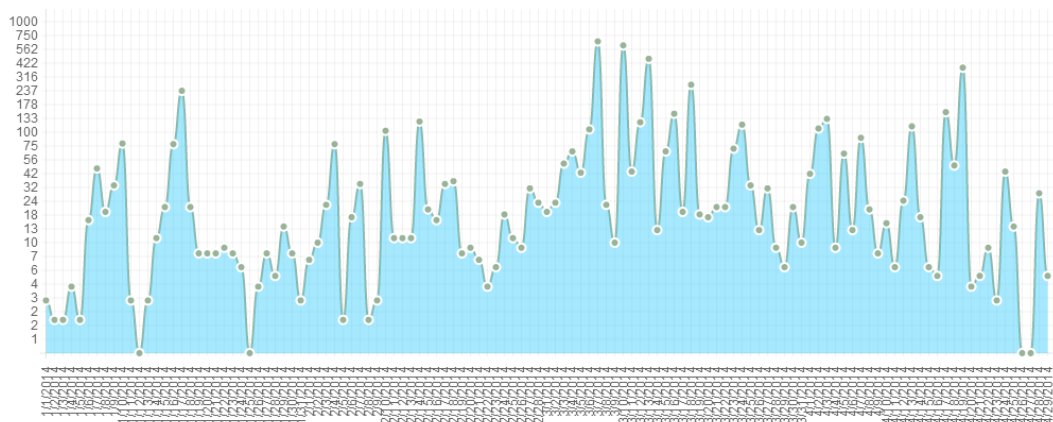
Najprej smo si ogledali, koliko sej je bilo aktivnih na dan. Rezultate prikazuje slika 4.2. Z izjemo zadnjih dveh tednov je raba aplikacije počasi rasla. Velik padec v zadnjih dveh tednih prikazuje izpad strežnika ali njegovo preobremenjenost.

Pri pregledu števila poslanih datotek na dan (slika 4.3) smo opazili, da se s časom število prenesenih datotek ni znatno povečalo, čeprav lahko opazimo, da je bil začetni trend pozitiven. Zadnji mesec in pol pa je trend negativen, kar je za poslovanje zaskrbljujoče, saj lahko pomeni, da aplikacije ljudje niso



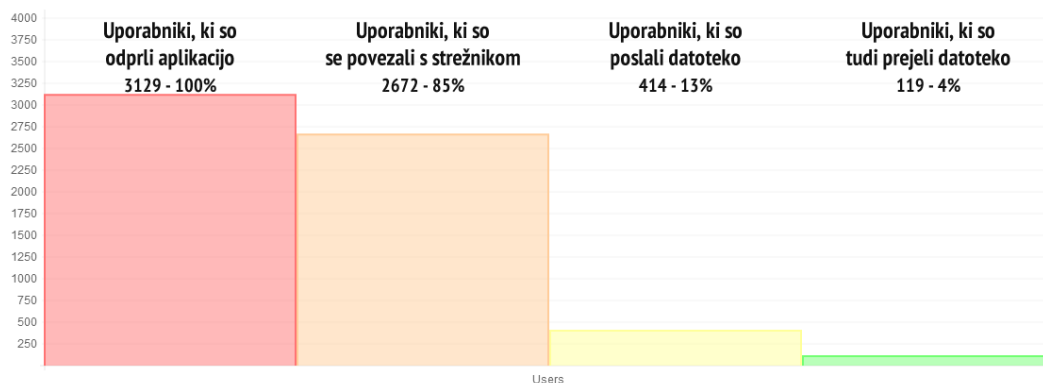
Slika 4.2: Število sej na dan tekom obdobja testiranja.

pripravljeni uporabljati večkrat. Točnega zaključka, zakaj je upadlo število prenosov, ne moremo odkriti iz tega grafa (slika 4.2).



Slika 4.3: Število prenesenih datotek na dan tekom obdobja testiranja.

Zato smo naredili še eno analizo (slika 4.4), ki nam predstavlja delež uporabnikov, ki so opravili nek cilj v aplikaciji. S tem grafom lahko opazimo, kje izgubljam uporabnike. Ugotovili smo, da se uporabniki izgubijo, ko bi morali datoteko poslati, saj jih od vseh uporabnikov le 13% pošlje datoteko drugemu uporabniku. To je zaskrbljujoče nizka številka.



Slika 4.4: Prikaz deleža uporabnikov glede na opravljene cilje.

Vprašamo se, zakaj je tako. Razlogov je lahko več in na naši strani je, da izberemo najverjetnejši razlog, ga odpravimo in objavimo novo verzijo aplikacije, ki bo ta razlog izničila. Oglejmo si možne razloge:

1. Opis aplikacije in spletna stran nista dovolj jasni. Rešitev je preoblikovanje spletne strani in opisa aplikacije, ki bolje nakazuje njen namen.
2. Oblika in izgled aplikacije ne vodita uporabnika dovolj dobro do cilja (nejasnosti pri uporabi aplikacije). Rešitev je preoblikovanje aplikacije na način, ki je uporabniku bolj jasen.
3. Povezava med napravama ni dovolj stabilna za pošiljanje datotek.

Prva točka je manj zahtevna in hitreje izvedljiva, vendar je druga veliko bolj verjetna. Privzemimo, da uporabnik hoče poslati datoteko prijatelju. Preiti mora skozi 4 korake: zagon aplikacije, povezava s strežnikom, vpis prijateljeve kode ter nato izbor datotek za pošiljanje. Uporabnik ob odprtju aplikacije ve, katere datoteke želi poslati, vendar ne ve prijateljeve šifre. Zato se uporabnik ustavi pri tretjem koraku in ne more naprej. Ostala sta mu dva koraka. Če zadnja dva zamenjamo, se uporabnik ustavi le na zadnjem in tako ima večjo motivacijo, da dokonča še zadnji korak namesto kot prej, ko sta mu ostala še dva.

Pri zadnji točki se napaka skriva v modulu, ki skrbi za vzpostavitev povezave med napravama. Modul je obsežnejši in bolj zapleten. Poleg obsega modula so tu tudi razni zunanji vplivi, ki lahko vplivajo na vzpostavitev povezave (požarni zidi, nestabilno mobilno omrežje). Te vplive se težko preverja in bi zato ponovno potrebovali ogromno časa, da preverimo, ali bi naši popravki delovali.

Drugo točko lahko preverimo s pregledom sej, v katerih so se uporabniki uspešno povezali na strežnik, vendar niso opravili nobenega prenosa. Seznam vseh sej lahko pridobimo s pomočjo pripomočka za pregledovanje sej (Slika 4.5).

Po podrobnejšem pregledu nekaterih sej smo odkrili, da so si zelo podobne in izgledajo približno tako, kot prikazuje slika 4.6. S pomočjo podrobnega pregleda smo natančneje ugotovili, kako uporabniki uporabljajo aplikacijo. Uporabnik odpre aplikacijo in se poveže s strežnikom. Nato ne ve, kako naprej. Potreboval bi vodenje.

S pomočjo podrobnega pregleda uporabnikov lahko sedaj določimo rešitev našega problema. Potrebno bo redefinirati korake, ki jih morajo uporabniki narediti, da pošljejo datoteko drugi napravi. Po popravku sledimo, če se odstotek uporabnikov, ki so poslali oziroma prejeli datoteko, povečuje ali manjša. Če se povečuje, smo s tem izboljšali uporabniško izkušnjo. Če se zmanjšuje, smo naredili napako in je bolje, da popravke odstranimo. Če se odstotek ne spreminja, potem se rešitev problema skriva v 1. ali 3. točki.

Vsekakor nam je lastna rešitev pomagala pri iskanju problema in njegove rešitve.

Session list - Simple Analy x

analytics.mesko.co/Sessions/BsDlvAuT

Simple Analytics Applications Help Hello Mesko! Log off

Session list for application

Filter results by custom fields

Field name Field value (ex: '> 3', '<= 4.0') Add/Remove

is_connected true Remove

received_files 0 Remove

sent_size 0 Add

Apply filter

Time	Sid	Userid	
2014-05-21 08:19:15.000	sw64MrWM	4c8a43718eccf9b0	Details
2014-05-21 08:04:30.000	6Elcsqf8	18fdc7f88afeea4f	Details
2014-05-21 07:57:54.000	IKzHfZ0T	2bf2dcaacbd4de8a	Details
2014-05-21 07:50:23.000	owsbmhPu	b50a27411f3d4887	Details
2014-05-21 07:05:34.000	RY12nlvk	122e01bd1737c350	Details
2014-05-21 06:49:57.000	g7gNH4cc	2a509e530e4bde32	Details
2014-05-21 06:01:38.000	K8XAZdum	8e83217b12a109ec	Details
2014-05-21 05:36:46.000	GerlcvV4	e0ca27e8185e6123	Details

Slika 4.5: Prikaz seznama sej

Time	Code Name	Code Description	Value
2014-05-21 07:50:23.810	Start	Application has started	b50a27411f3d4887
2014-05-21 07:50:23.810	Version Info	Version Info	FFS 1.9
2014-05-21 07:50:24.093	Device Info	Device Info	M470BSA,Android,4.2.1
2014-05-21 07:50:30.397	Connect Button	Connect Button Pressed	
2014-05-21 07:50:30.730	Connected	Device connected with code	72337
2014-05-21 07:50:33.447	Location	Location info	USA,United States,Oak Forest
2014-05-21 07:51:49.033	Exit	Application closed	

Slika 4.6: Prikaz seje nekega uporabnika

Poglavje 5

Zaključek

S pomočjo testiranja smo prišli do ugotovitve, da je naš sistem dovolj prilagodljiv, da lahko zbira podatke s poljubnega programskega ogrodja ter naprave. Lahko obvladuje več kot 300 MB podatkov brez večje izgube na kvaliteti ter hitrosti obdelave podatkov. Za uporabo nam ni potrebno pošiljati uporabniških podatkov tretjim osebam, kar pomeni, da morajo stranke zaupati le nam. Sistem je tako prilagodljiv, da lahko za vsako aplikacijo pridobimo dovolj natančne podatke o uporabnikih, za rezultate pa nam ni potrebno čakati 4 do 24 ur, kot čakamo pri drugih ponudnikih.

Čeprav podatkov ne posredujemo tretjim osebam, jih moramo shranjevati na svojih strežnikih. Naš sistem je odličen pri malem številu uporabnikov. Pri večjih količinah zbiranja podatkov pa bi morali naš sistem strežnikov nadgraditi.

Naša rešitev je najbolj primerna za mlada in majhna zagonska podjetja, ki bi rada o svojih uporabnikih izvedela čim več. Naša rešitev je bolj prilagodljiva od ostalih in zagonskim podjetjem ponuja popolno kontrolo nad podatki.

Slike

2.1	Stran, ki se prikaže ob vnosu prve aplikacije za sledenje v Google Analizah	11
2.2	Spletna stran ob prvi prijavi v Flurry	14
2.3	Uporabniki so zelo skopi z informacijami o napakah	17
3.1	Prikaz obdelave uporabnikove zahteve po modelu MVC	22
3.2	Rezultat knjižnice Chart.js	30
3.3	Dodajanje nove aplikacije	41
3.4	Deli spletne strani o strukturi aplikacije.	42
3.5	Okno za urejanje skripte dogodka.	43
3.6	Primer filtriranja sej.	44
3.7	Primer izdelave linijskega grafikona.	45
3.8	Rezultat filtriranja iz slike 3.7.	46
4.1	Prvi zaslon testne aplikacije.	48
4.2	Število sej na dan tekom obdobja testiranja.	52
4.3	Število prenesenih datotek na dan tekom obdobja testiranja.	52
4.4	Prikaz deleža uporabnikov glede na opravljene cilje.	53
4.5	Prikaz seznama sej	55
4.6	Prikaz seje nekega uporabnika	56

Tabele

3.1	Struktura razreda Application	31
3.2	Struktura razreda Code	32
3.3	Struktura razreda ApplicationCustomField	32
3.4	Struktura razreda Session	33
3.5	Struktura razreda SessionCustomField	34
3.6	Struktura razreda SessionData	35
3.7	Struktura razreda User	35

Izvorne kode

2.1	JavaScript koda za implementacijo Google Analize v našo spletno aplikacijo	10
2.2	Manifest Android aplikacije	14
2.3	Implementacija Flurry v Android aplikacijo	15
3.1	Razred, ki opisuje vrstico tabele Aplikacije	24
3.2	Primer razreda, ki predstavlja podatkovno bazo v Entity Framework	24
3.3	Primer pridobivanja seznama Aplikacij , katerih lastnik je Uporabnik z imenom Ivan	25
3.4	JavaScript je dinamični jezik, zato se tip spremenljivke določi ob prirejanju vrednosti.	26
3.5	Primer dodajanja atributov in metod objektu	26
3.6	Primer uporabe ogrodja jQuery za dodelitev obnašanja ob kliku na gumb z id-jem TestniGumb	27
3.7	Primer Ajax klika s pomočjo jQuery ogrodja	28
3.8	Primer inicializacije preprostega grafa v Chart.js knjižnici, rezultat viden na sliki 3.2	29
3.9	Povečanje vrednosti v dodatnem polju z imenom "skupaj_komentarji"	37
4.1	Implementacija začetka seje v C#	50
4.2	Primer metode, ki doda dogodek seji v C#	50

Literatura

- [1] Clifton, B. *Advanced Web Metrics with Google Analytics*. 3rd Edition. Indianapolis, John Wiley & Sons Inc, 2012. ISBN 978-1-118-16844-8
- [2] Galloway, J. et al. *Professional ASP.NET MVC 4*. Indianapolis, John Wiley & Sons Inc, 2012. ISBN 978-1-118-34846-8
- [3] Crockford, D. *JavaScript: The Good Parts*. 1st Edition. Sebastopol, O'Reilly Media, 2008. ISBN 978-0-596-51774-8
- [4] Chaffer, J. in Swedberg K. *Learning jQuery*. 4th Edition. Birmigham, Packt Publishing Ltd., 2013. ISBN 978-1-78216-314-5
- [5] Driscoll, B. et al. *Entity Framework 6 Recipes*. 2nd Edition. New York, Apress Media LLC, 2013. ISBN 978-1-4303-5788-2
- [6] Block, G. et al. *Designing Evolvable Web APIs with ASP.NET*. 1st Edition. Sebastopol, O'Reilly Media, 2014. ISBN 978-1-449-33771-1
- [7] Holdener, A. T. III. *Ajax: The Definitive Guide*. Sebastopol, O'Reilly Media, 2008. ISBN 978-0-596-52838-6
- [8] Internet World Stats. *World Internet Users and Population Stats*. Dostopno na: <http://www.internetworldstats.com/stats.htm> (marec 2014)
- [9] Nick Downie. *Chart.js Documentation*. Dostopno na: <http://www.chartjs.org/docs> (marec 2014)

- [10] Microsoft. *ASP.NET MVC 5*. Dostopno na:
<http://www.asp.net/mvc/mvc5> (maj 2014)
- [11] Flurry. *Flurry Analytics*. Dostopno na:
<http://www.flurry.com/solutions/analytics> (marec 2014)
- [12] Flurry. *Flurry Analytics Support*. Dostopno na:
<http://support.flurry.com/index.php?title=Analytics> (marec 2014)