Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Simon Repar

Razvoj CRM aplikacije na podlagi platforme v oblaku

diplomsko delo
na univerzitetnem študiju

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Force.com je platforma v oblaku, ki omogoča hiter in učinkovit razvoj CRM aplikacij. Proučite platformo Force.com in pri tem dajte poseben poudarek integraciji med aplikacijami. Na platformi Force.com nato tudi razvijte prototip CRM aplikacije, ki omogoča izvajanje tržnih akcij preko razpošiljanje personaliziranih sporočil elektronske pošte.

Izjava o avtorstvu diplomskega dela

Podpisani Simon Repar, z vpisno številko **63070168**, sem avtor diplomskega dela z naslovom:

Razvoj CRM aplikacije na podlagi platforme v oblaku

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Roka Rupnika,
- 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 27. maja 2014 Podpis avtorja:

Ob zaključku študija se želim za strokovne in koristne nasvete najprej zahvaliti mentorju doc. dr. Roku Rupniku.

Prav tako se zahvaljujem podjetju Agilcon za pomoč in podporo pri izdelavi tehnične rešitve.

Za prijetno preživljanje študijskega vsakdana izrekam zahvalo tudi vsem mojim študijskim kolegom.

Ter Esteli Emanueli, ki je kljub moji zamaknjenosti ob pisanju diplomskega dela, ohranila svojo jezo na sprejemljivi ravni.

Nazadnje pa se iskreno zahvaljujem svojim staršem, bratu in sestri, da so mi dajali motivacijo in moralno podporo za dokončanje študija.

Kazalo

Seznam kratic

ACID	Atomicity, Consistency, Isolation and Durability
AJAX	Asynchronous JavaScript and XML
APEX	Object-oriented programming language used on Force.com
API	Application Programming Interface
CMS	Content Management System
CRM	Customer Relationship Management
CRUD	Create, Read, Update, Delete
CSA	Cloud Security Alliance
CSS	Cascading Style Sheets
DML	Data Manipulation Language
DoS	Denial of Service
ERP	Enterprise Resource Planning
ESP	Email service provider
HRM	Human Resource Management
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
ISV	Independent Software Vendor
JAX-RS	Java API for RESTful Web Services
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MIME	Multipurpose Internet Mail Extensions
MVCC	Multi Version Concurrency Control

NaaS	Network as a Service
OAuth	An Open Standard for Authorization
ORDBMS	Object Relational Database Management System
PaaS	Platform as a Service
PDF	Portable Document Format
REST	Representional State Transfer
SaaS	Software as a Service
SFA	Sales Force Automation
SOAP	Simple Object Access Protocol
SOQL	Salesforce Object Query Language
SOSL	Salesforce Object Search Language
SQL	Structured Query Language
WS	Webservice
WSDL	Web Services Description Language
XSD	XML Schema Defenition
XSS	Cross-Site Scripting

Povzetek

Z razmahom računalništva v oblaku se spreminja način uporabe tehnologij in hkrati ponuja možnost za razvoj sodobnejših storitev. Namen diplomske naloge je raziskati možnost razvoja aplikacije na tem področju.

V delu sta opisana razvoj poslovne aplikacije na platformi Force.com. Integracija med različnimi platformami, ki je primarna naloga aplikacije, zahteva kar nekaj poznavanja sodobnih tehnologij. Raziskali bomo tovrstne tehnologije in jih ovrednotili v povezavi s Force.com. Glavni del aplikacije poganja Java na platformi Heroku. Postavljena je na oblačni platformi, zato so na kratko opisani tudi koncepti, kot so SaaS, PaaS in IaaS. Še posebej se dotaknemo CRM sistemov, saj je CRM, katerega z našo aplikacijo povezujemo s fokusiranim sistemom za izvajanje trženjskih akcij, središče njene uporabe. Poleg tehnične rešitve pa bomo opisali tudi postopek priprave aplikacije za objavo na spletno tržišče AppExchange, ki je del opisanega CRM-ja.

Ključne besede:

CRM, platforma kot storitev, računalništvo v oblaku, AppExchange, integracija, Force.com

Abstract

With the rapid cloud computing growth, the way we use technologies is changing and at the same time it offers a great possibility to develop new modern services.

The purpose of the diploma thesis is to research the possibility of developing applications in this area. The thesis describes the development of business applications on the Force.com platform. Integration between different platforms, which is the primary task of application, requires wide knowledge of modern technologies. We are going to evaluate the relation to the Force.com platform. The main part of the developed application is Java powered, hosting by Heroku platform. Later on we briefly describe cloudy concepts such as SaaS, PaaS and IaaS. Specifically we describe the CRM system, as it is the center of our application, which associates CRM with focused system for e-mail marketing. In addition to the technical solution, we describe the steps for publishing application to the AppExchange marketplace.

Key words:

CRM, Platform as a Service, cloud computing, AppExchange, integration, Force.com

Poglavje 1

Uvod

Vsak izmed nas je na svoj elektronski naslov že kdaj prejel sporočilo, ki je bilo namenjeno širšemu krogu prejemnikov pa naj je bilo to marketinškega ali kakšnega drugega namena. Ste se že kdaj vprašali kako podjetja organizirajo svoje stranke ali kako različne organizacije hranijo bazo e-naslovov svojih članov ter nato redno pošiljajo e-sporočila veliki množici ljudi?

Po podatkih raziskave podjetja Return path povprečni e-poštni odjemalec dobi vsak mesec 416 sporočil. V povprečju je bilo samo 10 % teh sporočil odprtih. Govorimo o tem, kdo je sporočilo odprl ter kaj je s tem sporočilom počel. Zagotovo vse pošiljatelje zanima, v kolikšni meri je to sporočilo vplivalo na naslovnika.

Zamislimo si podjetje, katerega glavno poslanstvo je prodaja izdelkov preko e-poštnih marketinških kampanj. Imajo 20.000 naročnikov, ki jih hranijo v CRM-ju. Temu podjetju želimo omogočiti najboljša orodja, ki mu bodo pomagala pri osnovni dejavnosti. Podjetje pričakuje, da bo v vsakem trenutku vedelo, kateri so tisti njihovi naročniki, ki so najbolj dejavni pri njihovih e-poštnih kampanjah.

Kot omenjeno podjetje uporablja CRM za stranke in t.i. 360° pogleda na stranko. V našem primeru se bomo odločili za sistem Salesforce CRM, ki je kot eden od vodilnih platform za upravljanje odnosov s strankami. V letu 2012 je imel 14 % tržni delež.

Za ponudnika e-poštnih storitev bomo izbrali MailChimp, ki se je za naše zahteve izkazal za najprimernejšega zaradi tehničnih značilnosti ter je po tržnem deležu na drugem mestu za storitvijo AWeber.

Za cilj si bomo torej zastavili, kako preko teh dveh vodilnih orodij podjetju

zagotoviti potrebne informacije, ki jih po pošiljanju e-poštnih kampanij želi videti.

1.1 Tehnologija

V tehničnem smislu imamo tako dve platformi, ki ju moramo povezati, da bi pridobili želeni produkt. S stališča uporabnika moramo za središče uporabe vzeti CRM. Uporabnika v tem primeru razumemo kot nekoga, ki upravlja s podatki, je v stiku ali na kakšen drug način dela s strankami. Vse podatke, ki nas o stranki zanimajo, hranimo v CRM-ju in s tem centraliziramo podatke ter posledično olajšamo in poenostavimo delo za uporabnike tega sistema. Obravnavan CRM je grajen v t.i. oblaku, kar lahko rečemo tudi za ESP MailChimp. Obe orodji spadata v IaaS kategorijo, katero bomo podrobneje opisali kasneje.

V slogu obeh bomo našo rešitev prav tako gradili v oblaku.

1.2 Pregled zastavljenih ciljev

Primarni cilji diplomske naloge:

- Izdelava storitve, opisane v uvodu.
- Aplikacijo preko vseh korakov pripeljati na spletno tržišče AppExchange.
- Kako lahko z omenjenimi platformami in obstoječimi tehnologijami nadgradimo aplikacijo za celostno izkušnjo?
- Na primeru Fakultete računalništva in informatike narediti izračun uporabe te storitve.

Diplomsko delo je strukturirano na naslednji način. Poglavje 2 opisuje računalništvo v oblaku ter se dotakne njegovih glavnih modelov. Tukaj opišemo tudi glavne varnostne pomisleke, ki jih računalništvo v oblaku prinaša. V poglavju 3 obravnavamo CRM na splošno, saj ga bomo kasneje uporabili za integracijo z e-poštnim marketinškim sistemom. V poglavju 4 predstavimo najbolj razširjen CRM, Salesforce in njegove ključne poslovne modele ter platformo Force.com, na kateri temeljijo Salesforcovi produkti. Pri razvoju aplikacije je uporabljenih veliko tehnologij, katere opišemo v poglavju 5. Sledijo tehnične podrobnosti in opis razvoja aplikacije v poglavju 6. Za konec se v poglavju 7 dotaknemo spletnega tržišča AppExchange in zahtevanim postopkom za objavo aplikacije na tem spletnem tržišču.

Poglavje 2

Računalništvo v oblaku

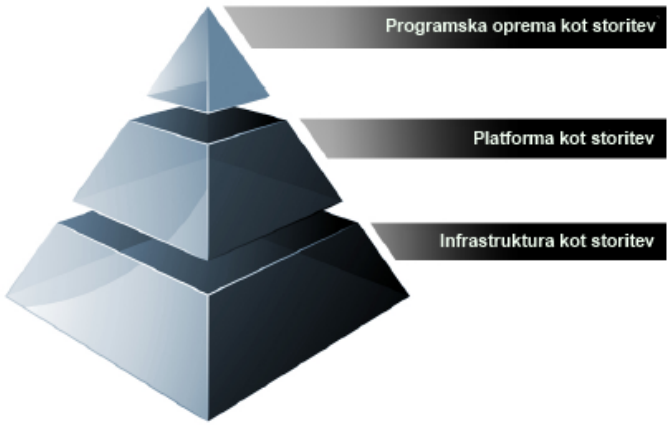
Preprosta definicija termina "računalništvo v oblaku"(RO) opisuje tip računalništva, ki se v nasprotju z lokalno namestitvijo na strežnike oziroma osebne naprave opira na deljenje računalniških virov. Izraz zajema številne računalniške koncepte, ki vključujejo veliko število računalnikov, ki so povezani v komunikacijsko omrežje, kot je npr. internet. Prvotna motivacija RO je bila v tem, da vire, ki so uporabljeni le del časa, uporabimo tudi za druge procese.

2.1 Zgodovina

Računalništvo v oblaku ima že zelo zgodnje začetke. Koncepti nudenja računalniških virov skozi globalno omrežje so se oblikovali v šestdesetih letih prejšnjega stoletja. Leta 1969 je bil vzpostavljen razvoj ARPANET-a [12], katerega vizija je bila, da bi lahko vsak na svetu dostopal do vseh programov in podatkov od kjerkoli. To pa zveni podobno kot definicija računalništva v oblaku danes.

2.2 Vrste modelov

Kot smo že omenili, je računalništvo v oblaku širok pojem, ki zajema mnogo različnih servisov. V času, ko je razvoj računalništva v oblaku zelo bliskovit, so ponudniki tovrstnih rešitev ta pojem razširili še na druge produkte, ki ne spadajo pod običajno definicijo RO. Preden razumemo, kakšno dodano vrednost lahko



Slika 2.1: Sklad računalništva v oblaku

organizaciji prinese t.i. oblak, moramo najprej razumeti, kaj oblak sploh je in kakšne komponente nam ponuja. Najprej se bomo osredotočili na klasične servise, ki so znani pod kraticami IaaS, PaaS, SaaS, NaaS, itd. Podali bomo tudi nekaj primerov uporabe in predstavili situacije, v katerih pa ti sistemi le niso najboljši za organizacijo.

2.2.1 IaaS, PaaS, SaaS

RO je pogosto opisan kot sklad več različnih storitev (Slika 2.1), ki so zgrajene ena na drugi pod sinonimom "oblak" [11]. Splošno sprejeta definicija računalništva v oblaku prihaja od ameriškega Nacionalnega inštituta za standarde in tehnologijo (NIST). Če povzamemo celotno definicijo, je računalništvo v oblaku *model, ki nudi priročen, na zahtevo omogočen omrežni dostop do deljenih računskih virov (omrežja, strežniki, prostor, shranjevanje datotek, aplikacije in storitve), ki so hitro omogočena in dana v uporabo z minimalnim trudom in pomočjo ponudnika storitev.*

Diagram prikazuje tri značilne kategorije znotraj računalništva v oblaku: programska oprema kot storitev (SaaS), platforma kot storitev (PaaS) in infrastruktura kot storitev (IaaS). V nadaljevanju bomo vsako kategorijo podrobno razčlenili. Kljub temu lahko poenostavljeno zaključimo, da je:

- SaaS aplikacija namenjena končnim uporabnikom, dosegljiva preko spleta,

- PaaS platforma ključna hitrem in učinkovitem razvoju SaaS aplikacij,
- IaaS "motoršestavljen iz strojne in programske opreme, ki poganja obe višji komponenti.

Za boljše razumevanje si oglejmo naslednjo analogijo.

Infrastruktura sama po sebi ni pretirano uporabna. Tam stoji in čaka na nekoga, da jo uporabi. Predstavljajmo si državne ceste, ki so zgrajene in bi brez avtomobilov ali tovornjakov za prevoz ljudi ali dobrin bile neuporabne. V tej analogiji so ceste IaaS, tovornjaki in avtomobili PaaS in ljudje ter dobrine SaaS, če govorimo v tehničnem smislu.

Vendar je treba opozoriti na to, da kljub jasnemu razlikovanju, ki nam jo ponuja analogija, razlike postajajo vedno manj očitne, še posebno pri PaaS in SaaS in tako se bo tudi nadaljevalo.

Programska oprema kot storitev

Poenostavljena definicija Programske opreme kot storitve opisuje ta del kot programsko opremo, ki je dosegljiva preko spleta. Ponudnik storitve lahko omogoča dostop do aplikacij kot t.i. storitev na zahtevo, preko naročnin, predplačniško ali povsem brezplačno. Kot se izkaže, to ni povsem res, saj se take storitve financirajo preko drugih kanalov, npr. oglaševanje ali prodaja podatkov. V preteklih študijah so SaaS modelu napovedali več kot 10 odstotno letno rast in ta odstotek se je kar podvojil. Zelo hitra rast nakazuje, da bo SaaS kmalu znotraj vsake organizacije, zato je pomembno, da dobro razumemo, kaj to sploh je.

Rešitve, ki se prodajajo kot SaaS, morajo zadoščati splošno sprejetim značilnostim. Glavne značilnosti:

- spletni dostop do aplikacije,
- programska oprema je upravljana centralno,
- "one-to-many"model,
- uporabnikom ni treba upravljati z nadgradnjami in popravki,
- API za integracijo med različnimi sistemi.

Kot smo že rekli, je računalništvo v oblaku in še posebej SaaS metoda v bliskovitem razmahu. Organizacije, ki načrtujejo selitev v oblak, morajo napraviti odločitev, katero storitev bodo selili v oblak. Tako obstajajo nekatere smernice za primarne kandidate, ki so primerni za prvi premik v SaaS.

Poslovnemu svetu je bil SaaS model na široko predstavljen s prihodom Salesforca in njihovim CRM produktom. Kot eden prvih ponudnikov CRM-ja ni presenetljivo, da je danes Salesforce najpopularnejši SaaS ponudnik.

Kljub temu obstajajo področja, pri katerih SaaS ni najboljša rešitev, to so npr. aplikacije, pri katerih je zahtevana visoka hitrost procesiranja podatkov v realnem času, aplikacije, pri kateri zakonodaja določa, da podatki ne smejo biti shranjeni zunaj države (npr. javna uprava ter bančni sektor) ter aplikacije, za katere že obstoječe aplikacije zadovoljijo vse potrebe organizacije.

Prišli smo torej do sklepa, da je SaaS najbolj prepoznaven model računalništva v oblaku, vednar se v zadnjem času vedno bolj omenja PaaS, ki ga razvijalci in organizacije uporabljajo za inovativne rešitve. Ta model omogoča enostavnost razvoja programske opreme z močjo IaaS. SaaS ima namreč mnoge omejitve, ki jih PaaS nima. Razlike bomo pogledali v nadaljevanju.

Platforma kot storitev

Platformo kot storitev (PaaS) lahko opišemo kot računsko platformo, ki hitro in enostavno omogoča razvoj spletnih aplikacij brez potrebe po kupovanju in vzdrževanju infrastrukture, saj za to poskrbi ponudnik PaaS sistema.

Poenostavljeno rečeno je PaaS analogen SaaS s pomembno razliko, da je PaaS programska oprema, namenjena razvoju aplikacij, ki so dostavljene preko spleta.

Čeprav obstaja veliko različnih mnenj glede točnejših značilnosti PaaS modela, saj le-ta ni tako definiran kot SaaS, podajmo nekaj značilnosti:

- storitve za razvoj, testiranje, uvajanje in gostovanje aplikacij v istem razvojnem okolju - razvoj aplikacij,
- spletno osnovano orodje za razvoj, spreminjanje, testiranje različnih uporabniških vmesnikov,
- arhitektura, ki omogoča hkraten dostop za več uporabnikov, ki delajo na istem razvojnem procesu,

- vgrajena razširljivost programske opreme vključno z uravnoteženjem obremenitev in samodejnim preklopom v primeru izpada,
- integracija z ostalimi spletnimi servisi in podatkovnimi bazami z uporabo običajnih standardov,
- podpora za sodelovanje razvojne skupine,
- orodja za upravljanje naročnin in plačevanja.

PaaS je v grobem v marsičem zelo podoben IaaS modelu, ki ga bomo kasneje podrobno opisali, in nastopa v dveh značilnih oblikah:

1. Kot platforma za razvoj programske opreme z glavnim poudarkom na procesih znotraj programa, neodvisna od podatkovnega vira, ki služi aplikaciji. Zelo dober primer te vrste je Heroku, ki je v začetku uporabljal Ruby on Rails kot razvojni jezik. Heroku je bil razvit leta 2007 in je kasneje dodal podporo še številnim pomembnim programskim jezikom: Java, Node.js, Scala, Python, PHP in Perl.
2. Kot platforma, ki omogoča razvoj programske opreme hkrati z uporabo podatkov, ki se nahajajo v tem sistemu. Tak sistem nudi metode za ustvarjanje aplikacij s podatki, ki so skupnega tipa ali oblike. Poenostavljeno, podatki so odvisni od sistema, ki ponuja tak model. Primer takega modela predstavlja platforma Force.com od podjetja Salesforce, na kateri je možen razvoj izključno z njihovim programskim jezikom.

Tudi pri PaaS modelu lahko definiramo področja, na katerih je PaaS primeren, ter tudi področja, kjer PaaS vseeno ni najboljša rešitev. PaaS je izjemno uporaben v situacijah, kjer več razvijalcev sodeluje na istem razvojnem procesu ter obstajajo zunanji člani, ki so povezani z razvojnim procesom. Ta model je izjemno uporaben v tistih primerih, kjer že obstaja podatkovni vir kot npr. prodajne informacije iz CRM in bi radi z aplikacijo izkoristili te podatke. Nenazadnje je PaaS uporaben v primerih, ko je potrebno visoko avtomatizirano testiranje in uvajanje storitev. Primeri: Force.com, Google App Engine, Heroku. Nakazuje se, da bo v prihodnjih letih rast PaaS kar 30 % [10], kar je zavoljo hitrejšega razvojnega procesa in zmanjševanja infrakstrukturnih stroškov zelo realno. Kljub prepričljivo

dobrim lastnostim tega modela pa naštejmo nekaj možnosti, ko PaaS ni najboljša možnost:

- kadar morajo aplikacije zagotoviti prenosljivost v primerih, da so gostovane,
- kadar (lastni) razvojni jeziki in pristopi močno vplivajo na razvojni proces,
- kadar (lastni) razvojni jeziki ovirajo kasnejši prestop k drugemu ponudniku. Tak primer je lahko Force.com, čigar razvojni jezik (APEX) je uporaben le na tej platformi,
- kadar aplikacija zahteva prilagoditev strojne in programske opreme, na kateri leži.

Infrastruktura kot storitev

Infrastruktura kot storitev (IaaS) je standardiziran model, kjer je oblačna infrastuktura - strežniki, prostor, omrežja in operacijski sistemi možna kot storitev na zahtevo. Glavna prednost za uporabnike predstavlja dejstvo, da jim naštete opreme ni treba kupiti, ampak so v celoti v domeni zunanjih izvajalcev. Tudi IaaS lahko razdelimo v dve večji skupini. V splošnem lahko ločimo javni ali privatni del, lahko pa tudi kot kombinacijo obeh. "Javni oblak"predstavlja tisti del, ki nudi deljene vire in se nahaja zunaj podjetja. Uporabnik svojo aplikacijo namesti preko oddaljene povezave (lahko tudi preko brskalnika). V nasprotju s tem pa so viri v žasebnem oblakužnotraj podjetja, katere si lahko priskrbi podjetje samo ali pa za to poskrbi drugo podjetje (CISCO, Microsoft, Oracle). Take vrste oblak posnema vrsto značilnosti običajnega računalništva v oblaku. Če želimo združiti prednosti obeh vrst oblakov, se lahko odločimo za "Hibridni oblak". Že ime jasno nakazuje, da je to kombinacija zasebnega in javnega oblaka. Zasebni del skrbi za varno hrambo notranjih virov, zunanje vire pa zagotavlja ponudnik. Največkrat se take rešitve uporabljajo v primerih, v katerih je glavni dejavnik varnost podatkov znotraj podjetja ter mora rešitev zagotavljati ustrezno nadgradljivost, kar pa nam učinkovito lahko nudi le javni oblak. Primer takega oblaka je Amazon VPC, ki je del Amazonovega oblaka (AWS) [9]. Amazon daje uporabniku ves nadzor nad privatnim oblakom, hkrati pa preko spletnih storitev (WS) omogoča dostop do javnega oblaka.

Značilnosti:

- viri so na voljo v obliki storitve,
- dinamično lahko povečujemo zmogljivost virov,
- spremenljivi stroški glede na koriščenje virov,
- večuporabništvo na enem kosu strojne opreme.

Na trgu obstaja velika množica ponudnikov IaaS storitve, med katerimi prevladujeta Amazon z AWS in Rackspace. Kot že omenjeno, se meja med IaaS in PaaS močno zabrisuje, saj ponudniki IaaS vgrajujejo orodja, ki omogočajo enostavno postavitev različnih okolij, kar je po definiciji že v domeni PaaS modela.

Področja uporabnosti PaaS:

- področja z zelo spremenljivimi zahtevami, pri katerih se pojavljajo vrhovi in padci na ravni obremenjenosti aplikacij,
- zlasti za mlade organizacije, ki nimajo sredstev za nakup lastne infrastrukture,
- za hitro rastoče organizacije, kjer je stalno povečevanje zmogljivosti virov problematično,
- za posebne vrste organizacij, kjer je določena storitev le v preizkusni dobi ali le v začasnem delovanju.

Področja, kjer PaaS ni najboljša možnost:

- kjer so regulative za organizacijo takšne, da je skladiščenje podatkov zunaj države in povečevanje zmogljivosti težavno,
- kjer so zahtevani izjemno visoki zmogljivostni nivoji delovanja,
- kjer obstoječa infrakstruktura že zadovolji potrebe organizacije.

Sklep

V opisih različnih modelov računalništva v oblaku smo ugotovili, da le-tega ne moremo opisati kot določeno storitev, ampak da je to splošni pojem za storitve, ki zajemajo večplastno arhitekturo aplikacij od infrastrukture, ki je temelj vsega,

do platforme, ki ponuja razvojno okolje za programsko opremo, ki nadomešča nameščeno programsko opremo na sistemu organizacije.

Za vsako organizacijo je pomembno, da razume različne modele in poglede na računalništvo v oblaku in oceni situacijo ter tako ugotovi, kateri model je najprimernejši za njene potrebe.

2.2.2 Heroku

Heroku je tipičen primer PaaS modela, ki samega sebe opisuje kot oblačna aplikacijska platforma [7]. Vizija te storitve je ta, da razvijalcem omogoča razvoj aplikacij brez ukvarjanja s strežniki, uvajanjem programske opreme in spremljajočih procesov ter povečevanje zmogljivosti. Omenjena platforma podpira številne programske jezike in zato ji pravimo kar večjezična platforma (ang. polyglot platform) [8]. Prve platforme (PaaS) so stremele k temu, da se razvijalci programske opreme pridružijo tej skupnosti, ki namenoma podpira le en programski jezik. V naslednjih letih se je, kljub vztrajanju nekaterih, da obdržijo število programskih jezikov, ki jih podpira določena platforma, pokazala realnost, da moderne aplikacije zahtevajo vse prej kot homogenost. To pomeni, da vedno več aplikacij zaradi različnih prednosti posameznih programskih jezikov uporablja različne le-te istočasno, tako da je platforma, ki podpira vse te jezike, logična posledica. Prav to je Heroku omogočil že zelo zgodaj in si pridobil prednost na trgu.

V osnovi je Heroku začel z Ruby on Rails, vendar je v naslednjih letih dodal podporo še za Java, Python, Clojure, Scala, Node.js ter PHP in Perl, ki sta uradno nedokumentirana. Heroku uporablja operacijski sistem Debian, ki je v zadnji nadgradnji prešel na Ubuntu, ki še vedno bazira na Debianu.

Dejstva, ki postavljajo platformo Heroku na eno izmed vodilnih med PaaS ponudniki so:

- Izjemno enostavna namestitev aplikacije. Za namestitev se uporablja "git repozitorij", ki je tako ali tako že močno razširjen med razvijalci.
- Zanesljivost podpore za Ruby (kot osrednji programski jezik na Heroku), saj se je organizaciji, ki skrbi za Heroku, pridružil izumitelj jezika Ruby.
- Heroku je bil prevzet s strani Salesforca ter sklenil partnerstvo s poslovno svetovalno organizacijo Accenture.

Delovanje Heroku platforme

Jedro te platforme predstavlja sklad, ki je poimenovan Čedar". Uporablja orodje git za vodenje različic kode za katerikoli jezik in na ta način ustvarja repozitorije na Heroku-ju. Za osnovno enoto delovanja uporablja enoto, ki je poimenovana *dyno*. Dyno je enota računske moči, ki zagotavlja enostavno in izolirano okolje, ki poganja aplikacijo. Razširljivost zmogljivosti platforme je samo še vprašanje dodajanja dodatnih enot. Heroku omogoča dve vrsti dyno-tov:

1. **Web Dyno** - lahko izvede vse zahteve, še posebej je namenjen HTTP zahtevam. Uporaben je za vse vrste spletnih strani. Zahteva mora biti izvedena v času 30 sekund.

2. **Worker Dyno** - izvaja procese v ozadju *(background jobs)*, tipično izvaja zahteve v čakalni vrsti. Uporaben je predvsem za zahtevne izračune, ki trajajo dlje časa. Ni časovne omejitve glede trajanja zahteve.

Ne glede na vrsto dyno-ta, ki ga izberemo, je cenovna politika Heroku-ja taka, da je sedaj en dyno brezplačen za uporabo (2014).

2.2.3 Force.com

Platforma Force.com, ki je le del celotnega ekosistema Salesforce, je še en primer PaaS modela. V nasprotju s Heroku-jem je Force.com zaprtega značaja, kar se tiče programskih jezikov. Kljub temu, da je enako kot Heroku last Salesforca, lahko na tej platformi razvijamo v enem programskem jeziku, ki se imenuje APEX. Platforma je zasnovana tako, da poenostavi razvoj in uvajanje oblačno baziranih aplikacij in spletnih strani. Glavna razlika med ostalimi PaaS platformami, ki so bolj odprtega značaja, je ta, da je Force.com usmerjen v poslovno okolje z močno integracijo s Salesforce CRM orodjem. Opredelimo lahko glavna področja, za katera je platforma namenjena:

- za podporo poslovnim procesom organizacije,
- za razvoj aplikacij po meri z možnostjo objave na spletno tržišče AppExchange.

V tej diplomski nalogi nas predvsem slednje, saj smo razvili aplikacijo, ki jo lahko uporabi vsaka organizacija z namestitvijo preko AppExchange.

Vse, kar nam Salesforce nudi s *Sales Cloud*, *Service Cloud* ali *Data.com*, lahko zgradimo na Force.com platformi, kar pomeni, da vse temeljijo na tej platformi. Treba je opozoriti, da Force.com kot platforma ne nudi CRM funkcionalnosti, vendar imamo vse možnosti, da si jo razvijemo po meri. Kot smo že omenili, se bomo v tej diplomski nalogi osredotočili tistemu delu, ki omogoča razvoj poslovnih aplikacij in objavo na spletno tržišče AppExchange.

Ena vidnih razlik, ki Force.com ločuje od ostalih PaaS platform je ta, da omogoča visoko prilagodljivost z množico t.i. "out-of-the-box"orodij in funkcij, katere lahko končni uporabnik brez znanja programiranja enostavno prilagodi. Obstajata dve vidnejši vlogi pri prilagoditvah: administrator ter razvijalec.

Administrator

Vloga administratorja je bdenje nad varnostnimi nastavitvami, omogočanje/neomogočanje funkcij, skrb za delovne tokove, odobritvene procese, delitvena pravila, dodajanje uporabnikov ter skrb za okolje v splošnem. Izjemno pomembno je, da imamo izkušenega administratorja za to vlogo.

Razvijalec

V nasprotju z administratorjem je naloga razvijalca, da izkoristi pravo moč platforme z uporabo t.i. Apex razredov in sprožilcev. Apex je programski jezik, posebej razvit za platformo Force.com, ki skrbi za interakcijo med objekti (tabelami), v katerih so shranjeni podatki, z ostalimi funkcijami platforme. Salesforcova platforma je zelo stabilna in zanesljiva, a kljub temu ima svoje omejitve, ki jih preseže Apex, saj poskrbi za kompleksne poslovne scenarije, ki so lastni vsaki organizaciji posebej. Implementacijo takih procesov na platformi Force.com rešujemo z Apex razredi, prožilci podatkovne baze, Visualforce stranmi, kontrolerji in ostalimi funkcijami. Poleg tega omenimo še možnost integracije z drugimi obstoječimi aplikacijami (AppExchange) ali s pisanjem spletnih servisov (ang. webservice) za povezovanje z zunanjimi spletnimi aplikacijami.

2.3 Vprašanje varnosti

Ena izmed bolj izpostavljenih problematik glede računalništva v oblaku je zagotovo vprašanje varnosti. S pojavom računalništva v oblaku se s tempom brez primere preoblikuje poslovno okolje in prav to prinaša s seboj nove varnostne izzive. Prihod

oblačnih modelov, ki smo jih že opisali, učinkovitejše rešuje poslovne procese kot kadarkoli prej. Pri vseh dodanih vrednostih, ki jih taki modeli omogočajo, se toliko bolj pojavljajo varnostne ranljivosti, ki odpirajo nova varnostna vprašanja.

Cloud Security Alliance (CSA) kot neprofitna organizacija je v ta namen sprožila postopke sprejetja splošnih standardov za učinkovito oblačno varnost. Sprva so izdali priročnik *Security Guidance for Critical Areas of Focus in Cloud Computing* [2]. Kmalu je priročnik postal razširjen standard pri uporabi najboljših praks, ki zadevajo oblačno varnost. V vseh letih obstoja od leta 2008 je njihov cilj promoviranje uporabe dobrih praks za zagotavljanje varnosti pri računalništvu v oblaku. Leta 2013 so od posameznikov iz tovrstne panoge najprej pridobili strokovno mnenje in z delovno skupino izdali končno poročilo za leto 2013, ki zajema 9 največjih varnostnih groženj [3].

1. **podatki v rokah tretje osebe**

   Podatki v oblačnih sistemih so shranjeni na virtualnih napravah, ki so lahko na istem fizičnem strežniku. V primeru, da večodjemalski oblačni sistemi niso pravilno zasnovani, lahko napaka v določeni aplikaciji pomeni priložnost za napadalca, da pridobi dostop do vseh podatkov, ki so shranjeni na takem napačno načrtovanem oblaku. Napor, ki ga zaradi nevarnosti izgube podatkov vlagamo v varnost, nam lahko povzroči druge težave, npr. šifriranje s šifrirnim ključem nam v primeru izgube le-tega onemogoči dostop do lastnih podatkov, itd.

2. **izguba podatkov**

   Za človeka ali podjetje je misel na izgubo podatkov zastrašujoča. Obstaja veliko primerov, ko so napadalci zlorabili uporabniške račune in iz njih izbrisali vse shranjene podatke. Prav tako ni mogoče izključiti napake ponudnika storitev. V letu 2011 je eden izmed spletnih velikanov (Google) zaradi sistemske napake resetiral kar 150.000 uporabniških računov, od katerih so jih le tretjino lahko povrnili. Zaradi takšnih možnih scenarijev večina ponudnikov oblačnih storitev uporablja varnostne kopije.

3. **ugrabitev računa ali storitve**

   To je starejša metoda, ki uporablja metode, kot so spletno ribarjenje, goljufije ali izkoriščanje lukenj programske opreme za pridobitev dostopa do

uporabniškega računa. Obenem je v splošni praksi razširjeno recikliranje gesel, kar pomeni, da enako geslo uporabljamo na različnih storitvah, kar še povečuje možnost napada. Oblačne rešitve predstavljajo novo nevarnost za take napade, saj omogočajo še obsežnejšo manipulacijo s podatki.

4. **nezanesljivi vmesniki in API-ji**

   Ponudniki oblačnih storitev omogočajo dostop navzven (API) za komunikacijo z drugimi aplikacijami. Varnost in dostop do celotnega oblačnega sistema je odvisna od zanesljivosti in varnosti le-teh. Ponudniki morajo poskrbeti od avtentikacije, kontrole dostopa do šifriranja in spremljanja aktivnosti za zaščito pred slučajnimi ali namernimi poskusi pridobitve dostopa. V nadaljevanju bomo opisali razširjen standard OAuth 2.0, ki velja za zanesljivega. Zelo je pomembno, da se tudi uporabnik zaveda varnostnih posledic ob uporabi teh vmesnikov in na ta način naredi vse, da zmanjša to nevarnost.

5. **zavrnitev storitve (DoS)**

   Enostavno povedano DoS napad povzroči nedosegljivost storitve s tem, da pošilja neobičajno veliko število zahtev, kar povzroči veliko porabo procesorske moči, spomina ali pasovne širine in tako sistem začne delovati počasi. Tako uporabnik postane nezadovoljen zaradi nedelujoče storitve. Z ozirom na to, da veliko ponudnikov uporabnikom zaračunava stroške glede na porabo procesorske moči ali drugih parametrov, se lahko pojavijo visoki stroški zaradi tovrstnih napadov. Tako je uporabnik prisiljen opustiti storitev zaradi stroškov.

6. **zlonamerni uporabniki**

   Okoli tega se v varnostni stroki veliko govori, vendar kljub temu ne moremo določiti prave stopnje te nevarnosti. Dejstvo je, da je ta grožnja prisotna. To so lahko sedanji ali bivši zaposleni podjetja ali poslovni partner, ki je vzpostavil sistem in je tudi imel podatke za dostop, kar je lahko izkoristil za namerno zlorabo pravic dostopa. Organizacijski sistemi, ki se pri varnosti zanašajo le na ponudnika oblačne storitve so bolj izpostavljeni tej nevarnosti. Za dostopne podatke mora poskrbeti vsaka organizacija.

7. **zloraba oblačnih storitev**

Prednost oblačnih sistemov je vsekakor velikost računske moči, do katere ima lahko dostop tudi majhna organizacija. Razširjen dostop do strežnikov v oblačnem sistemu je veliko bolj realen. In ravno v tem je prednost za napadalce, ki lahko uporabijo to ogromno računsko moč za npr. razbitje šifrirnega ključa, kar z vzporedno uporabo več takih sistemov lahko postane realna grožnja. Druga težava so DoS napadi ali tudi razpečevanje nelegalne programske opreme. Ta nevarnost bolj preži na ponudnike oblačnih storitev kot na uporabnike.

8. **nezadostno razumevanje**

   Oblačni sistemi obljubljajo celo vrsto pozitivnih učinkov, kot so zmanjšanje stroškov, učinkovitost procesov in izboljšanje varnosti. Velikokrat se zgodi, da se organizacije, ki sicer imajo vire za uvedbo tehnologije, prehitro podajo v to, brez celostnega razumevanja. To prinese številne težave, kot so varnostna tveganja in neskladja med ponudnikom in stranko. Stranka razvije aplikacijo, ki jo želi namestiti v oblak ter se hkrati ne ozira na arhitekturo ali omejitve. Ključno je, da stranka popolnoma razume, kaj za njegovo organizacijo pomeni nov tehnološki model.

Poglavje 3

CRM

3.1 Definicija in vrste CRM-ja

Upravljanje odnosov s strankami (ang. CRM - customer relationship management) je široko izvedena strategija, ki se osredotoča na odnose s strankami ter kupce in manj na produkte [1]. Tako s strategijo maksimizira informacije o stranki, želi ugotoviti strankine kupne navade, povečati lojalnost stranke in uspešno komunicirati s stranko. V splošnem je glavni cilj poiskati čim več dobičkonosnih strank in jih obdržati. To strategijo že v večini podpirajo s CRM sistemi, t.j. programskimi rešitvami, ki olajšajo in avtomatizirajo delo s strankami.

Obstaja veliko definicij, kaj CRM je, a kljub temu je večini skupna osredotočenost na poznavanje strank. Naštejmo nekaj najbolj znanih defenicij:

- CRM je prva in najodličnejša strategija ter korporacijska filozofija, ki postavlja stranko v ospredje poslovnega dogajanja tako, da bi povečala zaslužek s povečanjem pridobivanja strank in njihovim zadržanjem. Vključuje iskanje najboljših strank in avtomatizacijo procesa, da bi prodaja, marketing in storitve bile učinkovitejše. CRM skrbi za 360 stopinjski pogled na stranko in povezuje vse potrebne informacije o stranki. Glavni integrirani sistem, ki sestavlja CRM, vključuje upravljanje s podjetji in osebami, avtomatizacijo prodaje, e-poštni marketing, sistem za podporo strankam in integracijo z zalednimi aplikacijami (Doshi R., 2006).

- CRM ni samo skupek tehnoloških orodij, ampak ga lahko definiramo kot

poslovni proces, s katerim skušamo povečati dobiček, povečati tržni delež in zadovoljstvo stranke. CRM se prične s celovito analizo, kaj stranka potrebuje in kaj želi in s tem na koncu prinaša celovito informacijo o stranki (Zigarelli, 2001).

- V poslovnem svetu se pojem zadovoljstvo stranke pojavlja vse pogosteje, zdi se, kot da gre za neko vrsto fiksne ideje, da podjetje potrebuje le zadovoljno stranko, ki je potem običajno lojalna podjetju in edina zanimiva (Lowenstein, 1997).

- Uvedba CRM-ja za normalno in koristo uporabo zahteva kar nekaj ukrepov, kar pomeni integracijo oddelkov v podjetju, ki so kakorkoli povezani s strankami oz. v stiku z njimi. To združevanje ima velik vpliv na odnos s strankami, s čimer se povečuje tudi konkurenčna prednost(Chiablo, 2004).

V splošni literaturi je mogoče zaslediti najpogostejšo delitev na:

1. Operativni CRM - nudi podporo različnim poslovnim funkcijam (ERP sistemi, klicni centri, avtomatizacija prodaje ...), ki zajemajo neposredni stik s stranko, navzkrižno prodajo, trženje in podporo strankam. Zagotavlja podporo poslovnim procesom, kot sta trženje in marketing. Vsak stik s stranko je dodan k njegovi bazi podatkov, tako da lahko zaposleni kadarkoli dobi informacijo o določeni stranki iz baze podatkov.

2. Analitični CRM - omogoča vpogled v stranko, razumevanje njenega vedenja, napovedovanje trendov, analizo dobičkonostnosti in v podatke ter analize. Vsebuje rešitve za upravljanje poslovne učinkovitosti, kot recimo analiza povpraševanja, dobičkonosnost produktov in storitev ter analize tržnih kampanj.

3. Kolaborativni CRM - omogoča in vključuje sodobna orodja za sodelovanje in komunikacijo s strankami, med uslužbenci in s poslovnimi partnerji. V to skupino štejemo elektronsko pošto, spletne strani in klicne centre.

3.2 Trendi

Trend, ki je trenutno v ospredju in se bo bolj pokazal v naslednjih letih, je zagotovo mobilni CRM. Poleg tega vstopa v ospredje tudi *socialni CRM*, ki se vedno bolj integrira z *mobilnim CRM-jem* [17].

Mobilni CRM

Že trendi na področju uporabe mobilnih naprav nakazujejo, da je koncept mobilnega CRM-ja v razmahu. Podjetja, ki želijo konkurirati na trgu, se zavedajo, da morajo njihovi zaposleni biti vedno v realnem času dosegljivi strankam in hitro reagirati na njihove zahteve ter hkrati sodelovati z ostalimi člani prodajne skupine. Prodajalci, ključni uporabniki CRM sistemov, so vedno bolj na poti, zato potrebujejo mobilna orodja za pridobivanje novih priložnosti in ustvarjanju bodočih strank. Uporabniki CRM-ja se zavedajo prednosti, ki jih imajo z uporabo in zato nadaljujejo z razširjenjem teh rešitev tudi na mobilno platformo, s katero lahko nadzirajo, upravljajo z bodočimi strankami, skrajšajo prodajni proces in izboljšajo podporo strankam.

Socialni CRM

Ta tip CRM-ja se vedno bolj razvija in bo imel v prihodnosti močan vpliv na razvoj CRM industrije. Za izboljšanje uporabniške izkušnje bodo morale organizacije pridobiti še več podatkov o uporabnikih. Vsebina, objavljena na socialnih omrežjih strank, pripomore k natančnejšemu določanju pravih storitev za stranko. Organizacije se zavedajo, da združevanje teh podatkovnih kanalov z notranjimi viri predstavljajo velik potencial in tako CRM ni več omejen le na podatke, ki so znotraj njega. Pravo vrednost teh podatkov nam da kombinacija zunanjih virov, virov iz socialnih omrežjih, javno dostopnih virov ter seveda notranjih virov.

Integracija

CRM sistemi so na poti razvoja ubrali neodvisno in ločeno pot od ostalih informacijskih sistemov (npr. ERP, poslovna inteligenca). Toda za učinkovito analizo strank se potrebuje vedno več in več podatkov, kar CRM sili v povezovanje z drugimi sistemi. Učinkovit CRM se mora povezovati z ostalimi rešitvami v podjetju, da lahko zagotovi kar najboljše informacije o strankah. Primeri povezovanja: računovodski program, ERP, podatkovne baze, e-poštni marketing, spletne trgovine.

3.3 Najbolj znani CRM sistemi

Trenutno je moč najti številna orodja CRM, vendar bomo izpostavili le štiri najbolj znane po Gartnerjevem diagramu (Slika 4.1).

Salesforce CRM

Je zelo močno orodje na področju prodaje. Njegova posebnost je chatter, ki omogoča preprosto komunikacijo med prodajnim osebjem in managementom in tako pomaga, da so vsi na tekočem z informacijami ves delovni dan. Je najbolj znan CRM sistem in že dve leti zaporedoma prvi na lestvici po priljubljenosti. Ima Namenjen je tako za majhna kot za velika podjetja. Dostopen je izključno preko spletnega brskalnika. Cenovno se rangira od 55 - 125 $ na uporabnika na mesec za tipično licenco, čeprav je lahko cena tudi nižja ali višja glede na dodane module.

Oracle CRM

Oracle CRM je širok pojem za CRM, ki vsebuje tri rešitve.

- Oracle CRM
- PeopleSoft
- Siebel V letih razvoja Oracle CRM je podjetje Oracle prevzelo PeopleSoft CRM.

SAP CRM

Microsoft Dynamics CRM

Microsoft ponuja njihov CRM v oblačnem modelu kot tudi "na mestu uporabe". Oba produkta sta več ali manj funkcionalno enaka, razen seveda v načinu uporabe. Največja prednost pred ostalimi je polna združljivost z ostalimi Microsoftovimi aplikacijami, kar je zelo priročno za podjetja, ki že uporabljajo večino Microsoftovih programov. Uporabniški vmesnik je preprost in enostaven za uporabo. Zagotavljajo 99.9 % neprekinjenega delovanja, kar se v praksi velikokrat izkaže za neresnico. Ne vključuje podpore za socialne medije in slabo podpira dodatke od tretjih oseb. Podpira le Internet Explorer. Za oblačni tip je cena 65 $ na uporabnika na mesec.

3.4 Integracija CRM sistemov

Sistemi za upravljanje s strankami le redko delujejo ločeno od ostalih sistemov. Ob uvedbi CRM-ja organizacija uporablja druge sisteme, ki ji služijo za hranjenje podatkov o stranki. Cilj je, da te podatke iz različnih virov pripeljemo v CRM, ki nam služi kot centralno skladišče podatkov o stranki. Najbolj značilna je integracija z obstoječo IT infrastrukturo, klicnimi centri, ERP sistemi, dokumentnimi sistemi, HR sistemi in računovodskimi sistemi. Integracija z drugimi sistemi vedno bolj postaja ena od najpomembnejših funkcij v CRM-ju. Zelo zgovoren je podatek, da je od 1,3 milijarde transakcij dnevno na Salesforce CRM kar 60 odstotkov takih, ki pridejo preko integracije (API). Značilnosti CRM integracije:

- ni opazno, da se aplikacija povezuje navzven,
- delovanje je avtomatizirano,
- potrebna je zavarovana povezava (OAuth 2.0),
- podatki se sinhronizirajo v realnem času (ni vedno nujno),
- integracija lahko poteka v obe smeri (navznoter ali navzven).

Pasti:

- majhna sprememba v enem od sistemov lahko povzroči, da integracija pade (potrebne so spremembe na obeh sistemih),
- pomanjkljiva izvedba lahko privede do varnostnih težav (podatki se nezavarovano prenašajo po omrežju),
- nestrukturirani podatki, ki jih želimo pripeljati v CRM povzročijo ogromno dela,
- podvajanje podatkov (težavno dedupliciranje),
- težavna implementacija zaradi številnih protokolov in aplikacij.

Poglavje 4

Salesforce

Osrednje področje diplomske naloge se v precejšnji meri dotika sistema Salesforce CRM, zato ga bomo tudi podrobnejše opisali. Po Gartnerjevem diagramu 4.1 je Salesforce vodilni na področju avtomatizacije prodaje (SFA) in to dokazuje z rastjo poslovanja in inovacij, kljub temu da ima produkt visok cenovni vstopni prag. Prepoznavna znamka, uporabnost, inovacije in dokazne reference prepričajo vedno več strank za Salesforce [18]. Njihovi ISV (neodvisni ponudniki programske opreme) tvorijo močan ekosistem, s katerimi Salesforce prodira na trg in pridobiva nove stranke. Kljub temu lahko prilagoditve in dopolnitve njihovega osnovnega produkta s strani ISV stanejo kar precej. Najbolj je razširjen v Severni Ameriki, zato Salesforce potrebuje geografski prodor še v druga območja. Nedavni nakup podjetja ExactTarget bo verjetno pospešil proces od priložnosti do potencialne stranke. Salesforce svoje produkte združuje v tri glavne skupine, to so prodajni oblak (ang. Service cloud), storitveni oblak (ang. Service cloud) in platforma Force.com.

4.1 Prodajni oblak

Ta oblak je najbolj znan produkt, ki ga ponuja Salesforce. Vsebuje SFA (Sales Force Automation) modul [26] in na prvi pogled ni bistveno drugačen od ostalih CRM rešitev. Namenjen je avtomatizaciji prodajnega procesa in vsebuje orodja, ki so intuitivna in preprosta za uporabo ter imajo enostaven uporabniški vmesnik.

Prodajni oblak vključuje osnovne in napredne funkcionalnosti, kot so upravlja-

Slika 4.1: CRM Gartnerjev diagram

nje s podjetji, kontakti in iskanje novih sledi, mobilni dostop, integracija z naprednim e-poštnim marketingom, socialna platforma Chatter, upravljanje s priložnostmi in ustvarjanje prilagojenih poslovnih poročil. Pri dražjih različicah vsebuje tudi napovedovalne metodologije, funkcionalnosti delovnega toka in potrditvene procese ter razvoj aplikacij po meri. Cene se gibljejo od 5 $(licenca Contact Manager) do 300 $(licenca Performance Edition) na uporabnika.

4.2 Storitveni oblak

Storitveni oblak [27] štejemo v SaaS model in je znan kot aplikacija za storitve za stranke. Pomaga podjetjem, da upravljajo z zahtevami, mnenji ali pritožbami strank, ki prihajajo iz različnih kanalov, na eni priročni platformi. V današnjem času hiperpovezav podjetja komaj sledijo vsem informacijam, ki jih puščajo stranke in je za njih zelo pomembno, da je ravnanje s strankami priročno in hitro. Ta oblak je namenjen dvigu uporabniške izkušnje na najvišji možni nivo. Vsebuje orodja za spremljanje in sprejemanje zadev preko različnih socialnih omrežij (Facebook, Twitter) direktno v oblak.

Tudi v tem oblaku so funkcionalnosti odvisne od izbire izmed treh različic. V splošnem storitveni oblak omogoča integracijo s telefonskimi sistemi, upravljanje z zadevami, zajem zadev iz komunikacijskih kanalov, mobilni dostop, integracijo s socialnimi omrežji, spletni klepet za stranke, bazo znanja, delovne tokove in potrditvene procese ter tudi veliko funkcij s Force.com platformo.

4.3 Oblak po meri - Force.com

Force.com je platforma za razvoj aplikacij po meri [6]. Omogoča razvijalcem, da zgradijo optimalne aplikacije za posebne poslovne potrebe. Razvoj poteka direktno na platformi in take aplikacije se lahko povezujejo z ostalimi sistemi. Administratorski vmesnik je povsem enak tistemu na prodajnem oblaku, vendar ne vsebuje njegovih funkcionalnosti. Omogoča integracijo z rešitvami tretjih oseb, kot so Oracle, SAP, Microsoft, Google AppEngine, Amazon WebServices v realnem času. Kljub temu da ne vsebuje standardnih CRM funkcionalnosti, je možnosti za razvoj praktično neomejeno. S pomočjo dodatkov, ki si jih lahko naložimo preko spletnega

tržišča AppEchange, lahko v kratkem času izdelamo uporabno aplikacijo.

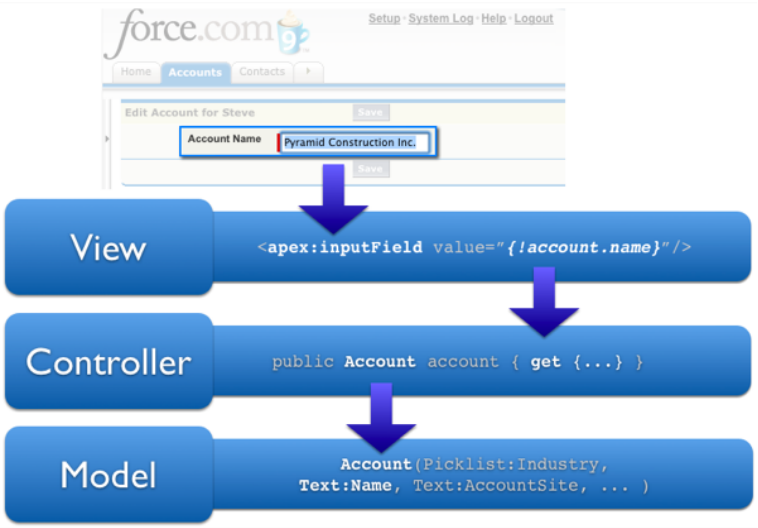
4.3.1 Arhitektura

Temelj Force.com platforme je metapodatkovno grajena programska arhitektura, ki omogoča večodjemalske aplikacije. Vse nastavite in konfiguracije znotraj platforme so takoj na voljo v XML obliki z natančno določeno shemo. Dostopni so preko spletnih storitev (metadata API) z IDE vtičnikom za Eclipse. Lahko jih uporabimo kot vir za nadzor izvorne kode ter prenos nastavitev na drugo okolje. Večodjemalski model je temeljni princip [25] pri računalništvu v oblaku in se uporablja za stroškovno optimalno in varno deljenje IT virov. Lahko si ga predstavljamo kot stanovanjski blok, ki uporablja isto infrastrukturo, kljub temu pa ima stene in vrata, ki omogočajo zasebnost vsakemu odjemalcu. Force.com platforma deluje dobro in učinkovito zaradi tega, ker so razvijalci na Salesforce-u razvili to storitev na podlagi dveh pomembnih principov:

- narediti moramo vse, kar se da učinkovito,
- platforma mora pomagati razvijalcem, da naredijo vse, kar se da učinkovito.

4.3.2 Podatkovna baza

Platforma uporablja relacijsko podatkovno bazo, ki dovoljuje definiranje tabel po meri (objekti). Tipi podatkov v poljih so natančno določeni, uporabljajo se standardni podatkovni tipi. Na objektih se lahko uporabljajo validacijska pravila za zagotovitev pravilnosti podatkov pred vnosom v bazo, delovne tokove in potrditvene procese za kompleksne poslovne procese. Omogoča tudi definiranje formul na poljih, ki se obnašajo kot celice na preglednicah in vsebujejo različne funkcije in sumarizacijo na povezanih tabelah. Nad podatkovno bazo se poizvedbe izvajajo s povpraševalnim jezikom SOQL (ang. Salesforce Object Query Language), iskanje besedila po vseh poljih pa z iskalnim jezikom SOSL (ang. Salesforce Object Search Language). Podatki so lahko dostopni tudi preko poročil po meri (ang. reports). Objekti z vklopom sledenja zgodovini omogočajo sledenje spremembam na podatkih, kar pri poslovanju velikokrat zahteva zakonodaja. Podatkovna baza je tesno povezana z vsemi lastnostmi na platformi in je največkrat uporabljen način



Slika 4.2: Paradigma MVC

shranjevanja. Pri uporabi razvijalcem ni treba nastavljati podatkovne baze ali nameščati gonilnikov za dostop. Force.com podatkovna baza uporablja Oraclove podatkovne strežnike in za vso konfiguracijo poskrbi platforma sama.

4.3.3 Visualforce

Visualforce opisujemo kot ogrodje za izdelavo uporabniških vmesnikov, ki uporablja lastne komponente, podobne HTML oznakam. Trenutno vsebuje več kot 100 vgrajenih komponent, vendar ima razvijalec možnost razvoj lastnih komponent po meri. Uporablja se tradicionalna paradigma (MVC), z možnostjo avtomatično generiranih kontrolerjev za podatkovne objekte, ki zagotavljajo enostavno in tesno integracijo s podatkovno bazo. Visualforce strani oblikujemo z uporabo vgrajenih značk (<apex>), s standardnimi oznakami HTML ali z ostalimi standardnimi spletnimi tehnologijami za izboljšanje uporabniškega vmesnika (JavaScript, CSS, Flash). Vsaka stran je dostopna preko unikatnega URL naslova (/apex/mypage). Pri dostopu na stran strežnik generira vsebino in v odvisnosti od logike, ki lahko komunicira s podatkovno bazo ali upravlja s spletnimi servisi, prikaže vsebino uporabniku. Logiko lahko definiramo s pomočjo Apex programskega jezika, ki ga na stran vključimo v obliki kontrolerja. Interakcija med stranmi in kontrolerji je avtomatična in ni potrebe po implementaciji kompleksne JavaScript logike. Na

elementih, ki podpirajo interakcijo z Apex razredi, le določimo metodo znotraj razreda z atributom "action"in s tem dosežemo komunikacijo. Če želimo uporabiti funkcionalnosti principa AJAX, uporabimo rerender atribut, ki določa, kateri element na strani mora biti posodobljen in za vse to poskrbi vgrajena logika. Visualforce strani so v osnovi le del zaprtega okolja, ki zahteva uporabniški račun Force.com, vendar lahko omogočimo funkcionalnost Force.com Sites in s tem spletnim uporabnikom omogočimo dostop brez uporabniškega računa.

4.3.4 Apex

Programski jezik APEX je v veliko pogledih podoben Javi in je bil razvit za razvijalce, ki želijo izdelovati poslovne aplikacije po meri. Apex lahko nastavlja programske spremenljivke in konstante, izvaja tradicionalne kontrolne stavke (if-else, for zanke, itd.), upravlja z DML (ang. data manipulation language: insert, update, upsert, delete) postopki, uporablja povpraševalna jezika SOQL in SOSL, izvaja spletne klice, itd. Uporabljamo ga lahko v dveh oblikah: kot Apex razred z metodami, ki se izvaja na zahtevo ali kot prožilec (ang. trigger), ki se avtomatično izvede po določeni spremembi v podatkovni bazi. Ima tudi dobro podporo spletnim storitvam. Metode pisane v Apexu lahko izpostavimo kot REST ali SOAP spletne storitve ali jih nastavimo, da se asinhrono prožijo po določenem urniku. Vsebuje tudi avtomatična preverjanja povpraševalnih stavkov v podatkovno bazo, ki lahko povzročijo napako ob zagonu programa. Apex hrani tudi soodvisnosti med objekti in razredi, tako da prepreči spremembo metapodatkov, kar bi sicer povzročilo nepravilno delovanje soodvisnih razredov. Apex kodo lahko pišemo kar preko internetnega brskalnika v omogočenem razvijalskem načinu ali preko namenskih okolij, kot je npr. Eclipse vtičnik ali samostojen Force.com IDE. V začetnih fazah razvoja tega programskega jezika so mu mnogi očitali, da še ni dovolj zmogljiv za resno uporabo, kar se je z leti spremenilo. Pridobil je na zanesljivosti, hitrosti ter predvsem na zmanjšanju omejitev, ki razvijalcu omogočajo razvoj kompleksnejših in zahtevnejših algoritmov.

4.3.5 Omejitve

Kljub neštetim možnostim, ki jih ponuja platforma, moramo omeniti tudi omejitve. Ko govorimo o omejitvah, se to največkrat nanaša na tehnične omejitve [28], katere je treba upoštevati pri načrtovanju postavitve ali dodatnih aplikacij znotraj CRM-ja. Uporabnik se bo sam le redko srečal z omejitvami, to pa ne velja za razvijalce in arhitekte prilagojenih aplikacij znotraj platforme. Večinoma so te omejitve vezane na licenco, ki jo uporabljamo. Obstajajo tudi takšne, ki so splošno sprejete in se jim lahko izognemo le s premišljenim načrtovanjem. Največkrat se srečamo z naslednjimi omejitvami (veljajo za eno transakcijo):

- čas za izvajanje kode (do nedavnega je bila omejitev 200.000 izvedenih programskih vrstic, sedaj je to omejeno na 10 sekund procesorskega časa),
- število zapisov (records) v poizvedovalnem stavku (query) na transakcijo: 50.000 v read/write načinu, pri read-only načinu je ta omejitev: 1.000.000,
- število poizvedovalnih stavkov: 100,
- število t.i. DML stavkov: 150 in število vrstic, ki jih lahko obdelamo v DML stavkih: 10.000,
- velikost programsko shranjene datoteke: 5 MB (priloga), 10 MB (dokument),
- največja dolžina formule: 5000 B,
- API klici: 5000/24h.

Našteli smo limite, ki so splošne za platformo. Poleg tega obstaja še veliko drugih omejitev, ki so običajno vezane na uporabljeno licenco [28].

Poglavje 5

Tehnologije pri razvoju

Ugotovili smo, da je integracija običajno vedno del vsakega CRM sistema. Dandanes obstaja veliko število teh sistemov, od katerih ima vsak svoje značilnosti in pravila povezovanja. Na drugi strani je ta množica različnih aplikacij še veliko večja. V prejšnjem poglavju smo našteli nekaj primerov sistemov, s katerimi se CRM največkrat povezuje. Zamislimo si, da ima vsak od teh primerov še različne implementacije in rešitve. Tako lahko pridemo do izjemno veliko možnosti, ki jih to področje prinaša. Prav iz tega razloga lahko integracija predstavlja velik tehnološki izziv za organizacijo, ki želi tak sistem implementirati. V diplomski nalogi smo se lotili integracije dveh znanih sistemov: platforme Force.com in MailChimp. Tehnologije, ki smo jih pri tem obravnavali, bomo na kratko predstavili v naslednjih poglavjih.

5.1 Java

Jedro integracije predstavlja aplikacija napisana v programskem jeziku Java. To je visoko nivojski programski jezik razvit s strani Sun Microsystems. Je objektno orientiran jezik, podoben kot C++, vendar poenostavljen z namenom odprave pogostih programskih napak ter ima manj nizko nivojskih možnosti. Izvorna koda je napisana v java datotekah, ki se prevedejo v t.i. štrojno kodo", ki jo zmore poganjati interpreter, kateremu pravimo tudi navidezni stroj (JVM). Moč jave se skriva v paradigmi "write once, run anywhere", ki nakazuje, da je to platforma, na kateri programi tečejo na vsaki strojni opremi z nameščenim javinim izvajalnim

okoljem (JRE).

V diplomski nalogi lahko aplikacijo v Javi razdelimo na dva dela:

1. spletna aplikacija,

2. aplikacija, ki se izvaja v ozadju (worker).

Razdelitev na dva dela je skladna s platformo Heroku, ki uporablja dve vrsti izvajalnih okolij: web dyno in worker dyno.

Naloga spletne aplikacije je detektiranje in sprejemanje HTTP zahtev, ki jih proži na platformi Force.com nameščena aplikacija in posredovanje zahtev v zaledje aplikacije.

5.2 HTML, CSS, JavaScript

Vsak spletni dokument (stran) sestoji iz osnove, katero imenujemo HTML. To je označevalni jezik sestavljen iz HTML elementov - značk (ang. tag), ki so zapisani v špičastih oklepajih (<>). HTML je edina obvezna sestavina vsakega spletnega dokumenta, vendar v sedanjem času ni več moč najti spletne strani, ki ne bi vsebovala še dveh ključnih elementov, ki sta zaslužna predvsem za predstavitev in interaktivnost. CSS skrbi za barve, velikost, odmike, obrobe, pozicije ter še za druge atribute pri prikazovanju HTML elementov. Bistvo je tako definicija pravil in ločitev strukture strani od njene predstavitve. Za HTML in CSS velja, da ju ne moremo šteti med programske jezike, saj sta zelo statična. Programski jezik sicer definiramo kot tehnologijo, ki lahko izraža izračune, katere lahko izvaja računalnik. Če predhodnika ne moremo šteti v to skupino, lahko to rečemo za JavaScript. Predhodnika ne moremo šteti v to skupino, jezik Javascript pa lahko. Z njim si pomagamo pri ustvarjanju interaktivnih spletnih strani. JavaScript ni odvisen od HTML-ja in se ga uporablja tudi v drugih orodjih (npr. Adobe Reader ga uporablja v datotekah PDF). V našem primeru, ga bomo vgradili v HTML za potrebe komuniciranja z Java aplikacijo ter hkrati za boljšo izkušnjo z upravljanjem aplikacije.

5.3 Apache Tomcat

Za poganjanje programa v spletu potrebujemo spletni strežnik. Termin se navezuje na program, ki obiskovalcem štrežešpletne strani ob obisku določenega spletnega mesta. Od tod tudi ime spletni strežnik. Za množico programskih jezikov, ki se uporabljajo v te namene obstaja kopica spletnih strežnikov. V našem primeru je to Apache Tomcat (na kratko Tomcat). Naj naštejemo še nekaj primerov drugih spletnih strežnikov: Apache (PHP, Perl, Python), NginX, Jetty, lighttpd. Največkrat se spletni strežnik namesti znotraj operacijskega sistema na računalniku, na katerega se potem prenese aplikacijo, ki jo strežnik ob zahtevi poganja in brskalniku pošilja rezultat te zahteve. Vendar na PaaS modele ne moremo namestiti spletnih strežnikov po klasični poti, ampak za to poskrbi platforma sama. V našem primeru smo strežnik Tomcat uporabili kot t.i. vgrajen spletni strežnik (ang. embedded web server) [20], ki implementira HTTP protokol (Koda 5.1).

```
public class Main {
    public static void main(String[] args) hrows Exception {
        String webappDirLocation = "src/main/webapp/";
        Tomcat tomcat = new Tomcat();
        // The port that we should run on can be set into an environment
            variable
        String webPort = System.getenv("PORT");
        if (webPort == null || webPort.isEmpty()) {
            webPort = "8080";
        }

        tomcat.setPort(Integer.valueOf(webPort));
        Connector httpsConnector = new Connector();
        httpsConnector.setPort(443);
        httpsConnector.setSecure(true);
        httpsConnector.setScheme("https");
        httpsConnector.setAttribute("keyAlias", "keyAlias");
        httpsConnector.setAttribute("keystorePass", "password");
        httpsConnector.setAttribute("keystoreFile", "keystore");
```

```
        httpsConnector.setAttribute("clientAuth", "false");
        httpsConnector.setAttribute("sslProtocol", "TLS");
        httpsConnector.setAttribute("SSLEnabled", true);

        tomcat.addWebapp("/", new File(webappDirLocation).getAbsolutePath
            ());
        tomcat.getService().addConnector(httpsConnector);
        tomcat.start();
        tomcat.getServer().await();
    }
}
```

Koda 5.1: Vgrajen spletni strežnik Tomcat

Za delovanje zgornjega je treba dodati naslednje knjižnice v pom.xml, ki je del ogrodja Maven:

- org.apache.tomcat.embed:tomcat-embed-core
- org.apache.tomcat:tomcat-jasper
- org.apache.tomcat.embed:tomcat-embed-jasper
- org.apache.tomcat:tomcat-jsp-api
- org.apache.tomcat:tomcat-jasper-el
- org.apache.tomcat.embed:tomcat-embed-logging-juli

BaseDir nastavimo s klicem funkcije addWebapp. Naš nastavljen primarni direktorij je šrc/main/webapp/". Če želimo uporabit SSL povezavo na spletno aplikacijo, moramo ustvariti še ustrezen keystore z orodjem keytool in ukazom keytool-genkey v ukazni vrstici. Treba je tudi razumeti, da brez klica await() na koncu kode server prekine svoje delovanje takoj po ukazu start(). Funkcija await() inicializira neskončno zanko (while true) in jo ob ustreznem ukazu stop prekine, tako se tudi strežnik izključi. Po klicu te funkcije je strežnik Tomcat zagnan in sprejema HTTP zahteve. Dostopne so le datoteke v baseDir mapi ter metode v arhitekturi RESTful, ki jih bomo opisali kasneje.

5.4 Maven

Maven je bil razvit z namenom poenostaviti proces nastavljanja projekta [21]. S tem orodjem za upravljanje s projektom definiramo pravila po katerih se .java datoteke prevedejo v .class, zapakirajo v .jar datoteko ter ostale naloge, ki so potrebne za uspešno gradnjo (ang. build) projekta. Maven je povsem samostojen, ki ne potrebuje dodatnih orodij ali skript, saj se vse samodejno konfigurira z namestitvijo samega Maven-a. Zasnovan je tako, da omogoča prenosljivost na druge naprave brez dodatnega nastavljanja projekta. Pri večjih projektih je navadno tudi tako, da ljudje, ki delajo na tem projektu, uporabljajo različna razvojna okolja in z Mavenom je ta problem enostavno rešljiv. Glavni cilji orodja Maven:

- izdelati preprost postopek gradnje programa,
- zagotavljati enotni sistem gradnje,
- zagotavljati kakovostne informacije o projektu,
- nuditi smernice za najboljše razvojne prakse,
- omogočati pregledno migracijo na nove funkcije.

Datoteka pom.xml je bistvena za delovanje tega orodja. Preko te datoteke Maven upravlja z vsemi knjižnicami in nastavitvami na temu projektu. V osnovi Maven prenese vse knjižnice nastavljene v pom.xml iz centralnega repozitorija. Maven je orodje, pri katerem se vse vrti okoli življenjskega cikla gradnje aplikacije. Ta cikel vsebuje naslednje faze:

- potrjevanje (ang. validate) - potrjevanje pravilnosti projekta ter vseh potrebnih informacij,
- prevajanje (ang. compile) - prevajanje izvorne kode projekta,
- testiranje (ang. test) - testiranje kode preko podanih unit testov,
- pakiranje (ang. package) - pakiranje izvajalne kode v jar datoteko,
- integracijski testi (ang. integration test) - procesiranje in prenos kode v okolje, kjer se lahko izvedejo integracijski testi,

- preverjanje (ang. verify) - zagon in preverjanje paketa, če je veljaven in če ustreza kriterijem,
- namestitev (ang. install) - namestitev paketa v lokalni repozitorij,
- prenos (ang. deploy) - prenos v produkcijsko okolje, kopiranje zadnjega paketa v oddaljeni repozitorij za v uporabo.

Primer odvisnosti v pom.xml datoteki.

```
<dependency>
        <groupId>com.force.api</groupId>
        <artifactId>force-partner-api</artifactId>
        <version>29.0.0</version>
</dependency>
```

Koda 5.2: Datoteka pom.xml

5.5 PostgreSQL

PostgreSQL (na kratko Postgres) je zmogljiv, odprtokodni, objektno-relacijski sistem za upravljanje s podatkovnimi bazami (ORDBMS) [22]. Ta sistem implementira večino točk standarda SQL:2011, vsebuje načela delovanja ACID, je transakcijski in se izogne zaklepanju podatkovne baze z metodo MVCC, zaradi katerega se loči od najbolj razširjene podatkovne baze MySQL. Ta metoda za zagotavljanje enovitosti podatkov ob hkratnih dostopih ne uporablja zaklepanja zapisov, marveč model zapisovanja, ki ob začetku transakcije le-tej zagotovi enovit pogled na stanje zbirke. Postgres teče tako na distribucijah Linuxa kot operacijskem sistemu Windows.

5.6 Protokoli in avtentikacija

V integriranih sistemih, kjer enote, ki smo jih povezali med seboj, delujejo povsem samostojno in neodvisno, je treba upoštevati protokole, ki zagotavljajo standard, red ter nenazadnje tudi varnost. Samo tako je možno učinkovito vzpostaviti komunikacijo med različnimi sistemi. Sprejeti standardi so delo množice strokovnjakov,

ki obvladajo to področje in so tako konceptualno varni. Vendar je od izvajalca odvisno, ali je protokol pravilno in dosledno uveden v integracijo.

5.6.1 OAuth 2.0

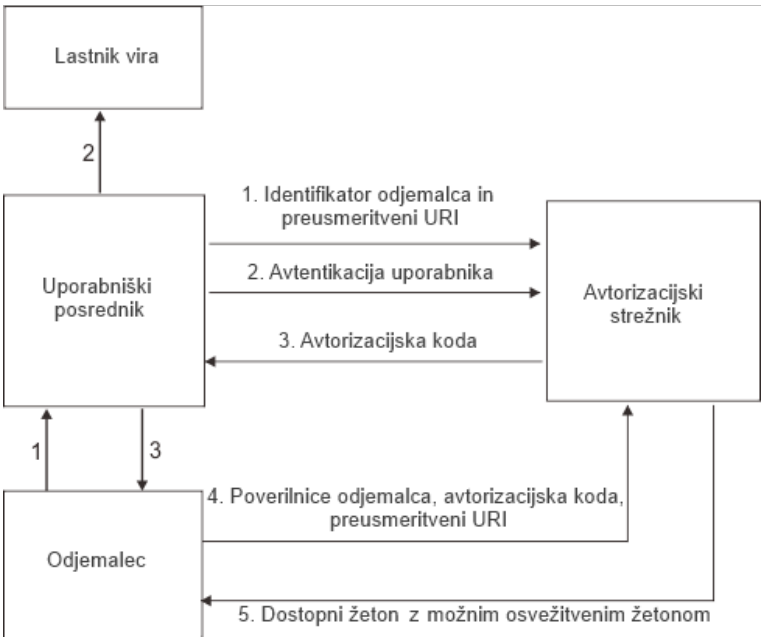
OAuth 2.0 [13] je naslednik OAuth protokola, ki je bil prvotno ustvarjen leta 2006. Druga generacija tega OAuth protokola se osredotoča na preprostost implementacije skupaj s posebnimi avtorizacijskimi tokovi za spletne aplikacije, namizne aplikacije, mobilne telefone in ostale naprave. Ta protokol uporabljajo zelo znana podjetja, kot so Facebook, Google, Twitter, Salesforce, Microsoft, itd. Avtorizacijsko ogrodje OAuth 2.0 omogoča aplikacijam drugih izvajalcev, da pridobijo omejen dostop do HTTP servisov bodisi v imenu lastnika vira bodisi v vlogi tretje osebe, ki pridobi dostop v svojem imenu. Tok protokola OAuth 2.0 [14]:

1. avtorizacijska zahteva na avtorizacijski strežnik,
2. avtentikacija lastnika vira,
3. potrditev ali zavrnitev zahteve ter preusmeritev na preusmeritveni URL z avtorizacijsko kodo,
4. zahteva za dostopni žeton z avtorizacijsko kodo in preusmeritvenim URL-jem,
5. avtorizacijski strežnik v primeru uspešnega preverjanja poverilnic vrne dostopni ter osvežitveni žeton.

Na Sliki 5.1 je opisan osnovni tok protokola OAuth 2.0. Obstajajo še drugi, za delovanje ravno tako pomembni tokovi, a niso predmet te diplomske naloge. Uporabljeni tokovi v naši aplikaciji so naslednji: pridobitev dostopnega žetona, osveževanje dostopnega žetona ter preklic osvežitvenega žetona.

5.6.2 REST

REST (ang. representional state transfer) je arhitekturni stil, ki sestavlja skupino omejitev, katere se nanašajo na komponente, funkcije arhitekturnih elementov ter povezav. REST je sestavljen iz odjemalcev in strežnikov ter nima predpisanih



Slika 5.1: OAuth 2.0 protokol

protokolov in standardov [19]. REST se pogosto omenja pri razvoju spletnih servisov kot alternativa ostalim tipom, še posebej najbolj razširjenemu tipu SOAP. Arhitekturne omejitve:

1. odjemalec - strežnik

   Enoten vmesnik ločuje odjemalca od strežnika. To pomeni, da odjemalca ne zanimajo procesi, ki se dogajajo na strežniku (podatkovna baza, povezovanje z drugimi spletnimi servisi, ipd.). Tako je zagotovljena boljša prenosljivost odjemalca. Prav tako tudi strežnika ne zanima uporabniško stanje ter vmesnik. Razvoj obeh lahko poteka povsem neodvisno s to omejitvijo, da vmesnik ostane nespremenjen.

2. brez stanja

   Podatki med komunikacijo o odjemalcu niso shranjeni na strežniku. Vsaka zahteva nosi vse potrebne podatke, ki jih strežnik potrebuje za izvršitev. Stanje je shranjeno pri odjemalcu.

3. zmožnost predpomnjenja

Odjemalec ima zmožnost predpomnjenja odgovorov. Odgovori morajo definirati, ali naj se shranijo v predpomnilnik ali ne, da preprečijo hranjenje zastarelih ali nepravih podatkov, ki bi se potencialno uporabili v naslednjih zahtevah. Dobro zasnovano predpomnjenje lahko odpravi odvečno interakcijo med odjemalcem in strežnikom, kar poveča nadgradljivost in učinkovitost.

4. večplastni sistem

   Odjemalec ne more vedeti, ali je povezan s posrednikom ali direktno do končnega serverja. Posrednik lahko izboljša učinkovitost sistema s primernim uravnoteženjem obremenitev (ang. load balancing) ter nudenju deljenega predpomnilnika.

5. koda na zahtevo

   To je edina neobvezna omejitev pri REST arhitekturi. Strežniki lahko razširijo funkcionalnost odjemalca s prenosom izvršljive kode npr. java apleti ali javascript skripte.

6. enoten vmesnik

   Omejitev enotnega vmesnika je poglavitnega pomena REST arhitekture. Enoten vmesnik poenostavlja in ločuje posamezne elemente, kar omogoča, da se vsak del razvija neodvisno. Načela enotnega vmesnika so: identifikacija sredstev, samoopisna sporočila, hipermedij kot določevalec stanja aplikacije, upravljanje s sredstvi preko njihove predstavitve.

Primerjava REST in SOAP
REST:

- spletni servisi so povsem brez stanja (ang. stateless),
- dobro predpomnjenje pri HTTP GET zahtevah,
- odjemalec in strežnik dobro razumeta kontekst, vsebino ter obliko vsebine, saj ni splošnih pravil za REST vmesnik,
- uporabnejši za mobilne naprave, saj so dodatni parametri kot npr. glava sporočila pri SOAP povsem odveč v smislu vsebine,

- enostavnejši za vgradnjo v obstoječe spletne strani in ni potrebe po dodatnem spreminjanju arhitekture spletne strani,
- implementacija REST je enostavnejša od SOAP.

SOAP:

- podaja WSDL, ki vsebuje in opisuje skupen nabor pravil, katera definirajo sporočila, operatorje in lokacijo spletnih storitev,
- SOAP potrebuje manj namestitvene kode kot REST (transakcije, varnost, usklajevanje, naslovi, itd.),
- uporaben pri upravljanju z asinhronimi procesi,
- podpira številne protokole in tehnologije kot so WSDL, XSDs, SOAP in WS-Addressing.

V splošnem je tako, da je SOAP primernejši pri kompleksnih aplikacijah, ki preko vmesnikov komunicirajo z zunanjim svetom. Kadar pa imamo manj časa za učni proces in si želimo hitrih rezultatov ter enostavnih transakcij (CRUD), se največkrat poslužujemo REST arhitekture.

JAX-RS To je eden izmed številnih vmesnikov za razvoj spletnih storitev pisan v Javi in je narejen v arhitekturnem stilu REST. Uporablja anotacije za definicijo pomena javanskih razredov. Primeri najpogostejših anotacij:

- @PATH(pot) - definira URL pot,
- @POST, @GET, @PUT, @DELETE - označuje metodo, ki se odzove na tovrstno HTTP zahtevo,
- @Produces(tip) - definira, katero vrsto MIME tipa vrača metoda,
- @Consumes(tip) - definira, katero vrsto MIME tipa sprejema,
- @PathParam - se uporablja za prepis vrednosti iz URL-ja v spremenljivke metode.

Jersey [23] je izvedba za JAX-RS [24] specifikacijo, katero uporablja tudi naša izdelana aplikacija. V osnovi Jersey vsebuje REST strežnik in REST odjemalca. Odjemalec priskrbi knjižnico, ki komunicira s strežnikom. Na strežniški strani Jersey uporablja javanski razred (ang. servlet), ki preišče vse razrede z namenom identifikacije RESTful virov. Ta razred analizira prihajajoče HTTP zahteve in izbere pravi razred ter metodo, ki se nato odzoveta. Izbor je narejen .

Poglavje 6

Razvoj aplikacije

Odločitev za razvoj aplikacije je nastala na podlagi več dejavnikov. Glavni izmed dejavnikov je preučitev možnosti izdelave integracijske aplikacije med storitvama dveh največjih ponudnikov, Salesforca kot vodilnega na CRM področju ter MailChimpa kot najbolj priljubljenega e-poštnega marketing orodja v zadnjem času.

Razvoj aplikacije je potekal v več fazah:

- Analiza
- Načrtovanje
- Implementacija
- Vzdrževanje

V fazi **analize** smo preučili trenutno situacijo na področju integracije Salesforce CRM in MailChimp. Na tem področju deluje že podobna aplikacija MailChimp for Salesforce [29], ki ne dosega postavljenih ciljev razvoja naše aplikacije. V začetku smo zastavili cilje, ki jih mora aplikacija izpolniti. Cilji:

Preprostost

Večina integracijskih orodij, ki povezujejo Salesforce CRM in e-poštna marketinška orodja je zelo kompleksnih in od uporabnika zahtevajo veliko razumevanja ter večkrat tudi poznavana dogajanja v ozadju. Aplikacija mora imeti kar najmanj možnosti, da se uporabnik ne zmede, zato, če je možno, v osnovi potrebujemo le en gumb, ki ob pritisku zažene vse potrebne procese.

Varčna z viri

Oblačni sistemi vsebujejo množico omejitev pri prenosu podatkov. V našem primeru so te še posebej značilne za CRM, ki jih opisujemo v poglavju 4.3.5. Kot omenjeno, je na celotni platformi Salesforce CRM že 60 % transakcij narejenih preko integracije (API). Cilj je tako porabiti kar najmanj API klicev pri sinhronizaciji.

Zmogljivost

Aplikacija mora omogočati sinhornizacijo do 1.000.000 kontaktov. Največja težava pri doseganju tega cilja je omejitev delovnega pomnilnika (RAM) na PaaS platformi Heroku, ki je omejena na 512 MB ter omejitev 5000 API klicev na platformi.

Hitrost

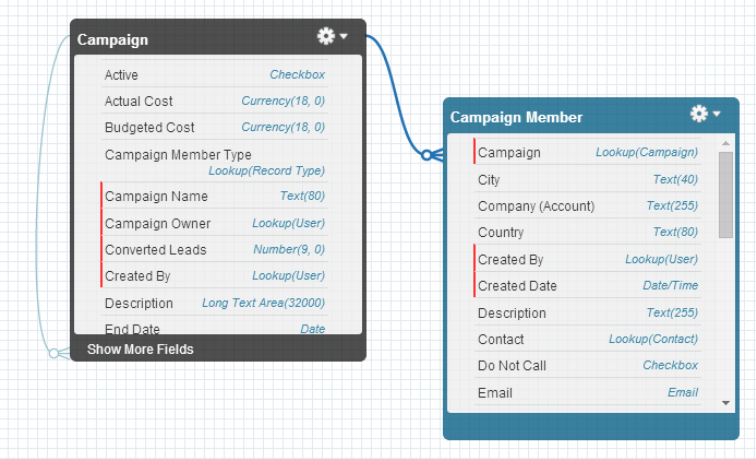
Kljub temu da je povprečno število prenesenih kontaktov več 10.000, je bilo treba zagotoviti hiter prenos podatkov iz ene na drugo stran zaradi dveh razlogov. Najpomembnejši je ta, da se viri aplikacije sprostijo za naslednjo morebitno neodvisno sinhronizacijo. Drug razlog je uporabniška izkušnja pri uporabi aplikacije.

6.1 Faze razvoja

6.1.1 Analiza

Salesforce CRM

Najprej je bilo treba analizirati vse sisteme, ki jih bo aplikacija uporabljala. Na začetku dela je center uporabe te aplikacije Salesforce CRM. Seznanili smo se z delovanjem samih marketinških kampanj ter že vgrajenih funkcionalnostih, ki jih vključuje. Slika 6.1 prikazuje povezavo med Čampaign Member"in Čampaign". Kampanja je torej nadrejeni objekt, vezana s člani kampanje po konceptu one-to-many. Kampanja vsebuje osnovne informacije: ime, datum kampanje, status, število članov, tip ter podatek o nadrejeni kampanji. Član kampanje vsebuje polja: ime, priimek, e-pošto, povezavo na kampanjo ter povezavo na kontakt (many-to-one). Pomembno je poudariti, da v CRM-ju obstajata dva koncepta ljudi (strank). Kontakti so ljudje, ki imajo s to organizacijo že določeno poslovno razmerje in so



Slika 6.1: Podatkovni model kampanje

vključeni v prodajni proces. Na drugi strani obstaja t.i. sled (ang. lead). S tem pojmom označujemo tiste ljudi, ki še niso naše stranke, vendar so blizu tega, da to postanejo. To so lahko: ljudje, ki so izpolnili spletni vprašalnik na naši strani ali ljudje, od katerih smo dobili vizitko na kakšnem poslovnem dogodku. V kampanjo lahko vključimo obe vrsti ljudi, kar v Salesforce CRM-ju prikažemo z dvema objektoma.

V smislu marketinških e-poštnih kampanj nas torej zanimajo samo informacije o imenih, priimkih, e-poštnih naslovih ter imenu kampanje. Ker sinhronizacija ne poteka le v smeri Salesforce CRM - MailChimp, temveč tudi obratno, moramo ob tem poslati tudi referenco na kontakt (id kontakta), da vemo kateri entiteti bomo kasneje pripeli informacije o rezultatu kampanje.

Platforma na kateri sloni Salesforce CRM v osnovi omogoča kup programskih vmesnikov (ang. API - application programming interface) [30] in vsak od teh je uporaben v različnih primerih. Za prenos podatkov obstajajo glavni trije načini:

- REST API
- SOAP API
- BULK API

REST in SOAP protokola smo omenjali v poglavju 4.6, vendar tu nastopi še Bulk API [31]. Za ta API, ki bazira na REST protokolu, je značilno to, da poteka v

asinhroni komunikaciji ter je zmožen prenesti velike količine podatkov z malo porabljenimi API klici. Značilnosti: 1 API klic na 10.000 prenesenih vrstic (največja velikost ene serije - ang. batch), 5.000 serij v obdobju 24 ur, to pomeni 50.000.000 vrstic v 24-ih urah. Ta API je namenjen le prenosu velikih količin podatkov v ali izven Salesforce CRM-ja. Za ostalo komunikacijo (avtentikacija, potrditve, metapodatki) bomo uporabili SOAP API, za katerega imamo že definiran WSDL.

MailChimp

Na drugi strani imamo MailChimp, ki prav tako ponuja programski vmesnik [32] za programsko sinhronizacijo bazo strank iz CRM, CMS ali spletne trgovino v MailChimp. Zahteve, ki jih mora ta API izpolnjevati, da zadošča našim integracijskim potrebam:

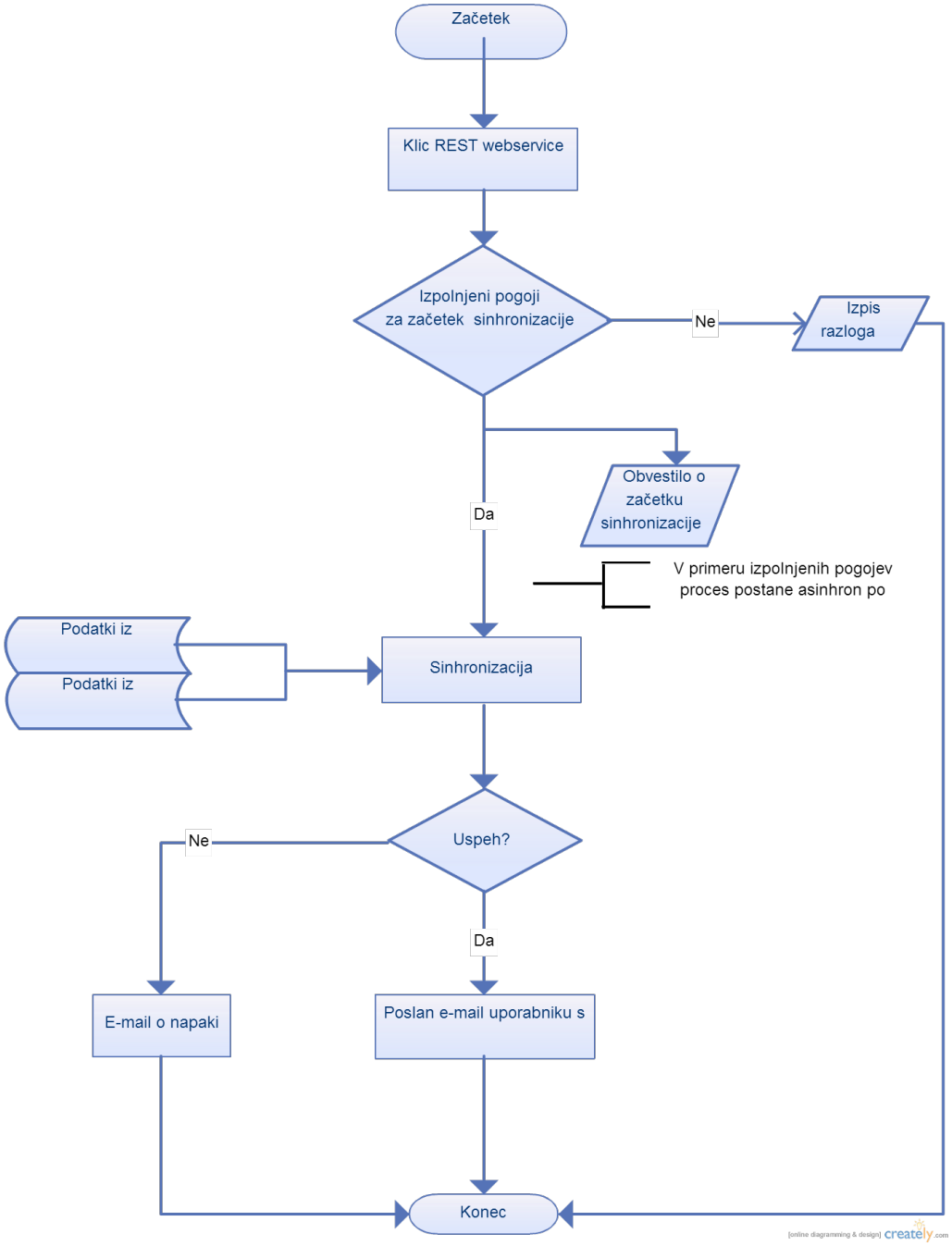
- vstavljanje kontaktov s polji (ime, priimek, e-poštni naslov, ID osebe) na določen seznam,
- ustvarjanje kampanj,
- segmentacija kontaktov,
- izvoz poročil.

MailChimp API je tipa RESTful in tako vsebuje že definirane URL naslove, ki pomenijo določeno akcijo. Ta API ne postavlja nobenih omejitev glede API klicev ter omogoča le eno vrsto API-ja, tako da izbire tukaj ni.

Heroku

V podpoglavju 2.2.2 smo na splošno opisali platformo Heroku, ki bo osrednja platforma delovanja naše aplikacije. Po analizi tega sistema v kontekstu naših potreb smo prišli do naslednjih ugotovitev:

- na voljo imamo 512 MB delovnega pomnilnika,
- podpira programski jezik Java,
- HTTP okno je odprto 30 sekund pri web dyno, worker dyno časovno ne omejuje delovanja.



Slika 6.2: Diagram poteka

6.1.2 Načrtovanje

Diagram poteka 6.2 prikazuje načrt delovanja aplikacije. Po fazi analize smo morali definirati obseg funkcionalnosti ter primer uporabe. Kot omenjeno, je bila preprostost uporabe ena glavnih zahtev, ki smo jo morali uresničiti. Zamislili smo si gumb po meri, ki bo v našo aplikacijo poslal zahtevo za prenos kontaktov. Uspeh procesa je odvisen tudi od vnešenih podatkov v Salesforce CRM-ju in predhodne avtentikacije OAuth 2.0. V sinhronem načinu preverjamo nastavljene parametre in uporabniku se na zaslonski maski izpiše obvestilo o uspešnosti/neuspešnosi začetka sinhronizacije. V primeru uspeha proces nadalje glede na podane parametre začne sinhronizacijo v asinhronem načinu, kar pomeni, da se je iz uporabnikovega vidika trenutna akcija končala, proces pa se nadaljuje v ozadju. Kljub temu uporabnik naloge še ne more nadaljevati, ker je prenos kontaktov ravno v teku. Odločili smo se za obvestilo v obliki e-pošte na uporabnikov e-naslov. V vmesnem času aplikacija izvaja naslednje procese: 1. spletni klic na Salesforce API in branje kontaktov, 2. procesiranje in pakiranje kontaktov, 3. spletni klic na MailChimp API in prenos kontaktov, 4. obvestilo o rezultatu prenosa.

6.1.3 Implementacija

Izdelavo rešitve načrtovane v prejšnji točki smo razdelili na dva dela. Vsak od teh delov vsebuje več modulov.

1. Aplikacija na Salesforce CRM:
    - spletni zavihek (Slika 6.3),
    - gumb Sinhroniziraj z MailChimpom (Slika 6.4),
    - nov objekt ter dodatna polja na večih objektih,
    - povezana aplikacija (ang. connected app).


```
{
        "id":"https://login.salesforce.com/id/00D50000000IZ3ZEAW/00550000001fg5OAAQ",
        "issued_at":"1296458209517",
        "scope": "id full api openid refresh_token chatter_api",
        "instance_url":"https://na1.salesforce.com",
        "token_type": "Bearer",
        "refresh_token":"5Aep862eWO5D.7wJBuW5aaARbbxQ8hssCnY1dw3qi59o1du7ob.
            lp23ba_3jMRnbFNT5R8X2GUKNA==",
```

```
        "id_token": "eyJhb...h97hc",
        "signature":"0/1Ldval/TIPf2tTgTKUAxRy44VwEJ7ffsFLMWFcNoA=",
        "access_token":"00D50000000IZ3Z!
            AQ0AQDpEDKYsn7ioKug2aSmgCjgrPjG9eRLza8jXWoW
        7uA90V39rvQaIy1FGxjFHN"
}
```


Koda 6.1: Podatki, pridobljeni pri avtentikaciji s Salesforce CRM

2. Aplikacija na platformi Heroku:

    - spletna stran za avtentikacijo,
    - REST spletni servis,
    - vgrajen spletni strežnik Tomcat,
    - modul za komunikacijo s podatkovno bazo PostgreSQL,
    - konektor za API vmesnik,
    - upravljanje z varnostnimi žetoni za dostop,
    - modul za obveščanje preko e-pošte,
    - modul za dnevno poročanje o aktivnosti.

Uporabniku je na prvi pogled viden le prvi del, vendar je glavnina delovanja ravno v drugem delu. Najprej smo ustvarili vstopno stran (JSP - Slika 6.3), ki je poskrbela za postopke v zvezi z avtentikacijo. Po kliku na gumb Avtenticiraj z MailChimpom se nam odprejo nove strani, ki nas vodijo po korakih do uspešne avtentikacije po protokolu OAuth 2.0.

V naslednji fazi smo izdelali gumb, s katerim sprožimo začetek procesa. Gumbi lahko prožijo več vrst dogodkov: povezava, VisualForce stran, izveden JavaScript. V našem primeru smo izbrali proženje JavaScripta.

```
$.ajax({
    url: 'https://chimpsync.herokuapp.com/launch/sync?callback=jsoncallback
        ',
    jsonpCallback: 'jsoncallback',
    dataType: 'jsonp',
    timeout: 15000,
```

```
    data: {
        'sfcmpg': SF_campaign_id,
        'sfid ': orgid,
        'oneway': one_way,
        'id': mc_campaign_id,
        'campaignName': campaignName,
        'dateCreated': dateCreated,
        'email': userEmail,
        'list ': mailchimpList
    },
    success: function (data) {
        //show success
    },
    error: function (jqXHR, textStatus, thrownError) {
        //show error
    }
})
```

Koda 6.2: JavaScript kliče REST metodo

V tej fazi je proces prešel iz prvega dela v drugi del. Proces se sedaj v asinhronem načinu odvija v aplikaciji na Heroku, na katerem že teče vgrajen strežnik Tomcat, opisan v prejšnjih poglavjih 5.1.

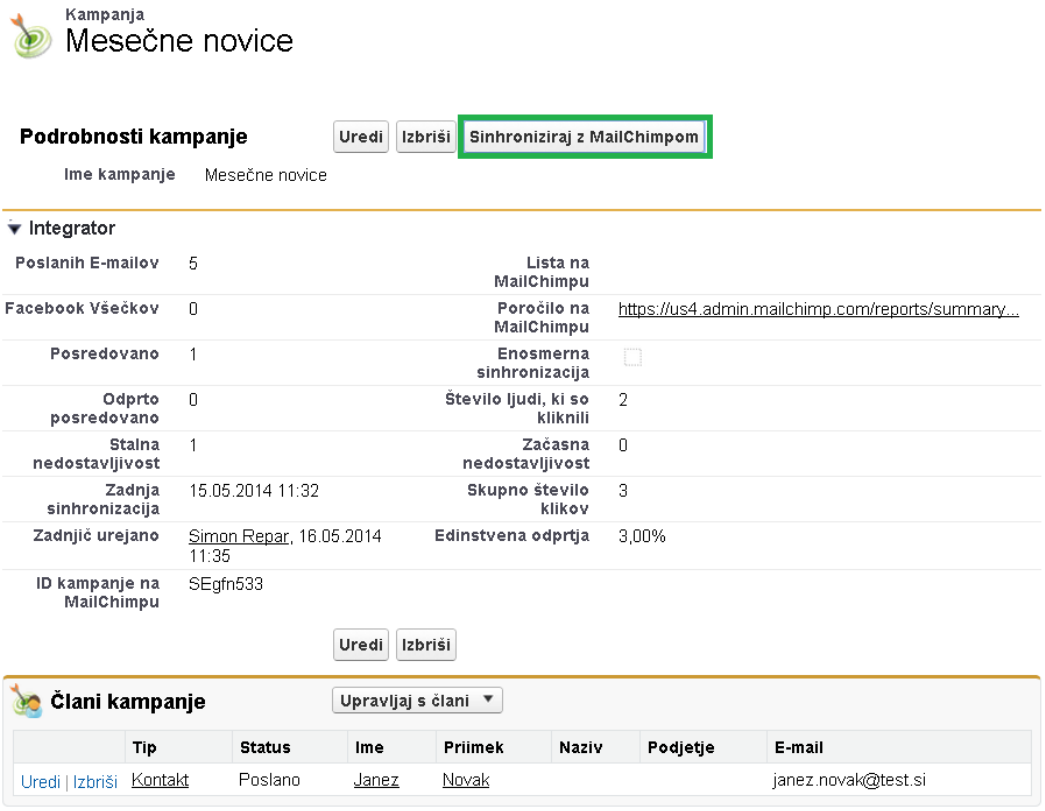
```
@Path("/sync")
public class Rest {
        public Rest() {}
        // defines a method which are used in REST based architecture
        @GET
        @Produces(MediaType.APPLICATION_JSON)
        public String startSync(@QueryParam("id") String mcid,
                        @QueryParam("oneway") Boolean mconeway,
                        @QueryParam("sfid") String orgid,
                        @QueryParam("sfcmpg") String sfcmpgID,
```

```
                @QueryParam("campaignName") String
                    campaignName,
                @QueryParam("dateCreated") String dateCreated,
                @QueryParam("email") String userEmail,
                @QueryParam("list") String mailchimpList,
                @Context HttpHeaders headers) throws Exception {
            // checks if authentication is valid
            if (!Database.isAuthenticated(orgid, false)) {
                return "jsoncallback({\"error\":\"Synchronizer is not
                    authenticated with your accounts\"})";
            }
            // start synchronization if authentication is valid
            Main.runSync(mcid, mconeway, orgid, sfcmpgID,
                campaignName, dateCreated, userEmail, mailchimpList);
            return "jsoncallback({\"success\":\"Your sync is in progress.
                We will send you an email when sync is completed!\"})";
        }
}
```

Koda 6.3: REST modul

Vstopni točka je REST spletni servis 6.3, ki proži nadaljnje procese. Najprej se proces spremeni v večnitnega in t.i. worker dyno prevzame nadaljnje naloge. To je zelo pomemben korak, saj v primeru več kot 30 sekund dolge zahteve web dyno zahteva prekinitev. Worker dyno poganja aplikacijo v večnitnem načinu in ta korak je najbolj časovno potraten. Sprožimo zahtevo za prenos kontaktov v BULK asinhronem načinu. Po zajemu kontaktov, ki smo jih ravnokar razčlenili iz XML datoteke, oblikujemo serijo kontaktov, katere še pred tem opremimo s parametri za segmentacijo. Serijo pošljemo v POST načinu in po uspešnem prenosu v MailChimp pošljemo e-pošto uporabniku, ki je uporabil to funkcijo v salesforce CRM.

Slika 6.3: Spletna stran za avtentikacijo



Slika 6.4: button začetek sinhronizacije

```
new Thread(new Runnable() {
    public void run() {
        try {
            WorkerProcess.runWorker(mcid, mconeway, sfid, sfcmpgID,
                campaignName, dateCreated, userEmail, mailchimpList);
            return;
        } catch (Exception e) {
            e.printStackTrace();
            return;
        }
    }
}).start();
```

Koda 6.4: Večnitno delovanje

Kljub temu da smo OAuth 2.0 avtentikacijo izvršili že na začetku uporabe, je bilo treba implementirati tudi logiko okoli tega med uporabo konektorja. Salesforce CRM uporablja t.i. osvežitveni žeton, kar pomeni dodatno logiko za osveževanje dostopnega žetona z veljavnostjo dveh ur.

6.1.4 Vzdrževanje

Prednost aplikacij, ki so nameščene na oblačnih platformah ali še bolje, so razvite prav za določeno platformo, je tudi v tem, da odpade potreba po dragem vzdrževanju infrastrukture in nadgradnja slojev. Za vse to poskrbi ponudnik platforme. V naši aplikaciji torej posebnega vzdrževanja sploh ni, razen na aplikativnem nivoju. Zaradi omejitev pri oblačnih platformah smo morali biti varčni z viri. Na Heroku je omejitev delovnega pomnilnika 512 MB, stranski učinek večnitnih procesov pa je povečana zasedenost le-tega. V primeru povečane uporabe aplikacije bi bilo treba preurediti in omejiti njeno uporabo. Med večjimi obremenitvenimi testi se je pojavila izjema *memory quota exceeded*, kar pomeni prekoračitev delovnega pomnilnika. Na tem področju je še prostor za izboljšave. Med izvajanjem aplikacije je možno veliko scenarijev in da bi sčasoma pokrili tudi tiste najredkejše, smo naredili tako, da se ob vsaki sprožitvi napake (ang. exception) sproži pošiljanje

e-pošte. Izkazalo se je za razmeroma dobro rešitev za sporočanje podrobnih napak v sistemu.

6.2 Integracija s platformama Force.com in MailChimp

Pri integraciji z MailChimp API-jem smo uporabili del obstoječih razredov [37]. S tem smo pridobili ogrodje za povezovanje z MailChimpom, vendar je ta knjižnica razredov za API verzijo 1.3 nepopolna, tako da smo tiste razrede, katere smo še potrebovali, napisali sami. V času izdelave aplikacije je bil API v1.3 opuščen in nadgrajen z API v2.0. V verziji 2.0 so želeli API približati trenutnim RESTful standardom. Kljub temu tega API-ja še ne moremo opredeliti kot RESTful protokol, saj za prenos podatkov vsebuje le GET in POST metodo. MailChimp API vsebuje množico metod, ki so same zase zelo omejene in izpisujejo malo povezanih podatkov v eni metodi. V izogib velikemu številu klicev pri veliki količini podatkov (še posebej pri izvozu) je na voljo tudi t.i. export API, ki omogoča celoten izvoz baze. Pri izvozu statistik iz MailChimpa in uvozu v Salesforce CRM uporabimo ta API, saj je časovno manj potraten predvsem zaradi manjšega števila vhodno/izhodnih klicev.

Na strani Salesforce CRM-ja so nam bili v pomoč številni primeri in obsežna in zelo jasna dokumentacija. Kljub temu da platforma Force.com, na kateri bazira tudi Salesforce CRM, ne podpira drugih jezikov razen lastnega (APEX), dokumentacija vsebuje primere integracije v drugih programskih jezikih (Java, C#). Integracijski vmesnik je tukaj zmožen veliko več in praktično ni stvari, ki je ne bi mogli narediti. Uporabljamo dva različna API vmesnika, tako za branje kot pisanje: BULK API in SOAP API. Prvega uporabljajo pri izvozu kontaktov in pri uvozu statistik po poslani kampanji, saj sta ti dve operaciji v smislu količine podatkovno zahtevni. Pri izvozu uporabljamo format XML (6.5) in pri uvozu CSV (6.6). Za ostale podatke, ki so v manjši meri, uporabljamo prilagodljivejši SOAP API.

```
<queryResult xmlns="http://www.force.com/2009/06/asyncapi/dataload"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
    <records xsi:type="sObject">
        <type>CampaignMember</type>
        <Id xsi:nil="true"/>
        <CampaignId>701G0000000n2SbIAI</CampaignId>
        <email__c>test@test.com</email__c>
        <name__c>Testni</name__c>
        <lastname__c>Kontakt</lastname__c>
        <member_id__c>c003G000001jeNzk</member_id__c>
    </records>
</queryResult>
```

Koda 6.5: Izvoz XML datoteke

```
Name,Campaign__c,Contact__c,Lead__c,MCidEmail__c,Times_opened__c,
    Time_opened__c,Links_clicked__c,Time_clicked__c
test@test. com701i00000009EWV,003i0000009FQQi,,5c7695ce3dtest@test.com
    ,2,2014-04-25T10:02:11Z,"http://www.example.com,http://www.example2.
    com,",2014-04-25T10:02:14Z
```

Koda 6.6: Uvoz CSV datoteke

6.3 Uporaba aplikacije

Aplikacijo uporablja nekaj organizacij, od katerih je še posebej znana Ekonomska fakulteta, ki uporablja tudi Salesforce CRM. Prve uporabniške izkušnje so pokazale, da bi jo za popolno uporabniško izkušnjo morali nekoliko prilagoditi. Gre namreč za to, da je zelo težko zajeti vse scenarije uporabe v samo en gumb.

6.4 Varnostni vidik

V poslovnih okoljih sta zelo pomembna varnost in načini hranjenja podatkov. V naši podatkovni bazi hranimo le tiste najnujnejše podatke, s katerimi lahko dostopamo do Salesforce CRM-ja ter MailChimpa, in podatke, brez katerih funkcije, ki smo jih opisali, ne bi bile mogoče. V okviru aplikacije so to dostopni žetoni

(pri Salesforce CRM-ju še osvežitveni žeton), podatki o organizaciji (ime, e-pošta, id organizacije) ter ostali podatki, namenjeni izključno statistiki. Podatki se hranijo v PostgreSQL bazi na Amazonovih strežnikih in so že v osnovi integrirani s platformo Heroku. Komunikacija poteka v SSL načinu, kar onemogoča razkritje podatkov. Prenešene podatke ne zapisujemo, ampak jih samo obdelujemo in prenašamo iz enega sistema v drugega.

6.5 Podobne aplikacije

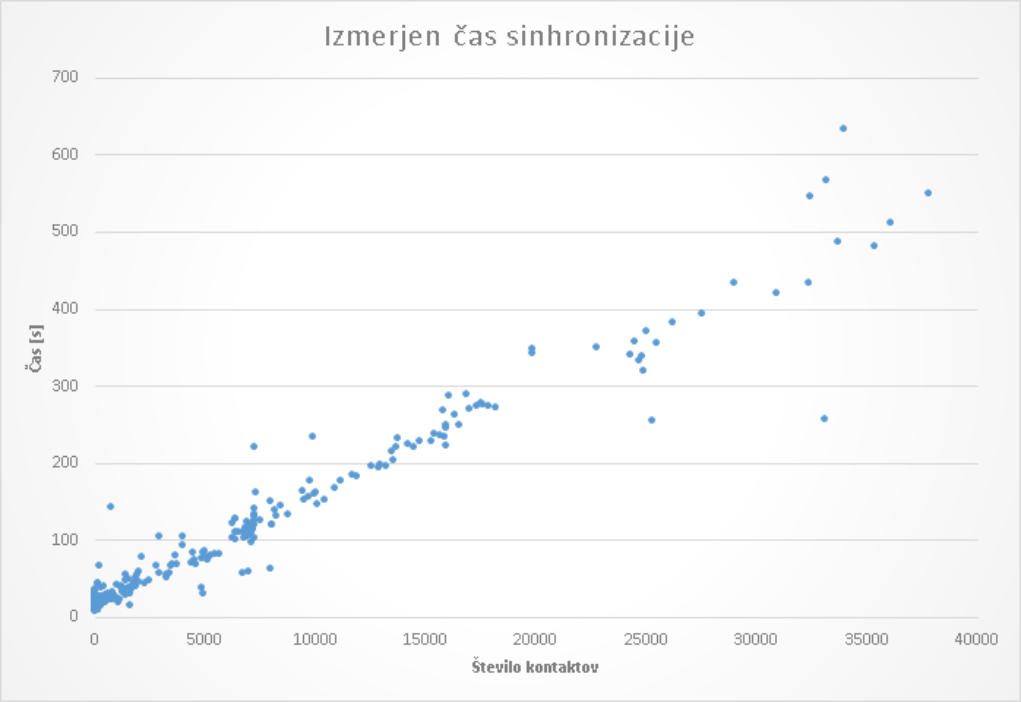
Obstaja kar nekaj orodij za sinhronizacijo z e-poštnimi marketing sistemi. Večina aplikacij je narejena na način, ki podpira celo vrsto platform in povezovanje je možno med vsemi kombinacijami platform. Pri oglaševanju se osredotočajo ravno na to velikansko podporo množici sistemov. Praksa pravi, da je to v poslovnem okolju največkrat odveč. Organizacija običajno uporablja le eno orodje za določeno funkcijo, kar pomeni, da so tudi integracije le med določenimi sistemi. Take aplikacije zato vsebujejo preveč generičnosti in ne omogočajo takšne fleksibilnosti kot namenska orodja za konkretne sisteme. Aplikaciji, ki sodita v to skupino, sta Cazoomi in OneSaas.

Mailchimp for Salesforce V času razvoja naše aplikacije smo stalno raziskovali podobna orodja in ugotovili, da je podjetje MailChimp razvilo svojo integracijsko aplikacijo, ki pa se razlikuje od ostalih. Aplikacija je zelo osredotočena na procese, ki se uporabljajo v praksi. Po globlji analizi smo ugotovili, da ne uporablja funkcij, ki bi omogočali prenose velikih količin podatkov, saj je jedro aplikacije grajeno na platformi Force.com.

6.6 Postopek vpeljave aplikacije na fakulteto

Za namestitev aplikacije mora fakulteta najprej uporabljati Salesforce CRM sistem ter MailChimp kot ESP. Nadaljnji koraki so sledeči:

- namestitev paketa,
- prilagoditev zaslonskih mask,
- avtentikacija preko spletnega zavihka.



Slika 6.5: Hitrost sinhronizacije

S tem se tehnična vpeljava v sistem zaključi. Pri tem smo predpostavljali, da fakulteta že uporablja omenjena sistema. Uvedba Salesforce CRM-ja kot celote je tehnično in organizacijsko v primerjavi z razvitim integratorjem zahtevnejša in presega okvire diplomske naloge. Kampanje, na katere se naša aplikacija priklaplja, so standardna funkcionalnost Salesforce CRM-ja in kot take povsem nezahtevne za vpeljavo. Treba je le odpreti preizkusni ali razvijalski račun, katerega lahko kasneje proti plačilu pretvorimo v trajnega. MailChimp račun lahko za potrebe preizkusa uporabljamo brezplačno, če imamo do 2000 kontaktov v e-poštnem seznamu.

6.7 Hitrost sinhronizacije

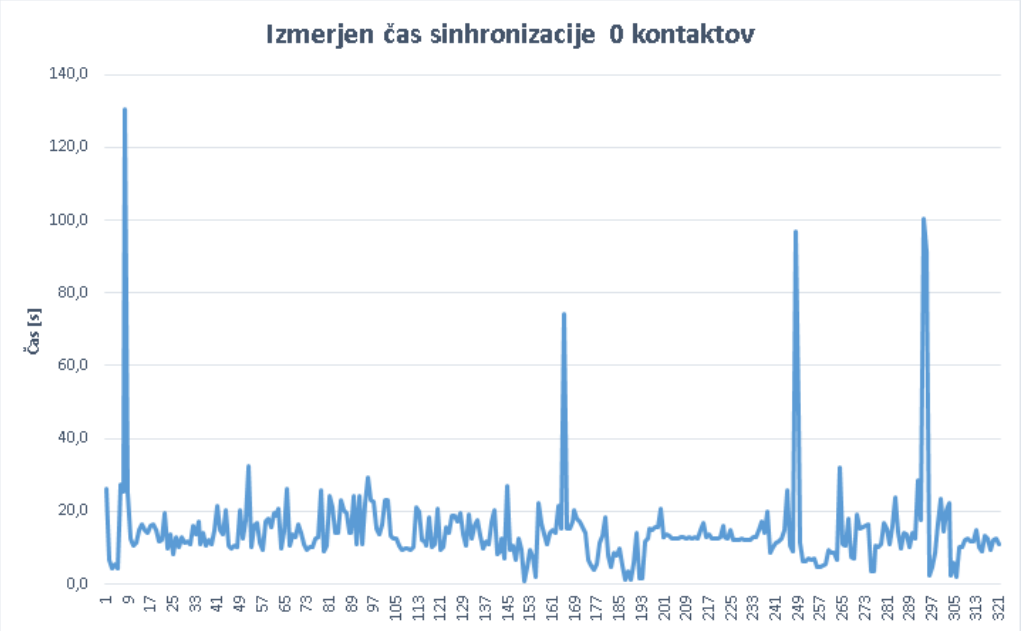
Graf 6.5 vizualizira hitrost sinhronizacije v odvisnosti od števila kontaktov. Prikazanih je 290 povprečnih časov po posameznih velikostnih razredih. Iz izmerjenih časov preko linearne regresije izračunamo enačbo:

$$y = 0,0139x + 18,07 \tag{6.1}$$

Kot smo predvidevali, je rast časovne komponente linearna in to potrjuje tudi

Število kontaktov	Čas
10000	157 s = 2,6 min
100000	1408 s = 23 min
1000000	15800 s = 230 min

Tabela 6.1: Izračun časa sinhronizacije



Slika 6.6: Hitrost sinhronizacije brez kontaktov

tabela 6.1 ter izpeljana enačba. Začetna vrednost je 18, kar v povprečju predstavlja čas, ki ga potrebujemo za inicializacijo sinhronizacije. Tako dobimo vrednost 0,0139 sekunde za prenos enega kontakta.

Skozi razvoj smo aplikacijo stalno izboljševali in optimizirali njeno delovanje. Graf 6.6 prikazuje množico sinhronizacij brez kontaktov. Vedeti moramo, da proces opravi več različnih preverjanj, izvede osvežitev dostopnega žetona in posodobi objekt na platformi Force.com, poleg tega ustvari tudi kampanjo na MailChimpu ter na koncu pošlje e-pošto. Vse to zahteva določen čas, kar lahko razberemo iz grafa.

Poglavje 7

Poslovne aplikacije

7.1 AppExchange

AppExchange je spletno tržišče poslovnih spletnih aplikacij, katere izdelovalci so predvsem neodvisni ponudniki programske opreme (ISV). Največkrat so to tudi partnerji podjetja Salesforce, ki že imajo poslovne stike na tem področju. Aplikacije so zgrajene na Force.com platformi ter na tehnologijah, ki niso sestavni del platforme in se z njo povezujejo preko spletnih storitev. Približno polovica aplikacij je brezplačnih in vsakdo si določeno aplikacijo lahko naloži na svoj ORG. Za razvoj aplikacije na Force.com platformi si lahko vsak odpre razvijalski račun.

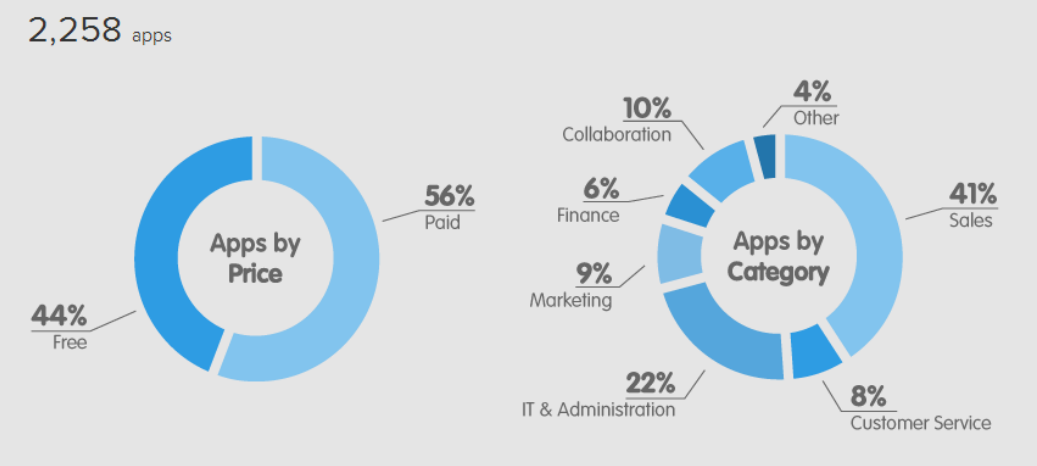
7.2 Priprava na varnostni pregled

Pot za objavo aplikacije na AppExchange v splošnem obsega tri korake varnostnega pregleda [33]:

- priprava aplikacije,
- varnostni pregled,
- objava aplikacije.

Priprava aplikacije

Implementacija standardnih varnostnih protokolov [34].



Slika 7.1: Spletno tržišče AppExchange

Pregled 10 največjih varnostnih ranljivosti [35].

Pregled aplikacije z definiranimi orodji [36].

Varnostni pregled

Vključitev aplikacije v ISVforce program.

Pošiljanje v varnostni pregled preko AppExchange konzole.

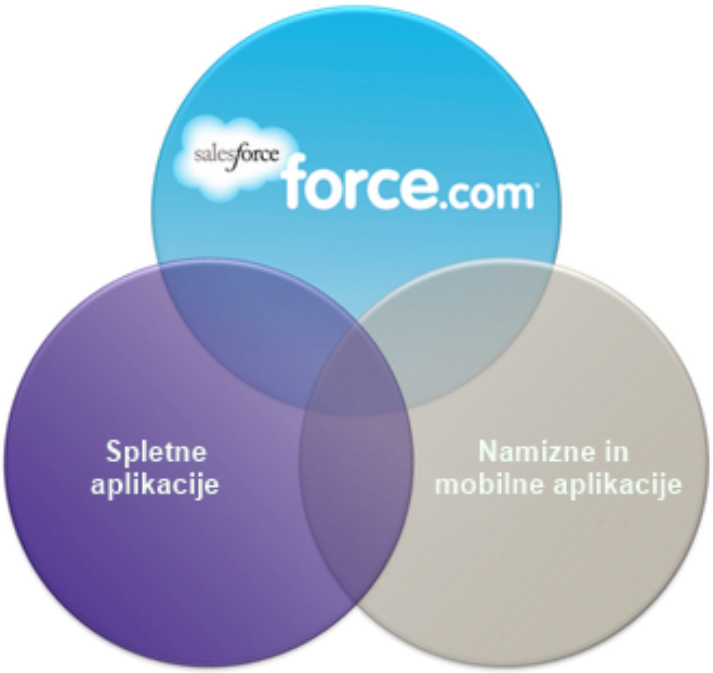
Zagotovitev testnega okolja in dokumentacije. Ročno in avtomatsko testiranje skupine za varnostne preglede. Prejem rezultatov.

Objava aplikacije

Objava aplikacije na spletno tržišče AppExchange.

Znotraj posameznega koraka je definiranih več različnih akcij, ki še natančneje opisujejo obvezne postopke. Kljub temu je preverjanje aplikacije nekoliko subjektiven proces ter odvisen od uporabljenih tehnologij, zato se lahko kakšen korak tudi izpusti ali doda. Obstajajo trije deli aplikacije (slika 7.2), ki jih lahko klasificiramo kot:

- lastne,
- sestavljene,
- klient/mobilne aplikacije.



Slika 7.2: Deli aplikacije

Večina aplikacij vsebuje eno ali več teh delov, kar pomeni različno testiranje za različne aplikacije.

7.2.1 Testiranje z Burp suite

Burp suite je javanska aplikacija, katere namen je zaščititi ali zlomiti spletno aplikacijo. Sestavljena je iz različnih orodij: proxy strežnik, spletni pajek, napadalec (ang. intruder), repetiror (ang. repeater), skener in sekvencer.

Proxy strežnik

Spletni brskalnik ob uporabi tovrstnega načina uporabi proxy strežnik, ki ga ustvari Burp suite. Tako ima aplikacija ves nadzor nad prometom, ki se izmenja med brskalnikom in spletnim strežnikom, kjer gostuje naša aplikacija. Tako Burp omogoča manipulacijo s podatki, preden se pošljejo na strežnik. Znotraj tega orodja lahko nastavimo pravila, po katerih se poslani podatki generirajo. S tem načinom lahko odkrijemo hrošče in ranljivosti naše spletne aplikacije.

Spletni pajek

S tem orodjem lahko pregledujemo spletno aplikacijo in tako znotraj nje sledimo vsem povezavam, ki so potem uporabljene pri napadu.

Napadalec

To je orodje za izvajanje avtomatičnih napadov na aplikacijo. Napadi so mogoče le, če Burp pozna pregledovano aplikacijo v podrobnosti ter HTTP protokol. Vsebuje funkcionalnost s številnimi prilagodljivimi algoritmi, ki generirajo zlonamerne HTTP zahteve. Na ta način lahko odkrijemo ranljivosti, kot so: vrivanje stavkov SQL, vstavljanje škodljive JavaScript kode (XSS), manipulacija poslanih parametrov, brute-force napadi.

Repetitor

To enostavno orodje omogoča spreminjanje zahtev, poslanih na strežnik ter njihovo ponavljajoče pošiljanje. Uporablja se za ročno testiranje aplikacije.

7.3 Težave pri pregledu

Prvi korak tega procesa je zajemal vprašalnik v obliki dokumenta, preko katerega so odgovorni za pregled aplikacij dobili začetni vpogled in njen osnovni namen. V nadaljevanju samega procesa smo se srečali tudi s pregledom v obliki spletne konference, na kateri je bilo treba podrobno razložiti posamezne dele aplikacije ter tehnično osnovo, na kateri je aplikacija grajena. Pri tem delu smo ugotovili, da aplikacija ne ustreza zahtevanim kriterijem in jo bo treba dopolniti. Kritika se je dotikala predvsem ločenosti uporabe aplikacije, saj je ni bilo mogoče uporabljati povsem znotraj CRM-ja. Oblikovanje in pošiljanje kampanje sta se še vedno vršila na strani za e-poštni marketing. To lahko razumemo kot težnjo dotičnega podjetja, da je njihov CRM središče uporabe in ne kot uporabnejšo funkcijo. Pred začetkom implementacije smo namreč raziskali način uporabe pri samih uporabnikih in ugotovili, da je uporaba v t.i. iframe težja, ker le-ta zoži vidno polje. Omenjene težave smo rešili na predlagan način, a še vedno verjamemo, da bi aplikacijo na tak način uporabljal le majhen del uporabnikov. Ostalih težav ni bilo, vendar se je varnostni pregled kljub temu zelo zavlekel in bil odobren šele po nekaj mesecih vstopa v ta proces.

Poglavje 8

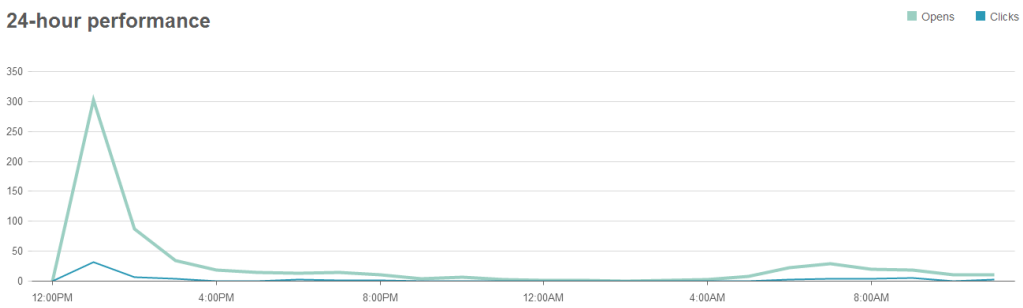
Sklep in nadaljnje delo

V diplomski nalogi smo se približali oblačnim platformam in še posebej raziskali vodilne oblačne sisteme za upravljanje odnosov s strankami. Kot smo ugotovili, je računalništvo v oblaku tudi na tem področju v velikem razmahu. Povsem realno je pričakovati, da se bo v prihodnosti vedno več podjetij odločalo za aplikacije v tem načinu. Posledično to pomeni upad klasičnih tehnologij in razvoja aplikacij na lokalnih omrežjih.

8.1 Težave pri izdelavi

Pri samem teoretskem delu diplomske naloge in analiziranju oblačnih platform se je izkazalo, da so nove tehnologije v resnično hitrem razvoju in tako je praktično skoraj nemogoče v realnem času spremljati nadgradnje in nove funkcionalnosti vseh ponudnikov. Salesforce konkretno na vsake štiri mesece izda novo različico CRM-ja, ki obsega množico novih funkcionalnosti. Če k temu prištejemo še vse ostale ponudnike in ponudnike, ki so novi na trgu, pridemo do množice tehnologij in platform, ki se nenehno razvijajo.

Implementacija prototipa aplikacije je tezo o nenehnem in težavnem spremljanju vseh posodobitev še bolj utrdila. V času razvoja so dotične platforme dozorevale in dobivale nove funkcionalnosti, ki bi v začetku lahko pospešile razvoj. Pomembno spoznanje pri povezovanju različnih oblačnih platform je to, da obstaja množica tehničnih omejitev, ki jih pri klasičnem razvoju v takšni obliki ne poznamo. To sicer ne pomeni, da so platforme manj zmogljive, ampak jih je treba



Slika 8.1: Časovni graf

uporabljati na bolj premišljen način.

8.2 Nadaljnji razvoj in priložnosti

Možnosti pri tem integratorju je izjemno veliko. Zaznali smo potrebo po avtomatičnem delovanju. Sedaj se vse akcije izvršijo po uporabnikovem ukazu. Ideja je sinhronizacijo avtomatizirati tam, kjer je to mogoče. Zapisovanje uspešnosti kampanje nazaj v CRM bi lahko izvajali po vnaprej določenem urniku. Podatki na grafu 8.1 kažejo na to, da večina ljudi odpre e-pošto v prvi uri po prejetju, potem se ta trend naglo manjša. Uvedli bi torej časovni režim, ki določa pogosto sinhronizacijo podatkov v prvih urah po poslani e-pošti in zmanjševanje sinhronizacije v odvisnosti od padanja števila odprtih e-pošt. Uporabnik bi potem v vsakem trenutku lahko videl trenutno stanje uspešnosti brez potrebe po ročnem zagonu. Aplikacija uporablja mnogo različnih API klicev na obeh straneh. Čeprav so iz ponudnika CRM-ja in e-poštnega marketing ponudnika zagotovljene visoko razpoložljivostne storitve, je pričakovati, da spletni servisi kdaj ne bodo dostopni. Izboljšali bi lahko detektiranje takšnih situacij in si v podatkovno bazo shranili trenutno stanje. S periodičnimi poskusi bi nadaljevali, dokler se operacija ne bi uspešno izvršila. Ob koncu diplomskega dela naj omenimo, da je ob raziskovanju in iskanju informacij bilo zaznati veliko število ljudi, ki se srečujejo s podobnimi vsebinskimi težavami. Tako bi bilo ob morebitnem dokončnem razvoju te aplikacije smiselno pričakovati uprabo tega integratorja v mnogih organizacijah.

Literatura

[1] F. Buttle, "Customer Relationship Management," *Elsevier Butterworth-Heinemann*, Oxford, 2004.

[2] "Security Guidance for Critical Areas of Focus in Cloud Computing" [Online]. Dostopno na: `https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf`, [2011]

[3] "The Notorious Nine Cloud Computing Top Threats in 2013"[Online]. Dostopno na: `https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf`, [2013]

[4] "Top Threats to Cloud Computing V1.0"[Online]. Dostopno na: `https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf`, [2010]

[5] "Why Salesforce Is Winning The Cloud Platform War"[Online]. Dostopno na: `http://www.forbes.com/sites/louiscolumbus/2013/10/07/why-salesforce-is-winning-the-cloud-platform-war/`, [2013]

[6] "Force.com Platform"[Online]. Dostopno na: `http://www.salesforce.com/platform/what/`, [2014]

[7] "Heroku platform"[Online]. Dostopno na: `http://www.heroku.com`,[2014]

[8] "The Rise of the Polyglot Cloud"[Online]. Dostopno na: http://readwrite.com/2011/10/28/heroku-ceo-byron-sebastian-the, [2011]

[9] "The Best Private Cloud Providers"[Online]. Dostopno na: http://www.businessnewsdaily.com/5259-best-private-cloud-providers.html, [2013]

[10] "PaaS market to reach $14 billion by 2017, IDC says"[Online]. Dostopno na: http://www.infoworld.com/d/cloud-computing/paas-market-reach-14-billion-2017-idc-says-230440, [2013]

[11] "Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS"[Online]. Dostopno na: http://www.rackspace.com/knowledge_center/whitepaper/understanding-the-cloud-computing-stack-saas-paas-iaas, [2013]

[12] "A Complete History of Cloud Computing"[Online]. Dostopno na: http://www.salesforce.com/uk/socialsuccess/cloud-computing/the-complete-history-of-cloud-computing.jsp, [2014]

[13] "OAuth 2.0"[Online]. Dostopno na: http://oauth.net/2/, [2014]

[14] "OAuth 2.0 workflow"[Online]. Dostopno na: http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=%2Fcom.ibm.tivoli.fim.doc_6226%2Fconfig%2Fconcept%2FOAuth20Workflow.html, [2014]

[15] "The OAuth 2.0 Authorization Framework"[Online]. Dostopno na: http://tools.ietf.org/html/rfc6749, [2012]

[16] "What is CRM?"[Online]. Dostopno na: http://www.salesforce.com/eu/crm/what-is-crm.jsp, [2014]

[17] "7 CRM Trends for 2014"[Online]. Dostopno na: http://www.enterpriseappstoday.com/crm/7-crm-trends-for-2014.html, [2014]

[18] "Salesforce Review 2014" [Online]. Dostopno na: http://crm-software-review.toptenreviews.com/salesforce-review.html, [2014]

[19] L. richardson and S. Ruby, "RESTful web services,"O'Reilly Media, 2007.

[20] "Create a Java Web Application using Embedded Tomcat" [Online]. Dostopno na: https://devcenter.heroku.com/articles/create-a-java-web-application-using-embedded-tomcat, [2014]

[21] "Apache Maven"[Online]. Dostopno na: http://maven.apache.org/, [2014]

[22] "PostgreSQL: Documentation 9.3"[Online]. Dostopno na: http://www.postgresql.org/docs/9.3/static/index.html, [2014]

[23] "Jersey"[Online]. Dostopno na: https://jersey.java.net/, [2014]

[24] "REST with Java (JAX-RS) using Jersey"[Online]. Dostopno na: http://www.vogella.com/tutorials/REST/article.html, [2012]

[25] "The Force.com Multitenant Architecture"[Online]. Dostopno na: https://developer.salesforce.com/page/Multi_Tenant_Architecture, [2014]

[26] "Salesforce Sales Cloud" [Online]. Dostopno na: http://www.egafutura.com/en/wiki/salesforce-sales-cloud, [2014]

[27] "Salesforce Service Cloud" [Online]. Dostopno na: http://www.egafutura.com/en/wiki/salesforce-service-cloud, [2014]

[28] "SalesForce Limits Quick Reference Guide" [Online]. Dostopno na: https://login.salesforce.com/help/pdfs/en/salesforce_app_limits_cheatsheet.pdf, [2014]

[29] "MailChimp for Salesforce"[Online]. Dostopno na: https://appexchange.salesforce.com/listingDetail?listingId=a0N3000000B3byfEAB, [2014]

[30] "Which API should I use?"[Online]. Dostopno na:
http://help.salesforce.com/HTViewHelpDoc?id=integrate_what_is_api.htm&language=en_US, [2014]

[31] "Bulk API"[Online]. Dostopno na:
https://www.salesforce.com/us/developer/docs/api_asynch/, [2014]

[32] "MailChimp API"[Online]. Dostopno na:
http://apidocs.mailchimp.com/api/, [2014]

[33] "Force.com ISV Security Review"[Online]. Dostopno na:
https://developer.salesforce.com/page/Security_Review, [2014]

[34] "Force.com Security Resources"[Online]. Dostopno na:
https://developer.salesforce.com/page/Security, [2014]

[35] "Top 10 2010 Security Risks"[Online]. Dostopno na:
https://www.owasp.org/index.php/Top_10_2010-Main, [2014]

[36] "Burp Suite"[Online]. Dostopno na:
http://portswigger.net/burp/, [2014]

[37] "MailChimp API Wrapper for Java"[Online]. Dostopno na:
https://github.com/Ecwid/ecwid-mailchimp, [2014]

Slike

Seznam kode