Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Lesjak

Uporabniški portal za upravljanje virov v oblaku

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentorica: doc. dr. Mojca Ciglarič

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00430/2013
Datum: 10.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATIC LESJAK**

Naslov: **UPORABNIŠKI PORTAL ZA UPRAVLJANJE VIROV V OBLAKU**
USER PORTAL FOR CLOUD RESOURCES MANAGEMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Opišite infrastrukturo oblaka in njegove storitvene modele ter primerjajte upravljalska orodja, ki se uporabljajo v bolj razširjenih oblačnih platformah. Analizirajte potrebe podjetja po upravljalskem portalu za uporabnike in preučite, ali se za ta namen lahko uporabi obstoječa orodja. Izberite primerne tehnologije za zasnovo in razvoj lastnega uporabniškega portala za upravljanje oblačnih virov. Portal implementirajte in kritično ovrednotite.

Mentor:
M. Ciglarič
doc. dr. Mojca Ciglarič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matic Lesjak, z vpisno številko **63100430**, sem avtor diplomskega dela z naslovom:

Uporabniški portal za upravljanje virov v oblaku

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mojce Ciglarič.
- 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 15. septembra 2013 Podpis avtorja:

Zahvaljujem se Janu Bervarju in podjetju NIL Podatkovne komunikacije, d.o.o., ki sta mi dala idejo in mi omogočila izdelavo te diplomske naloge. Posebej bi se zahvalil sodelavcu Marku Zagožnu za vso pomoč pri izvedbi, Poloni in Primožu za vso podporo, ki sta mi jo nudila v času mojega dela na NILu, Aljažu in vsem ostalim sodelavcem.

Zahvalil bi se tudi staršema - Ireni in Miljkotu, ki sta mi nesebično stala ob strani vsa ta leta in me podpirala pri vseh mojih dogodivščinah, sestri Petri, Gregorju, prijatelju Maticu za vso motivacijo, Simonu in Aljošu za vse preživete ure na fakulteti ter vsem ostalim, ki ste mi na kakršenkoli način pomagali.

Zahvaljujem se tudi mentorici, doc. dr. Mojci Ciglarič, za mentorstvo in pomoč pri izdelavi diplomske naloge.

Za konec pa bi se rad zahvalil moji dragi Neži za vso podporo in razumevanje.

Kazalo

Slike

Povzetek

Računalništvo v oblaku je trenutno en najhitreje rastočih trgov v računalniškem svetu. Veliko podjetij želi s svojimi rešitvami prodreti nanj. V okviru te diplomske naloge je nastal uporabniški portal za upravljanje virov v oblaku. Uporabniški portal je namenjen upravljanju oblačnih storitev, ki temeljijo na rešitvah podjetja VMware. Uporabniški portal uporablja orodje Cacti za spremljanje porabe dodeljenih virov, varnostni pregledovalec Nessus skrbi za preverjanje varnostnih ranljivosti ter VMware vCenter za upravljanje z infrastrukturo. Preko portala je mogoče upravljati virtualne naprave in se nanje povezati s pomočjo konzolne povezave. Za razvoj portala je bilo uporabljeno ogrodje AngularJS, ki zadnje čase postaja vedno bolj priljubljeno. Ogrodje se je izkazalo za izredno zmogljivo in enostavno za uporabo. Omogoča enostavno izgradnjo interaktivnih spletnih strani.

Ključne besede

AngularJS, oblak, spletni portal, Cacti, .NET, VMware

Abstract

Cloud computing is currently one of the fastest growing markets in computer world. Multiple companies would like to make a break trough with their solutions. For purpose of this thesis a user portal for cloud resource management was developed. User portal provides management tools for cloud, based on VMware solutions. User portal uses Cacti to monitor usage of allocated resources, Nessus Vulnerability Scanner for security testing and VMware vCenter for infrastructure management. User portal offers resource management and remote access console connection. User portal is build with increasingly popular AngularJS framework. Proved to be a very powerful tool and simple to use. It provides simple way to build interactive web page.

Keywords

AngularJS, cloud, web portal, Cacti, .NET, VMware

Poglavje 1

Uvod

V podjetju Nil d.o.o (v nadaljevanju Nil) se že nekaj let ukvarjajo z oblačnimi storitvami (stran 3). Kot prvi v Sloveniji so začeli ponujati pisarno v oblaku imenovano FlipIT. Storitev nudi virtualizirana namizna okolja na platformi Windows. Uporabniki dobijo oddaljeni dostop do svojih namizij. Povežejo se lahko preko namiznih računalnikov, prenosnikov, pametnih telefonov, tablic in lahkih odjemalcev (angl. Thin client). Nadrejeni pa dobijo dostop do portala, ki jim nudi upravljanje z zakupljenimi virtualni namizji. Vidijo lahko trenutno zakupljene vire in po potrebi zahtevajo nove, ter dobijo tudi celovit stroškovni pregled nad njihovo informacijsko tehnologijo. Podjetje želi razširiti svojo ponudbo oblačnih storitev tudi na IaaS (Infrastruktura kot storitev, v nadaljevanju IaaS) ter PaaS (Platforma kot storitev, v nadaljevanju PaaS). Storitev FlexIT je zasnovana z jasnim ciljem zagotavljanja aplikacijske platforme, ki je od primerljivejših storitev varnejša, zanesljivejša, bolj prilagodljiva, standardizirana in podprta z asistenco. Obstoječi portal za storitev FlipIT je osredotočen na upravljanje namizij in ne omogoča učinkovitega upravljanja strežniške infrastrukture. Podjetje potrebuje rešitev, ki bo omogočala spremljanje delovanja infrastrukture, njeno upravljanje in zaračunavanje. Strankam želimo ponuditi portal, ki bi jim ponujal samo osnovne nastavitve brez možnosti nastavljanja nižjih plasti. To bo v domeni podjetja in tam zaposlenih strokovnjakov. Rešitev mora biti ce-

novno ugodna in mora omogočati visoko stopnjo prilagodljivosti. Po pregledu obstoječih rešitev so se v podjetju odločili za nadgradnjo obstoječega FlipIT portala, saj obstoječe rešitve ponujajo uporabnikom preveč globoke nastavitve, prezahtevno jih je prilagoditi našim potrebam, so morda plačljive ali pa od nas zahtevajo prehod iz obstoječih VMware rešitev na kakšne druge. Vse prilagajanje, ki bi ga morali izvajati za naše stranke, bi nas ločilo od glavne razvojne veje, ki pa prinese veliko problemov. Če se za posodobitev ne bi odločili, bi to pomenilo potencialno izpostavljenost varnostnim grožjam. Posodobitev na novejšo verzijo pa bi prinesla vsakokratno združevanje programske kode in popravljanje združitvenih napak. Zaradi tega smo se odločili za lastno rešitev, implementacija le te pa je bila zaupana meni.

Poglavje 2

Računalništvo v oblaku

Računalništvo v oblaku (angl. Clound Computing) je koncept delovanja računalniških storitev, ki po potrebi omogoča mrežni dostop do deljenih računalniških virov, ki jih lahko hitro, enostavno in brez intervencije ponudnika, pripravimo in uporabimo. Računalništvo v oblaku je tako pogojeno z uporabo interneta kot medija za komunikacijo in prenos podatkov. Takšen model oblaka temelji na dosegljivosti storitve in je sestavljen iz 5 osnovnih značilnosti, 3 storitvenih modelov in 4 postavitvenih modelov. [1] Temelj vseh oblačnih storitev je virtualizacija.

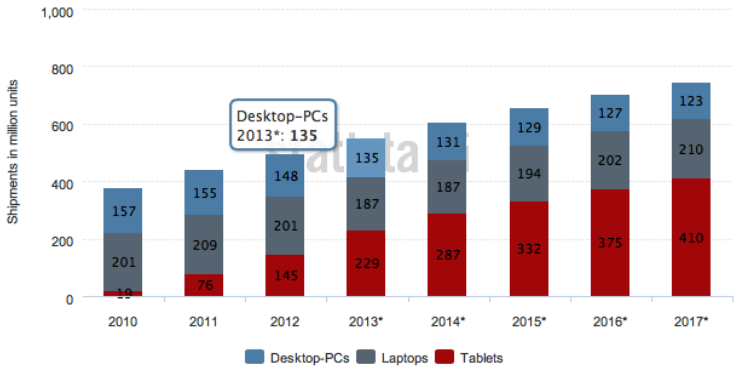
2.1 Virtualizacija

Računalništvo v oblaku si težko predstavljamo brez virtualizacijskih tehnologij. Po definiciji računalništva v oblaku virtualizacijske tehnologije niso pogoj za oblačne storitve, so pa le te še vedno najpogostejša in skoraj edina uporabljena izbira pri oblačnih ponudnikih.

2.1.1 Osnove virtualizacije

Virtualizacija je tehnologija, ki omogoča delovanje več operacijskih sistemov na enem fizičnem strežniku. Virtualizacija je drugi izraz za abstrakcijo. V praksi virtualizacija omogoča zmanjševanje stroškov, porabe in združevanje

sistemov na način, ki prej nikoli ni bil mogoč. Omogoča boljšo izrabo strojne opreme in pomaga graditi učinkovitejše sisteme. Ideja in izvedba virtualizacije ni nova. Njeni zametki so se začeli v 60. letih prejšnjega stoletja, ko so se trudili čimbolje izkoristiti takratne centralne računalnike (angl. Mainframe). Tudi takrat je več uporabnikov naenkrat uporabljalo en računalnik z enim operacijskim sistemom in se trudilo čimbolje izkoristiti njegove računske zmogljivosti. Takrat so začeli razvijati operacijske sisteme, ki so znali procesorski čas razdeljevati med več procesi ali uporabniki in so tako omogočili večopravilnost. Ko se je računalniška oprema začela ceniti in postajati vedno zmogljivejša, je uporaba virtualiziranja upadla. Zadnje čase pa pospešeno pridobiva na popularnosti, saj imamo sedaj že tako zmogljivo računalniško opremo, da jo en uporabnik ali operacijski sistem redko polno obremeni. Glede na raziskave *International Data Corporation* je povprečna obremenjenost strežnika med 10 – 15%. Okoli 85 – 90% računske moči teh naprav je tako neizkoriščene. Cilj virtualizacije je polno izkoristiti te naprave. [16]



Slika 2.1: Napoved prodaje računalniških naprav do leta 2017. [15]

Trenutni trend pri navadnih uporabnikih računalniških naprav se premika proti manj zmogljivim mobilnim napravam kot so tablice in mobilni telefoni. Te naprave se v veliki meri zanašajo na uporabo oblačnih storitev, saj same niso tako zmogljive. [16]

Z uporabo virtualizacije zmanjšamo stroške, saj lahko bolje izkoristimo strojno

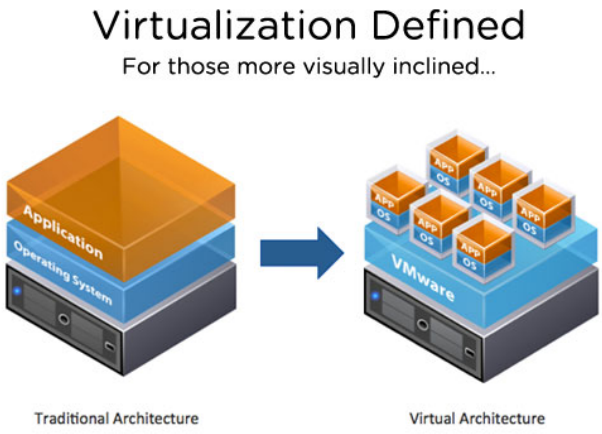
opremo in lahko ustvarimo več virtualnih naprav tako, da ni potrebno kupovati nove strojne opreme za vsako napravo. Bolje izkoristimo večjederne procesorje. Z uporabo virtualizacije lahko ustvarimo več testnih virualnih naprav in na njih testiramo programsko opremo. V primeru, da testiranje ne gre po načrtu, lahko virtualno napravo enostavno in hitro obnovimo na predhodnje stanje.[17]

2.1.2 Delovanje virtualizacije

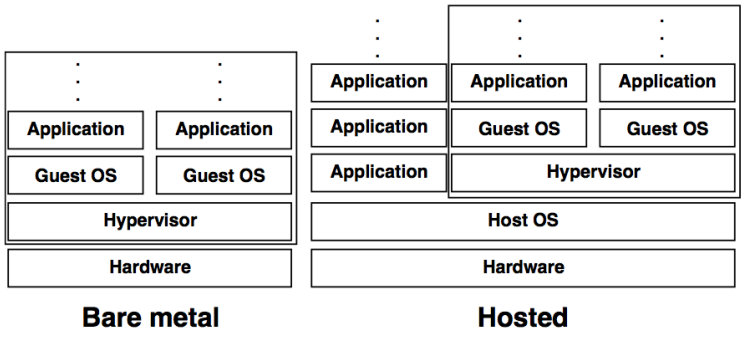
Delovanje računalniških sistemov na klasičen način bi lahko opisali na način en fizični računalnik, en operacijski sistem. Osnovni gradnik virtualizacije pa je virtualna naprava (angl. Virtual machine, v nadaljevanu VM). Virtualna naprava je ločena od strojne opreme in vsebuje operacijski sistem z aplikacijami. Ker je vsaka VM popolnoma ločena in neodvisna od strojne opreme, jih lahko na istem računalniškem sistemu teče več hkrati. Med VM in strojno opremo deluje lahek operacijski sistem, imenovan hipernadzornik (angl. Hypervisor). Naloga hipernadzornika je ločitev VM od strojne opreme ter razporejanje računskih virov med gostujočimi VM. Tako lahko na eni fizični napravi hkrati poganjamo več aplikacij in operacijskih sistemov. Z uporabo virtualizacije zmanjšamo število naprav, obstoječim pa povečamo izkoriščenost. Ker VM delujejo kot zaključena programska celota, jih lahko upravljamo na način kopiraj in prilepi. VM lahko prenašamo med fizičnimi napravami tudi med delovanjem.

Ločimo dve vrsti virtualizacijske arhitekture. Prva je domorodna in se imenuje Bare Metal, druga pa je gostujoča (angl. Hosted). Pri prvi hipervizor teče neposredno na strojni opremi, medtem ko pri drugi hipervizor deluje kot gost v operacijskem sistem, ki deluje na strojni opremi. Gostitelji hipervizorja so lahko trenutno razširjeni operacijski sistemi, kot so Linux, Windows in Mac OS. Pri gostujoči arhitekturi lahko poleg hipervizorja tečejo tudi druge aplikacije. [19]

V strežniških okoljih se običajno uporablja Bare metal pristop, za uporabo na namizju pa gostujoči. Ne glede na način virtualizacijske arhitekture,



Slika 2.2: Vizualni prikaz virtualizacije. [18]



Slika 2.3: Grafični prikaz virtualizacijske arhitekture. [19]

pa je vsem gostujočim operacijskim sistemom skupno to, da imajo svojo virtualizirano strojno opremo. Ta vključuje:

- CPE,
- pomnilnik,
- pomnilniške naprave,
- pomnilniške krmilnike,

- mrežne krmilnike,
- zaslon in zvočno kartico,
- miško in tipkovnico.[19]

Fizične strežnike, ki uporabljajo Bare metal pristop ponavadi združimo v večji bazen virov (angl. Resource pool), tako združene vire pa kasneje lahko delimo na nove bazene, glede na različne kriterije (namembnost, organizacijska struktura, oddelki). Posameznemu bazenu določimo količino virov, ki jih ima na voljo. Količina virov, ki so na voljo, praviloma presega računske zmogljivosti enega strežnika. V ta bazen lahko postavimo VM, ki pa ne more imeti določenih več računskih virov, kot posamezen fizični strežnik.
Pri uporabi virtualizacije virtualiziramo tudi pomnilniške diske. Vsaka VM ima tudi svoj virtualizirani disk, te pa hipernadzornik vidi kot običajne datoteke. To prinaša veliko prednosti, saj je tako mogoče lažje in hitreje prenašati diskovne datoteke med različnimi diskovnimi polji. Najnovejše tehnologije omogočajo tudi dinamično povečevanje datotek. Tako nam med prenašanjem slik med diskovnimi polji ni potrebno prenašati tudi praznega prostora. Zaradi tega so prenosi hitrejši.[20][21]

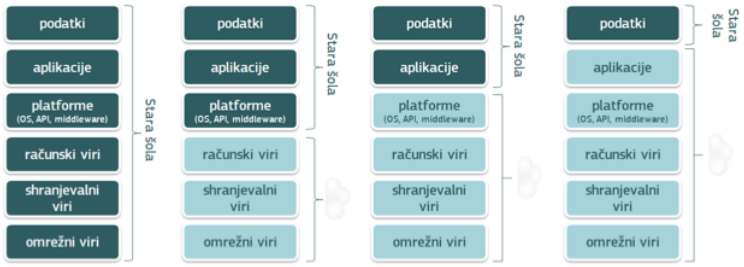
2.2 Osnovne karakteristike oblaka

Uporaba virtualiziranih računalniških sistemov je že ustaljena praksa, da lahko virtualizirane sisteme označimo kot računalniški oblak (v nadaljevanju oblak), pa moramo zadostiti naslednjim zahtevam:

- **Samopostrežba na zahtevo:** Uporabnik lahko samostojno in samovoljno določa računsko zmogljivost brez potrebe po udeležbi osebja ponudnika.
- **Univerzalen mrežni dostop:** Viri so na voljo prek standardnih omrežij in so preko standardnih mrežnih mehanizmov na voljo uporabnikom in uporabniškim napravam.

- **Združevanje virov:** Ponudnikovi viri (pomnilnik, procesor, hramba podatkov, platforme, aplikacije, namizja) so zbrani v homogeno celoto. Virov ne ločujemo glede na namen; lahko jih uporabljajo vsi informacijski procesi. Uporabnik oblaka nad fizično lokacijo dodeljenih virov nima kontrole, ne ve kje se dodeljeni viri nahajajo. Na isti infrastrukturi tako sobiva več uporabnikov (angl. multitenancy).
- **Visoka elastičnost:** Vire lahko hitro in elastično določimo, v nekaterih primerih jih avtomatično prilagodimo trenutnim potrebam. Razpoložljivi viri uporabniku običajno delujejo neomejeno in so na voljo za takojšen zakup.
- **Merjena storitev:** Oblačni sistemi merijo porabo virov in njihovo uporabo zaračunavajo uporabnikom. Ponudnik in uporabnik lahko pregledujeta porabo virov.[1]

2.3 Tipi storitvenih modelov



Slika 2.4: Primerjava oblačnih storitvenih modelov s klasičnim pristopom.[2]

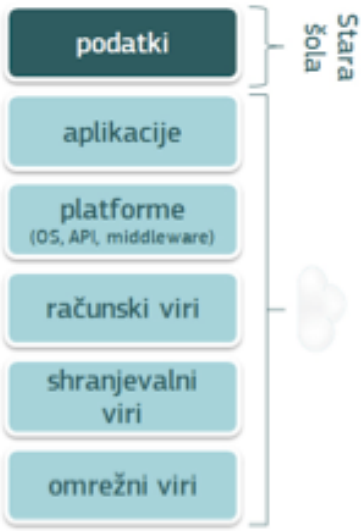
Za zagon storitev na klasični način potrebujemo naslednje:

- računske vire,
- shranjevalne vire,
- omrežne vire,

- platformo,
- aplikacije
- in podatke.

Poznamo tri različne tipe storitvenih modelov. Ti se med seboj razlikujejo po tem, koliko kontrole nad klasičnimi viri imamo. Vsi storitveni modeli imajo dobro definirano ločnico med viri za katere skrbi ponudnik in tiste, za katere uporabnik.

2.3.1 Programska oprema kot storitev (SaaS)



Slika 2.5: Sklad storitve SaaS. [2]

Programska oprema kot storitev (v nadaljevanju SaaS, angl. Software as a Service) je definirana kot delujoča storitev v oblaku, kjer mora uporabnik za uporabo storitve samo naložiti podatke in lahko storitev že uporablja. Tako se uporabniku ni potrebno ukvarjati s strojno opremo, operacijskim sistemom in programsko opremo, saj za to skrbi ponudnik. Ponudnik lahko storitev

ponuja zastonj, preko naročnin ali zaračunava po porabi. Rešitev ustreza storitvenemu modelu SaaS, kadar se drži splošnih načel računalniškega oblaka. Osnovne karakteristike SaaS oblaka vključujejo:

- spletni dostop do programske opreme,
- centralni nadzor nad programsko opremo,
- programska oprema se uporablja na način »en - mnogim«,
- uporabniki ne skrbijo za posodobitve programske opreme
- in programski vmesnik (API), ki omogoča integracijo med različnimi programi. [3]

SaaS storitveni model je priporočljivo uporabiti v naslednjih primerih:

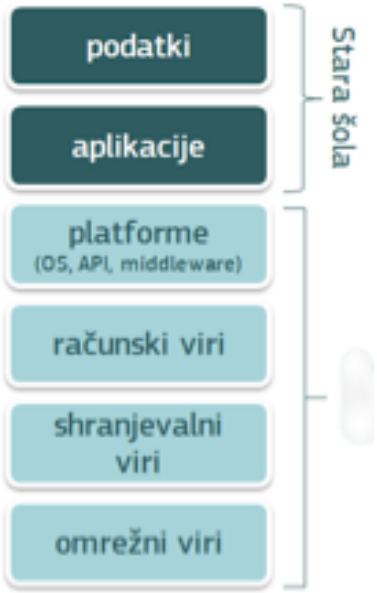
- za programsko opremo, ki se jo potrebuje redko, vendar ob uporabi potrebuje veliko računskih zmogljivosti (obračunski in davčni programi, ki se zaganjajo 1x mesečno),
- za programsko opremo, ki se uporablja krajši čas (orodja za sodelovanje na specifičnih projektih),
- za aplikacije, kjer uporabniki potrebujejo spletni ali mobilni dostop,
- za aplikacije, kjer je veliko prepletanja med organizacijo in zunanjimi uporabniki (e-poštno oglaševanje)
- in za aplikacije, ki jih lahko uporabljamo brez posebnih prilagoditev (e-pošta) in so za uporabnika zelo pomembne, vendar tehnologija sama po sebi ne prinaša nobene prednosti pred konkurenco.

SaaS storitveni model na naslednjih področjih ni najboljša izbira:

- za aplikacije, kjer potrebujemo ekstremno hitro procesiranje podatkov v realnem času,

- za aplikacije, kjer nam zakonodaja ali katera druga regulacija, ne dovoljuje gostovanja podatkov pri zunanjih ponudnikih,
- za aplikacije, ki trenutno zadovoljujejo vse potrebe organizacije.

2.3.2 Platforma kot storitev (PaaS)



Slika 2.6: Sklad storitve PaaS. [2]

Platforma kot storitev (v nadaljevanju PaaS, angl. Platform as a Service) se premika nižje po skladu IT storitev. Prednosti, ki jih je prinesel SaaS model na aplikacije, prinaša v svet razvoja programske opreme. PaaS lahko definiramo tudi kot računsko platformo, ki omogoča hitro in enostavno ustvarjanje spletnih aplikacij, brez kompleksnosti kupovanja ter vzdrževanja spodaj ležeče programske opreme in infrastrukture. Pri PaaS ponudnik oblačnih storitev skrbi za delovanje strojne opreme in operacijskega sistema, uporabnik pa mora sam namestiti ali razviti programsko opremo in poskrbeti za podatke.

Osnovne PaaS karakteristike vsebujejo:

- storitve za razvoj, testiranje, postavitev, gostovanje in vzdrževanje aplikacije v vgrajenem razvojnem okolju. Različne storitve morajo zadovoljiti proces razvoja aplikacije,
- spletno orodje, ki omogoča ustvarjanje, spreminjanje, testiranje in postavitev različnih uporab uporabniškega vmesnika,
- vgrajeno skalabilnost postavljene programske opreme, ki vključuje tudi izenačevanje obremenitve in zagotavljanje nadomestnega načina delovanja,
- integracijo s spletnimi servisi in podatkovnimi bazami,
- orodja za obračunavanje in urejanje naročnin.

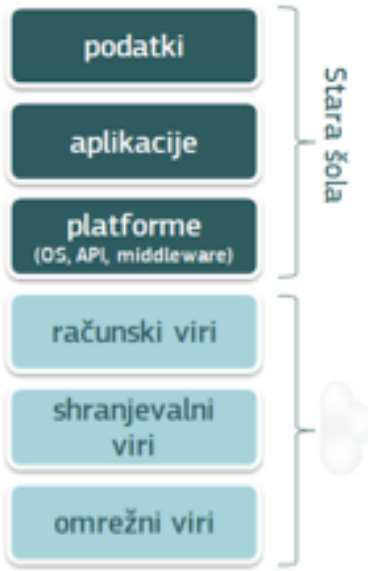
PaaS se najbolj uporablja v situacijah, kjer na enem projektu deluje več ljudi ter je potrebno avtomatsko testiranje in vzdrževanje.

PaaS se ne obnese najbolje v situacijah, kjer:

- mora biti aplikacija prenosljiva v smislu lokacije gostovanja,
- lastniški pristopi ali programski jeziki vplivajo na razvojni proces,
- lastniški jezik onemogoča kasnejši prestop k drugemu ponudniku (angl. vendor lock-in, slov. zaklep na ponudnika),
- povečanje hitrosti delovanja aplikacije zahteva prilagajanje nižje ležeče strojne in programske opreme.[3]

2.3.3 Infrastruktura kot storitev (IaaS)

Infrastrukturta kot storitev (v nadaljevanju IaaS, angl. Infrastructure as a Service) je ponudba oblačne infrastrukture (strežnikov, pomnilniškega prostora, omrežne opreme in operacijskih sistemov) na zahtevo. Uporabnikom ni potrebno kupiti fizičnih strežnikov in ostale infrastrukture, te storitve zakupijo v oblaku. Uporabnik tako zakupi virtualizirano strojno opremo za



Slika 2.7: Sklad storitve IaaS. [2]

katero skrbi upravljalec oblaka, sam pa mora poskrbeti za vso programsko opremo, ki jo želi uporabljati.

Osnovne karakteristike storitev IaaS:

- razdeljevanje virov kot storitev,
- možnost dinamičnega lestvičenja[36],
- plačilo po porabi,
- in na eni strojni opremi gostuje več uporabnikov.

IaaS se najbolje obnese:

- kjer prihaja do velikih in kratkih obremenitev na infrastrukturo,
- pri novih podjetjih, ki nimajo dovolj denarja za investicije v strojno opremo
- in pri hitro rastočih podjetjih, ki bi s težavo dohajale rast.[3]

IaaS ni priporočljivo uporabiti na področjih, kjer lokalna zakonodaja ne dopušča procesiranja podatkov izven podjetja, ali pa nam trenutna infrastruktura zadošča.

2.4 Postavitveni modeli

Pri postavitvenih modelih vedno nastopata dva tipa udeležencev; ponudnik in uporabnik virov. Postavitveni modeli določajo, kdo je ponudnik virov računalništva v oblaku. Glede na ponudnike in uporabnike ločimo štiri postavitvene pristope.

2.4.1 Privatni oblak (angl. Private Cloud)

Privatni oblak je oblak ,pri katerem sta ponudnik in uporabnik ista organizacija. Vsebuje lahko različne tehnologije in aplikacije. Najpomembneje pa je, da oblak ni na voljo javnosti.

2.4.2 Skupnostni oblak (angl. Community Cloud)

O skupnostnem oblaku govorimo kadar organizacija, ki ponuja oblak, tega ponuja določenem interesnem krogu drugih organizacij. Z oblakom upravlja ena ali več organizacij, zunanji izvajalec ali pa več njih.

2.4.3 Javni oblak (angl. Public Cloud)

Oblak je javen, kadar ga organizacija ponuja v najem poljubnim organizacijam in je oblak večstanovalski. Lastnik oblaka je lahko akademska skupnost ali vlada. Oblak se nahaja pri lastniku.

2.4.4 Hibridni Oblak (angl. Hibrid Cloud)

Kombiniranje postavitvenih modelov nas pripelje do hibridnega oblaka. Če uporabnik oblačnih storitev hkrati uporablja več postavitvenih modelov go-

vorimo o hibridnem oblaku. Vsaka od postavitev je ločena od druge. Občutljive podatke uporabniki pogosto gostijo v zasebnem oblaku, manj občutljive pa v javnem.

2.5 Prednosti računalniških oblakov

Računalniški oblaki prinašajo uporabnikom veliko prednosti. Izpostavil bi naslednje:

- **Mobilni dostop**: Zaradi načina delovanja oblakov se je nanje mogoče povezati kjerkoli. Potrebujemo samo internetno povezavo. Infrastrukturo tako lahko nadzorujemo in upravljamo kjerkoli.
- **Plačilo po porabi:** Pri oblačnih storitvah zaračunavanje praviloma poteka glede na uporabo virov. Zato vedno plačujemo toliko, kot porabimo. Zaradi nadgradljivosti virov nam teh ne zmanjka.
- **Stroškovna učinkovitost:** Največja prednost oblaka je prihranek pri dragih investicijah v strojno in programsko opremo ter stroških vzdrževanja le te. Z uporabo oblačnih storitev prihranimo pri licenčninah, stroških hrambe podatkov, programskih posodobitev, upravljanju, itd... Stroški uporabe oblaka so večinoma cenejši od tradicionalnih pristopov.
- **Mobilni dostop in stalna razpoložljivost:** Javni oblaki ponujajo storitve, ki so na voljo, neodvisno od uporabnikove lokacije. Tako se lahko uporabnik s pomočjo mrežne povezave poveže v oblak in ga upravlja. Takšen pristop olajša delo uporabnikov oblačnih storitev. Ponudniki oblačnih storitev zagotavljajo visoko razpoložljivost. Tipično imajo vgrajene varovalne sisteme, ki zagotavljajo redundančnost storitev, ki jih ponujajo. V primeru sistemske odpovedi se zaženejo alternativne instance na drugi lokaciji.
- **Varnostno kopiranje in obnovitve:** Računalništvo v oblaku poenostavi postopke izvajanja varnostnega kopiranja in obnavljanja podat-

kov.

2.6 Slabosti računalniških oblakov

Oblačno računalništvo ima tudi svoje slabosti. Nekatere veljajo za vse vrste oblakov, nekatere so specifične.

- **Manj kontrole:** Z uporabo oblaka izgubimo del kontrole, ki smo je bili vajeni. To kontrolo predamo tistim, ki skrbijo za naš oblak.

- **Tehnične težave:** Lastnost oblaka je tudi to, da je vedno in povsod dosegljiv. Ampak tako kot pri vseh sistemih, se lahko tudi pri oblaku zgodijo kakšne težave. Kadar pride do takšnih težav ostanemo odrezani od podatkov, saj se ti nahajajo pri ponudniku in ne pri nas lokalno. Tudi najboljšim oblačnim ponudnikom se dogajajo izpadi.

- **Priklenitev na ponudnika:** Pri uporabi oblačnih storitev moramo paziti, da se ne priklenemo na ponudnika storitev, saj nam to na dolgi rok prinese veliko problemov. Prenašanje večje količine podatkov med ponudniki je lahko zelo drago.

- **Zasebnost v oblaku:** Kadar uporabljamo oblačne storitve se moramo zavedati, da podatkov ne hranimo pri sebi in so tako dosegljivi tudi ponudniku storitev. Če podatkov pred izvozom v oblak ne šifriramo, jih lahko bere tudi ponudnik in mogoče tudi državne varnostne službe.

2.7 Varnost oblačnih storitev

Uporaba oblačnih storitev prinaša, poleg klasičnih problemov varnosti IT infrastrukture, nove varnostne izzive. Prinaša tudi nekatere izboljšave na področju varnosti.

2.7.1 Varnostni izivi

Ne glede na tip uporabljene oblačne storitve se postavljajo naslednji varnostni pomisleki:

1. **Priviligiran dostop do podatkov:** Upravljalci oblaka, ki imajo nadzor nad infrastrukuturo, predstavljajo varnostno tveganje lastniku podatkov.

2. **Skladnost s predpisi:** Uporabniki nosijo odgovornost za varnost in celovitost podatkov.

3. **Lokacija podatkov:** Pri uporabi oblaka ne vemo natančno, kje se podatki nahajajo. Pri nekaterih ponudnikih tudi ni jasno, v kateri državi se nahajajo in kakšen režim varovanja zasebnosti in podatkov je tam v veljavi.

4. **Ločevanje podatkov:** Viri v oblaku so deljeni in zato obstaja možnost, da bi jih kdo lahko nepoblaščeno prebral ali spremenil. Šifriranje podatkov pri tem pomaga, ampak ne reši vseh težav.

5. **Obnovitev:** Ne glede na lokacijo tvojih podatkov, ti mora ponudnik povedati, kaj se bo zgodilo s podatki v primeru nesreče.

6. **Preiskovalna pomoč:** Pri preiskovanju varnostnih incidentov nujno potrebujemo pomoč ponudnika, saj ima le ta dostop do nekaterih podatkov, ki bi nam pomagali pri preiskovanju.

7. **Dolgoročno preživetje:** Poslovanje ponudnika oblačnih storitev mora biti stabilno. Od ponudnika je potrebno zahtevati dostop do naših podatkov tudi v primeru, če bankrotirajo ali pa jih kupi drugo podjetje.

Dostop do podatkov

Do podatkov v oblaku ima, poleg lastnika podatkov, vpogled tudi ponudnik. Tako lahko osebe s priviligiranim računom pregledujejo naše podatke tudi,

če jih za to nismo pooblastili. Enkripcija podatkov pa v tem primeru ne reši vseh težav. Če želimo te podatke uporabljati, se morajo nekaj časa nahajati v delovnem pomnilniku nekriptirani. V tem času pa jih lahko kdo nepooblaščeno prebere. Po razkritju ameriškega tajnega programa PRISM, kjer je ameriška obveščevalna agencija nepooblaščeno in neomejeno dostopala do vseh uporabniških podatkov pri velikih ameriških ponudnikih oblačnih storitev kot so Google, Microsoft, Apple, Dropbox in še veliko drugih, dobijo te storitve zelo negativen prizvok.[4]

Skladnost s predpisi

Uporabnik se mora zavedati, da lokalna zakonodaja velja samo nad podatki, ki se lokalno hranijo v isti državi. Še vedno pa veljajo predpisi, ki od podjetja zahtevajo, da varuje osebne podatke in da vanje ne dovolijo nepooblaščenega vpogleda. Slovenska zakonodaja je glede varstva osebnih podatkov stroga in ima tudi člene, ki govorijo o omejitvi iznosa podatkov v tretje države. Urad informacijskega pooblaščenca definira iznos: »*O iznosu podatkov v tretje države govorimo takrat, ko upravljalec iz države Evropske unije osebne podatke iz takšnega ali drugačnega razloga posreduje v države izven Evropske unije, ali ko je dostop do podatkov omogočen organizacijam, posameznikom, ipd. iz tretjih držav izven EU, pa čeprav so podatki hranjeni znotraj EU(62. člen ZVOP-1).*«[5]

Lokacija podatkov

Izbira lokacije podatkov je glede na prejšnji odstavek zelo pomembna. Nekaterih večji IaaS ponudniki (Amazon, Microsoft, RackSpace, GoGrid,...) ponujajo izbiro podatkovnega središča. Pri večini PaaS in SaaS storitvah pa nimamo nadzora nad lokacijo podatkov. Ne vemo, kateri zakoni zanje veljajo in kdo vse ima vpogled vanje.

Ločevanje podatkov

Zaradi deljenja virov v oblaku in delovanja virtualizirane infrastrukture, se med dodeljevanjem in odvzemanjem virov, predvsem pomnilnika, ti fizično ne pobrišejo, zato lahko teoretično, do teh podatkov dostopa tisti, ki bo naslednji uporabil te vire. Zaželjeno je, da sami počistimo podatke za seboj.[6]

Obnovitev podatkov

Tako kot pri ostalih storitvah, moramo tudi pri oblačnih poskrbeti za pravilno varnostno kopiriranje in obnovo v primeru odpovedi. Oblak lahko uporabimo za hrambo varnostnih kopij in tako izboljšamo varnost naših podatkov. Podatke pa je pred hrambo zaželjeno šifrirati. Če hranimo podatke tudi v oblaku, se izboljša njihova varnost, saj se nahajajo na dveh različnih lokacijah.

Preiskovalna pomoč

Preiskovanje nezaželjenih ali ilegalnih aktivnosti je lahko brez sodelovanja ponudnika skoraj nemogoče. Težava pa leži v tem, da so lahko dnevniške datoteke deljene med več različnimi uporabniki in se lahko nahajajo v različnih podatkovnih centrih. V tem primeru uporabnik brez zagotovitve podatkov in asistence ponudnika ne more temeljito raziskati aktivnosti.

Dolgoročno preživetje

Pri prehodu v oblak postanemo odvisni od ponudnika storitev. Če ponudnik propade ali pa če mu strojno opremo iz kakršnega koli razloga zasežejo, izgubimo dostop do podatkov. Tako lahko tvegamo, da svojih podatkov nikoli več ne dobimo nazaj. Zgodba spletne strani za hrambo datotek Megaupload nam prikaže te slabosti, saj uporabniki že več kot leto čakajo na svoje podatke, ki so jih gostovali pri tem ponudniku. Sedaj je tudi jasno, da, vsaj Evropski uporabniki, teh podatkov nikoli ne bodo dobili nazaj, saj je podjetje LeaseWeb, kjer je gostoval evropski podatkovni center podjetja Megaupload

pobrisalo 630 strežnikov, ki so gostovali pri njih. [7] Težava nastopi tudi takrat, ko si uporabnik oblačnih storitev ne more več privoščiti plačevanja teh storitev. Ker so podatki pri ponudniku, lahko ta prepreči dostop do podatkov, dokler uporabnik ne poravna vseh obveznosti.

2.7.2 Varnostne izboljšave

Poleg nekaterih novih varnostnih izzivov pa je računalništvo v oblaku prineslo tudi izboljšave. Hranjenje podatkov v oblaku olajša njihovo kontrolo ob izgubi naprav, s katerimi se v oblak povezujemo. Izgubljeni napravi preprečimo dostop do oblaka in se tako izognemo uhajanju podatkov. Infrastrukturo v oblaku je hitreje in enostavneje očistiti zlonamerne programske opreme.[8]
Z uporabo oblačnih storitev se oddaljimo od problemov strojne opreme, saj je le ta v domeni ponudnika. V primeru odpovedi enega strežnika njegovo breme prevzame drugi. Nekatere virtualizacijske tehnologije omogočajo direktno replikacijo navidezne naprave in v primeru izpada primarne delo takoj prevzame sekundarna.[9]
Varnost podatkov v oblaku lahko izboljšamo z razpršitvijo podatkov med strežniki. To storimo tako, da datoteko d razdelimo na n fragmentov, ki jih podpišemo in razpršimo na n strežnikov. Uporabnik lahko rekonstruira datoteko d s sestavitvijo m izbranih fragmentov. Napadalec mora dobiti dostop do m strežnikov, da lahko obnovi datoteko. Varnost lahko še izboljšamo tako, da podatke pred fragmentiranjem šifriramo.[10]

Poglavje 3

Orodja za upravljanje oblakov

Rešitev za upravljanje oblakov je veliko. Zaradi izredne popularnosti in hitro rastočega trga ponudnikov oblačnih storitev, je na voljo veliko število odprtokodnih in zaprtokodnih rešitev. Ponudniki platform se običajno med seboj razlikujejo samo v nekaterih podrobnostih. Rešitve na tržišču so:

- VMware vCloud,
- OpenStack,
- CloudStack,
- OpenNebula,
- Eucalyptus,
- Cloupia,
- RightScale,
- Flexiant
- in še veliko drugih.

Pred odločitvijo za lastno implementacijo smo opravili pregled nekaterih obstoječih rešitev in jih primerjali z internimi zahtevami. Rešitve našim

zahtevam ne ustrezajo, ker nudijo uporabnikom portala več upravljalskih možnosti, kot bi jih mi želeli nuditi. Radi bi ohranili možnost prilagajanja uporabniškega portala željam naših strank. Spreminjanje in prilagajanje izvorne kode potrebam naših strank pa prinese veliko dolgoročnih stroškov. Pri nadgradnjah verzij moramo vsakič združevati naše popravke in prilagoditve z novo kodo. Če pa se za nadgradnjo ne odločimo, tvegamo izpostavljenost varnostnim luknjam.

3.1 VMware vCloud

Podjetje VMware nudi programsko rešitev vCloud, ki omogoča učinkovito upravljanje z oblačno infrastrukturo. Preko vCloud Directorja, ki je del programskega paketa vCloud, lahko upravljamo virtualne podatkovne centre. Omogoča kontrolo nad računskimi, shranjevalnimi in mrežnimi viri. Omogoča učinkovito porazdelitev virov med različne fizične strežnike in integracijo s tehnologijo vSphere Storage.

Hitro oskrbovanje virov omogočajo naslednje tehnologije:

- vApp predloge,
- vApp katalogi,
- vSphere povezani kloni,
- trenutni posnetki in obnovitve.

Programska oprema omogoča dostop do odprtega programskega vmesnika. Zaradi tega so mogoči enostavni prenosi med različnimi oblačnimi storitvami, saj se ohranjajo nastavitve in mrežne konfiguracije. Programski vmesnik omogoča tudi integracijo z drugimi sistemi, saj omogoča prejemanje in pošiljanje sporočil. Za uporabnike nudi dostop do spletnega portala, kjer lahko upravljajo z viri.[11]

Rešitev je napredna in omogoča uporabnikom precejšen nadzor nad njihovo infrastrukturo. Programska oprema vCloud Director ni odprtokodna in

tako ne nudi vseh možnosti modifikacije, ki bi jih sami želeli. Želimo uporabljati rešitev, ki jo lahko prilagodimo vsem zahtevam naših naročnikov, pri zaprtokodnih rešitvah pa te možnosti nimamo. Za uporabo te programske opreme je potrebno plačevati licenčnino.

3.2 OpenStack

OpenStack je odprtokodni oblačni operacijski sistem za nadzor večjih količin računskih, pomnilniških in mrežnih virov v podatkovnem centru. OpenStack vsebuje naslednja orodja:

- Compute (Nova),
- Object Storage (Swift),
- Block Storage (Cinder),
- Networking (Neutron),
- Dashboard (Horizon),
- Identity Services (Keystone),
- Image Services (Glance).

Projekt upravlja in razvija fundacija OpenStack, ki združuje velika podjetja, kot so AMD, Canonical, Cisco, Dell, EMC, HP, IBM, Intel, Rackspace, Red Hat, SUSE Linux, VMware in Yahoo. Podprti pa so naslednji hipernadzorniki:

- KVM (Kernel-based Virtual Machine),
- LXC (Linux Containers),
- QEMU (Quick EMUlator),
- ULM (User Mode Linux),

- VMware vSphere,
- Xen,
- PowerVm,
- Hyper-V,
- Bare Metal.

To nam omogoča veliko izbiro in nas ne priklene na enega ponudnika hipervizorjev. Nadzorna plošča imenovana OpenStack Dashboard nam omogoča nadzor in oskrbovanje vseh virov. Napisana je v jeziku Python in Ruby.[12] Težava paketa OpenStack je kompleksna namestitev in zahteva, da podjetje preide iz trenutnih VMware vSphere programskih rešitev. Spletni vmesnik bi bilo potrebno modificirati, da bi ustrezal zahtevam podjetja. OpenStack podpira Nagios rešitev za monitoriranje infrastrukture v podjetju pa uporabljamo orodje Cacti za katerega smo razvili posebne vtičnike, ki omogočajo napredno spremljanje in obveščanje o porabi virov virtualnih naprav. Na orodje Cacti sta vezani tudi storitvi Assist in Monitor, ki imata pomembno vlogo v podjetju. Prehod na drugi sistem obveščanja bi bil preveč zahteven in časovno potraten.

3.3 CloudStack

CloudStack je odprtokodna programska oprema za ustvarjanje, upravljanje in postavljanje infrastrukturnih oblačnih servisov. Deluje na naslednjih hipernadzornikih:

- KVM,
- vSphere,
- XenServer/XCP.[13]

Podpira tudi Amazon Web Services programski vmesnik. Prvotni razvijalec je bilo podjetje Cloud.com. V letu 2010 je večino programske kode odprlo pod licenco GPLv3. V letu 2011 je Citrix kupil Cloud.com in tudi preostalo kodo objavil pod GPLv3 licenco. CloudStack vsebuje tudi spletni vmesnik za nadzor oblaka. Tudi ta rešitev zahteva spremembo trenutnega produkcijskega okolja in ne vsebuje vseh želenih funkcionalnosti. Uporabniku ponuja več možnosti modificiranja in ustvarjanja, kot bi mu jih mi želeli dati.

3.4 OpenNebula

OpenNebula je odprtokodno ogrodje za upravljanje porazdeljene in heterogene infrastrukture podatkovnih centrov. Temelji na operacijskem sistemu Linux in podpira naslednje hipernadzornike:

- Xen,
- KVM,
- VMware.[14]

OpenNebula vsebuje spletni vmesnik za upravljanje z infrastrukturo. Ne podpira monitoriranja platforme VMware, ampak samo Xen in KVM. Rešitev, ki ne omogoča spremljanja in obveščanja o porabi virov in morebitnih napakah, ne ustreza zahtevam podjetja. Integracija orodja Cacti v OpenNebulo pa bi zahtevala več razvijalskega časa.

Poglavje 4

Izbrane tehnologije in orodja

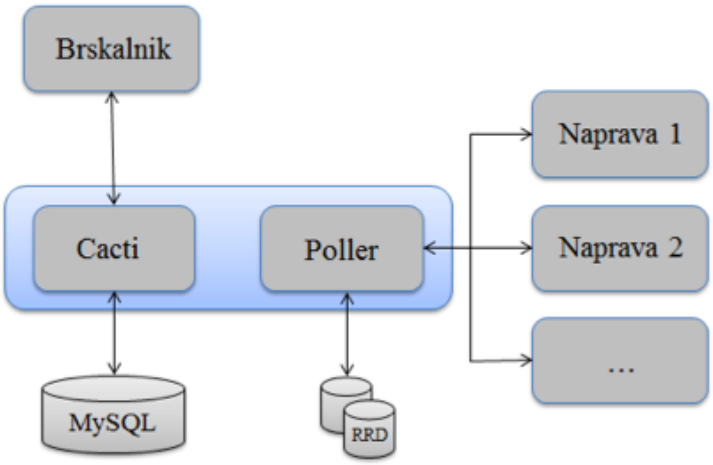
Pri implementaciji uporabniškega portala sem moral izbrati nekatere programe in knjižnice, ki so mi omogočile lažjo in učinkovitejšo implementacijo. Pri izdelavi sem moral uporabiti orodje Cacti in preiskovalec Nessus.

4.1 Orodje za nadzor omrežij Cacti

Cacti je odportokodno orodje za monitoriranje omrežji. Z njegovo pomočjo v podjetju izvajamo storitev Monitor. Orodje je sestavljeno iz naslednjih komponent:

- podatkovna baza MySQL,
- strežnik s »pollerjem«,
- brskalnik,
- RRD datoteke (Round Robin Database).[22]

Uporabljamo in nadzorujemo ga preko spletnega vmesnika, ki podatke pridobiva in shranjuje v bazo MySQL. Preko omrežja s pomočjo »pollerja« periodično zbira podatke o nadzorovanih napravah in jih shranjuje v datoteke RRD. Poller praviloma poganjamo na 5 minut. Običajen zajem podatkov



Slika 4.1: Gradniki orodja Cacti. [23]

poteka preko protokola SNMP, vendar orodje omogoča tudi poganjanje raznih skript, ki lahko podatke pridobijo iz drugih virov. Podatke za risanje grafov črpa iz datotek RRD. Prikazujemo lahko več vrednosti hkrati in izbiramo različne časovne intervale. Cacti uporablja RRDtool za shranjevanje podatkov, ki jih pridobi od omrežnih naprav. RRDtool podatke shranjuje v datoteke RRD (angl. Round Robin Database). *Zbirko podatkov si lahko predstavljamo kot vrtiljak, kjer se novi podatki zapisujejo na obod. Ko je krog sklenjen, začnejo novi podatki prepisovati stare. Velikost zbirke RRD je torej vedno enaka oz. konstantna. RRDtool lahko uporabimo tudi za izračun spremembe glede na prejšnjo vrednost in nato v zbirko podatkov shranimo izračunano vrednost glede na prej določene časovne intervale.*[23] Cacti je mogoče razširiti s pomočjo vtičnikov, preko katerih mu lahko dodajamo nove funkcionalnosti ali pa razširjamo obstoječe. V podjetju imamo razvit vtičnik, ki nam omogoča spremljanje virtualnih naprav. Podpira spremljanje virtualnih naprav, bazenov in strežnikov, ki poganjajo programsko opremo proizvajalca VMware preko nadzornega centra vCenter (del vSphere). Preko vtičnika se podatki shranjujejo v datoteke RRD, zadnje vrednosti pa hranimo tudi v podatkovni bazi MySQL.

4.2 Varnostni preiskovalec Nessus

Nessus je celovit mrežni preiskovalec varnostnih lukenj. Razvijajo ga v podjetju Tenable Network Security. Program je na voljo brezplačno za domače uporabnike, podjetja pa morajo za njegovo uporabo plačevati licenčnino. Program testira naslednje tipe ranljivosti:

- možnosti oddaljenega dostopa,
- nepravilne konfiguracije,
- tovarniško nastavljena in ostala šibka gesla
- in ohromitev storitve z DoS napadi.[25]

Program izvaja penetracijska testiranja naprav s preizkušanjem že odkritih varnostnih lukenj. Podpira pregledovanje mrežnih naprav večjih proizvajalcev, virtualne infrastrukture, operacijskih sistemov, podatkovnih baz, industrijskih kontrolnih sistemov in spletnih aplikacij. Omogoča tudi spremljanje stanja mobilnih naprav, nevarne programske opreme in namestitev popravkov.[24] Prepozna botnete in ostale nevarne procese ter nadzoruje delo protivirusnih programov. Nessus je izredno zmogljiva programska oprema, ki pomaga pri vzdrževanju varnosti sistemov.
V podjetju Nessus uporabljamo za varnostno pregledovanje produkcijskih strežnikov. To storitev pa bi sedaj želeli ponuditi tudi našim strankam.

4.3 Twitter Bootstrap

Twitter Bootstrap je zbirka orodij, ki nam omogočajo enostavno zasnovo ogrodja sodobnih spletnih strani. Izdan je pod licenco Apache Licence 2.0. Ima veliko lepo in sodobno oblikovanih HTML (angl. HyperText Markup Language) komponent, kot so tabele, obrazce, ikone, gumbi, tipografijo in navigacijo. Vsebuje tudi JavaScript komponente, kot so modalna okna, spustni meniji, zaslonski namigi in še nekaj drugih. Omogoča tudi uporabo

mreže za enostavno razporejanje elementov. Trenutno je ena najbolj popularnih zbirk ogrodij med razvijalci spletnih strani, saj omogoča enoten izgled v vseh popularnih spletnih brskalnikih (Chrome, Firefox, Opera, Safari, Internet Explorer). To pa razvijalcem skrajša in poenostavi razvoj.[26]
Pri projektu sem se odločil za uporabo Twitter Bootstrapa zaradi preproste izdelave spletnih strani z modernim videzom. Od vseh orodji zbirke uporabljam samo datoteko CSS (Cascade Style Sheet).

4.4 AngularJS

AngularJS je odprtokodno programsko ogrodje napisano v programskem jeziku JavaScript. Pod okrilje ga je vzelo podjetje Google. Cilj ogrodja je bogatenje brskalniško orientiranih aplikacij. Ogrodje je nekakšna mešanica načinov MVC (angl. Model-View-Controler, slov. Model-pogled-krmilnik), MVVM (angl. Model-View-ViewModel) in MVP (Model–View–Presenter), zato ga je avtor razglasil za model MVW (angl. Mode-View-Whatever, slov. Model-pogled-karkoli).[32] AngularJS prinaša hitrejše in enostavnejše razvijanje ter testiranje spletnih aplikacij.

Knjižnica na uporabniški strani loči aplikacijsko logiko in podatke. To si enostavneje lahko razlagamo kot ločitev statičnega dela strani (predloge HTML), dinamičnih podatkov in aplikacijske logike (krmilnik). Posledično lahko v predpomnilniku brskalnika dlje časa hranimo statične dele strani in zmanjšamo količino prenešenih podatkov, saj je dinamični del strani manjši. Knjižnica prebira posebne direktive in atribute iz HTML dokumenta ter na podlagi tega povezuje vhodne ali izhodne dele spletne strani z modelom predstavljenim v standardnih JavaScript spremenljivkah.
Ogrodje sprejme in razširi tradicionalni označevalni jezik HTML tako, da ta bolje deluje pri prikazovanju dinamičnih spletnih strani. Vsebuje dvosmerno povezovanje podatkov, kar omogoča avtomatično sinhronizacijo modela in pogleda. AngularJS tako zmanjšuje pomen manipulacije drevesa DOM (Document Object Model) in izboljšuje testne zmožnosti. Omogoča iz-

delavo svojih HTML elementov imenovanih direktive. V datoteki JavaScript določimo predvideno delovanje elementa in ga uporabimo v označevalnem jeziku HTML. Zaradi kasnejšega dinamičnega nalaganja vsebine strani, teh spletni iskalniki še ne znajo prebrati in ovrednotiti njihove vsebine.[33]

4.4.1 Uporabniški vmesnik Angular (AngularUI)

Poleg ogrodja AngularJS lahko uporabimo že zgrajene elemente grafičnega vmesnika imenovane AngularUI. Ta vsebuje s pomočjo AngularJS že zgrajene standardne prikazne elemente strani, ki jih lahko prosto uporabimo na naši strani. Na voljo so nam naslednji sklopi gradnikov:

- UI-Utils,
- UI-Modules,
- UI-Alias,
- UI-Bootstrap,
- NG-Grid,
- UI-Router.[27]

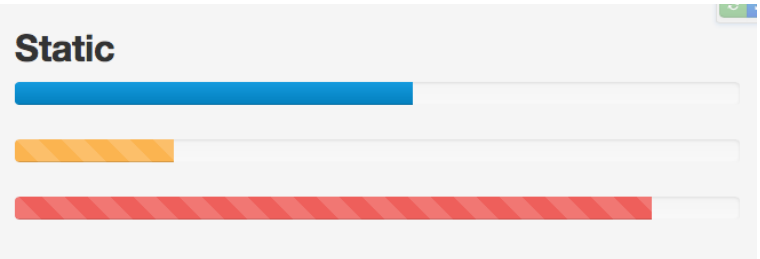
UI-Bootstrap

Sklop gradnikov UI-Bootstrap prinaša funkcionalnosti Twitter Bootstrap napisane s pomočjo AngularJS. Gradniki za delovanje potrebujejo samo bootstrap CSS datoteko in niso odvisni od zunanjih JavaScript knjižnic. Uporabo gradnikov si lahko ogledamo na spodnjem primeru.[28]

Izvorna koda 4.1: Uporaba gradnikov v datoteki HTML.

```
<div ng-controller="ProgressDemoCtrl" class="well">
    <h2>Static</h2>
    <div class="row-fluid">
        <div class="span4">
                <progress percent="55">
                </progress>
        </div>
        <div class="span4">
```

```
                <progress percent="22" class="progress-warning progress-striped">
                </progress>
        </div>
        <div class="span4">
                <progress percent="88" class="progress-danger progress-striped active">
                </progress>
        </div>
    </div>
</div>
```



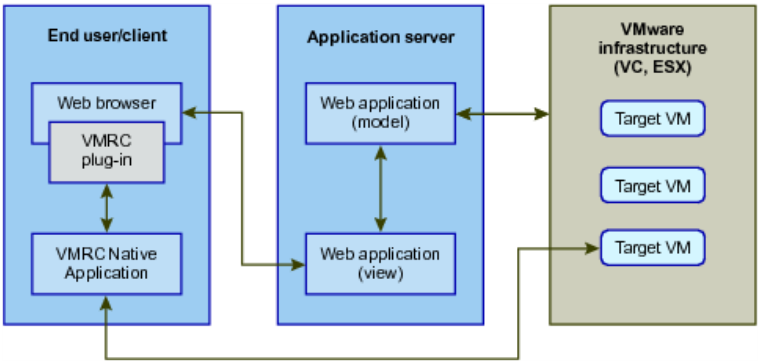
Slika 4.2: Rezultat zgoraj napisane kode. [28]

4.4.2 Vmware Remote Console

Vmware Remote Console (v nadaljevanju VMRC) je aplikacija, ki omogoča ustvarjanje spletnih vmesnikov za interakcijo z upravljalnikom VM. Aplikacija deluje v operacijskih sistemih Microsoft Windows in Linux. Poleg aplikacije pa dobimo tudi vtičnik za spletni brskalnik. Trenutno podprti brskalniki so Mozilla Firefox, Internet Explorer in Google Chrome. Če želimo VMRC uporabiti za povezavo na VM, moramo napisati spletno aplikacijo. Ta aplikacija mora naložiti brskalniški vtičnik ter se preko vtičnika povezati na VMRC, ki smo jo namestili na računalnik. Po vzpostavitvi povezave z vtičnikom preko JavaScripta podajamo ukaze za vzpostavitev konzolne povezave. Ko vzpostavljamo povezavo, imamo 2 možnosti prijave:

- standardno (uporabniško ime in geslo),
- enkratno (sejni identifikator).

Pri standardni prijavi je potrebno ustvariti uporabniška imena in gesla za vsakega uprabnika posebej in zato ni najbolj optimalna rešitev za spletni



Slika 4.3: Prikaz arhitekture VMRC.[30]

portal. Enkratni sejni identifikator je v tem primeru dosti boljša rešitev, saj vedno sproti generira sejne identifikatorje s katerimi se lahko prijavimo samo enkrat in še to v omejenem času. Ko imamo vzpostavljeno povezavo z VM, lahko preko vtičnika aplikacijo poženemo v celozaslonskem načinu in dovolimo uporabitev miške in tipkovnice. Če želimo, lahko v VM priključimo tudi USB naprave.[31]

Poglavje 5

Spletni portal za upravljanje oblačnih storitev FlexIT

5.1 Funkcijske zahteve

Projekt je razdeljen na 3 stopnje. Prva stopnja zagotavlja spremljanje infrastrukture, druga upravljanje in tretja zaračunavanje. Aplikacija mora prilagoditi in nadgraditi trenutni FlipIT spletni portal za upravljanje s pisarno v oblaku. V sklopu diplomske naloge sem implementiral prvi dve stopnji, tretja pa ostaja kot prihodnji projekt.

5.1.1 Prva stopnja: Spremljanje infrastrukture

V prvem razvojnem koraku želimo omogočiti spremljanje delovanja infrastrukture. Uporabnikom portala bi radi omogočili vpogled v delovanje njihove infrastrukture. Za vsak strežnik bi želeli prikazati trenutno in preteklo porabo virov (CPE, pomnilnik in diskovni prostor) in število zakupljenih virov. Prikazati bi želeli v kakšnem stanju je posamezni strežnik, pod kakšen SLA (angl. Service Level Agreement) spada, rezultate varnostnega pregleda z orodjem Nessus ter tip operacijskega sistema. Podatke o infrastrukturi želimo pridobiti z orodja Cacti. Za orodje Cacti bo potrebno razviti API preko katerega bomo pridobivali podatke.

5.1.2 Druga stopnja: Upravljanje z infrastrukturo

V drugem razvojnem koraku, bi želeli uporabnikom omogočiti enostavno upravljanje z infrastrukturo. Uporabnik bo imel možnost direktne konzolne povezave na strežnik. Prijavne podatke bo potrebno pridobiti direktno iz vCentra. Uporabnik bo lahko strežnike zagnal, ugasnil ali ponovno zagnal. Uporabnik količine zakupljenih podatkov ne bo mogel spreminjati, saj je to še vedno vezano na pogodbo.

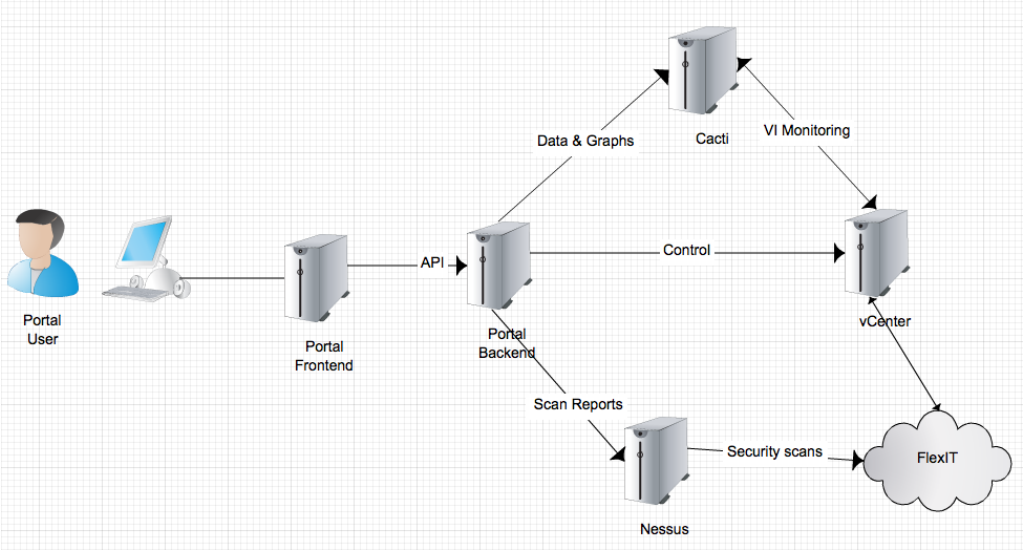
5.1.3 Tretja stopnja: Zaračunavanje porabe

V tretjem razvojnem koraku bi želeli implementirati avtomatično zaračunavanje glede na porabo virov. Podatke o porabi bi pridobili iz vCentra. Na tej stopnji bi uporabnikom dovolili spreminjati količino zakupljenih virov ter dodajati nove strežnike. Informacije o strežnikih bi shranjevali v lokalno podatkovno bazo. Parametre delovanja pa bi še vedno pridobivali iz orodja Cacti.

5.2 Zasnova

Uporabniški portal bo združeval nekaj že obstoječih orodij, ki jih bo potrebno razširiti, da bodo ustrezale našim zahtevam. Temelj uporabniškega portala bo že obstoječi FlipIT portal. Ta je napisan v programskem ogrodju .NET in programskih jezikih C# in Javascript. Obstoječi portal je izdelan na način ASP.NET Web Forms in je trenutno v procesu prehoda na MVC način. Prehod na MVC način je časovno zelo zamuden in še ne bo kmalu zaključen, zato sem moral svoj del portala zasnovati na način, ki ne bo odvisen od ostalih delov strani. Zaradi tega sem se odločil uporabiti ogrodje AngularJS, ki bo poskrbelo za prikaz in logiko spletne strani na strani uporabnika. Podatke pa bom uporabniku nudil preko APIja. Od obstoječih funkcij portala bom uporabil samo pregled uporabniških pravic.
Za orodje Cacti je potrebno implementirati API preko katerega bodo na voljo

različne informacije o VM ter grafi porabe virov. Varnostni pregledovalnik Nessus šele v najnovejši verziji podpira komunikacijo preko RESTful (angl. Representational State Transfer) APIja, zato smo v podjetju razvili poenostavljeno verzijo APIja. Kontrolo virtualnih naprav pa bomo izvajali preko vCentra.



Slika 5.1: Prikaz povezanosti aplikacije.

Uporabnik bo s pomočjo AngularJS delal zahtevke na API, ki ga bom napisal v zaledju portala. Ta bo izvajal preverjanje, če je uporabnik prijavljen na portal in če ima pravico dostopa do podatkov. Nato bo zaledje zahtevalo podatke iz ostalih APIjev. Uporabnik tako, za primer, ne bo mogel dostopati direktno do orodja Cacti. Na tak način bomo preprečili nepooblaščen dostop do virov drugih uporabnikov.

5.2.1 Podatkovna baza

Za potrebe uporabniškega portala je bilo potrebno v obstoječo podatkovno bazo Microsoft SQL vstaviti nekaj nastavitevnih podatkov in ustvariti nekaj dodatnih podatkovnih tabel. Za vsako stranko v vCentru ustvarimo

ločen bazen virov. Bazene virov med seboj ločimo s pomočjo unikatnega identifikatorja, ki ga določi vCenter. Preko tega identifikatorja lahko izvajamo poizvedbe v orodju Cacti. Ustvariti je potrebno tabelo, ki bo vsebovala identifikator in ga povezovala s stranko. Za potrebe povezovanja z napravami preko VMRC konzole je v podatkovni bazi potrebno hraniti uporabniško ime in geslo. Zato je potrebno ustvariti novo tabelo, ki hrani ta dva podatka in ju povezuje z podjetjem.

Zaradi potreb beleženja aktivnosti je potrebno ustvariti novo tabelo, ki bo hranila podatke v zvezi z upravljanjem infrastrukture. V tabelo bi želeli shraniti tip dogodka, id uporabnika, id podjetja, čas dogodka in na katere naprave je vplivalo. V tretji stopnji razvoja bo v podatkovni bazi potrebno hraniti tudi vse podatke o virtualnih napravah. Zato je potrebno ustvariti podatkovno tabelo v kateri bomo hranili te podatke.

5.3 Implementacija

Pri implementaciji sem začel z razvojem vmesnika API za orodje Cacti. Nadaljeval sem z izgradnjo zaledja portala in za konec zgradil še uporabniški vmesnik. Pri gradnji zaledja sem moral implementirat dva različna tipa API-jev. Posebnega za prenos grafov iz orodja Cactija in enega za ostalo komunikacijo. Napisati sem moral tudi funkcije za zagon, ponovni zagon in ustavitev virtualnih naprav.

5.3.1 Vmesnik API za orodje Cacti

Za orodje Cacti imamo razviti vtičnik, kateri se preko skript napisanih v programskem jeziku Perl, vsake 5 minut poveže na vCenter. Iz njega pridobiva podatke o izbranih virtualnih napravah in bazenih virov. Za vsako virtualno napravo pridobljene podatke shrani v bazo MySQL. V orodju Cacti pa se na vsake 5 minut požene "poller", ki s pomočjo skript v programskem jeziku PHP prebere podatke iz te podatkovne baze in podatke shrani v datoteke RRD. Iz datotek RRD pa črpa podatke za risanje grafov.

Za potrebe portala sem razvil programski vmesnik API, ki preko zahtevkov GET sprejema zahteve in podatke vrača v formatu JSON (angl. JavaScript Object Notation). Pri razvoju sem si pomagal z knjižnico GluePHP, ki ponuja enostavno povezovanje med naslovi URL (angl. Uniform Resource Locator) in programskimi razredi. V tabelo je potrebno shraniti naslov URL in ime razreda, ki se mora ob klicu naslova URL izvršiti. Pri primerjanju naslovov lahko uporabimo regularne izraze.

Izvorna koda 5.1: Prikaz vseh povezanih naslovov URL.

```
$base_url = '/api/v1';
$uname = '(?P<username>[\w\-_.@]+)';
$resourcepool = '(?P<rpid>[\w\Q-_\%#.@\E]+)';
$vm_id = '(?P<vmid>[\w\Q-_\%#.@\E]+)';

$urls = array(
        '/' => 'index',
        '/favicon' => 'favicon',
        "/user/$uname/rpool/$resourcepool/list" => 'all_vms_from_rp',
        "/user/$uname/vmid/$vm_id/graphs" => 'all_vm_graph_ids',
        "/user/$uname/vmid/$vm_id/tholds" => 'all_vm_tholds',
        "/user/$uname/vmid/$vm_id/health" => 'vm_health',
        "/user/$uname/graphid/(?P<graphid>\d+)/image" => 'get_graph',
        "/user/$uname/rpool/$resourcepool/vmshealth" => 'all_vm_health_from_rp',
        "/user/$uname/rpool/$resourcepool/vmid/$vm_id" => 'vm_from_rp',
);
```

Naredil sem osnovni razred, ki v konstruktorju vzpostavi povezavo s podatkovno bazo in jo v destruktorju zapre. Osnovni razred ima napisano tudi funkcijo, ki za napisani stavek SQL (angl. Structured Query Language), izvede vse potrebne procedure, izvrši poizvedbo ter vrne rezultate.

Vsi ostali razredi razširjajo ta osnovni razred ter mu razširjajo funkcionalnost. Razredi s pomočjo knjižnice GluePHP[34] iz klicanega naslova URL pridobijo parametre, ki jih uporabijo pri poizvedbah SQL. Vse poizvedbe SQL so napisane s pomočjo vnaprej pripravljenih stavkov (angl. prepared statements) in so tako varne pred napadi SQL-vrinjanja (angl. SQL injection). Rezultate poizvedb vračam v formatu JSON. Za vzpostavitev povezave med naslovi URL, razredi in knjižnico je potrebno poklicati funkcijo *stick*.

Največ težav pri implementaciji vmesnika API sem imel pri posredovanju grafov. Teh podatkov ni mogoče vrniti v formatu JSON ampak samo kot sliko v formatu "*image/png*". Sliko sem moral najprej pridobiti iz orodja Cacti kot datoteko, jo prebrati in pretvoriti v podatkovni tok ter ga vrniti.

Z vmesnikom bo lahko komuniciral samo zaledni sistem portala.

5.3.2 Zaledni sistem uporabniškega portala

Po izgradnji vmesnika API za orodje Cacti sem začel z implementacijo zaledja portala. Najprej sem napisal nov programski vmesnik, ki bo sprejemal zahteve uporabnika portala, izvajal nadaljne poizvedbe in vračal uporabniku primerne rezultate. Za prenos grafov je bilo potrebno uporabiti drugačno vrsto vmesnika.

Podatkovni vmesnik

V zalednem sistemu sem ustvaril nov spletni vmesnik (angl. Web API Controller Class) namenjen izmenjavi podatkov med uporabnikom in zaledjem. Ko uporabnik naredi zahtevo na ta vmesnik, ta preveri ali je uporabnik v portal prijavljen in ali ima pravico videti te vire. Vsaka podana zahteva vsebuje unikatni identifikator podjetja preko katerega iz podatkovne baze pridobim identifikator bazena virov. Ta identifikator se uporablja pri vseh zahtevah na programski vmesnik Cacti. Uporabnik identifikatorja nikoli ne vidi in ga ne more spreminjati, zato ne more pridobiti podatkov o virih, ki ne pripadajo njegovemu podjetju. Glede na tip zahteve se izvršijo določene funkcije. Te funkcije uporabijo povezovalni razred, ki izvaja zahteve na različne vmesnike API in vrača pridobljene podatke.

Povezovalni razred

Povezovalni razred komunicira z vmesniki API in vrača rezultate. Na vmesnike pošilja *GET* zahteve, odgovore pa shrani v objekte. Pri implementaciji tega razreda sem si pomagal z programsko knjižnico RestSharp.[35] RestSharp pošlje zahtevek *GET* na vmesnik, vrnjen rezultat pa pretvori v prej podani objekt. Podatke, ki se sedaj nahajajo v objektih, ustrezno obdelam in vrnem podatkovnemu vmesniku. Povezovalni razred zna komunicirati z orodjem Cacti in varnostnim pregledovalnikom Nessus.

Vmesnik za Cacti grafe

Podatkovni vmesnik zna vračati rezultate samo v formatih XML (angl. Extensible Markup Language) in JSON, zato ni primeren za slike tipa png. To je bil razlog, da je bilo potrebno ustvariti nov generični vmesnik (angl. Generic Handler), ki omogoča tudi prenašanje slik.

Izvorna koda 5.2: Vtičnik za posredovanje grafov iz orodja Cacti.

```
public class CactiGraph : IHttpHandler
    {
        private HttpWebRequest request;
        private HttpWebResponse response;

        public void ProcessRequest(HttpContext context)
        {
            string TYPE_PNG = "image/png";
            string URL, baseURL;
            string contentType;
            string companyHash = context.Request.QueryString["companyHash"];

            if (context.Request.QueryString["graphId"].Length > 0)
            {
                contentType = TYPE_PNG;
                string graphID = string.Format("graphid/{0}/image",
                     context.Request.QueryString["graphId"]);
                URL = string.Format("{0}{1}", baseURL, graphID);
            }
            else
            {
                return;
            }

            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(URL);
            request.Method = "GET";
            request.ContentType = contentType;

            response = (HttpWebResponse)request.GetResponse();

            context.Response.ContentType = contentType;
            Stream stream = response.GetResponseStream();

            //Read the stream to memory.
            byte[] buffer = new byte[4096];

            int byteSeq = stream.Read(buffer, 0, 4096);

            while (byteSeq > 0)
            {
                context.Response.OutputStream.Write(buffer, 0, byteSeq);
                byteSeq = stream.Read(buffer, 0, 4096);
            }
        }

        public bool IsReusable
        {
            get
            {
                return false;
            }
```

```
        }
    }
```

Za izvršitev zahteve potrebujemo identifikacijsko številko grafa. Te lahko dobimo s pomočjo podatkovnega APIja. Vmesnik tako naredi zahtevo na Cacti in prebere dobljeno sliko ter jo posreduje naprej.

Razred za komunikacijo z vCentrom

Programska zbirka vSphere vsebuje razvojno programsko orodje (v nadaljevanju SDK, angl. Software Development Kit), ki nam omogoča izgradnjo aplikacij. SDK ima napisanih veliko razredov in funkcionalnosti, ki jih lahko uporabimo v aplikacijah. S pomočjo SDK sem napisal funkcije za povezovanje na vCenter, zagon VM, ponovni zagon VM, zaustavitev VM in za pridobitev konzolnih prijavnih podatkov.

Izvorna koda 5.3: Funkcija za zagon virtualne naprave.

```
public bool StartVmGuest(VirtualMachine vm)
        {
            try
            {
                vm.UpdateViewData(new[] { "runtime.powerState",
                    "guest.toolsRunningStatus" });
                if (vm.Runtime.PowerState == VirtualMachinePowerState.poweredOn)
                {
                    logger.Info("VM already in powerOn state!");
                    return true;
                }

                vm.PowerOnVM(host: null);
                WaitForVmTools(vm);

                return true;
            }
            catch (VimException ex)
            {
                logger.Error(ex);
                return false;
            }
        }
```

Največ težav mi je povzročalo pridobivanje prijavnih podatkov za vzpostavitev konzolne povezave z virtualno napravo. Za vzpostavitev povezave sta na voljo dva različna načina. Pri prvem načinu je potrebno uporabniku sporočiti uporabniško ime in geslo, ter z njima vzpostaviti povezavo do naprave. Drugi način pa je vzpostavitev povezave s pomočjo enkratne "vsto-

pnice"(angl. ticket). Vstopnica velja za enkratno povezavo v omejenem času od izdaje. Ta rešitev je bolj varna, saj uporabniku ni potrebno posredovati uporabniškega imena in gesla.

Problem se pojavi pri pridobivanju vstopnice. Ta je na voljo preko spletnega uporabniškega vmesnika in preko APIja. Vstopnica, ki jo dobiš preko APIja, je veljavna toliko časa kot je aktivna sejna povezava. Pri normalni zahtevi preko APIja se seja prekine takoj, ko prejmemo podatke, kar pomeni, da vstopnica ni več veljavna. Zato moramo najprej ročno vzpostaviti sejo z APIjem in potem poslati zahtevo za vstopnico. Tako pridobljena vstopnica ima daljši čas obstoja.

Izvorna koda 5.4: Funkcija za pridobitev vstopnice.

```
public string AcquireCloneTicket()
{
        var sessionManager =
            (SessionManager)VimClient.GetView(VimClient.ServiceContent.SessionManager,
            new[] { "message" });
        var ticket = sessionManager.AcquireCloneTicket();

        return ticket;
}
```

5.3.3 Grafični vmesnik

Grafični vmesnik je zgrajen s pomočjo JavaScript MVW ogrodja AngularJS. Ločen je na 3 dele:

- model,
- krmilnik,
- podatki.

Pod model spada datoteka HTML, ki jo uporabnik dobi ob obisku spletne strani. V datoteki se nahaja predloga strani. Predloga vsebuje posebne značke, ki jih prebere AngularJS ter jih nadomesti s podatki.

Izvorna koda 5.5: Koda HTML za izris stanja zasedenosti vseh particij s pomočjo AngularJS.

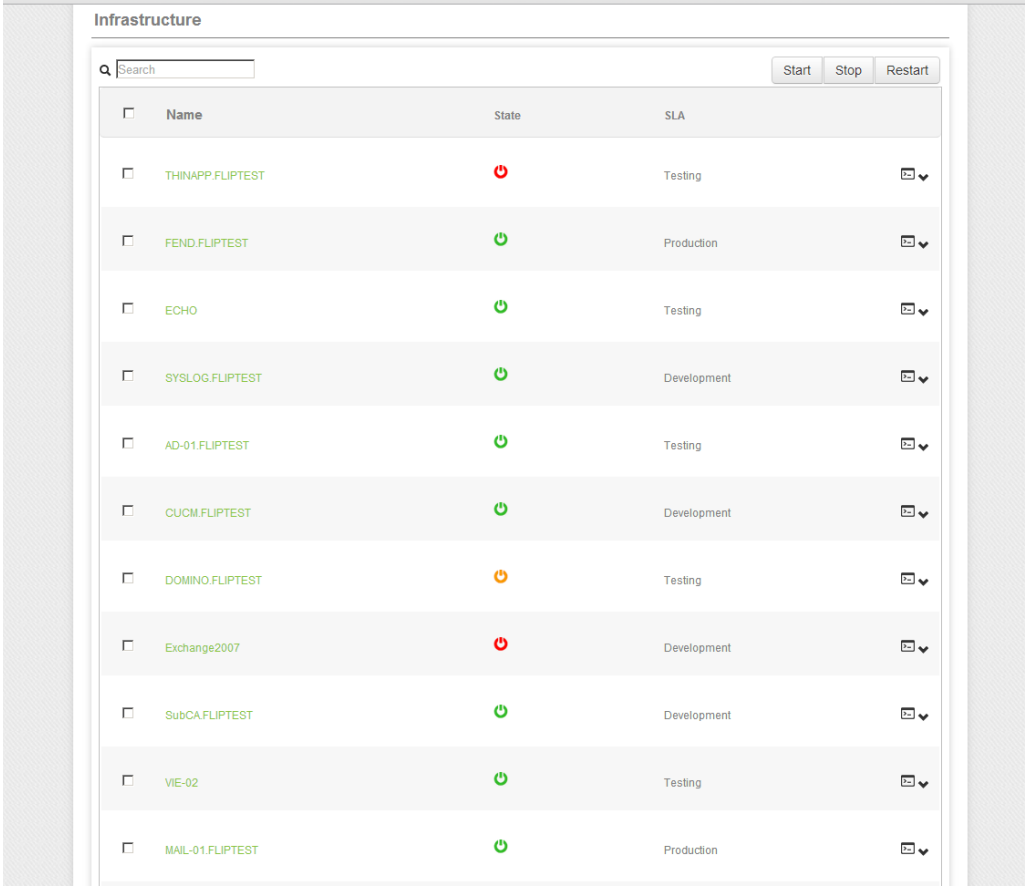
```
<tr data-ng-repeat="partition in vm.disk">
        <td class="image disk labelW">{{partition.PartitionName}}</td>
        <td>{{partition.SizeGB}} GB</td>
        <td style="width: 200px" data-ng-switch on="partition.status">
                <progress data-ng-switch-when="success"
                     data-percent="partition.UsageDisk" class="tb-progress-success"
                     data-animate="false">
                </progress>
                <progress data-ng-switch-when="warning"
                     data-percent="partition.UsageDisk" class="tb-progress-warning"
                     data-animate="false">
                </progress>
                <progress data-ng-switch-when="danger"
                     data-percent="partition.UsageDisk" class="tb-progress-danger"
                     data-animate="false">
                </progress>
        </td>
        <td style="text-align: right; padding-right: 20px">{{partition.UsageDisk}}
            %</td>
        <td>
                <img data-ng-click="showGraph(vm.vCenterId, 'disk', vm.powerState)"
                     class="kpi-icon clickable"
                     src="/skin/default/images/statistics.png" />
        </td>
</tr>
```

Dinamični del strani pa se nahaja v datoteki JavaScript. V našem primeru gre za krmilnik preko katerega upravljamo z našo spletno aplikacijo. V krmilniku je zapisano, kam se mora povezati, da bo dobil potrebne podatke. Tudi datoteke dinamičnega dela se redko spreminjajo, zato jih lahko brskalnik skupaj z datoteko HTML shrani v predpomnilnik in s tem zmanjšuje število prenešenih podatkov.

Zadnji del pa so aplikacijski podatki, ki jih je potrebno pridobiti. Podatkovni del aplikacije se nahaja v prej opisanih zalednih sistemih.

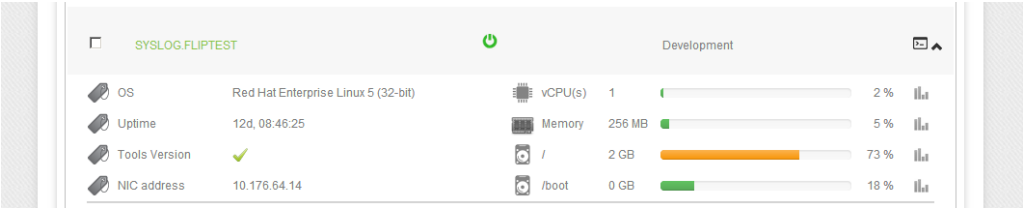
Pri gradnji grafičnega vmesnika sem najprej moral prikazati imena vseh virtualnih naprav ter nekaj njihovih osnovnih lastnosti. Takšne podatke je najlažje prikazati v tabeli. Pri implementaciji pa nisem mogel uporabiti klasičnih HTML elementov, saj ti ne omogočajo enostavnega razširjanja vrstic v slogu harmonike. Zato sem moral tabelo zgraditi s pomočjo HTML elementov *div*.

V tabeli prikazujem ime, stanje in tip SLA virtualne naprave. Nad tabelo je iskalno polje, ki išče po imenu virtualne naprave. Iskalnik pri delovanju strani ne naloži ponovno, ampak takoj prikaže rezultate.



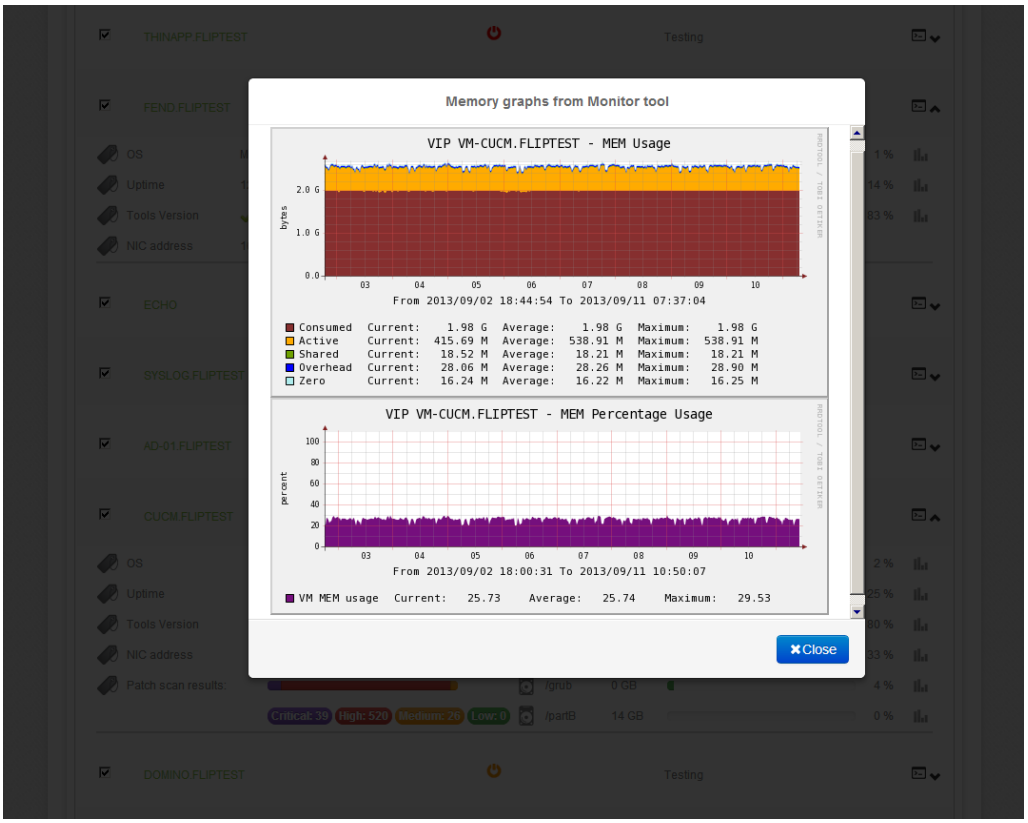
Slika 5.2: Tabela, ki prikazuje pregled infrastrukture.

Ob kliku na vrstico se ta razširi in prikaže več podrobnosti o virtualni napravi. Prikažejo se podatki o tipu operacijskega sistema, času delovanja, verziji VMware orodij, IP naslov ter podatki o trenutni porabi virov.



Slika 5.3: Razširjen tabelarični vnos.

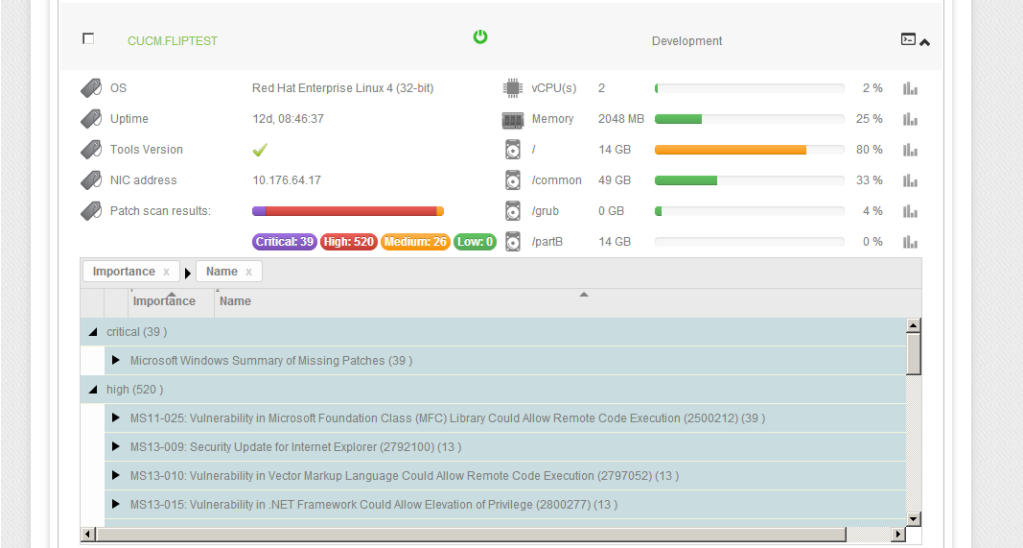
Ob kliku na ikono zraven histograma trenutne porabe virov pa se nam odpre modalno okno v katerem vidimo grafe z zgodovino iz orodja cacti.



Slika 5.4: Prikaz grafov iz orodja Cacti.

Če za napravo izvajamo varnostna skeniranja, potem se ob razširitvi tabele prikažejo tudi rezultati. Ob kliku na naloženi histogram varnostnih opozoril, pa se ti prikažejo v tabeli s krajšimi opisi. Ob kliku se pošlje zahteva v zaledje portala, od tam pa se preko povezovalnega razreda pridobi podatke iz strežnika Nessus. Tabela je zgrajena s pomočjo knjižnice NG-Grid, ki je del paketa AngularUI. Tabela nam omogoča združevanje po več stolpcih.

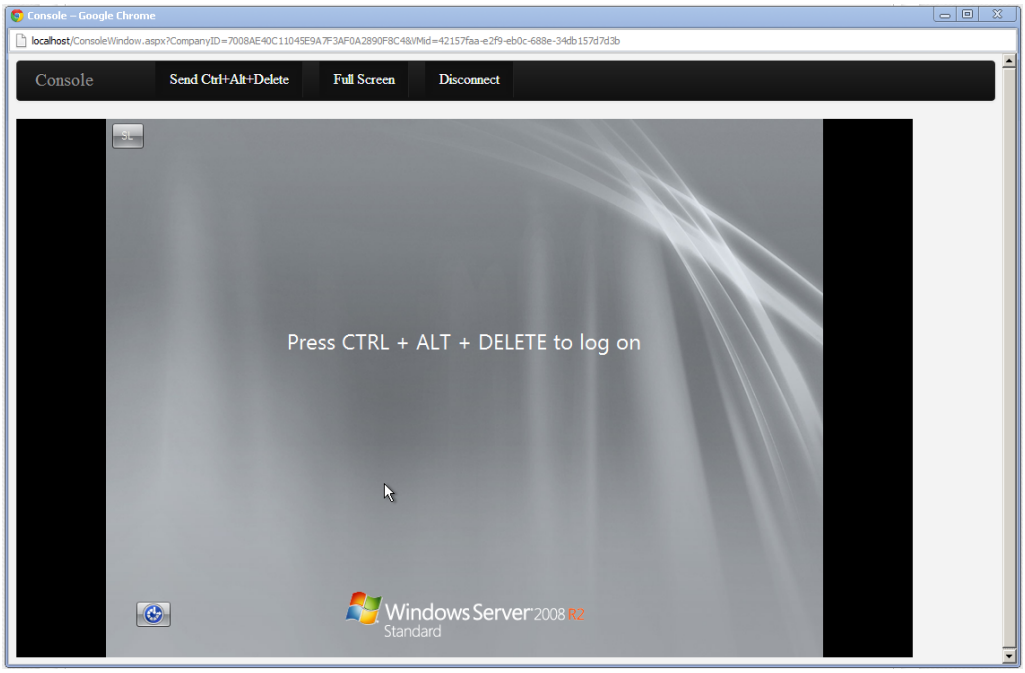
V osnovni vrstici imamo ikono za vzpostavitev konzolne povezave z virtualno napravo. Ob kliku na ikono se nam odpre novo okno, v katerem se vzpostavi konzolna povezava z virtualno napravo. Za delovanje konzole moramo imeti nameščen VMRC brskalniški vtičnik in glavni program. Če



Slika 5.5: Prikaz rezultatov varnostnih pregledov orodja Nessus.

aplikacija zazna, da uporabnik nima nameščenih potrebnih programov in vtičnikov, mu ponudi prenos potrebnih datotek.

Uporabnik lahko označi virtualne naprave in jih preko gumbov, ki se nahajajo nad tabelo zažene, ponovno zažene ali zaustavi.



Slika 5.6: Okno z vzpostavljeno konzolno povezavo na virtualni strežnik.

Poglavje 6

Sklepne ugotovitve

Oblačne storitve so trenutno ena najbolj popularnih tem v svetu računalništva. Številne storitve se selijo v oblak. Veliko podjetij po svetu želi prodreti na trg oblačnih storitev, kjer je trenutno nesporni zmagovalec Amazon z oblakom AWS (Amazon Web Services). Podjetja iščejo svojo zgodbo o uspehu tako, da ponudijo nekaj novega, drugačnega, saj je prodreti s klasičnimi storitvami bistveno težje. Tudi podjetje Nil želi ponuditi nekaj drugačnega, zato je prišlo do odločitve, da bo podjetje za svoje oblačne storitve implementiralo lasten uporabniški portal. Implementacija je bila zaupana meni in se sedaj odraža v tem diplomskem delu. Pred pričetkom implementacije sem pregledal nekaj obstoječih rešitev in jih primerjal z zahtevami podjetja. Ugotovil sem, da vse rešitve ponujajo uporabnikom preveč nastavitev in tako tehnično slabše podkovane uporabnike odvrnejo od uporabe. Izdelan uporabniški portal je enostaven in omogoča dober vpogled v delovanje infrastrukture. Za bolj podrobne nastavitve pa poskrbijo tehnično ustrezno podkovani inženirji.

Med težavami, ki sem jih imel pri razvoju portala bi izpostavil dejstvo, da sem moral med razvojem zavreči precej narejenega dela, ker je prišlo do odločitve, da se večino na spletni strani uporabljenih gradnikov zavrže. Pri projektu sem bil primoran uporabljati gradnike uporabniškega vmesnika podjetja Telerik imenovane RadGrid. Privajanje na uporabo le teh mi je vzelo ogromno časa in napredek je bil temu primeren. Po opustitvi pa sem

sodeloval pri izbiri novih orodij in gradnikov. Tako sem lahko izbral orodja, ki so mi bližje. Motivacija za delo je narasla in napredek pri delu je postal očitnejši.

Z uporabo JavaScript ogrodja AngularJS sem poglobil svoje znanje na področju izdelave spletnih strani in aplikacij, saj ogrodje prinaša nove koncepte pri izdelavi spletnih strani. Za potrebe izdelave diplomske naloge sem se naučil uporabljati programski jezik C# in razvojno ogrodje .NET.

Vsekakor delo na portalu še ni zaključeno, saj ostaja še veliko odprtih vprašanj, ki jih bo potrebno rešiti v prihodnosti. Če bi na projektu kaj spremenil, bi se že na začetku dela odločil za opustitev uporabe gradnikov podjetja Telerik in takoj prešel na uporabo AngularJS, Bootstrap in ostalih tehnologij. Tako bi lahko v istem času razvil več funkcionalnosti. Veseli me, da sem spoznal delo na večjem projektu in s tem pridobil pomembne izkušnje za nadaljno programersko pot.

Literatura

[1] T. Grance, P. Mell. "The NIST Definition of Cloud Computing". Dostopno dne 15.09.2013 na: http://www.cloudbook.net/resources/stories/the-nist-definition-of-cloud-computing

[2] J. Bervar. "Nilov vodnik po oblaku". Dostopno dne 15.09.2013 na: http://www.nil.si/assets3554/wp-content/uploads/2013/04/nil-ov-vodnik-po-oblaku.pdf?5e05f5

[3] B. Kepes. "Understanding the Cloud Computing Stack". Dostopno dne 15.09.2013 na: http://bit.ly/n3iFjT

[4] G. Greenwald, E. MacAskill. "NSA Prism program taps in to user data of Apple, Google and others". Dostopno dne 15.09.2013 na: http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data

[5] Urad informacijskega pooblaščenca RS. "Varstvo osebnih podatkov in računalništvo v oblaku". Dostopno dne 15.09.2013 na: http://bit.ly/1aD6oBZ

[6] V. Winkler. "Cloud Computing: Virtual Cloud Security Concerns". Dostopno dne 15.09.2013 na: http://technet.microsoft.com/en-us/magazine/hh641415.aspx

[7] J. Brodkin. "Kim Dotcom: Megaupload data in Europe wiped out by hosting company". Dostopno dne 15.09.2013 na:

http://arstechnica.com/tech-policy/2013/06/kim-dotcom-megaupload-data-in-europe-wiped-out-by-hosting-company/

[8] "Cloud Computing: Benefits - Better Security". Dostopno dne 15.09.2013 na: http://bit.ly/1egqohP

[9] O. Ingthorsson. "Security Benefits of Cloud Computing". Dostopno dne 15.09.2013 na: http://cloudcomputingtopics.com/2013/03/security-benefits-of-cloud-computing/

[10] Cloud Security Aliance."Security Guidance For Critical Areas of Focus in Cloud Computing v3.0". Poglavje 5.2 str. 52. Dostopno dne 15.09.2013 na: http://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf

[11] VMware. "vCloud Suit spletna stran". Dosegljiva dne 15.09.2013 na: http://www.vmware.com/products/vcloud-suite

[12] OpenStack. "OpenStack spletna stran". Dostopno dne 15.09.2013 na: http://www.openstack.org/software/

[13] CloudStack. "CloudStack spletna stran". Dostopno dne 15.09.2013 na: https://cloudstack.apache.org/

[14] OpenNebula. "OpenNebula spletna stran". Dostopno dne 15.09.2013 na: http://opennebula.org/about:keyfeatures

[15] Statista. "Forecast for global shipments of tablets, laptops and desktop PCs from 2010 to 2017 (in million units)". Dostopno dne 15.09.2013 na: http://www.statista.com/statistics/183419/forecast-of-global-sales-of-pcs-by-category/

[16] Virtual Management Technologies"Virtualization Basics – Back to the Future". Dostopno dne 15.09.2013 na: http://bit.ly/1aMwC8u

[17] National Instruments. "Virtualization Basics". Dostopno dne 15.09.2013 na: http://www.ni.com/white-paper/8708/en/

[18] VMware. "Virtualization Basics". Dosegljivo dne 15.09.2013 na: http://www.vmware.com/virtualization/virtualization-basics/how-virtualization-works.html

[19] K. Scarfone, M. Souppaya, P. Hoffman. "Guide to Security for Full Virtualization Technologies". Dostopno dne 15.09.2013 na: http://csrc.nist.gov/publications/nistpubs/800-125/SP800-125-final.pdf

[20] S. D. Lowe. "vSphere Storage (Part 1) - A VAAI Primer". Dostopno dne 15.09.2013 na: http://www.virtualizationadmin.com/articles-tutorials/vmware-esx-and-vsphere-articles/storage-management/vsphere-storage-part1.html

[21] S. D. Lowe. "Back to basics (Part 3): Virtualization 101: A Storage Primer (Cont.)". Dostopno dne 15.09.2013 na: http://www.virtualizationadmin.com/articles-tutorials/general-virtualization-articles/back-to-basics-part3.html

[22] Cacti. "Cacti Documentation". Dostopno dne 15.09.2013 na: http://bit.ly/19VSAS9

[23] P. Antončič. "Monitoriranje računalniških omrežij", Ljubljana, 2012. Dostopno dne 15.09.2013 na: http://bit.ly/1aMzoKV

[24] "Nessus spletna stran". Dosegljivo dne 15.09.2013 na: http://www.tenable.com/products/nessus/features

[25] "Nessus Wikipedia ". Dostopno dne 15.09.2013 na: http://bit.ly/bcwmDP

[26] N. Župec. "Twitter Bootstap in razvoj spletnega repozitorija za Cacti". Ljubljana, 2012. Dostopno dne 15.09.2013 na: http://bit.ly/18pSh4L

[27] "AngularUI spletna stran". Dostopno dne 15.09.2013 na: http://angular-ui.github.io

[28] "UI-Bootstraps spletna stran". Dostopno dne 15.09.2013 na: http://angular-ui.github.io/bootstrap/

[29] "NG-Grid spletna stran". Dostopno dne 15.09.2013 na: https://github.com/angular-ui/ng-grid

[30] "VMRC Documentation". Dostopno dne 15.09.2013 na: http://bit.ly/16w7RJu

[31] "VMRC Documentation". Dostopno dne 15.09.2013 na: http://bit.ly/1eTqc6P

[32] I. Milnar, 2012. "MVC vs MVVM vs MVP". Dostopno dne 17.09.2013 na: https://plus.google.com/+AngularJS/posts/aZNVhj355G2

[33] "Use AngularJS to Power Your Web Application". Dostopno dne 17.09.2013 na: http://www.yearofmoo.com/2012/08/use-angularjs-to-power-your-web-application.html

[34] "GluePHP - Documentation". Dostopno dne 17.09.2013 na: http://gluephp.com/documentation.html

[35] "Simple REST and HTTP API Client for .NET". Dostopno dne 17.09.2013 na: http://restsharp.org/

[36] "Prevod besede scaling". Dostopno dne 17.09.2013 na: http://www.islovar.org/izpisclanka.asp?id=4130