OULUN AMMATTIKORKEAKOULU

Veli-Matti Jaakola

WINDOWS AZUREN PERUSTEET JA KÄYTTÖ ASIAKAS-PALVELINSOVELLUKSESSA

WINDOWS AZUREN PERUSTEET JA KÄYTTÖ ASIAKAS-PALVELINSOVELLUKSESSA

Veli-Matti Jaakola
Opinnäytetyö
Kevät 2014
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Veli-Matti Jaakola
Opinnäytetyön nimi: Windows Azuren perusteet ja käyttö asiakas-palvelinsovelluksessa
Työn ohjaaja: Tuomo Tikkanen
Työn valmistumislukukausi ja -vuosi: Kevät 2014 Sivumäärä: 52 + 2 liitettä

Tämä opinnäytetyö tehtiin BelleGames Oy:lle. Työssä selvitetään pilvipalveluiden määritelmä, Windows Azuren perusteet ja sen komponenttien käyttö yksinkertaisessa asiakas-palvelinsovelluksessa. Dokumentissa ei mennä pintaa syvemmälle eri teknologioihin, vaan se toimii perusteina sekä ohjeena, miten saadaan ensimmäinen asiakas-palvelinsovellus toimimaan Windows Azure -alustalla.

Dokumentista tehtiin mahdollisimman yksityiskohtainen ja helposti ymmärrettävä, jotta se toimisi kaikille lukijoille pohjana pilvipalveluihin ja Windows Azuren käyttöön. Asiakas-palvelinsovellus, joka tehtiin tätä dokumentaatiota varten ja jonka lähdekoodi löytyy liitteistä, pidettiin mahdollisimman yksinkertaisena. Sovelluksen rakenne ja toiminnot eivät ole tämän opinnäytetyön tärkein aihe, vaan sen toimivuus Windows Azuren kanssa.

Opinnäytetyöstä valmistui dokumentaatio, josta selviää, mitä pilvipalvelut ja Windows Azure ovat ja mitä niillä voidaan tehdä. Työssä esitetty asiakas-palvelinsovellus on toimiva esimerkki, mitä Windows Azuren avulla voi rakentaa yksinkertaisimmillaan. Tuloksena on kattava teoriatietopaketti sekä yksityiskohtainen ohje, miten saadaan ensimmäinen sovellus Windows Azure -alustalle.

Asiasanat:
Windows Azure, client, server, pilvipalvelut, asiakassovellus, palvelin

ALKULAUSE

Haluan kiittää yhteistyöstä BelleGames Oy:tä ja sen toimitusjohtajaa Pirjo Ritokangas-Huttusta, joka antoi tilaisuuden mielenkiintoiseen opinnäytetyöhön pilvipalveluihin liittyen. Haluan myös kiittää työpariani Tanja Henttusta hyvästä yhteistyöstä opinnäytetyön aikana.

Kiitokset myös opettajille Tuomo Tikkaselle, joka toimi ohjaavana opettajana opinnäytetyön ajan, ja Tuula Hopeavuorelle, joka auttoi opinnäytetyötekstin hiomisessa.

Oulussa 3.2.2014

Veli-Matti Jaakola

SISÄLLYS

SANASTO

.NET	Microsoft kehittämä ohjelmistokomponenttikirjasto, joka tukee n. 20:tä ohjelmointikieltä, joista käytetyimpiä ovat C# ja VB.Net.
Asiakasohjelma	Asiakasohjelma (engl. client) on käyttöliittymäsovellus, jolla otetaan yleensä verkon yli yhteys palvelimeen ja käytetään etäkäyttöisesti palvelimella olevia palveluita.
Blob	Binary Large Object on yleisesti käytetty termi tietokannoissa, kun suuri määrä tietoa, esimerkiksi kuva, tallennetaan binääridatana tietokantaan. Pilvipalvelujen storaget käyttävät myös blobia tiedon tallentamiseen.
Container	Windows Azure Storagen nimitys kansiolle, jonka sisälle voi laittaa blobeja.
CPU	Central Processing Unit eli suoritin on tietokoneen osa, joka suorittaa tietokonesovelluksen sisältämiä konekielisiä käskyjä.
Infrastructure as a Service	IaaS on pilvipalveluiden palvelumalli, joka antaa tietokoneen laskentatehoa ja tallennustilaa.
Infrastruktuuri	Pohjana oleva rakenne, johon laajempi ja moninaisempi toiminta tukeutuu.

Internet Information Services	IIS on Microsoftin kehittämä palvelinohjelmistokokonaisuus, joka on tarkoitettu käytettäväksi Windows-pohjaisissa palvelimissa.
Java	Ohjelmointikieli, joka on laitteistoriippumaton sekä oliopohjainen.
Laskentateho	Tietokoneen kyky suorittaa laskutoimituksia aikayksikössä.
Lähiverkko	Englanniksi local area network eli LAN on rajoitetulla maantieteellisellä alueella toimiva tietoliikenneverkko. Esimerkiksi yhden talon koneiden muodostama tietokoneverkko.
NIST	National Institute of Standards and Technology on yhdysvaltalainen kauppaministeriön alainen virasto, jonka tehtävänä on kehittää ja edistää mittaustekniikoita, standardeja ja tekniikkaa.
Ohjelmistokehys	Ohjelmistokehys (engl. framework) muodostaa rungon sen päälle rakennettavalle tietokoneohjelmalle. Se on apuväline, jonka tarkoituksena on nopeuttaa uusien ohjelmistotuotteiden valmistusta.
Palvelin	Palvelimella (engl. server) tarkoitetaan tietoliikenteen yhteydessä tietokoneessa suoritettavaa palvelinohjelmistoa tai ohjelmistoa suorittavaa tietokonetta.

Palvelinkeskus	Palvelinkeskus on huone tai rakennus, jossa voi olla suurimmillaan satoja tuhansia palvelimia.
Palvelutasosopimus	Englanniksi service-level agreement eli SLA on palvelun laadun takaava sopimus.
Platform as a Service	PaaS on pilvipalveluiden palvelumalli, joka antaa laskentatehoa kehitysalustan kautta.
Resurssien yhdistäminen	Palveluntarjoajan tietojenkäsittelyresursse- ja yhdistetään palvelemaan useita asiakkaita, jossa erilaiset fyysiset ja virtu- aaliset resurssit jaetaan dynaamisesti asiakkaille tarpeiden mukaan. Resurssien yhdistäminen on englanniksi resource poo- ling.
Service oriented architecture	SOA eli palvelukeskeinen arkkitehtuuri on ohjelmistotekniikassa käytetty arkkitehtuuritason suunnittelutapa, jolla eri tietojärjestelmien toiminnot ja prosessit on suunniteltu toimimaan itsenäisinä, avoimina ja joustavina palveluina.
Software as a Service	SaaS on pilvipalveluiden palvelumalli, joka antaa käyttäjälle kolmannen osapuolen so- velluksia, joihin päästään käsiksi internetin avulla.
Structured Query Language	SQL eli standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä
Synkroituminen pilvessä	Palveluntarjoajan avulla synkronoidaan tiedostoja kaikkien tietokoneiden ja

	kannettavien laitteiden kanssa, jotka käyttäjä omistaa.
Tietokanta	Tietotekniikassa käytetty termi tietovarastolle (engl. database).
Virtuaalikone	Ohjelmallisesti toteutettu tietokone, jossa voidaan ajaa sovelluksia kuten oikeassa tietokoneessa.
Virtual private network	VPN eli virtuaalinen erillisverkko on tapa, jolla kaksi tai useampaa yksityistä verkko voidaan yhdistää julkisen verkon yli muodostamaan näennäisesti yhden yksityisen verkon. Voi tarkoittaa myös yksittäisten etätyöasemien liittämistä yksityiseen verkkoon.
Web role instance	Windows Azure Cloud Services virtuaalikoneesta luotu olio, joka antaa IIS:n, jota käytetään web-sovellusten käyttöliittymissä.
Wide area network	WAN eli laajaverkko on tiedonsiirtoverkko, joka kattaa laajan alueen käyttäen yksityisiä tai julkisia verkkoja kuljetukseen. Se yhdistää lähiverkot sekä kaupunkiverkot yhdeksi suureksi verkoksi.
Worker role instance	Windows Azure Cloud Services -virtuaalikoneesta luotu olio, joka tekee asynkronisia, pitkäkestoisia tai ikuisia tehtäviä, jotka eivät ole riippuvaisia käyttäjästä

1 JOHDANTO

Opinnäytetyöni tehtävänä oli jatkaa aiempien työntekijöiden tekemää peliprojektia nimeltä Year Of Eden ja saada se julkaistavaksi Facebook-sivustolla. Peli oli toteutettu luomalla Flash-pohjainen asiakasohjelma, joka kommunikoi Windows Azuressa olevan palvelinohjelman kanssa. Pelistä oli olemassa toimiva versio kehitysympäristössä, mutta siitä puuttui tärkeitä ominaisuuksia sekä sen toiminen Facebook-sivustolla vaati työtä. Roolini oli selvittää ja työstää Windows Azuren osuutta.

Projektin parissa ei enää työskennellyt henkilöitä, jotka olivat sitä aikaisemmin tehneet, ja dokumentaatio oli hajanaista ja puutteellista monilta osin. Tästä huolimatta jatkokehitimme yhdessä toisen opinnäytetyötä tekevän henkilön kanssa Year of Edenistä toimivan version Facebookiin, jatkokehitimme peliin uusia ominaisuuksia ja parantelimme peliin liittyvää dokumentaatiota.

Päätimme työnantajan kanssa, että teen opinnäytetyöni Windows Azuren perusteista ja ohjeista yksinkertaiselle esimerkkisovellukselle, joka käyttää hyödyksi Windows Azuren komponentteja. En käyttänyt Year of Eden -peliä esimerkkinä, koska siinä oli paljon ominaisuuksia, joista en ollut vastuussa, sekä siinä on myös paljon ylimääräistä, mikä hankaloittaisi perusteiden selittämistä.

Pilvipalvelut ovat viime aikoina nousseet muotisanaksi, mutta harva ymmärtää, mitä se oikeasti tarkoittaa ja mihin se soveltuu. Tämän dokumentin tarkoitus on koota tilaajan käyttöön pilvipalveluiden määritelmä, Windows Azuren perusteet ja esimerkki, miten Windows Azurea voi käyttää hyödyksi Javalla toteutetussa palvelinsovelluksessa.

2 PILVIPALVELUT

Yksinkertaisimmillaan pilvipalvelut voivat tarkoittaa tiedon tallentamista, siihen käsiksi pääsemistä ja ohjelmien käyttämistä internetin kautta oman kovalevyn sijaan. Pilvipalveluiksi ei lasketa tiedon tallentamista kodin tai toimiston sisäisessä internet-verkossa. Jotta voidaan puhua pilvipalveluista, siihen pitää päästä käsiksi ainoastaan internetin välityksellä tai tiedon pitää vähintään synkronoitua internetin ylitse. Pilvipalveluita voidaan käyttää missä ja milloin vain. (1.)

Pilvipalvelut ovat uusi ja nopeasti kehittyvä teknologia. Kun joku on keksinyt hyvän määritelmän niille, ovat ne jo kehittyneet nykyisestä määritelmästä uuteen muotoon. Monien mielestä National Institute Of Standards and Technologyn (NIST) tekemä määritelmä pilvipalveluista on virallisin, mutta myös sekin on muuttunut ajan myötä. Muutoksista huolimatta monet käyttävät sitä vielä perustana, kun määrittelevät pilvipalveluita. Käytän sitä apunani kertoessani niiden pääpiirteitä. (2, luku Chapter I: What is the Cloud?.)

2.1 Keskeiset piirteet

Monet yritykset ovat yrittäneet rahastaa pilvipalveluiden suosiolla väittämällä tarjoavansa niitä, vaikka näin ei ole. Pilvipalveluksi ei riitä, että sovellus on käytettävissä internetin kautta. Pilvipalveluiden pitää osoittaa tietynlaisia piirteitä, ennen kuin niitä voidaan kutsua oikeiksi pilvipalveluiksi. NIST:n määritelmän mukaan pilvipalveluihin kuuluvat viisi keskeistä tunnistettavaa piirrettä: itsepalvelu, laaja pääsy verkkoon, resurssien yhdistäminen, nopea elastisuus ja mitattava palvelu. (2, luku Chapter I: What is the Cloud?.)

2.1.1 Itsepalvelu

Käyttäjä voi pyytää ja saada pääsyn palvelun tarjontaan ilman, että palvelun ylläpitäjä tai tukihenkilöt täyttävät näitä pyyntöjä manuaalisesti. Pyyntöjen prosessointi ja täyttäminen on näin täysin automatisoitua. Käyttäjän itsepalvelu tekee palveluihin käsiksi pääsemisestä nopea. Tätä kutsutaan on-demand self-

serviceksi eli itsepalveluksi tarpeen vaatiessa. (2, luku Chapter I: What is the Cloud?.)

2.1.2 Laaja pääsy verkkoon

Pilvipalveluiden tulisi olla helposti käytettävissä. Palveluiden ja sovellusten käyttämiseen tarvitaan ainoastaan tavallinen internet-yhteys. Vaikka internet-yhteyksien nopeudet kasvavat jatkuvasti, ovat ne edelleen suhteellisen hitaita verrattuna lähiverkkoon (LAN). Siksi palveluntarjoajien ei tulisi edellyttää käyttäjiltä nopeaa internet-yhteyttä, jotta palvelut toimisivat. (2, luku Chapter I: What is the Cloud?.)

Raskaan asiakasohjelman lataaminen voi kestää hitailla internet-yhteyksillä todella kauan ja palvelun kanssa paljon kommunikoiva asiakasohjelma voi aiheuttaa käyttäjälle viivettä palvelussa. Tämän vuoksi pilvipalvelujentarjoajien tulisi käyttää enimmillään kevyttä asiakasohjelmaa, jonka avulla käyttäjät pääsevät palveluun käsiksi. (2, luku Chapter I: What is the Cloud?.)

2.1.3 Resurssien yhdistäminen

Resurssien yhdistäminen (engl. resource pooling) auttaa säästämään kuluissa ja luo joustavuutta palveluntarjoajalle. Resurssien yhdistäminen perustuu siihen, ettei asiakas tarvitse jatkuvasti kaikkia resursseja, joita hänelle on tarjolla. Kun resurssit eivät ole käytössä asiakkaalla, tyhjäkäynnin sijaan ne voidaan antaa käyttöön toiselle asiakkaalle. Tämä antaa palveluntarjoajille mahdollisuuden palvella paljon enemmän asiakkaita, kuin että jokaisella asiakkaalla olisivat omat varatut resurssit. Resurssien yhdistäminen tapahtuu usein virtuaalisoinnin avulla. (2, luku Chapter I: What is the Cloud?.)

2.1.4 Nopea elastisuus

Nopea elastisuus (engl. rapid elasticity) tarkoittaa kykyä muuntaa palvelun resurssit nopeasti täyttämään käyttäjän tarpeita. Pilvipalveluiden tarjoajilla on jo laitteistoinfrastruktuurit valmiina käyttäjien tarpeiden täyttämiseen, mutta vaikka ne ovat valmiina, niitä ei käytetä, ennen kuin niille on tarvetta. (2, luku Chapter I: What is the Cloud?.)

Nopean elastisuuden avulla pystytään käsittelemään käyttäjämäärässä ilmeneviä lyhytkestoisia piikkejä. Esimerkiksi nettisivulla kävijämäärä on keskiarvoltaan sama pitkin viikkoa, mutta viikonloppuisin tulee tiettyinä aikoina iso kävijämäärä piikki. (2, luku Chapter I: What is the Cloud?.)

2.1.5 Mitattava palvelu

Pilvipalveluilla pitää olla kyky mitata käytön määrää, kuten käytettyä aikaa tai siirretyn datan määrää. Pilvipalveluiden tarjoajat suosivat pay as you go -laskutusta, jossa käyttäjä maksaa vain käytetyistä resursseista. Tästä johtuen käyttäjän ei tarvitse tehdä pidempiaikaista sitoutumista pilvipalveluihin. (2, luku Chapter I: What is the Cloud?.)

2.2 Käyttöönottomallit

Käyttöönottomalli määrittää, missä fyysiset palvelinlaitteet sijaitsevat ja kuka niitä ylläpitää. Käyttäjällä on omat tavoitteensa, mitä hän haluaa pilvipalveluilta, miten hän haluaa niitä käyttää ja kuinka paljon hallittavuutta pilvipalveluiden tulee antaa. Pilvipalveluissa on erilaisia käyttöönottomalleja eri tarpeiden mukaisesti, joista neljä yleisintä ovat

- julkinen pilvi
- yksityinen pilvi
- yhteisöllinen pilvi
- hybridipilvi. (2, luku Chapter I: What is the Cloud?.)

2.2.1 Julkinen pilvi

Julkisessa pilvessä (engl. public cloud) kaikki fyysiset laitteet ovat palveluntarjoajan tiloissa. Palveluntarjoaja on vastuussa laitteiston ylläpidosta ja huoltamisesta, ja käyttäjä on vastuussa vain ohjelmistoista tai asiakasohjelmista, jotka ovat asennettuna loppukäyttäjän laitteistoihin. Yhteydenotto julkiseen pilveen tapahtuu internetin välityksellä. (2, luku Chapter I: What is the Cloud?.)

2.2.2 Yksityinen pilvi

Yksityisessä pilvessä (engl. private cloud) kaikki fyysiset järjestelmät ja resurssit ovat käyttäjän omissa tiloissa. Hän on vastuussa laitteiston ylläpidosta ja huollosta sekä ohjelmistoista tai asiakasohjelmista, jotka ovat asennettuina loppukäyttäjän laitteistoihin. Yksityiseen pilveen otetaan yhteys yleensä lähiverkon (engl. local area network) tai laajaverkon (engl. wide area network) kautta. Etäkäyttäjät pääsevät käsiksi palveluun internetin kautta tai virtuaalisen erillisverkon (engl. virtual private network) kautta. (2, luku Chapter I: What is the Cloud?.)

2.2.3 Yhteisöllinen pilvi

Yhteisöllisessä pilvessä (engl. community cloud) on puolijulkinen pilvimuoto, joka on jaettu käyttäjäyhteisön kesken. Sillä on yhteinen tehtävä tai tarkoitus. Yhteisö ei halua käyttää julkista pilveä, joka on avoin kaikille, vaan he haluavat enemmän yksityisyyttä tiedoilleen, kuin mitä julkinen pilvi antaa. Lisäksi yhteisön jäsenet eivät halua yksin vastuuta pilvipalveluiden ylläpitämisestä, vaan he haluavat jakaa vastuun toisten kanssa. (2, luku Chapter I: What is the Cloud?.)

2.2.4 Hybridipilvi

Hybridipilvi (engl. hybrid cloud) on yhdistelmä kahdesta tai useammasta mallista. Pilvipalveluiden käyttöönottomalleja ei ole sekoitettu yhdeksi, vaan ne ovat kaikki erillään, mutta yhdistettyinä toisiinsa. Hybridipilvi voi tehdä ympäristöstä monimutkaisemman, mutta se tuo myös enemmän joustavuutta pilvipalveluihin. (2, luku Chapter I: What is the Cloud?.)

2.3 Palvelumallit: SaaS, PaaS ja IaaS

Pilvipalveluiden palvelumallit (engl. service models) määrittelevät, millaista palvelua tarjotaan käyttäjille. Yleisimmät palvelumallit ovat

- Software as a Service (SaaS)
- Platform as a Service (PaaS)

- Infrastructure as a Service (IaaS). (3.)

2.3.1 Software as a Service

Software as a Service (SaaS) on eniten käytetty ja yksinkertaisin kolmesta palvelumallista. Se antaa käyttäjälle kolmannen osapuolen sovelluksia, joihin päästään käsiksi asiakasohjelman avulla. Suurin osa SaaS-sovelluksista voidaan suorittaa suoraan internetselaimessa ilman latauksia tai asennuksia. (3.)

SaaS poistaa tarpeen asentaa ja suorittaa sovelluksia yksittäisillä tietokoneilla. Sen avulla yritykset voivat suoraviivaistaa ylläpidon ja teknisen tuen, koska palveluntarjoaja huolehtii kaikesta, kuten sovellus, suoritusaika ja tiedon tallennus. Esimerkiksi sovellukset Microsoft Office 365 ja Gmail ovat SaaS-sovelluksia. (3.)

2.3.2 Platform as a Service

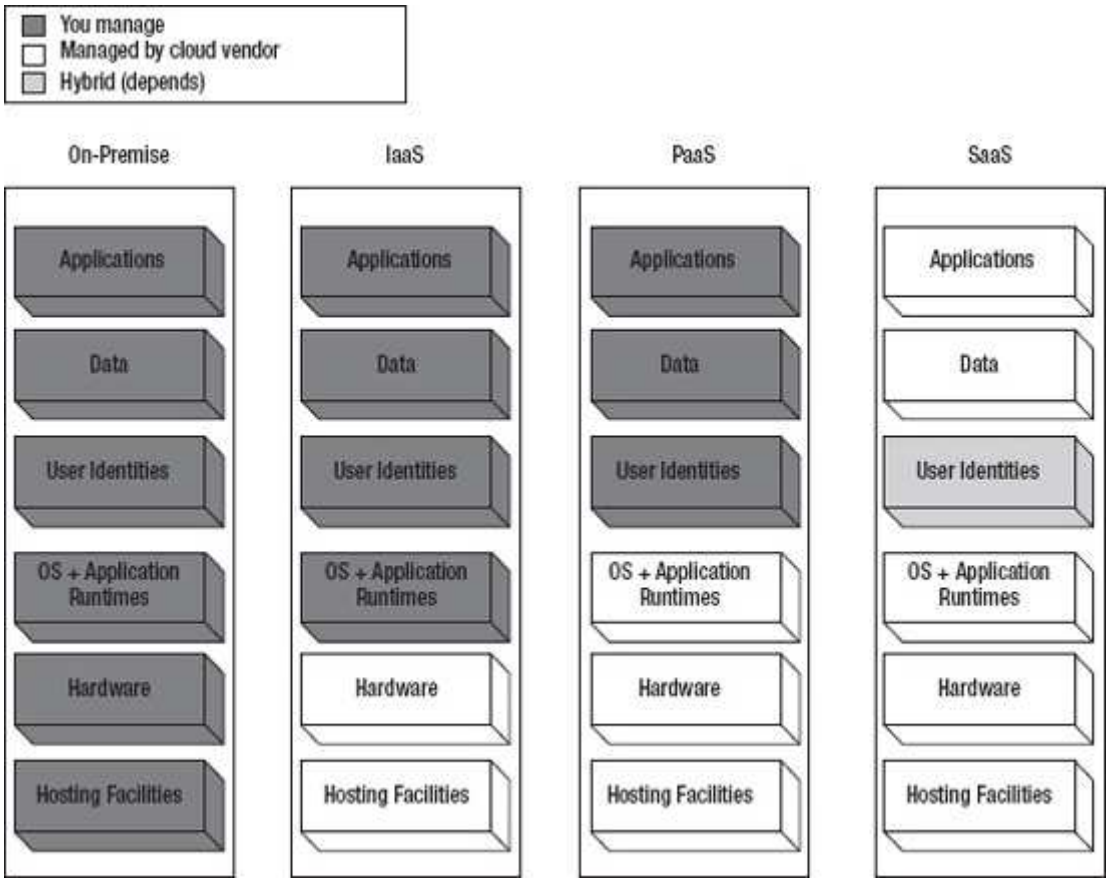
Platform as a Service (PaaS) antaa laskentatehoa kehitysalustan kautta. Sovelluksien kehittäjät saavat PaaSista käyttöönsä ohjelmistokehyksen, johon voidaan kehittää sovelluksia. PaaS tekee sovelluksien kehityksestä, testauksesta ja julkaisemisesta kustannustehokasta ja poistaa tarpeen ostaa tarvittavia palvelinlaitteistoja ja ohjelmistoja. (3.)

Ero SaaSin ja PaaSin välillä on siinä, mitä asioita käyttäjä joutuu ylläpitämään. PaaSissa palveluntarjoaja ylläpitää ajoaikaa, käyttöjärjestelmää, virtualisointia, palvelimia, tiedon tallennustilaa ja verkkoyhteyksiä, mutta käyttäjän ylläpitää sovellusta ja tietoa. PaaS on skaalattavissa ja käyttäjien ei tarvitse huolehtia alustan päivittämisestä ja palvelun katkeamisesta huoltotoimenpiteiden aikana. (3.)

2.3.3 Infrastructure as a Service

Infrastructure as a Service (IaaS) antaa tietokoneinfrastruktuurin, kuten alustan virtualisoinnin, tiedon tallennustilan ja verkkoyhteydet. Käytännössä se tarkoittaa palvelimien ja palvelinsalien ulkoistamista eli palveluntarjoaja antaa käyttäjän asentaa virtuaalipalvelimet heidän laitteistoihinsa. (3.)

Muihin palvelumalleihin verrattuna käyttäjällä on paljon enemmän vastuuta ylläpitämisessä, mutta myös enemmän mahdollisuuksia muokata palvelua. Käyttäjän pitää ylläpitää IaaS-palvelussa sovellusta, tietoa, ajoaikaa ja käyttöjärjestelmää. Mahdolliset päivitykset ja niiden käyttöönotto ovat käyttäjän vastuulla. (3.) Kuvasta 1 nähdään, mitkä asiat ovat käyttäjän vastuulla ja mitkä palveluntarjoajan eri palvelumalleissa.



KUVA 1. Yhteenveto hallittavista komponenteista eri palveluissa (4)

2.4 Pilvipalveluiden hyvät puolet

Pilvipalvelut antavat kehittäjille monia uusia mahdollisuuksia. Ennen pilvipalveluita yritysten piti käyttää paljon rahaa omiin laitteistoihin ja henkilöstön koulutukseen ennen sovelluskehittämisen aloittamista. Pilvipalveluita käytettäessä budjetti ei ole läheskään yhtä suuri kuin omien laitteiden hankinta. (2, luku Chapter I: Cloud Drivers.)

2.4.1 Laitteistoarkkitehtuuri

Käyttäjä voi haluta järjestelmäarkkitehtuurin, jota hän ei saa aikaiseksi nykyisellä laitteistollaan, hänellä ei ole osaamista tai rahoitusta luoda tarvittavia pilvipalveluympäristön ominaisuuksia itse. Haluttuja pilvipalveluiden ominaisuuksia ovat muun muassa ketteryys, luotettavuus, skaalautuvuus, suorituskyky ja helppo ylläpidettävyys. (2, luku Chapter I: Cloud Drivers.)

Ketteryys

Pilvipalvelut nopeuttavat sovelluskehitystä. Resursseja voidaan helposti järjestellä uudelleen tarpeiden mukaan, eli voidaan lisätä resursseja järjestelmiin, jotka tarvitsevat niitä, ja ottaa pois resursseja järjestelmiltä, jotka eivät tarvitse niitä. (2, luku Chapter I: Cloud Drivers.)

Luotettavuus

Luotettavan pilvialustan rakentaminen on todella kallista, koska se vaatii tarvittavan laitteiston lisäksi myös varalaitteistoja ja mahdollisesti useamman palvelinkeskuksen (engl. data center). Lisäksi pitää varautua katastrofien, kuten tulipalon varalta, joiden seurauksena voidaan menettää paljon arvokasta dataa. Monilla pilvipalveluiden tarjoajilla on valmiina useita palvelinkeskuksia eri sijainneissa, joten niitä käyttämällä pilvialustaan saadaan luotettavuutta. Tiedon tallentamista moneen paikkaan pitää mahdollisesti pyytää erikseen pilvipalveluiden tarjoajalta tai se pitää valita erikseen asetuksista. (2, luku Chapter I: Cloud Drivers.)

Skaalautuvuus

Pilvialusta pystyy automaattisesti skaalautumaan käyttäjän tarpeisiin. Uusia resursseja voidaan dynaamisesti lisätä tarpeen vaatiessa. Dynaaminen lisäys tarkoittaa sitä, ettei kaikkien resurssien tarvitse olla varattuina käyttäjälle. Tämä poistaa resursseista tarpeettoman tyhjäkäynnin, mikä säästää laitteistojen sähkö- ja jäähdytyskuluissa. (2, luku Chapter I: Cloud Drivers.)

Suorituskyky

Pilvipalveluiden suorituskykyä mitataan ja monitoroidaan jatkuvasti. Jos suorituskyky putoaa minimitason alle, järjestelmä pystyy automaattisesti lisäämään resursseja tarvittaessa. (2, luku Chapter I: Cloud Drivers.)

Palvelutasosopimus (engl. service-level agreement) takaa käyttäjille, että pilvipalvelut toimivat määritetyllä tasolla. Jos tasoon ei ylletä, palveluntarjoajan tulee myöntyä korvauksiin käyttäjille. Korvaus on yleensä takaisinmaksu kyseiseltä ajalta, jolloin tasoon ei ylletty, tai palvelu hintojen alentaminen. (2, luku Chapter I: Cloud Drivers.)

Helppokäyttöisyys

Pilvipalveluiden helppokäyttöisyys on merkittävä etu. Palveluntarjoaja, joka hallinnoi järjestelmää ja laitteistoa, on myös vastuussa niiden huoltamisesta ja päivittämisestä. Käyttäjän ei tarvitse pysyä ajan tasalla uusimmista laitteista ja niiden ohjelmistopäivityksistä, eikä hänen tarvitse huolehtia palvelun seisahduksesta huoltokatkon ajaksi. (2, luku Chapter I: Cloud Drivers.)

2.4.2 Pilvipalveluiden tietoturvallisuus

Turvallisuus on olennaisin osa palvelun käyttöä, sillä ilman sitä asiakkaan tieto on helposti varastettavissa. Monien ammattilaisten mielestä pilvipalveluiden tarjoajien ympäristöt ovat turvallisempia kuin heidän omat laitteensa.

Henkilöt, jotka ylläpitävät pilvipalveluja, eivät ole vain osa-aikaisesti ylläpitämässä järjestelmää, kuten on monissa yrityksissä tapana, vaan he voivat keskittyä pelkästään turvaamaan tietyntyyppistä ympäristöä tai tietotyyppiä. Keskittyminen antaa heille aikaa ja resursseja kehittää ratkaisuja turvamenetelmiin. Pilvipalveluiden tarjoajille on myös kannattavampaa laittaa enemmän rahaa ongelmien ratkaisemiseen, koska he ratkaisevat ongelman kaikille käyttäjille eivätkä vain yhdelle yritykselle. (2, luku Chapter I: Cloud Drivers.)

2.4.3 Liiketoimintamahdollisuudet

Pilvipalvelut voivat auttaa käyttäjää kehittämään ja julkaisemaan sovelluksensa nopeammin. Muita keskeisiä etuja ovat muun muassa kustannusten määrä. (2, luku Chapter I: Cloud Drivers.)

Pilvipalveluympäristöt voivat olla ratkaisuna kustannusten vähentämiselle. Kun perustetaan perinteinen palvelinympäristö omiin tiloihin, infrastruktuuri ja laitteet pitää ostaa etuajassa ennen muita osia. Pilvipalveluissa ei tarvitse huolehtia laitteiden ostosta, koska palveluntarjoaja huolehtii laitteiden hankinnasta ja päivittämisestä. Käyttäjä maksaa vain käyttämistään palveluista ja resursseista. (2, luku Chapter I: Cloud Drivers.)

2.4.4 Kehittyvä teknologia

Nopea teknologian kehittyminen on yksi syy, miksi pilvipalvelut ovat nousevassa suosiossa. Aikaisemmin pilvipalvelut olivat hyvä idea, mutta teknologia ei ollut vielä valmis tekemään siitä totta. (2, luku Chapter I: Cloud Drivers.)

Ennen oli liian kallista saada tarpeeksi palvelimia täyttämään asiakkaiden tarpeet, koska jokaiselle asiakkaalle olisi pitänyt olla yksityiset palvelimet. Monien soveluksien piti siirtää suuria määriä dataa asiakasohjelman ja sovelluksen välillä. Nyt teknologia on kehittynyt tarpeeksi ja korjannut nämä puutteet. (2, luku Chapter I: Cloud Drivers.)

Virtualisointi

Virtualisointi on ollut yksi tärkeä edistysaskel, joka on auttanut pilvipalveluita menestymään. Sen avulla voidaan ylläpitää useita virtuaalisia järjestelmiä yhden fyysisen järjestelmän sisällä. Tämä on vähentänyt käyttöönoton kustannuksia. Enää ei tarvita erillisiä palvelinlaitetta jokaiselle asiakkaalle. Lisäksi virtuaalisoinnin avulla voidaan yhdistää resursseja ja tehostaa fyysisten järjestelmien hyödyntämistä. Vaikka virtualisointi olikin tärkeä edistysaskel, ei se siltikään ole pakollinen vaatimus pilvipalveluiden tarjoajalle. (2, luku Chapter I: Cloud Drivers.)

Sovelluksen arkkitehtuuri

Ennen yksittäinen palvelinsovellus ei pystynyt palvelemaan monia asiakasohjelmia. Ei ollut keinoa estää yhtä asiakasta pääsemästä käsiksi toisen asiakkaan tietoihin tai sovelluksen osiin. Nyt monta asiakasta voi yhdistyä yhteen sovelluksen olioon, mutta heidän vuorovaikutuksensa on hajautettua. (2, luku Chapter I: Cloud Drivers.)

Sovellukset käyttävät nykyään SOA-mallia eli palvelukeskeistä arkkitehtuuria (engl. service oriented achitehture). SOA:n avulla sovellukset voidaan jakaa useampiin komponentteihin ja niihin päästään käsiksi yksitellen. SOA:n avulla sovellukset voivat myös jakaa komponentteja. (2, luku Chapter I: Cloud Drivers.)

Web-sovellusten standardointi on myös kehittynyt, ja sen avulla yhteensopivuus ja yhteentoimivuus on kasvanut. Se on myös lisännyt web-sovellusten kehitystä. Tämä on johtanut kevyempiin asiakasohjelmistoihin. (2, luku Chapter I: Cloud Drivers.)

Kasvanut kaistanleveys

Internet-yhteyksien nopeudet (kaistanleveys) ovat kasvaneet merkittävästi, mikä on lisännyt nopeutta sovelluksiin. Monissa tapauksissa internet-pohjaista yhteyttä voidaan verrata lähiverkkopohjaiseen yhteyteen. Yleisesi ottaen kaistanleveyden kasvaminen on parantanut web-sovellusten käytettävyyttä. (2, luku Chapter I: Cloud Drivers.)

2.5 Pilvipalvelun huonot puolet

Pilvipalveluilla on monia hyviä puolia, mutta mikään ei ole täydellistä. Huonojen puolien lisäksi on myös ongelmia, jotka voivat pitää käyttäjät loitolla pilvipalveluista. (2, luku Chapter I: Cloud Adoption Inhibitors.)

2.5.1 Epäselvyys

Suurin haitta, joka pitää käyttäjät loitolla pilvipalveluista, on ymmärryksen puute siitä, mitä pilvipalvelut ovat ja mitä ne tarjoavat. Ymmärtämättömyys aiheuttaa pelkoa, joka kohdistuu piilokustannuksiin, hallittavuuden puutteeseen, integroin-

tiongelmiin ja turvallisuuteen liittyviin huolenaiheisiin. Kuitenkin kaikki nämä pelot lieventyvät, kun opitaan, mitä palveluntarjoajalta tulee etsiä ja odottaa. (2, luku Chapter I: Cloud Adoption Inhibitors.)

2.5.2 Kypsymättömyys

Pilvipalvelut ja niiden tarjoajat ovat vielä lastenkengissä, joten huolenaihe niiden kypsyydestä on aito. Palveluntarjoajien tulee antaa sekä haluttua että tasokasta palvelua ja asiakastukea. Monet uudet palveluntarjoajat eivät täytä kuitenkaan asiakkaiden tarpeita, mikä aiheuttaa epäluuloisuutta. (2, luku Chapter I: Cloud Adoption Inhibitors.)

Pilvipalveluiden tarjoajien palvelut eivät ole tarpeeksi laajoja täyttämään käyttäjien kaikkia tarpeita. Jos asiakkaalla ei ole tarvetta tietylle palvelulle, hän ei pysty hyödyntämään sitä. Palveluntarjoajat pyrkivät lisäämään ja päivittämään palvelujaan jatkuvasti, jotta ne voisivat paremmin vastata käyttäjien tarpeisiin. (2, luku Chapter I: Cloud Adoption Inhibitors.)

Kaikki palveluntarjoajat eivät pysty antamaan luvussa 2.4.1 esille tullutta palvelutasosopimus, joka on palvelun laadun takaava sopimus. Asiakas voi tarvita palvelun tai sovelluksen käytettävyyttä vuorokauden ympäri, mutta palveluntarjoajat eivät voi välttämättä taata sitä. (2, luku Chapter I: Cloud Adoption Inhibitors.)

2.5.3 Pilvipalveluiden tietoturvallisuus

Vaikka monet ammattilaiset pitävät pilvipalveluja turvallisempana kuin omilla palvelinlaitteistoilla toteutettua ympäristöä, on olemassa riskejä, jotka tekevät pilvipalveluista vähemmän turvallisen vaihtoehdon. Riskit tulevat pääosin siitä, ettei käyttäjä pysty täysin hallitsemaan järjestelmiä ja tietoja. Käyttäjän pitää luottaa siihen, että palveluntarjoaja osaa työnsä. (2, luku Chapter I: Cloud Adoption Inhibitors.)

Tiedon omistus

Tiedon omistuksesta pilvipalveluissa on monia vastaamattomia kysymyksiä. Yksi suuri kysymys on talletetun tiedon omistuksesta. Käyttäjä on voinut luoda tiedon, mutta se on tallennettuna palveluntarjoajan laitteistoihin. Onko käyttäjällä vielä omistusoikeus siihen? (2, luku Chapter I: Cloud Adoption Inhibitors.)

Palvelua valittaessa tulisi ottaa huomioon seuraavat kysymykset:

- Mitä tapahtuu jos palveluntarjoaja menee konkurssiin?
- Kuinka käyttäjä pääsee tietoon käsiksi?
- Onko konkurssipesästä laitteiston ostaneella omistusoikeudet käyttäjän tietoihin?
- Onko uusi omistaja velvollinen antamaan tiedon käyttäjälle?
- Jos on ongelmatilanne ja käyttäjä ei maksa laskuaan, voidaanko hänen tietojaan pitää panttina?

Eri palveluntarjoajat antavat eri vastauksia, joten on oltava tietoinen siitä, mitä heiltä voi odottaa. (2, luku Chapter I: Cloud Adoption Inhibitors.)

Yksityisyys, lakiasiat ja niiden noudattaminen

Yksityisyys on suuri huolenaihe pilvipalveluissa, koska palveluntarjoajilla on suora pääsy käyttäjän tietoihin. Käyttäjä voi mahdollisesti rikkoa yksityisyyden standardeja tallentamalla omien asiakkaiden tietoja, kuten esimerkiksi kirjautumistietoja, pilvipalveluiden tarjoajan tietokantoihin eikä kerro siitä asiakkailleen. Jos tiedon on tarkoitus olla salaista, pitää käyttäjän varmistaa, mitä palveluntarjoaja tekee asian hyväksi. (2, luku Chapter I: Cloud Adoption Inhibitors.)

Lakiasiat ja niiden noudattaminen ovat monimutkaisia pilvipalveluissa, koska toimivaltaa ei ole vielä tarkkaan määritelty. Jos käyttäjä sijaitsee Suomessa ja hän käyttää palvelimia, jotka ovat Yhdysvalloissa, minkä maan lakeja tulee soveltaa? Yleisenä ohjeena on, että käyttäjän tulisi varmistaa noudattavansa lakeja molemmilla lainkäyttöalueella. (2, luku Chapter I: Cloud Adoption Inhibitors.)

3 WINDOWS AZUREN PALVELUT

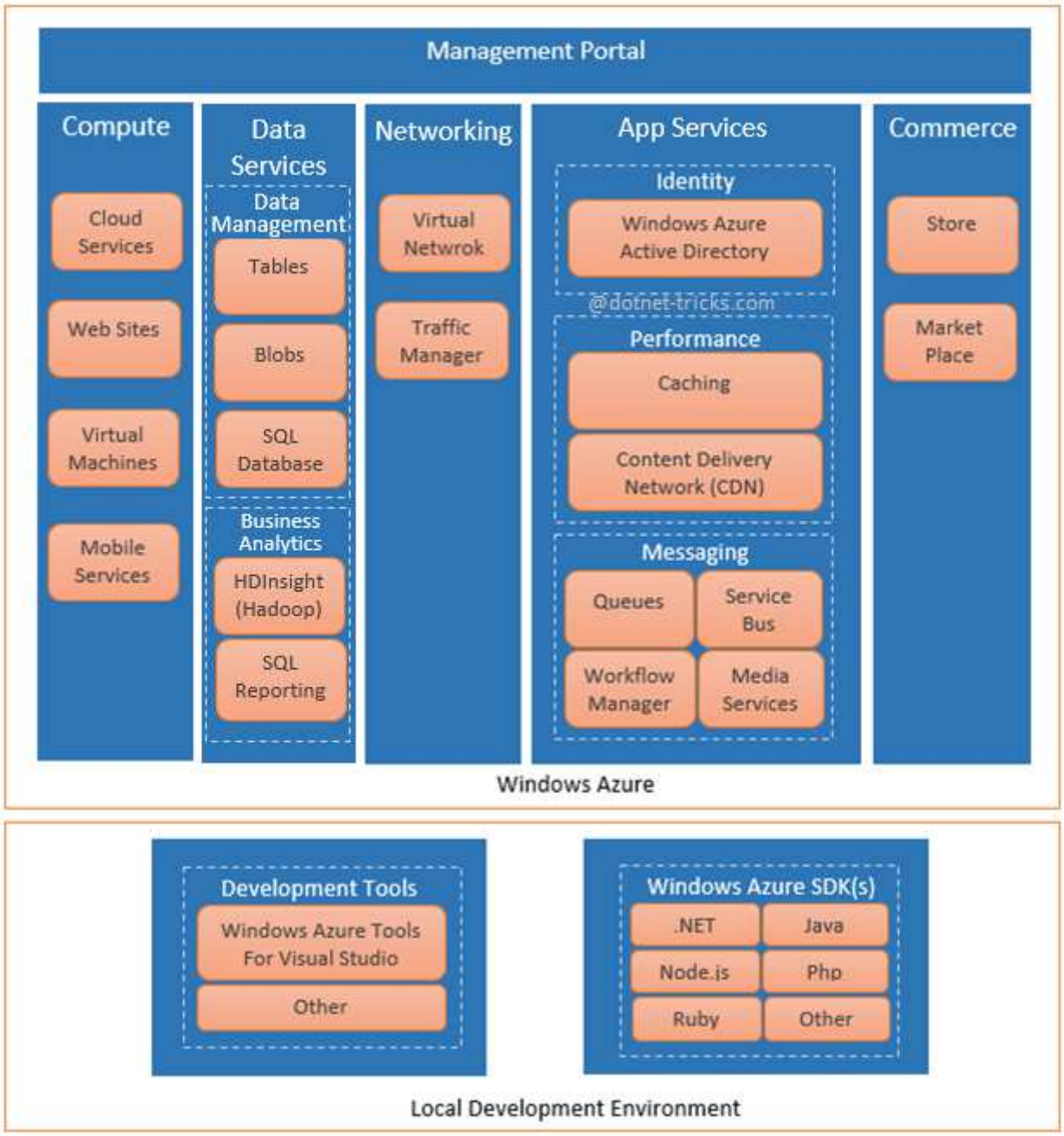
Windows Azure on Microsoftin pilvipalvelujen sovellusalusta, joka antaa kehitysympäristön, palveluita ja palvelun hallintaympäristön Windows Azure -alustalle. Windows Azure antaa kehittäjille tarpeen mukaan laskentatehoa, tiedon varastointitilaa, skaalautuvuutta, web- ja pilvisovellusten hallinnointia sekä palveluita internetin välityksellä. (5.)

Windows Azure on avoin alusta, joka tukee Microsoftin ja muiden kehittäjien ohjelmointikieliä ja -ympäristöjä. Se tukee erilaisia työkaluja ja kieliä, kuten Eclipse, Ruby, PHP, ja Python. (5.)

Windows Azure palveluiden hahmottamiseksi on hyvä jakaa ne neljään pääryhmään: laskentapalvelut, verkkopalvelut, datapalvelut ja sovelluspalvelut (kuva 2). Käyn myöhemmin luvussa 4 läpi tarkemmin laskentapalvelut-ryhmän, joka on tärkein ryhmä pilvisovelluksessa.

Lyhyesti esitettynä pääryhmien määritykset ovat seuraavat:

- Laskentapalvelut antavat prosessointitehoa, jonka avulla suoritetaan pilvisovelluksia.
- Verkkopalvelut antavat vaihtoehtoja siihen, miten Windows Azuren sovelluksia voidaan toimittaa käyttäjille ja tietokeskuksiin.
- Datapalvelut antavat erilaisia tapoja tallentaa, hallita ja turvata tietoa sekä analysoida yrityksen datan käyttöä.
- Sovelluspalvelut antavat keinoja parantaa pilvisovellusten suorituskykyä, löydettävyyttä ja integroituvuutta. (6, luku Chapter 1: Windows Azure under the hood.)



KUVA 2. Windows Azuren palvelut jaettuina ryhmiin (7)

4 WINDOWS AZUREN LASKENTAPALVELUMALLIT

Luvussa perehdytään Windows Azuren kolmeen tärkeimpään laskentapalvelumalliin. Palvelumallien nimet ovat myös itsenään käytettynä yleisiä teknisiä termejä: Virtual Machines on virtuaalikoneet, Web Sites on nettisivut ja Cloud Services on pilvipalvelut. Tässä dokumentissa, kun käsitellään Windows Azuren laskentapalvelumallia, käytetään englanninkielistä termiä. Suomenkielistä sanaa käytetään, kun puhutaan yleisesti esimerkiksi virtuaalikoneesta.

4.1 Virtual Machines

Virtual Machinellä luodaan virtuaalikone. Virtuaalikoneen luominen alkaa virtuaalikoneen image-tiedostosta. Windows Azuren Virtual Machine -galleriassa löytyy valmiita image-tiedostoja tai voidaan käyttää myös omia image-tiedostoja. Image-tiedostoa valittaessa ensimmäinen tärkeä yksityiskohta on järjestelmän koko: kuinka monta CPU-ydintä, kuinka paljon muistia ja kuinka iso kovalevytila varataan. Valmiissa image-tiedostoissa on Microsoft- tai Linux-käyttöjärjestelmä. Vaihtoehtojen määrä kasvaa koko ajan, ja esimerkiksi seuraavista palvelinkäyttöjärjestelmistä on eri versiota:

- Windows Server
- Linux servers (esim. Suse), Ubuntu ja CentOS
- SQL Server
- BizTalk Server
- SharePoint Server. (8.)

Virtual Machinen avulla voidaan nopeasti kehittää sovelluksia ja ne voidaan poistaa, kun niitä ei enää tarvita. Niitä voidaan integroida myös jo valmiina oleviin laitteisiin ja näin lisätä niiden laskentatehoa. Virtual Machinen avulla käyttäjä voi helposti ja nopeasti palautua omien palvelinlaitteiden ongelmista. Se voidaan luoda pyörittämään tärkeimpiä sovelluksia ja sulkea, kun palvelinongelmat on ratkaistu. (8.)

4.2 Web Sites

Käyttäjä voi luoda Windows Azuren Virtual Machinen avulla internet-sivut, mutta se vaatii taitoa ja ylläpitämistä. Yksinkertaisempi tapa tehdä internet-sivut tai yksinkertainen web-sovellus on Windows Azure Web Sites. (8.)

Windows Azuren Web Sites ajetaan yksittäisellä virtuaalikoneella, mutta käyttäjän ei tarvitse alustaa tai ylläpitää virtuaalikonetta. Web Sites voi olla useamman henkilön jakamassa virtuaalikoneessa (vaihtoehto Shared), tai henkilö voi ostaa yksityisen virtuaalikoneen. (8.)

Aloitus Windows Azuren Web Sitesilla on helppoa, koska siinä on laaja valikoima valmiita sovelluksia, ohjelmistokehyksiä ja mallipohjia, joilla luodaan nettisivut nopeasti. Windows Azuren Web Sitesilla on tuki myös kolmannen osapuolen ohjelmointikielille, kuten PHP ja Python. (8.)

4.3 Cloud Services

Cloud Services on suunnattu tukemaan sovelluksia, jotka ovat skaalautuvia, luotettavia ja halpoja ylläpitää. Sen tarkoitus on vapauttaa ohjelmistokehittäjät huolehtimasta kehitysalustasta ja antaa heidän keskittyä pelkästään sovelluksiin. (8.)

Kuten muutkin Windows Azuren palvelumallit, Cloud Services käyttää virtuaalikoneita. Cloud Services on kaksi erilaista virtuaalikonetyyppiä:

- Web-rooli, joka antaa käyttöön Internet Information Services -web-palvelimen, jota käytetään web-sovelluksien käyttöliittymissä.
- Worker-rooli, joka voi suorittaa asynkronisia, pitkäkestoisia tai ikuisia tehtäviä, jotka eivät ole riippuvaisia käyttäjästä. (8.)

Cloud Servicesillä ei siis luoda virtuaalikonetta, vaan tehdään asetustiedostot, jotka kertovat, kuinka monta oliota luodaan ja mitä rooleja ne tekevät (8). Ase-

tustiedostoihin kuuluvat palvelupakettitiedosto (.cspkg) ja konfiguraatiotiedosto (.cscfg), jotka käydään läpi myöhemmin luvussa 7.

4.4 Sovellusmallien käyttötarkoitus

Dokumentissa aiheena olevaa asiakas-palvelinesimerkkisovellusta varten käytettiin Cloud Servicesiä, mutta kaikki kolme vaihtoehtoa antaisivat mahdollisuudet rakentaa skaalautuvan ja luotettavan sovelluksen pilveen.

Virtual Machine sopii lähes kaikkeen. Se antaa eniten mahdollisuuksia, mutta sen mukana tulee myös paljon vastuuta, joten tämä vaihtoehto vaatii eniten ylläpitoa käyttäjältä. (8.)

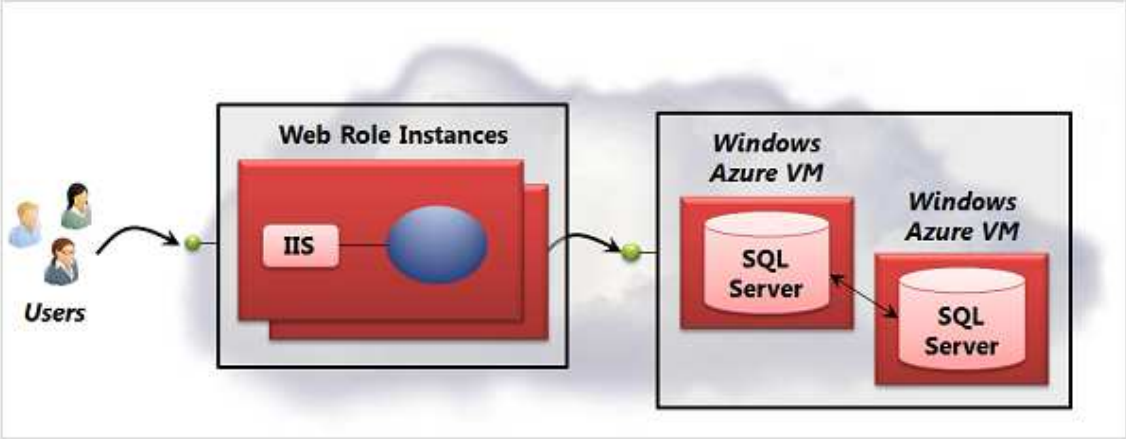
Web Sites on hyvä vaihtoehto yksinkertaisille nettisivusovelluksille. Se on käytännöllinen, jos haluaa käyttää paljon valmiita sovelluksia, kuten Joomlaa, WordPressiä tai Dupalia, vähän ylläpitoa vaativien internetsivujen luomiseen. Se on myös hyvä vaihtoehto vähäistä ylläpitoa vaativalle sovellukselle, jonka pitää olla skaalattavissa.

Cloud Services on hyvin samankaltainen kuin Web Sites, mutta ne eroavat toisistaan muutamissa tärkeissä asioissa:

- Cloud Services antaa virtuaalikoneeseen hallinnointityökaluja, jonka avulla voidaan asentaa monia ohjelmistoja. Tämä ei ole mahdollista Web Sitesilla.
- Cloud Services on parempi vaihtoehto monipuolisemmille sovelluksille, koska se sisältää sekä web- että worker-roolin olioita.
- Cloud Services sisältää erilliset Staging- ja Production-ympäristöt, jotka tekevät sovelluksen päivittämisestä sulavampaa kuin Web Sites vaihtoehdossa.
- Cloud Servicesissä pystytään Windows Azure Virtual Networkin ja Windows Azure Connectin avulla kytkemään omia laitteistoja Cloud Servicesin jatkeeksi.

- Cloud Servicesin avulla voidaan ottaa etäyhteys sovelluksen virtuaalikoneeseen. Tämä ei ole mahdollista Web Sitesilla. (8.)

Sovellusmalleja on mahdollista yhdistää tarpeiden mukaan. Esimerkiksi käyttäjä voisi haluta hyödyn Cloud Servicesin Web-rooleista, mutta haluaisi myös käyttää Virtual Machinen SQL-palvelimia yhteensopivuuden tai suorituskyvyn takia (kuva 3). (8.)



KUVA 3. Yksittäinen sovellus voi käyttää hyväkseen useampaa toimintamallia (8)

5 WINDOWS AZUREN KÄYTTÖÖNOTTO

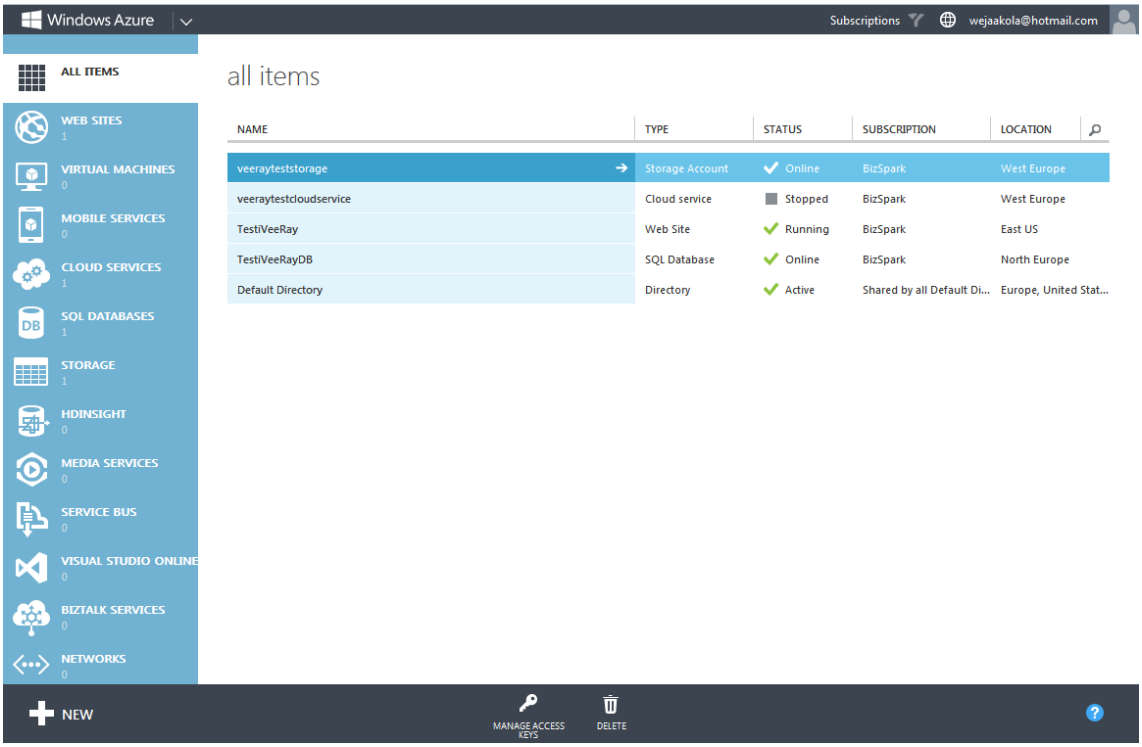
Tässä luvussa kerrotaan, mitä Windows Azuren Portaalissa tulee tehdä esimerkkisovellusta varten. Työssä käytettiin asiakas-palvelinsovelluksen ajamiseen Windows Azuren Cloud Services -palvelua. Tekstissä käytetään esimerkkisovellus-termiä, jolla tarkoitetaan tässä opinnäytetyössä tehtyä asiakas-palvelinsovelluskokonaisuutta, kun taas palvelinohjelma- tai asiakasohjelma-termiä käytettäessä tarkoitetaan esimerkkisovelluksen yksittäistä komponenttia.

Esimerkkisovelluksessa käytetään kolmea Windows Azure -komponenttia: Cloud Servicesiä, Storagea ja SQL Databasea. Cloud Services -palvelun virtuaalikone suorittaa palvelinohjelmaa, joka on tallennettuna Windows Azure Storageen. Asiakasohjelma ottaa yhteyden palvelinohjelmaan ja lähettää sille pyyntöjä. Pyydettäessä palvelinohjelma hakee Windows Azuren SQL-tietokannasta dataa ja lähettää sen asiakasohjelmalle tulostettavaksi.

5.1 Windows Azuren portaali ja komponenttien alustus

Windows Azuren Portaalia varten pitää luoda tunnukset, joiden avulla kirjaudutaan palveluun. Tehtäviin tunnuksiin tarvitaan luotto- tai pankkikortti, jolla varmistetaan henkilöllisyys ja jota käytetään maksusuorituksiin, joita komponenttien käytöstä tulee. Windows Azuren etusivut löytyvät osoitteesta http://www.windowsazure.com/en-us/.

Windows Azuren etusivun yläreunasta löytyy linkki Portal, josta päästään Windows Azuren Portaliin (kuva 4). Vaihtoehtoisesti Windows Azure Portaliin voidaan mennä suoraan osoitteesta https://manage.windowsazure.com/.

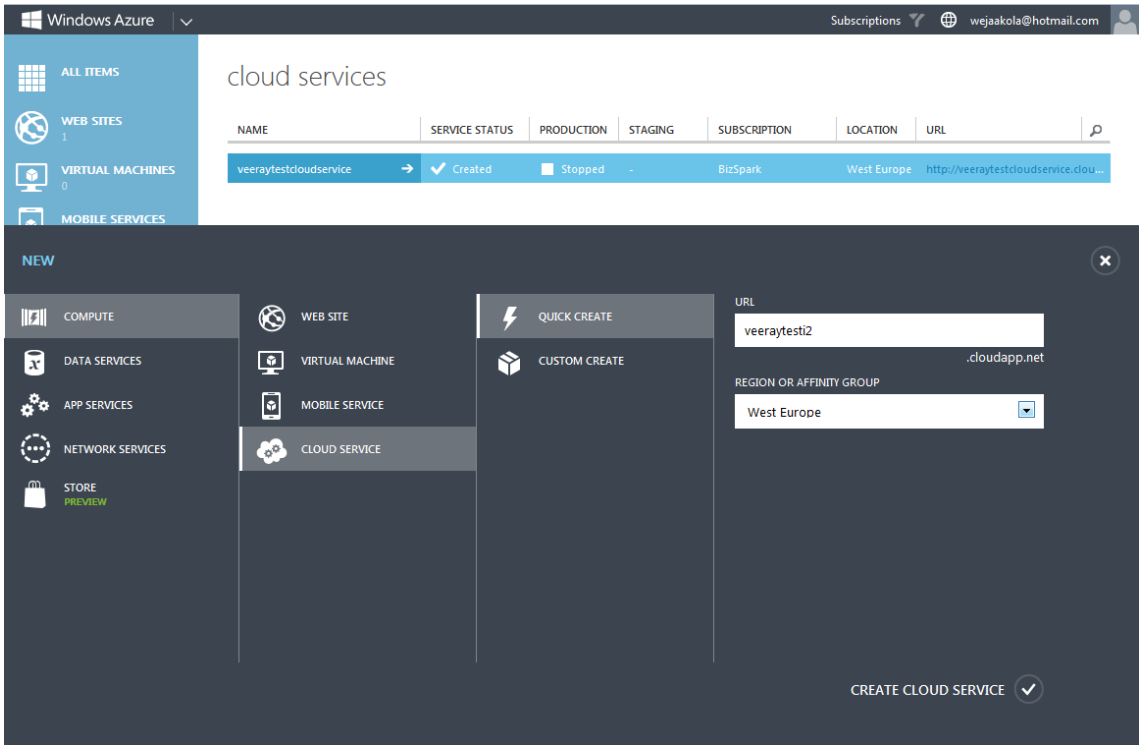


KUVA 4. Windows Azure Portalin etusivu

5.1.1 Cloud Services

Cloud Services luodaan valitsemalla kuvan 4 vasemmalla näkyvästä listasta Cloud Services ja painamalla sen jälkeen vasemmasta alareunasta NEW-painiketta. Quick Create riittää perustarpeisiin, mutta on mahdollista määritellä tarkemmin tilattavan palvelun ominaisuuksia valitsemalla Custom Create. Kun Quick Create on valittu, pitää valita palvelulle nimi ja maanosa, jonka palvelimia käytetään. On kannattavaa valita itseään lähimpänä sijaitseva paikka, kuten esimerkiksi West Europe. Kun kilkataan kuvan 5 oikeassa alareunassa näkyvää ok-painiketta (check-merkki), Windows Azure aloittaa palvelun luomisen, jossa kestää 5–10 minuuttia. Cloud Servicesille ei tässä vaiheessa tarvitse tehdä muuta.

Myöhemmin luvussa 7 opastetaan Cloud Services -palvelun käynnistys, jota varten luodaan palvelupakettitiedosto (.cspkg) ja konfiguraatiotiedosto (.cscfg). Ne määrittelevät workerien määrän, tyypin ja toiminnot.



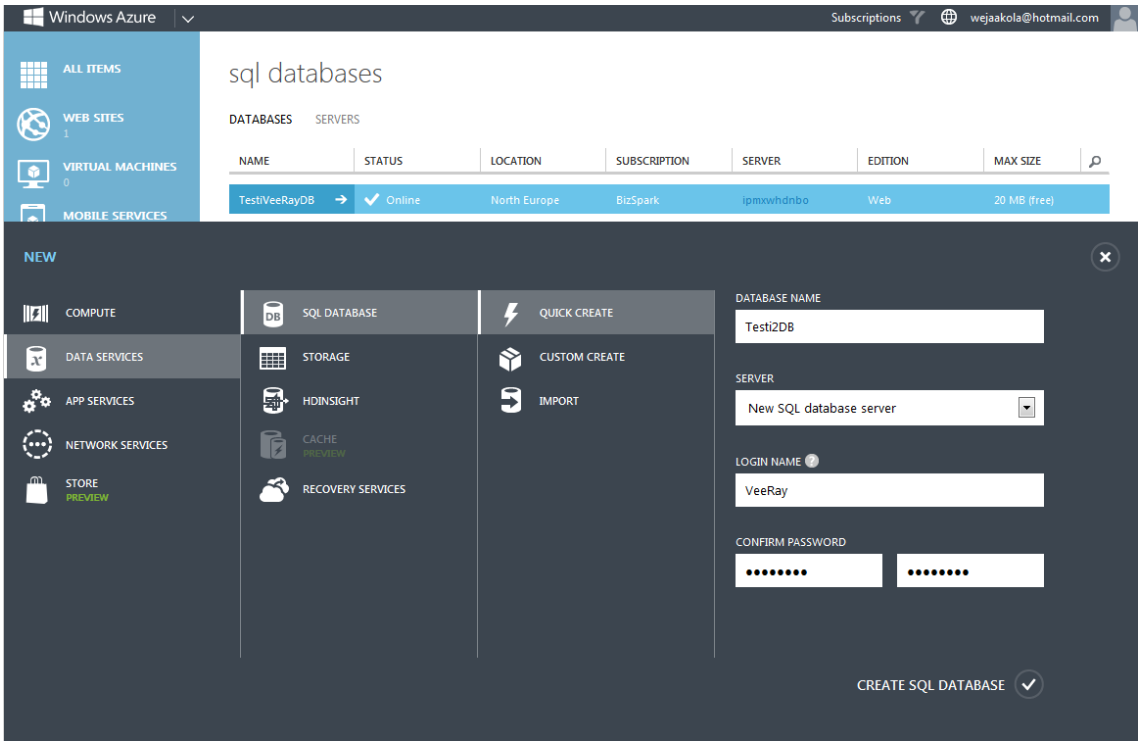
KUVA 5. Windows Azure Cloud Services palvelun luominen

5.1.2 SQL Database

SQL Databasen luominen tapahtuu samalla periaatteella kuin Cloud Services. Valitaan vasemmasta listasta ensin SQL Database ja sen jälkeen klikataan vasemmalta alhaalta NEW-painiketta.

SQL Databasen ja Cloud Servicesin luomisessa on eroja, sillä Quick Createn ja Custom Createn lisäksi on myös mahdollista tuoda jo olemassa oleva SQL-tietokanta Windows Azureen. Ensimmäistä tietokantaa luodessa tietokannan nimen lisäksi pitää asettaa kirjautumistiedot, joilla tietokantaan päästään käsiksi.

Luominen hyväksytään klikkaamalla kuvan 6 oikeassa alareunassa näkyvää ok-painiketta (check-merkki). Windows Azure aloittaa tietokannan luomisen, jossa voi kestää 2–10 minuuttia. SQL Databaselle ei tässä vaiheessa tarvitse tehdä muuta, mutta luvussa 5.2 luodaan esimerkkisovellusta varten tietokantataulu, joka sisältää tietoa.



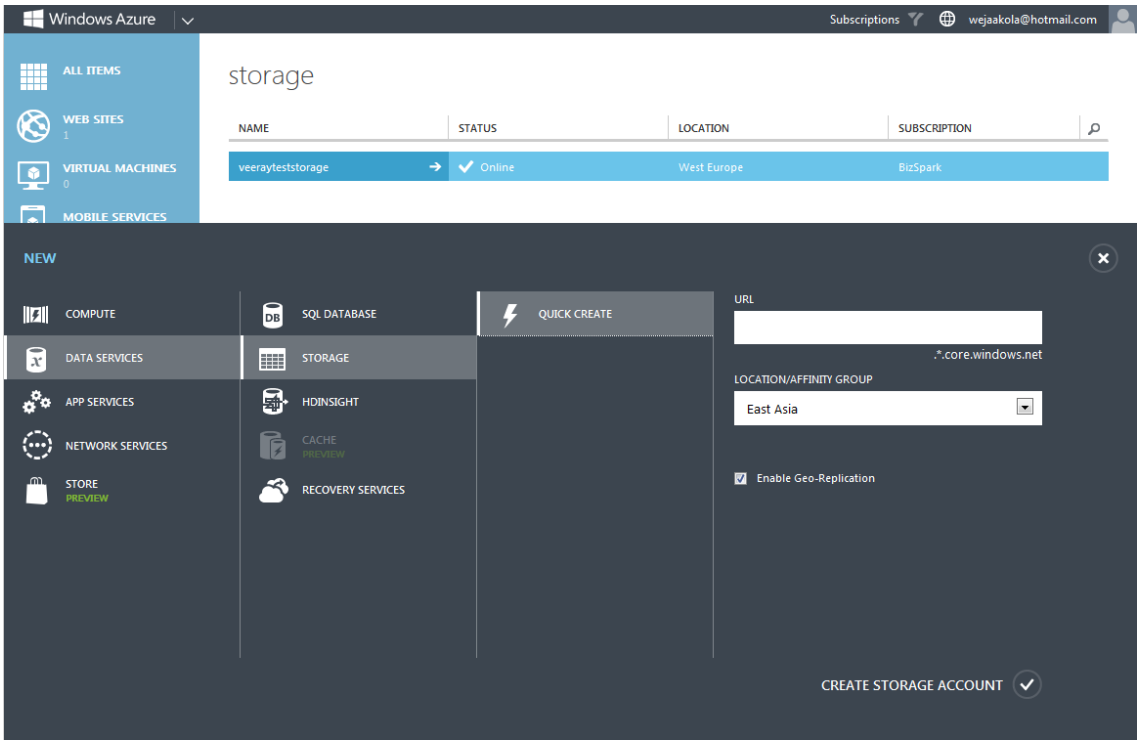
KUVA 6. Windows Azure SQL Databasen luominen

5.1.3 Storage

Storagen luominen tapahtuu samalla periaatteella kuin Cloud Services. Valitaan vasemmasta listasta ensin Storage ja sen jälkeen klikataan vasemmalta alhaalta NEW-painiketta.

Storagen ja Cloud Servicesin luomisessa on eroja. Valittavana on vain Quick Create -luominen ja sille voi valita lisävaihtoehdon Geo-Replication. Geo-Replication antaa enemmän luotettavuutta tiedon tallessa pysymiseen, sillä data pidetään tallessa kahdessa eri palvelinkeskuksessa. Esimerkiksi jos yksi palvelinsaleista tuhoutuisi täysin tulipalossa, tiedot ovat vielä tallessa toisessa sijainnissa. Kun klikataan kuvan 7 oikeassa alareunassa näkyvää ok-painiketta (check merkki), Windows Azure aloittaa Storagen luomisen, jossa voi kestää 2–10 minuuttia.

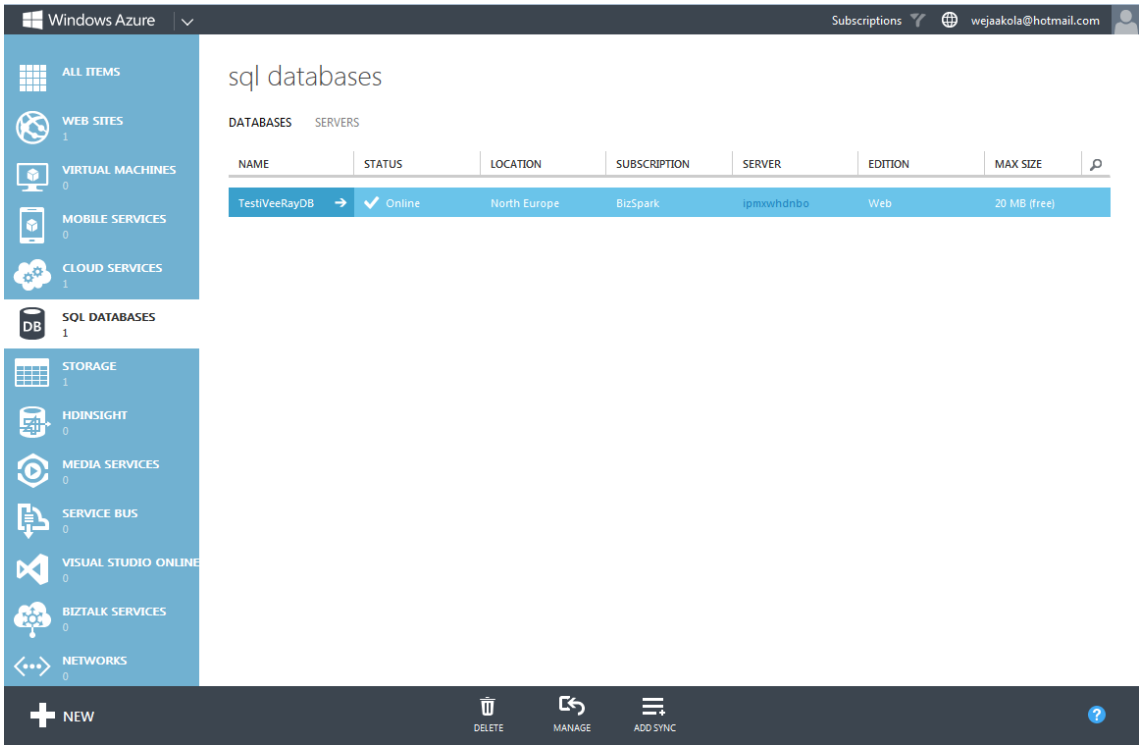
Storageen sisällön käsiksi pääsemiseen tarvitaan liitännäinen (engl. plugin) Microsoft Visual Studioon tai kolmannen osapuolen sovellus. Luvussa 5.3 käydään läpi kolmannen osapuolen sovellus, jota käytettiin esimerkkisovellusta tehdessä.



KUVA 7. Windows Azure Storagen luominen

5.2 SQL Databasen muokkaaminen

Tietokantaan pitää syöttää tietoa esimerkkisovellusta varten. Tietokannan sisältöä voidaan muokata valitsemalla ensin vasemmasta reunasta SQL Database. Tämän jälkeen valitaan aukeavasta listasta tietokanta, jota halutaan muokata, ja klikataan kuvan 8 alareunassa näkyvää Manage-painiketta. Painike ohjaa käyttäjän uudelle sivulle, jossa pitää syöttää tietokannan tunnus ja salasana, jotka päätettiin tietokannan luontivaiheessa luvussa 5.1.2.



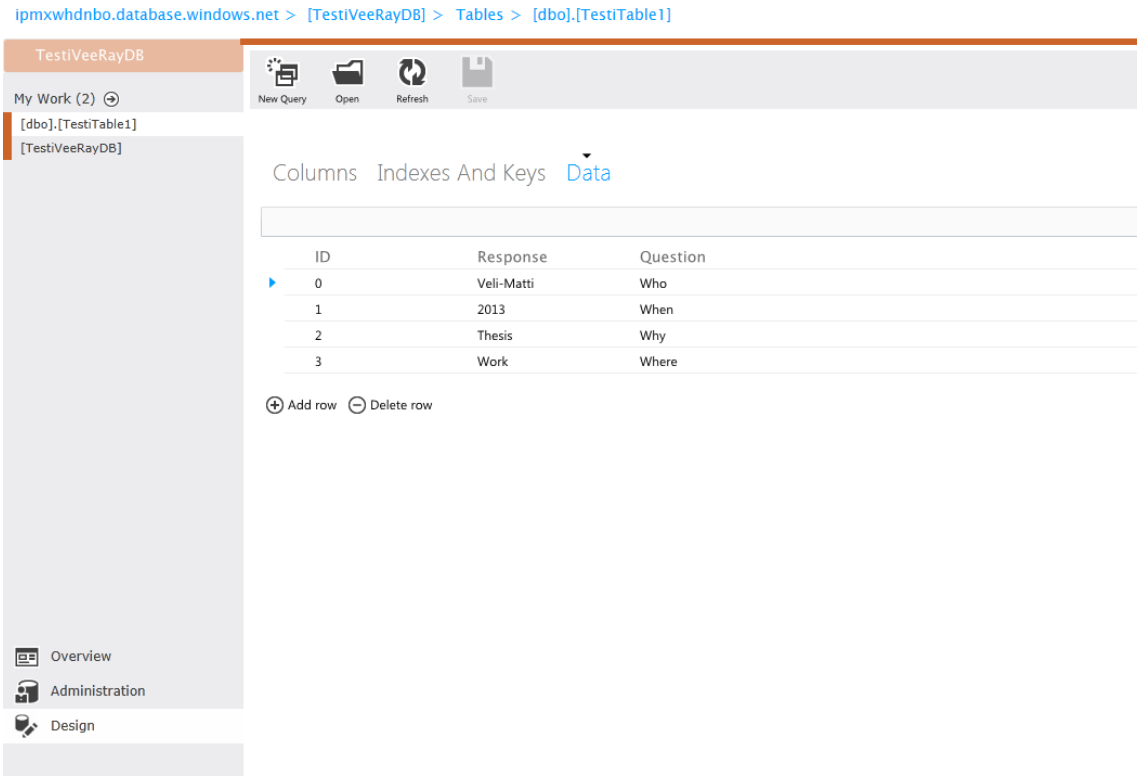
KUVA 8. Windows Azure Portalin SQL-tietokantalista

Tietokannan hallintaportaalissa voi hallinnoida kahdella tavalla. Joko käytetään SQL-kyselyjä painamalla ylhäältä New Query -painiketta, tai voidaan käyttää graafista käyttöliittymää klikkaamalla alhaalta vasemmalta Design-painiketta. Graafisella käyttöliittymällä ei voi tehdä kaikkea, mitä SQL-kyselyt mahdollistavat, mutta graafinen käyttöliittymä riittää yksinkertaisten taulujen luomiseen ja tietojen lisäykseen.

Design-painikeen klikkaamisen jälkeen aukeaa lista olemassa olevista tauluista, johon voidaan luoda uusia tauluja painamalla New table -painiketta. Taulussa on jo valmiiksi muutama ehdotettu sarake, mutta ne voidaan poistaa tai niitä voidaan muokata omiin tarpeisiin. Kun halutaan luoda taulu tai tallentaa siihen tehdyt muutokset, klikataan yläreunasta löytyvää Save-painiketta.

Tauluille ja sarakkeille tulee antaa kuvaava nimi, koska niitä joudutaan käyttämään palvelinohjelman SQL-kyselyissä. Esimerkkisovellusta varten tehtiin taulu nimeltä TestiTable1 ja siihen kolme saraketta nimeltä: ID, Question ja Response. Taulua voi myös myöhemmin muokata lisäämällä sarakkeita tai nimeämällä niitä uudelleen.

Tauluun voidaan syöttää tietoa siirtymällä Data-välilehteen (kuva 9), jossa tieto syötetään käsin. Tieto lisätään yksi solu kerrallaan ja se lopuksi tallennetaan Save-painikkella. Tiedoilla ei ole esimerkkisovelluksen kannalta muuta merkitystä, kuin että voidaan tunnistaa, että palvelinohjelma hakee juuri siellä sijaitsevan tiedon.



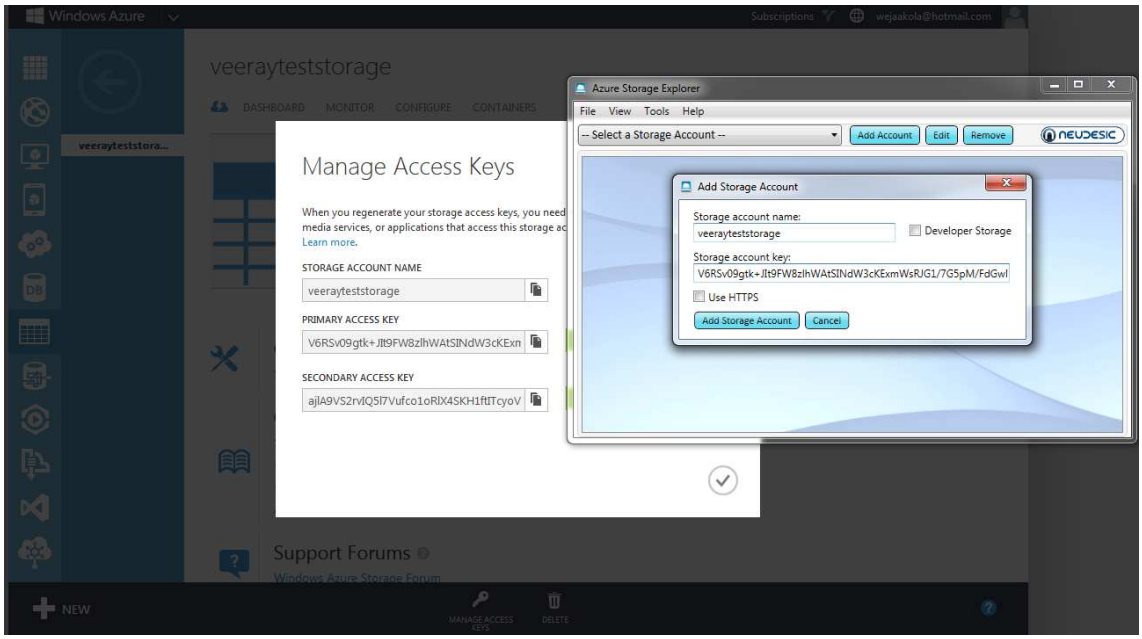
KUVA 9. Tietokannan hallintaruutu

5.3 Storagen hallinta

Storagen hallintaan tarvitaan joko Microsoft Visual Studio 2012 tai uudempi versio ja Windows Azure SDK -liitännäinen, tai voidaan käyttää kolmannen osapuolen ohjelmistoa. Esimerkkisovellusta tehdessä käytettiin kolmannen osapuolen tekemää Azure Storage Explorer -sovellusta, johon Windows Azuren -sivut ohjasivat. Azure Storage Explorer -sovelluksen voi ladata osoitteesta http://azurestorageexplorer.codeplex.com/.

Asennuksen ja ohjelmiston käynnistämisen jälkeen pitää ohjelmistoon lisätä oma Windows Azure Storage -tili, johon pitää tietää Storagen nimi ja pääsyavain (engl. primary access key). Molemmat voidaan selvittää Windows Azuren

Portaalista valitsemalla ensin vasemmasta reunasta Storage ja avautuvasta listasta haluttu Storage-tili. Tämän jälkeen klikataan ruudun alareunassa löytyvää Manage Access Key -painiketta, joka avaa ponnahdusikkunan, josta löytyy storage account name ja primary access key -kohdat (kuva 10). Storage-tili jää Azure Storage Explorer -sovelluksen muistiin ja seuraavilla kerroilla siihen voidaan päästä käsiksi alasvetoikkunasta.



KUVA 10. Storage-tilin lisäys Azure Storage Explorer -sovellukseen

Esimerkkisovellusta varten luotiin container nimeltä packages, jonka sisälle ladataan ajettava palvelinohjelma luvussa 7.2.1. Container on Windows Azuren nimitys blob storage -kansiolle.

6 ASIAKAS-PALVELINSOVELLUS

Asiakas-palvelin kuvaa sovellusarkkitehtuuria, jossa asiakasohjelma lähettää pyyntöjä ja odottaa niihin vastausta palvelinohjelmalta. Verkkoselain ja verkkopalvelin ovat yksinkertainen esimerkki, jonka kaikki internettiä käyttäneet ovat kokeneet. Kun syöttää URL-osoitteen verkkoselaimessa (asiakasohjelma), lähettää se pyynnön nettisivusta verkkopalvelimelle. Verkkopalvelin palauttaa verkkoselaimelle html-sivun, joka muodostuu luettavaan muotoon jäsennyksen (engl. parsing) jälkeen. (9.)

6.1 Esimerkkisovelluksen kuvaus

Esimerkkisovellus koostuu kahdesta osasta: asiakas- ja palvelinohjelmasta. Palvelinohjelma luo käynnistyttyään itsestään olion, joka suorittaa loputtomiin omaa run()-funktiota. Funktion sisällä odotetaan yhteydenottoa asiakasohjelmalta. Yhteyden saatuaan palvelinohjelma odottaa viestejä asiakasohjelmalta. Yksi viesteistä pyytää palvelinohjelmaa hakemaan SQL-tietokannasta tietueen ja lähettämään sen asiakasohjelmalle tulostettavaksi.

Asiakasohjelma sisältää yksinkertaisen graafisen käyttöliittymän, jossa on painikkeita. Painikkeilla voidaan yhdistää palvelinohjelmaan tai lähettää sille viestejä. Lisäksi asiakasohjelman graafisessa käyttöliittymässä on tekstilaatikko, johon lähtevät ja vastaanotetut viestit tulostetaan.

Sovelluksen tarkoitus on esitellä, miten mahdollisimman yksinkertainen asiakas-palvelinsovellus voisi toimia Windows Azuren avulla. Sovelluksessa on vain välttämättömimmät kirjastot. Ohjelmakoodit ovat yhden tiedoston sisässä, jotta ei tarvitsisi hahmottaa monimutkaista luokkarakennetta. Palvelinohjelman lähdekoodi löytyy liitteestä 1 ja asiakasohjelman lähdekoodi liitteestä 2.

6.2 Ohjelmointiympäristö ja ohjelmointikieli

Ohjelmointiympäristöksi valittiin NetBeans IDE ja ohjelmointikieleksi Java, koska opinnäytetyön projektissa käytettiin näitä työkaluja. Saman

esimerkkisovelluksen voisi tehdä monilla muilla ohjelmointityökaluilla ja ohjelmointikielellä Windows Azureen.

NetBeans IDE on ilmainen ohjelmointiympäristö, joka toimii monien eri ohjelmointikielien kanssa. Ohjelmointiympäristöstä on olemassa monta eri asennuspakettia, joissa on tuet eri ohjelmointikielille. Tähän esimerkkisovellukseen riitti Java SE -versio. NetBeansin voi ladata osoitteesta https://netbeans.org/downloads/. Tietokoneella pitää olla myös asennettuna uusin Java Development Kit (JDK). JDK:n voi ladata osoitteesta http://www.oracle.com/technetwork/java/javase/downloads/index.html.

6.3 Palvelinohjelma

Palvelinohjelman toiminnot koostuvat alussa kutsuttavasta main()-funktiosta, ikuisessa loopissa kutsuttavasta run()-funktiosta ja kolmesta muusta funktiosta: sendMessage(String msgString)-, getConnection()- ja getDataFromTable()-funktioista.

Funktiot lyhyesti kuvattuina:

- main()-funktio, joka suoritetaan kerran alussa
- run()-funktio, joka sisältää pääkoodin
- sendMessage(String msgString)-funktio, jonka avulla lähetetään asiakasohjelmalle viestejä
- getConnection()-funktio, joka yhdistää SQL tietokantaan
- getDataFromTable()-funktio, joka suorittaa SQL-kyselyn.

Tärkeimmät luokat lyhyesti kuvattuina:

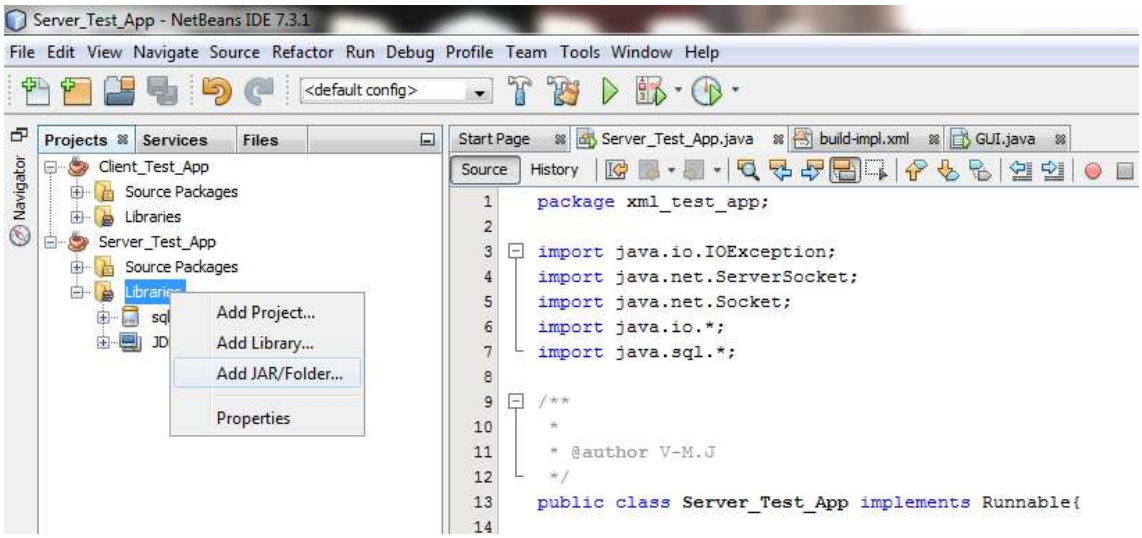
- ServerSocket, joka odottaan pyyntöjä verkon ylitse
- Socket, joka on endpoint viestinnälle kahden koneen välillä

- ObjectOutputStream, jota käytetään viestien lähettämiseen kahden koneen välillä
- ObjectInputStream, joka lukee ObjectOutputStreamin luomat viestit
- Connection, jota käytetään yhdistettäessä tietokantaan
- ResultSet-datataulukko, joka edustaa tietokannasta saatua tulosjoukkoa, joka syntyy, kun suoritetaan SQL-kysely.

Tarvittavat kirjastot

Yhteys Windows Azuren SQL -tietokantaan vaatii sqljdbc4.jar-nimisen luokkakirjaston, joka pitää lisätä projektin Libraries-kansioon. Luokkakirjasto on tarkoitettu JRE (Java Runtime Enviroment) -versioille 6 ja 7. On olemassa myös vanhempi versio sqljdbc.jar, joka antaa tuen versioille 5, mutta se ei toimi versioiden 6 ja 7 kanssa. Luokkakirjasto sqljdbc4.jar voi ladata Microsoftin virallisilta sivuilta http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774.

Luokkakirjasto lisätään NetBeansissä Projects-välilehdessä, jossa on projektikansion rakennenäkymä. Klikkaamalla hiiren oikealla painikkeella Libraries-kansiota avautuu valikko, josta valitaan Add JAR/Folder... vaihtoehto (kuva 11), joka aukaisee tiedostonhallinta ikkunan. Tiedostonhallintaikkunan avulla voi hakea sqljdbc4.jar-tiedoston ja lisätä sen klikkaamalla Open-painiketta.



KUVA 11. Luokkakirjasto sqljdbc4.jar-tiedoston lisääminen

Palvelinohjelman suorituksen kuvaus

Käynnistyttyään palvelinohjelma luo aluksi itsestään olion ja alkaa suorittamaan run()-funktiota ikuisesti while-silmukan sisällä. Funktion run() alussa luodaan ServerSocket-olio porttiin 8080, jonka jälkeen se odottaa yhteydenottoa asiakasohjelmalta.

Kun yhteys on luotu, alustetaan lähteviin ja saapuviin viesteihin tarvittavat ObjectOutputStream ja ObjectInputStream sekä lähetään asiakasohjelmalle viesti. Tämän jälkeen siirrytään do-while-silmukkaan, jonka sisällä luetaan asiakasohjelmalta tulevat viestit ja vastataan niihin if-rakenteessa. Do-while-silmukkaa suoritetaan, kunnes asiakasohjelma lähettää bye-viestin.

Kun ohjelmakoodi pääsee ulos do-while-silmukasta eli se on lopettanut keskustelun asiakasohjelman kanssa, suljetaan stream- ja socket-olioiden yhteydet. Koska run()-funktio on ikuisen while-silmukan sisässä, alkaa run()-funktio alusta sen loputtua.

6.4 Asiakasohjelma

Asiakasohjelman toiminnot koostuvat alussa kutsuttavasta main()-funktiosta, painikkeiden painamisesti syntyvistä eventeistä, sendMessage() ja receiveMessage(). Lisäksi asiakasohjelmaan kuuluu graafinen näkymä, joka helpottaa sen hallintaa sekä lähtevien että tulevien viestien tulkitsemista.

Funktiot lyhyesti kuvattuna:

- main()-funktio, joka suoritetaan kerran alussa
- sendMessage(String msgString)-funktio, jonka avulla lähetetään palvelinohjelmalle viestejä
- receiveMessage()-funktio, joka lukee palvelinohjelmalta tulevat viestit.

Tärkeimmät luokat lyhyesti kuvattuna:

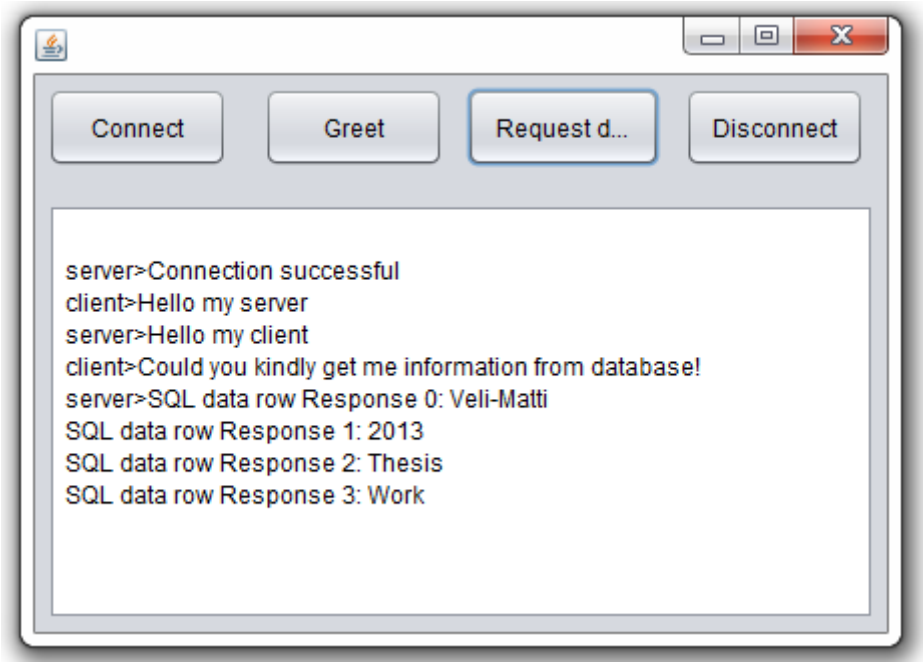
- Socket, joka on endpoint viestinnälle kahden koneen välillä
- ObjectOutputStream, jota käytetään viestien lähettämiseen kahden koneen välillä
- ObjectInputStream, joka lukee ObjectOutputStreamin luomat viestit.

Asiakasohjelman suorituksen kuvaus

Asiakasohjelman käynnistyttyä main()-funktio luo olion pääluokastaan, joka on rakennettu JFrame-pohjalla graafisen näkymän vuoksi (kuva 12). JFramen-painikkeet ja tekstikenttä on lisätty Design-moodissa, jolloin NetBeans on lisännyt niille tarvittavat koodit automaattisesti.

Connect-painike luo yhteyden palvelinohjelmaan tekemällä Socket-olion, joka yhdistetään palvelinohjelman osoitteeseen ja porttiin. Tämän jälkeen liitetään ObjectOutStream- ja ObjectInStream-oliot tähän yhteyteen, jotta viestien lähetys ja vastaanottaminen onnistuisi.

Kolme muuta painiketta ohjelmassa, Greet, Request ja Disconnect, toimivat kaikki samalla periaatteella. Ne lähettävät sendMessage(String messageString)-funktion avulla viestin palvelinohjelmalle ja jäävät sen jälkeen odottamaan vastausta siltä. Ainoana erona on Disconnect-painike, jossa palvelinohjelman vastauksen jälkeen suljetaan sekä stream- että socket-olioiden yhteydet.



KUVA 12. Valmis asiakasohjelma

6.5 Puutteet esimerkkisovelluksessa

Esimerkkisovelluksesta on tehty mahdollisimman yksinkertainen, sillä sen päätarkoitus oli esitellä, miten päästään nopeasti käsiksi Windows Azuren komponentteihin. Tästä johtuen sen toiminnassa ja ominaisuuksissa on puutteita. Yksi esimerkki tästä on, ettei palvelinohjelma pysty palvelemaan montaa asiakasohjelmaa yhtä aikaa.

7 PALVELINOHJELMAN KÄYTTÖÖNOTTO WINDOWS AZURESSA

Cloud Servicesin oliota varten pitää rakentaa projekti, jonka paketoinnista saadaan palvelupakettitiedosto (.cspkg) ja konfiguraatiotiedosto (.cscfg). Tiedostot sisältävät muun muassa workerien tyypit, määrän ja tehtävät.

Projekti voidaan rakentaa joko Eclipse IDE:llä ja siihen tarkoitetulla Windows Azure SDK -liitännäisellä tai Microsoft Visual Studio 2012:lla tai sitä uudemmalla versiolla ja siihen asennettavalla Windows Azure SDK -liitännäisellä. Esimerkkisovellusta varten valittiin Microsoft Visual Studio, koska sille löytyy lähes valmis projektipohja Windows Azure Cloud Servicesiä varten. Projektipohja nimeltä AzureRunMe käydään läpi luvussa 7.2.

7.1 Tarvittavat työkalut ja resurssit

Ohjelmointiympäristönä käytettiin Microsoft Visual Studio Express 2012 for Web -ohjelmistoa, jonka voi rekisteröidä ilmaiseksi omaa käyttöä varten. Sen voi ladata osoitteesta http://www.microsoft.com/en-us/download/details.aspx?id=30669.

Windows Azuren .NET SDK:n mukana tulee tärkeitä kirjastoja Visual Studioon sekä tyhjä projektipohja Windows Azure Cloud Servicesille, joka löytyy Cloud-tyypin alta projekteista. Windows Azure .NET SDK:n voi ladata osoitteesta http://www.windowsazure.com/en-us/downloads/.

7.2 Valmis projektipohja AzureRunMe

Vaikka Windows Azuren .NET SDK antaa projektipohjan, vaatii se vielä paljon .NET-osaamista ja työtä, ennen kuin saadaan Windows Azure suorittamaan palvelinohjelmaa. Rob Blackwellin kehittämä AzureRunMe-projektipohja on tarkoitettu kolmannen osapuolen ohjelmointikielien, kuten Javan, suorittamiseen Windows Azuren pilvipalveluissa. Sitä ovat käyttäneet monet harrastelijat, yritykset ja jopa Microsoft itse. Projektin saaminen toimintakuntoon vaatii vain muutamien asetustietojen muuttamista ja ajettavan sovelluksen .zip-pakettin

lataamista Windows Azure Storageen. AzureRunMe projektipohjan voi ladata osoitteesta https://github.com/robblackwell/AzureRunMe.

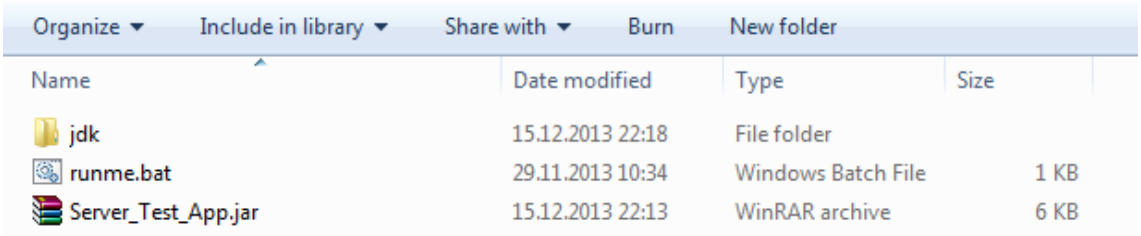
7.2.1 Palvelinohjelman paketointi ja lisääminen Storageen

Tarvittavien tiedostojen keräämiseksi yhteen paikkaan luotiin tietokoneelle kansio nimeltä Deployment. Palvelinohjelmaa varten tarvittiin kansioon kolme kokonaisuutta (kuva 13).

Ensiksi tarvitaan .jar-paketti palvelinohjelmasta. NetBeans IDE:ssä projektin buildaamisen jälkeen .jar-paketti ilmestyy projektikansion alle dist-nimiseen kansioon. Palvelinohjelman .jar-paketti kopioitiin Deployment-kansioon juureen.

Jotta palvelin pystyisi suorittamaan Javalla tehtyä .jar-pakettia, pitää palvelimella olla käytettävissä Java Development Kit (JDK). Asennettu JDK löytyy ..\Program File\Java\ -kansion alta, joka kopioidaan Deployment-kansion juureen. Deployment-kansioon kopioitu jdk-kansio nimettiin uudelleen ottamalla siitä versionumero pois. Tämä yksinkertaistaa myöhemmin käytettävää komentoriviä.

Lopuksi tarvitaan runme.bat-niminen tiedosto, joka sisältää komennot, jotka palvelin ajaa käynnistyttyään. Tiedosto voidaan luoda tekemällä uusi tekstitiedosto ja muuttamalla sen nimeä ja päätettä. Tiedoston sisältöä voidaan muokata millä tahansa tekstieditorilla. Palvelinohjelmaa varten riittää, että se sisältää seuraavan tekstin: jdk\jre\bin\java -jar XML_Test_App.jar. Komento käynnistää Java Runtime Enviromentin avulla .jar-pakatun tiedoston. Huomioitavaa on, että olen ottanut omasta jdk-kansiosta versionumeron pois.



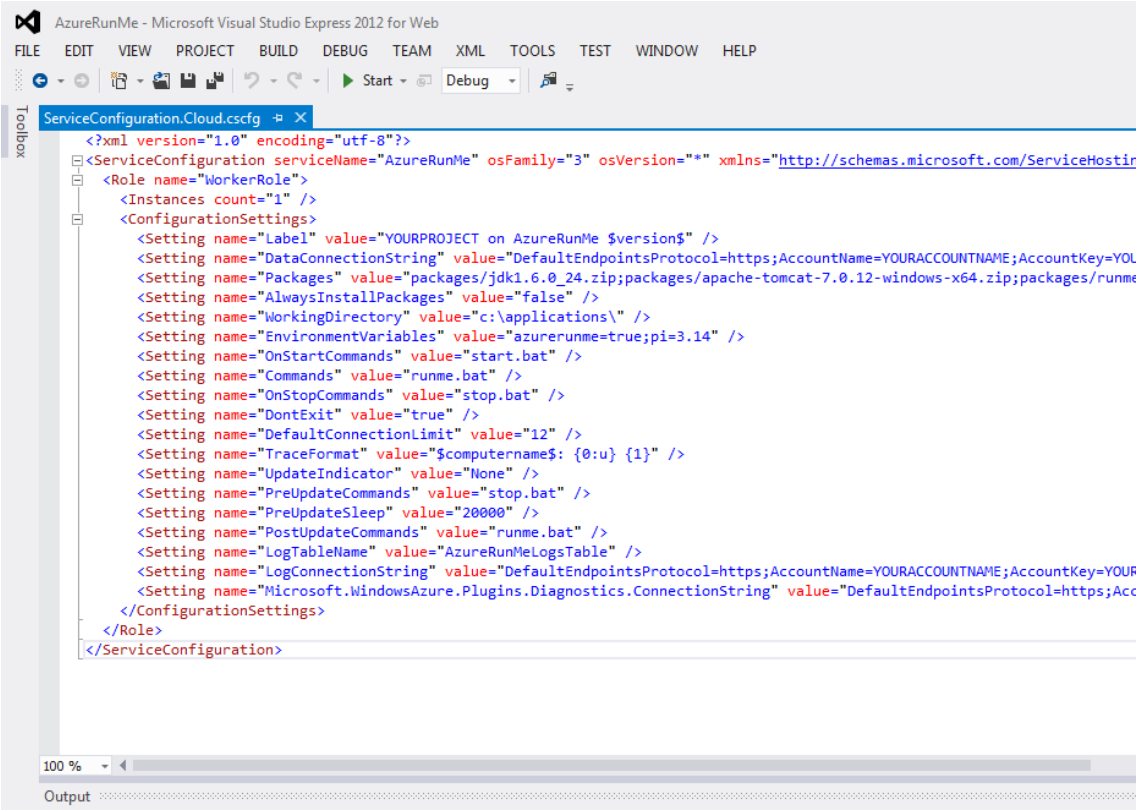
KUVA 13. Lopullinen Deployment-kansion rakenne ennen paketointia

AzureRunMe ottaa projektin ulkopuoliset tiedostot käyttöönsä Windows Azuren Storageen tallennetuista .zip-paketeista. Pakataan jdk-kansio jdk.zip-paketiksi sekä erilliseen runme.zip-pakettiin .jar- ja .bat-tiedosto. JDK-paketti pidetään erillään kokonsa takia, jotta sitä ei tarvitse lähettää Storageen joka kerta, kun tehdään uusi versio palvelinohjelmasta.

Tämän jälkeen tiedostot ovat valmiita siirrettäviksi Windows Azuren Storageen. Käytetään tähän aikaisemmin esille tullutta Azure Storage Explorer -ohjelmaa, jonka alasvetoikkunassa pitäisi olla muistissa yhteysasetukset, jotka lisättiin luvussa 5.3. Container-kohdassa pitäisi olla jo aikaisemmin luvussa 5.3 luotu kansio nimeltä packages, joka voidaan avata tuplaklikkaamalla sitä. Työkalupalkissa on Blob-niminen kohta ja siellä Upload-painike, jonka avulla .zip-paketit lisätään Storageen. Koska JDK on melkein 100 MB:n kokoinen, menee lataamisessa storageen 2–10 minuuttia.

7.2.2 AzureRunMen muokkaus

AzureRunMe on melkein valmis projekti paketoitavaksi, kun sen aukaisee Microsoft Visual Studiossa. Edellä tehtyä palvelinohjelmaa varten tarvitsee muokata vain ServiceConfiguration.Cloud.cscfg-nimistä tiedostoa (kuva 14).



KUVA 14. ServiceConfiguration.Cloud.cscfg-tiedosto aukaistuna Visual Studi-ossa

Ensimmäisenä pitää määritellä kolmeen kohtaan yhteysasetukset Windows Azure Storage -tiliä varten. Nämä kolme kohtaa ovat DataConnectionString, LogConnectionString ja Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString. Jokaisessa näissä on kohdat YOURACCOUNTNAME ja YOURACCOUNTKEY, jotka korvataan Windows Azure Portaalista löytyvillä tiedoilla. Näihin haetaan samat tiedot, joita tarvittiin Windows Storage Explore -sovellusta käyttäessä.

Windows Azure Portaalista valitaan vasemmalta Storage ja esiin tulevasta listasta haluttu Storage. Tämän jälkeen klikataan ruudun alareunasta Manage Access Key, josta löytyy Storage Account Name- ja Primary Access Key -kohdat, joilla korvataan YOURACCOUNTNAME- ja YOURACCOUNTKEY-kohdat.

Lopuksi editoidaan ServiceConfiguration.Cloud.cscfg-tiedostosta Packages-kohtaa, joka kertoo AzureRunMeelle, missä polussa ajettavat paketit ovat

Storagen sisässä. Polkuun tulee containerin nimi sekä .zip-paketin nimi. Eri paketit erotetaan ;-merkillä. Pakettien pitää olla .zip-muodossa, koska AzureRunMe käyttää niitä siinä muodossa. Poistetaan valuesta kaikki lainausmerkkien välistä ja lisätään siihen seuraava teksti: packages/jdk.zip;packages/runme.zip.

Cloud Services -olio purkaa käynnistyttyään .zip-paketit itselleen järjestyksessä vasemmalta oikealle. ServiceConfiguration.Cloud.cscfg-tiedostossa on valmiiksi määritelty Commands-kohtaan runme.bat. Tämä tiedosto ajetaan .zip-pakettien purkamisen jälkeen. Tässä projektissa runme.bat-tiedosto pakattiin runme.zip-pakettiin luvussa 7.2.1.

7.2.3 AzureRunMe:n buildaus

Palvelupakettitiedoston (.cspkg) ja konfiguraatiotiedoston (.cscfg) luomiseksi ei tavallinen buildaaminen riitä Visual Studiossa. Menemällä ServiceConfiguration.Cloud.cscfg-tiedoston juureen AzureRunMe-kohtaan ja klikkaamalla oikeaa hiiren painiketta aukeaa valikko. Valikon sisältä lötyy Package-painike, jota klikkaamalla aukeaa ponnahdusikkuna.

Ponnahdusikkunan oletusarvojen pitäisi olla oikeat. Oletusarvoina Service configuration -kohdassa on Cloud, Build configuration -kohdassa Release ja lisävaihtoehtoja ei ole valittuina. Klikkaamalla Package-painiketta alkaa projektin paketointi. Jos ei tule virheilmoituksia, Visual Studio aukaisee paketoinnin jälkeen kansion, jonka sisällä ovat .cspkg- ja .cscfg-tiedostot. Mikäli kansio ei aukea automaattisesti, tiedostojen pitäisi löytyä projektin bin\Release\app.publish\-kansiosta.

7.3 Cloud Services -olion käynnistäminen

Palvelupakettitiedoston (.cspkg) ja konfiguraatiotiedoston (.cscfg) avulla voidaan käynnistää Windows Azure Cloud Services -olio. Käynnistys tehdään Windows Azuren Portaalin kautta. Valitsemalla vasemmalta Cloud Services -kohdan aukeaa lista luoduista Cloud Services -olioista. Klikkaamalla listasta Cloud Services -oliota päästään käsiksi sen toimintoihin.

Cloud Services -olio käynnistetään klikkaamalla New production deployment -painiketta avautuneesta sivusta. Tämä aukaisee ponnahdusikkunan, johon pitää syöttää Deployment label, palvelupakettitiedosto, konfiguraatiotiedosto ja muutama muu lisäasetus. Deployment label -kohdalla ei ole merkitystä tämän esimerkkisovelluksen toiminnan kannalta, joten siihen voi keksiä mitä vain. Klikkaamalla From Local -painiketta palvelupakettitiedosto- ja konfiguraatiotiedosto-kohdassa päästään selaamaan omaa tietokonetta ja hakemaan kyseiset tiedostot.

Ponnahdusikkunan lopusta löytyy vielä muutama lisäasetus. Lisäasetusten oletusarvot ovat muuten oikeat, mutta valitaan päälle vaihtoehto Deploy even if one or more roles contain a single instance. Tämä tarkoittaa, ettei vaadita Windows Azuren kehotettua sääntöä. Säännöissä määritellään, että jokaista workeriä tulisi olla kaksinverroin tarvittava määrä. Tämän avulla Windows Azure voi taata 99,95 %:n toimivuuden ohjelman suoritukselle. AzureRunMe-projektipohjassa on määritelty vain yksi worker ja sitä ei ole muutettu esimerkkisovellusta varten. Cloud Services -olio alkaa käynnistyä painamalla check-merkkiä ponnahdusikkunan alareunasta. Tässä voi kestää 5–10 minuuttia.

Kun halutaan ottaa käyttöön uusi versio palvelinohjelmasta, pysäytetään Cloud Services -olion suoritus, korvataan vanha runme.zip-tiedosto Storagesta uudella ja lopuksi käynnistetään Cloud Services -olio uudelleen. Tästä tulee 5–10 minuutin katkos palvelimeen, jolloin palvelinohjelmaa ei suoriteta.

8 LOPPUSANAT

Työn tarkoituksena oli selvittää tilaajalle pilvipalveluiden määritelmä, Windows Azuren perusteet ja opastaa tekemään yksinkertainen esimerkkisovellus, joka käyttää Windows Azuren komponentteja hyödyksi. Tein dokumentista mahdollisimman yksityiskohtaisen ja helposti ymmärrettävän, jotta se toimisi kaikille lukijoille pohjana pilvipalveluihin ja Windows Azuren käyttöön.

Olen neljättä vuotta opiskellut tietotekniikan koulutusohjelmassa ohjelmistokehitystä Oulun ammattikorkeakoulun tekniikan yksikössä. Opintosuunnitelmaani ei kuulunut lainkaan internet-sovellusten kehitystä, joten lähes kaikki tässä dokumentissa käydyt asiat olivat myös itselleni uutta tätä dokumenttia kirjoittaessani. Opiskeluni aikana käytiin perusteiden kurssi tietokannoista ja lyhyesti socketin käytön perusteet toisella kurssilla. Näin opintotyön aiheen, johon liittyi pilvipalvelut ja Windows Azure, mahdollisuutena haastaa itseäni ja oppia uutta. En ole suinkaan vielä pilvipalveluiden asiantuntija tämän opinnäyteyön jälkeen, mutta minulla on kuitenkin hyvä pohja perusteista, joista voin jatkokehittyä pitemmälle.

Windows Azure antaa uuden ja ketterän tavan kehittää asiakas-palvelinsovelluksia. Lisäksi se madaltaa kynnystä uusille ja pienille yrityksille, koska niiden ei tarvitse ostaa omaa palvelinlaitteistoa ja ylläpitää sitä. Pivlipalvelut ovat vielä nopeasti kehittyvä teknologia, joten on mahdotonta ennustaa, mitä se tulee tarjoamaan ja miten sovellustenkehittäjät tulevat innovoimaan sen avulla tulevaisuudessa. Windows Azurella on kilpailijoita, kuten Google ja Amazon, mutta tässä opinnäytetyössä ei oteta kantaa palvelujen eroihin.

LÄHTEET

1. Griffith, Eric 2013. What is Cloud Computing? PC Magazine. Saatavissa: http://www.pcmag.com/article2/0,2817,2372163,00.asp. Hakupäivä 10.12.2013.

2. Rountree, Derrick - Castrillo, Ileana 2013. The Basics of Cloud Computing. Syngress. Saatavissa: http://proquest.safaribooksonline.com/book/information-technology-and-software-development/9780124059320. Hakupäivä 26.1.2014.

3. IaaS, PaaS, SaaS (Explained and Compared). Apprenda. Saatavissa: http://apprenda.com/library/paas/iaas-paas-saas-explained-compared/. Hakupäivä 10.12.2013

4. Redkar, Tejaswi - Guidici, Tony 2011. Windows Azure Platform. Apress. Saatavissa: http://proquest.safaribooksonline.com/book/web-development/9781430235637. Hakupäivä 26.1.2014.

5. Inside the Platform: Windows Azure. Microsoft. Saatavissa: http://msdn.microsoft.com/en-gb/ee514245.aspx. Hakupäivä 10.12.2013

6. Tulloch, Mitch 2013. Introducing Windows Azure for IT Professionals. Microsoft Press. Saatavissa: http://proquest.safaribooksonline.com/book/web-development/9780735682863. Hakupäivä 26.1.2014.

7. Chauhan, Shailendra 2013. Understanding Components of Windows Azure. Saatavissa: http://www.dotnet-tricks.com/Tutorial/windowsazure/3L0D011013-Undestanding-Components-of-Windows-Azure.html. Hakupäivä 10.12.2013.

8. Introducing Windows Azure. Microsoft. Saatavissa: http://www.windowsazure.com/en-us/develop/net/fundamentals/intro-to-windows-azure/. Hakupäivä 10.12.2013.

9. Understanding Client-Server Application. National Instruments. Saatavissa: http://www.ni.com/white-paper/4431/en/. Hakupäivä 10.12.2013.

```
package server_test_app;


import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.*;
import java.sql.*;

public class Server_Test_App implements Runnable
{
		private ServerSocket serverSocket;
		Socket socket = null;
		ObjectOutputStream outStream;
		ObjectInputStream inStream;
		String messageString;
		Connection connection;
		ResultSet result;

public void Server_Test_App()
{}

public static void main(String[] args)
{
		Server_Test_App server = new Server_Test_App();

		while(true)
		{server.run();}
}

public void run()
{
		try
		{
		 //1. creating server socket
		serverSocket = new ServerSocket(8080);
```

```
//2. Waiting for connection
System.out.println("Waiting for connection")
socket = serverSocket.accept();

System.out.println("Connection received from " + sock-
et.getInetAddress().getHostName());

//3. Initialize Input and Output streams
outStream = new ObjectOutputStream(socket.getOutputStream());
outStream.flush();
 inStream = new ObjectInputStream(socket.getInputStream());
sendMessage("Connection successful");

 //4. Do-while loop for communication
do
{

try
{
messageString = (String)inStream.readObject();
System.out.println(messageString);

if (messageString.equals("Hello my server"))
{sendMessage("Hello my client");}
else if (messageString.equals("Could you kindly get me information from data
base!"))
{

 try
 {
connection = getConnection();
getDataFromTable();
int i = 0;
String s = "";

while (result.next())
{
s += "SQL data row Response " + i + ": " + result.getString("response")+"\n";
i++;
}
```

```
                    sendMessage(s);
                    }
                    catch (SQLException sqlE)
                    {sendMessage("ERROR: " + sqlE.toString());}}

                    System.out.println("client>" + messageString);

                    if (messageString.equals("bye"))
                    {sendMessage("bye");}
                    }
                    catch(ClassNotFoundException classnot)
                    {System.err.println("Data received in unknown format");}
                    }while(!messageString.equals("bye"));
                    }
                    catch(IOException ioException)
                    {ioException.printStackTrace();}
                    finally
                    {
                    //5: Closing connection
                    try{
                    inStream.close();
                    outStream.close();
                    serverSocket.close();
                    }
                    catch(IOException ioException)
                    {ioException.printStackTrace();}
                    }
}

//Send messages to client
void sendMessage(String msgString)
{
                    try{
                    outStream.writeObject(msgString);
                    outStream.flush();
                    System.out.println("server>" + msgString);
                    }
                    catch(IOException ioException)
                    {ioException.printStackTrace();}
}
```

```
//Connect to database
private Connection getConnection() throws SQLException
{
		Connection conn = null;

		try
		{
		Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver").newInstance();

		conn = DriverMan		ag-
er.getConnection("jdbc:sqlserver://ipmxwhdnbo.database.windows.net:1433;d
		atabase=DATABASE NAME HERE;"
		+ "user=DATABASE USERNAME HERE@ipmxwhdnbo;password
		={DATABASE PASSWORD HERE};");

		System.out.println("Connected to database");

		}
		catch (Exception e)
		{System.out.println(e.toString());}

		return conn;
}

//Create SQL query, execute and get result
private void getDataFromTable() throws SQLException
{
		Statement stmt = null;
		stmt = connection.createStatement();
		result = stmt.executeQuery("select * from TestiTable1");
}

}
```

```
package client_client_app;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

public class GUI extends javax.swing.JFrame
{
Socket clientSocket;
ObjectOutputStream outStream;
ObjectInputStream inStream;
String messageString;

public static void main(String args[])
{
                new GUI().setVisible(true);
}

public GUI()
{
                initComponents();
}

/**
* This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code. The content of this method is always
* regenerated by the Form Editor.
*/
    @SuppressWarnings("unchecked")
/*NETBEANS FORM EDITOR AUTO-GENERATED CODE HERE
*REMOVED BECAUSE NOT RE-USABLE IN NEW PROJECT
*/
```

```
private void connectButtonMouseClicked(java.awt.event.MouseEvent evt)
{
                    try
                    {
                     clientSocket = new Socket("veeraytestcloudservice.cloudapp.net ", 8080);

                    if (clientSocket.isConnected())
                    {
                    //2. get Input and Output streams
                    outStream = new ObjectOutputStream(clientSocket.getOutputStream());
                    outStream.flush();
                    inStream = new ObjectInputStream(clientSocket.getInputStream());
                    receiveMessage();
                    }
                    else
                    {textAreaOutput.setText(textAreaOutput.getText() + "\nConnection failed");}
                    }
                    catch (Exception e)
                    {
                    textAreaOutput.append("Error: " + e.toString());
                    }
}

private void greetButtonMouseClicked(java.awt.event.MouseEvent evt)
{sendMessage("Hello my server");}

private void requestDataButtonMouseClicked(java.awt.event.MouseEvent evt)
{sendMessage("Could you kindly get me information from database!"); }

private void disconnectButtonMouseClicked(java.awt.event.MouseEvent evt)
{
                    sendMessage("bye");

                    try
                    {
                    inStream.close();
                    outStream.close();
                    clientSocket.close();
                    }
```

```
                catch(IOException ioException)
                {
                ioException.printStackTrace();
                }
}

void sendMessage(String msg)
{
                try
                {
                outStream.writeObject(msg);
                outStream.flush();
                textAreaOutput.setText(textAreaOutput.getText() + "\nclient>" + msg);
                }
                catch(IOException ioException)
                {textAreaOutput.append("Error: " + ioException.toString());}

                receiveMessage();
}

void receiveMessage()
{
                messageString = "";
                do
                {
                try
                {
                messageString = (String)inStream.readObject();
                textAreaOutput.setText(textAreaOutput.getText() + "\nserver>" + messag
                eString);
                }
                catch(Exception e){
                textAreaOutput.append("Error: " + e.toString());
                }
                }while (messageString.equals(""));
}

// Variables declaration - do not modify
private javax.swing.JButton connectButton;
private javax.swing.JButton disconnectButton;
```

```
private javax.swing.JButton greetButton;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JButton requestDataButton;
private javax.swing.JTextArea textAreaOutput;
// End of variables declaration
}
```