

TuulariApi & Testaustyökalut

ASP.NET & C# Harjoitustyö

Jussi Heikkinen
Niko Jokipalo

12 2015

Ohjelmistotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

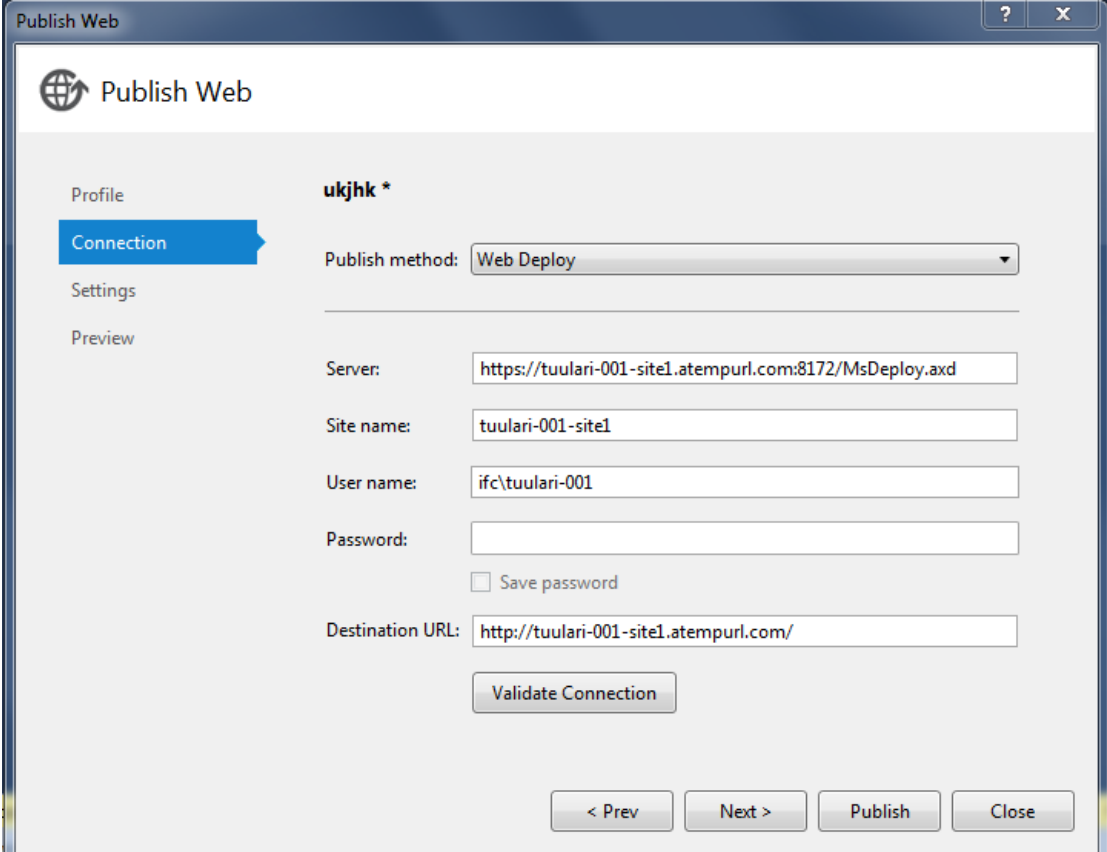


1 Asennus

1.1 API

Asensimme aspin toimimaan smarterASP.net palvelussa, josta saimme ilmaista serveritilaa 30 päiväksi. Api asennetaan palvelimelle siten että Visual Studiassa painetaan ylävalikosta **build-> publish + projektin nimi**

Julkaisutyökaluun pitää luoda uusi julkaisu profiili, valitaan Custom vaihtoehto, ja annetaan sille nimi. Connection välilehti aukeaa, jonne syötetään palveluntarjoajan antamat tiedot kenttiin: Server, Site name, User name, Password, sekä Destination URL. Yhteyttä voidaan nyt testata painamalla Validate connection. Seuraavaksi avataan Settings välilehti, jossa valitaan database connection kenttää klikkamalla avautuvasta kentästä projektissa käytetty connection string eli tietokantayhteys. Nyt projekti voidaan julkaista klikkaamalla Connection välilehdeltä löytyvää Publish nappia.



Publish Web

Profile

Connection

Settings

Preview

ukjhh *

Publish method: Web Deploy

Server: https://tuulari-001-site1.atempurl.com:8172/MsDeploy.axd

Site name: tuulari-001-site1

User name: ifc\tuulari-001

Password:

☐ Save password

Destination URL: http://tuulari-001-site1.atempurl.com/

Validate Connection

< Prev Next > Publish Close

Kyseisen toimenpiteen jälkeen palvelimella olevan nettisivun juurikansioon ilmestyy seuraavat kansiot, ja ohjelman pitäisi olla ajettavissa selaimella.



<input type="checkbox"/>	Name	Ext	Size	Modify Date
	.. [current directory=h:\root\home\tuulari-001\www\site1\]			
<input type="checkbox"/>	bin	< dir >		11/30/2015 10:05:00 AM
<input type="checkbox"/>	Models	< dir >		11/30/2015 10:05:00 AM
<input type="checkbox"/>	Views	< dir >		11/30/2015 10:05:00 AM
<input type="checkbox"/>	Global.asax	asax	0 kb	11/28/2015 12:40:21 AM
<input type="checkbox"/>	packages.config	config	1 kb	11/28/2015 12:40:23 AM
<input type="checkbox"/>	Web.config	config	5 kb	11/30/2015 12:04:29 PM

2 Konfiguroitavat asiat

SmarterASP.NET palvelusta on kytkettävä **“VS Web Deploy”** toiminto päälle. Käytimme palvelimelle konfiguroitavaa detail error palvelua, jotta virheilmoituksesta ymmärtää virheen syyn, joten sekin on hyvä kytkeä päälle.

Api pakotetaan tulostamaan kaiken datan JSON formaatissa lisäämällä App_Start hakemistossa sijaitsevaan WebApiConfig.cs tiedostoon seuraavat koodirivit Register-metodiin.

```
var json = config.Formatters.JsonFormatter;
json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
config.Formatters.Remove(config.Formatters.XmlFormatter);
```

Ennen kuin projekti voidaan julkaista täytyy web.config tiedostoon vaihtaa “compilerOptions=“/langversion:6” -> “compilerOptions=“/langversion:5”, sekä APS.NET versio täytyy muuttaa vanhempaan 4.5 versioon.

Käytimme SmarterASP:in omaa tietokanta palvelua, jonne teimme MSSQL tietokannan. Jos haluaa käyttää omaa vaihtoehtoista tietokantaa, niin TuulariApi projektin web.config tiedostoon täytyy käydä vaihtamassa oman tietokantapalvelimen yhteystunniste (ConnectionString).

käytetyt .NET Frameworkin ulkopuoliset kirjastot tai kilkkeet
Newtonsoft.Json -Version 7.0.1, MSSQL, Cryptography.BCrypt 1.0.0

3 Tietoa ohjelmasta

3.1 REST API (asp.net)

REST API:mme tarkoitus oli toimittaa **MS SQL**-tietokannasta dataa API:n endpoint URL-osoitteisiin, joista Android-äppimme tekee kyselyitä. MVC-mallilla toteutettu rajapinta palauttaa kyselystä riippuvaista dataa JSON-muodossa ja yhdistää HTTP-



protokollan metodit *POST*, *GET*, *PUT* ja *DELETE* tietokannoissa yleisesti käytettyihin *CREATE*, *READ*, *UPDATE* ja *DELETE* -kyselyihin. Lisäksi palautetaan HTTP:n status-numero (200, 201, 401, 404 jne.), joista kyselyn tilaa voi äppis myös tarkkailla.

Rajapinnan avulla voi tehdä kirjautumisen sähköpostiosoitteella ja salasanalla, joka palauttaa onnistuessaan session tokenin äppikselle. Tällä viestitään jatkossa pyyntöjen välillä, ettei kirjautumistietoja tarvitse jakaa joka pyynnöllä.

Lisäksi rajapinta voi käsitellä yleisemmin käyttäjien (Users) dataa ja äppiksessä käytettäviä asetuksia (UserData), hakea ja tallentaa tapahtumista (Events) ja tapahtumailmoituksista (Announcement) mm. tapahtuma-ajan ja osallistujat (Attendance). Ilmoituksesta jokainen käyttäjä voi tehdä oman tapahtumansa, jonka käynnistyttyä äppis lähettää sijaintia (Location) ja käyttäjän niin halutessa tulostaa suurpiirteisen sijainnin myös muiden samaan ilmoitukseen liittyneiden sijainnit kartalla. Käyttäjät voivat myös kommentoida ja ehdottaa (Comment) ilmoitusta, jotka näytetään kaikille muille.



```

← → C ■ http://tuulari-001-site1.atempurl.com/api/Users/1
Collapse all Expand all

{
  $id: "1",
  - Announcements: (0) [
  ],
  - Attendances: (0) [
  ],
  - AuthTokens: (0) [
  ],
  - Comments: (0) [
  ],
  - Events: (0) [
  ],
  - UserDatas: (1) [
    - {
      $id: "2",
      - User: {
        $ref: "1"
      },
      Id: 3,
      UserId: 1,
      Gender: null,
      Dob: null,
      Visibility: 1,
      Theme: 1,
      Sync: true,
      SyncFrequency: null,
      Interests: null,
      MarkerHue: null
    }
  ],
  Id: 1,
  Email: "matti@mattijateppo.fi",
  Password: "$1$c5GtIXk$VehGc5wcGwIc0diLwXAaC.",
  Nickname: "MäOomMatti"
}

```

Kaikki edellä mainittu kulkee rajapinnan kautta ja tallennetaan kantoihin. Rajapinta palauttaa onnistumisesta/epäonnistumisesta ilmoituksen, jonka jälkeen äppis tekee halutut toimet, kuten piirtää markerit kartalle, lisää kommentit tekstikenttään jne.

Rajapinnasta löytyy myös toiminnot (action), jotka palauttavat tekijöiden tiedot sekä kaikki mahdolliset endpointit API:ssa.

Rajapinta on koodattu C#:lla, ja .NET:stä löytyvien MVC- ja MVC API -frameworkkien templatejen pohjalta sekä käyttäen Entity Framework -kirjastoa kannan yhdistämiseen. Jokaista tietokannan taulua vastaa Entity, jolla on ns. Data Model, joka heijastelee tietokannan taulua ja taulujen välisiä riippuvuuksia ja rajoitteita (*FK*, *constraints*). Entityjen avulla tietokantaan ei tarvitse käsin kirjoitella SQL-kyselyitä, vaan koodi generoidaan valmiiksi.



MVC- ja API-templatejen mukana kaikki tieto on jaettu Controllereihin, Näkymiin ja Modeleihin. Näkymiä meillä ei periaatteessa ole erikseen, vaan kirjasto palauttaa haun tuloksen suoraa JSON-tekstinä. Controllerimme vastaavat pääasiassa Entityille (tauluille) tehtäviä CRUD-toimintoja poislukien muutaman controllerin, joissa piti käyttää taulun näkymiä (View) tai muuten yhdistettyjä tai minimoituja kyselyitä.

API:a varten tarvitsimme IIS- tai muun ASP.NET'iä ja sen versio 4.5:ssa tulevia kirjastoja tukevan serverin. Päädyimme smarterASP.NET-sivustoon.

Alunperin tarkoitus oli käyttää MySQL:ää, mutta ajureiden ym. kanssa kikkailu sai meidät vaihtamaan MSSQL:ään, jonka tuki on (luonnollisesti) valmiina muissa MS-ketjun työkaluissa. SQL:n rakenne ei loppupeleissä eroa niin kovasti muista murteista.

Rajapinnassa ei ole tarkistuksia, onko pyynnön tekijällä oikeudet tehdä jotain tiettyä toimintoa, esim. poistaa käyttäjää. Kun käyttäjä on kirjautunut oikeilla tunnuksilla ja käyttää saamaansa istuntoavainta, kaikkia pyyntöjä noudatetaan. Jos tokenia ei löydy, rajapinta palauttaa *401 Unauthorized*. Rajapinnassa ei myöskään ole tarkistusta, onko edellisen identtisen pyynnön jälkeen mikään muuttunut, jolloin voisi palauttaa *304 Not Modified*, vaan aina palautetaan data.

4 Testing tool (exe ja asp.net)

Työkalusta on kaksi eri versiota: työpöytäympäristössä ajettava ja verkossa käytettävä työkalu, jolla pystyy testaamaan joko tekemäämme tai mitä tahansa muuta APIa; HTTP:n POST, GET, PUT ja DELETE -metodeilla pystytään luomaan, hakemaan tai poistamaan kannasta tietoa ja tulostamaan työkalussa käyttäjälle tuloste/palaute.

Työkalun käyttöliittymässä saa valita kutsuttavan URL-osoitteen API:n, käytettävän metodin, syöttää itse key-value-muodossa parametrejä kutsuun ja tarkastelemaan palautettua dataa. Työkalumme näyttävät mitä tahansa tekstimuodossa tulevaa vastauksen dataa, mutta muotoilevat ainoastaan JSON-muotoista, koska tekemämme API palauttaa aina JSON-dataa.

Tavoitteena työkaluille oli saada toimivat testausohjelmat omalle APIllemme. Mitään suunniteltuja isompia osia ei jäänyt toteuttamatta eikä myöskään toteutettu mitään ylimääräisiä ominaisuuksia.

Ajettava ohjelma toimii Windows-ympäristössä, on ohjelmoitu C#-kielellä .NET-kirjaston (v. 4.5) komponentteja käyttäen. Web-työkalu on asp.NETillä, HTML:llä ja JavaScriptillä koodattu sivu, joka vaatii IIS- tai vastaavan, ASP.NET (v. 4.5), tukevan serverin.

Työkalut tarvitsevat internet-yhteyden, koska käytössä on HTTP-protokolla. Muita rajoituksia ei projekteilla ollut.



5 Kuvaruutukaappaukset tärkeimmistä käyttöliittymistä

MainWindow

https://httpbin.org/put

GET POST **PUT** DELETE

Lähetä

Lisää avain, arvo pareja

Avain	Arvo
arvo	pari

Lisää http otsakkeita

X-plaah	plaah

```
{
  "args": {},
  "data": "arvo=pari",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "9",
    "Host": "httpbin.org",
    "X-Plaah": "plaah"
  },
  "json": null,
  "origin": "195.148.26.20",
  "url": "https://httpbin.org/put"
}
```

ready

Application name Homepage/About Testing Tool

requested uri

POST GET **PUT** DELETE

Call API Reset form

Add new key-value pair

Response

Status **200 OK**

Response size: 0b

© 2015 Jussi Heikkinen and Niko Jokipalo

Ohjelman tiedostot/tietokannat

Tietokannasta on kuva dokumentin lopussa .
Tietokannan luontiskripti liiteenä, sekä Git repositorin wikissä.



JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

6 Tiedossa olevat ongelmat ja bugit sekä jatkokehitysidea

Ongelmia tai bugeja ei ole tullut vastaan. Projekti ja sen osaset toimivat perusvarmasti. Jos Android-äppiä lähdetään jatkokehittämään, myös API:a kehitetään mm. kirjoittamalla se kokonaan uudestaan jollain helpommin hostattavalla kielellä ja kevyemmällä kielellä/frameworkilla.

7 Opittua ja jatkossa opeteltavaa sekä haasteet

.NET-kirjastot, C# ja yleisperiaatteet ovat tulleet tutuiksi sekä VisualStudiossa olevien wizardien käyttö, kuten myös Entity Framework sekä ADO.NET. Haasteita oli alussa lähes kaikessa mitä lähdimme tekemään ja hakukonehakuja tuli tehtyä kovasti.

MySQL-tietokannasta ja Azure-ympäristöistä luopuminen tulivat eteen sekä sopivan hostauspalvelun löytäminen asp.net'lle ja tarvittavan .net-kirjaston version tukeminen olivat vaikeaa. Azurea ei voi kyllä opiskelijalisenssin perusteella suositella edes pahimmalle vihamiehelle. Palvelu oli äärimmäisen hidas, sivujen latautuminen kestää minuutteja, dreamspark lisenssiä mainostettiin sanalla ilmainen, mutta mikään muu kuin staattisten nettisivujen hostaaminen ei ollu ilmaista. Lisäksi visualstudio ei tue dreamspark lisenssiä, joten web appia ei voi julkaista ainakaan visual studiosta käsin. Haasteita aiheutti myös se, että modeleihin/ kantaan ei voinut tehdä muutoksia ilman että koko projekti hajoaisi. Tästä syystä projekti tuli tehtyä uudelleen n.10 kertaa.

8 Tekijät, vastuiden ja työmäärän jakautuminen sekä tekijöiden perusteltu ehdotus arvosanaksi

8.1 Tekijät

Ryhmään kuului Jussi Heikkinen, Niko Jokipalo.

Vastuut jakautuivat suunnitellun mukaisesti: molemmat tekivät omat testaus clientit. Api ja hostauspalvelut sumplittiin pääsääntöisesti yhdessä. Aikamääreet venyivät jonkin verran kun mm. työkalu- ja hostausasiat eivät sujuneet suunnitelmien mukaan. Lisäksi aikataulut venyivät kun ei tahtonut aika riittää kaikkiin koulutöihin samanaikaisesti.

Tuote toimii suunnitellusti, käyttää opintojakson sisältöjä ja oppimistavoitteet saavutettiin pääasiallisesti. Arvosanaksi ehdotamme 4.

9 Ajankäyttö



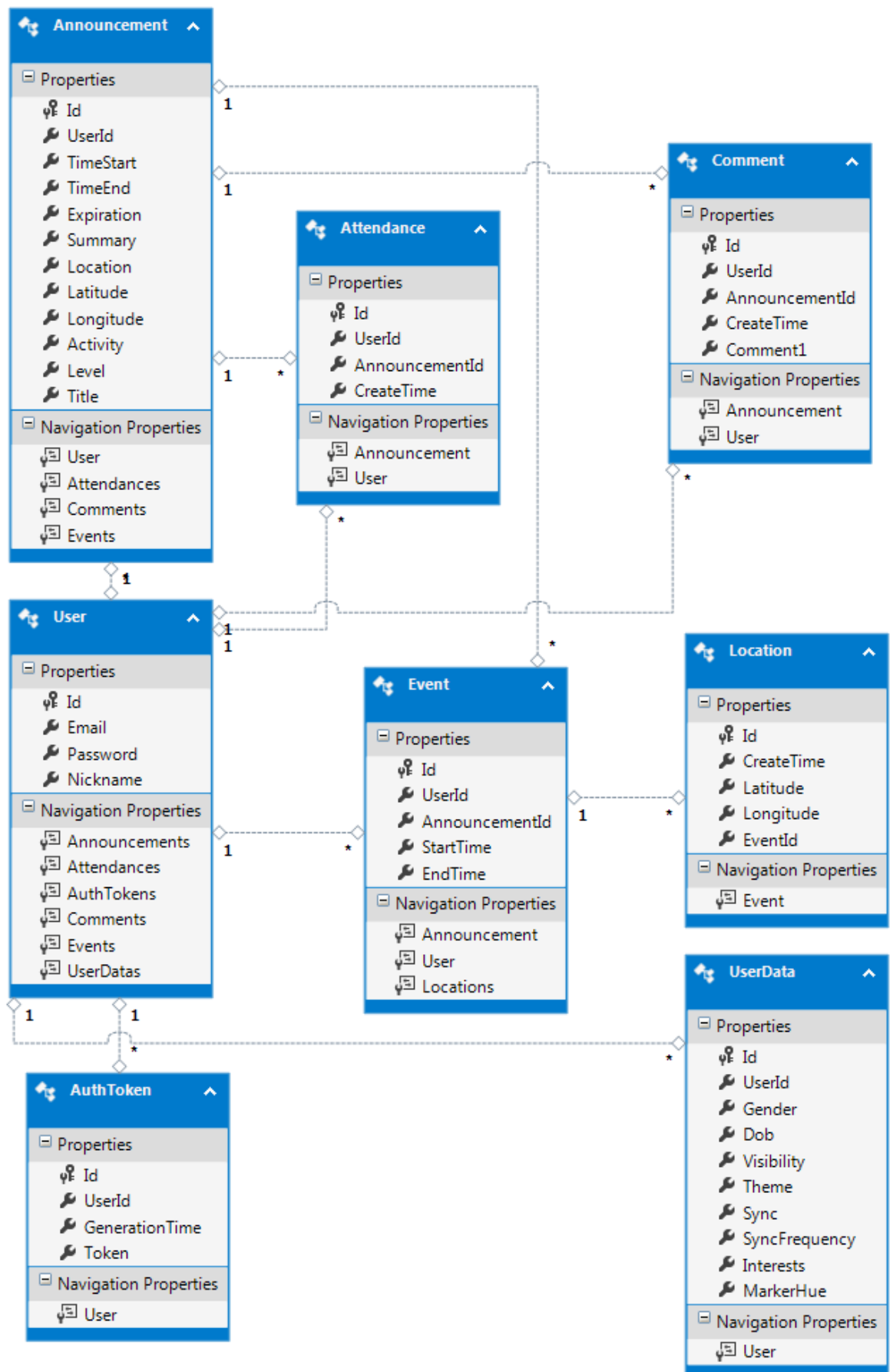
JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Tehtävät	Työmäärä (h)		Päiväys	Vastuuhenkilö(t)
Suunnittelu	5	6	vko 46	Molemmat
• Vaatimusmäärittely	2,5	4		
• Vuokaavio	0,5	0,5		
• Mockup	0,5	0,5		Jussi
• Tietokanta	1	2		Niko
• Käyttötapauskaavio	0,5	0,5		
Ympäristön pystytys			vko 46	Molemmat
• DB	1	2		
• Service (Azure?)	2 hukattua tuntia			
• SmarterASP	2	2		
• Hosting	3	1		
API:n controllerit			vko 46-47	Molemmat
• mock-api	5		vko 46	
• Controllerit kaikille modeleille	1	1	vko 47	Jussi
• Auth ja About		10		Niko
ADO.NET	5		vko 47	
Entity-luokat ja -modelit (EF)	5	5	vko 47	Jussi
WPF-käyttöliittymä	5	4	vko 48	Jussi
HTTP-custom headers	1	1	vko 49	Jussi
ASP.NET-käyttöliittymä	5	8	vko 48	Niko
.NET-bisneslogiikka	6	6	vko 47-48	Jussi
Viimeistelyä	3	2	vko 49	Molemmat



Työn palautus ja esittely			vko 49 (3.12.2015)	Molemmat
----------------------------------	--	--	-----------------------	----------







JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES