jamk.fi

Harjoitustyön raportti

Pasi Reini, YTC16S

Harjoitustyön raportti
IIO13200 ASP.NET -ohjelmointi, Esa Salmikangas
8.12.2016
Information technology

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

Sisältö

1 Johdanto 3

2 Ohjelman asennus 3

2.1 Asennuspaketin muodostaminen 4

2.2 Asennus palvelimelle 5

2.3 Connectionstrings 7

3 Tietoa ohjelmasta 8

3.1 Toteutetut toiminnalliset vaatimukset 8

3.2 Toteuttamatta jääneet toiminnalliset vaatimukset 9

4 Käyttöliittymä 9

5 Tietovarasto 10

5.1 Tietokantakaavio 11

5.2 Tietokantaskriptit 11

6 Tiedossa olevat ongelmat 12

7 Mitä ollaan opittu ja muuta pohdintaa 12

Lähteet 14

1 Johdanto

Tämä raportti on tehty IIO13200 ASP.NET –ohjelmointikurssin harjoitustyötä varten. Ohjelman arkkitehtuuri on esitelty aiemmin palautetussa määrittelydokumentissa (Reini 2016).

Raportissa ensimmäisessä luvussa käsitellään miten ohjelma saadaan asennettua ja mitä asioita on otettava huomioon asennuksessa. Listataan .NET Frameworkin ulkopuoliset kirjastot ja asennukseen liittyvät konfiguroitavat asiat.

Toisessa luvussa käsitellään ohjelman vaatimuksia ja toimintoja. Käydään läpi mitä asioita on toteutettu ja mitkä ovat jääneet keskeneräiseksi.

Kolmannessa luvussa esitellään ohjelman käyttöliittymän toiminnallisuutta ja avataan vähän käyttöä lyhyillä ohjeistuksella. Ohjelmasta esitellään myös kuvaruutukaappauksia.

Neljännessä luvussa käsitellään mitä tietovarastoa on hyödynnetty ohjelmassa ja esitellään tietokannasta kaavio. Lisäksi pohdintaa, että miten tietokantaa olisi kannattanut hyödyntää koodin generoimisessa.

Viidennessä luvussa käydään läpi tiedossa olevia ongelmia ja bugeja sekä pohditaan mahdollisia jatkokehitysideoita.

Kuudennessa luvussa kerrotaan mitä tekijä on oppinut harjoitustyötä tehdessä ja mitkä olivat suurimmat haasteet projektin kuluessa. Lisäksi mietitään, että mitä kannattaisi jatkotutkia.

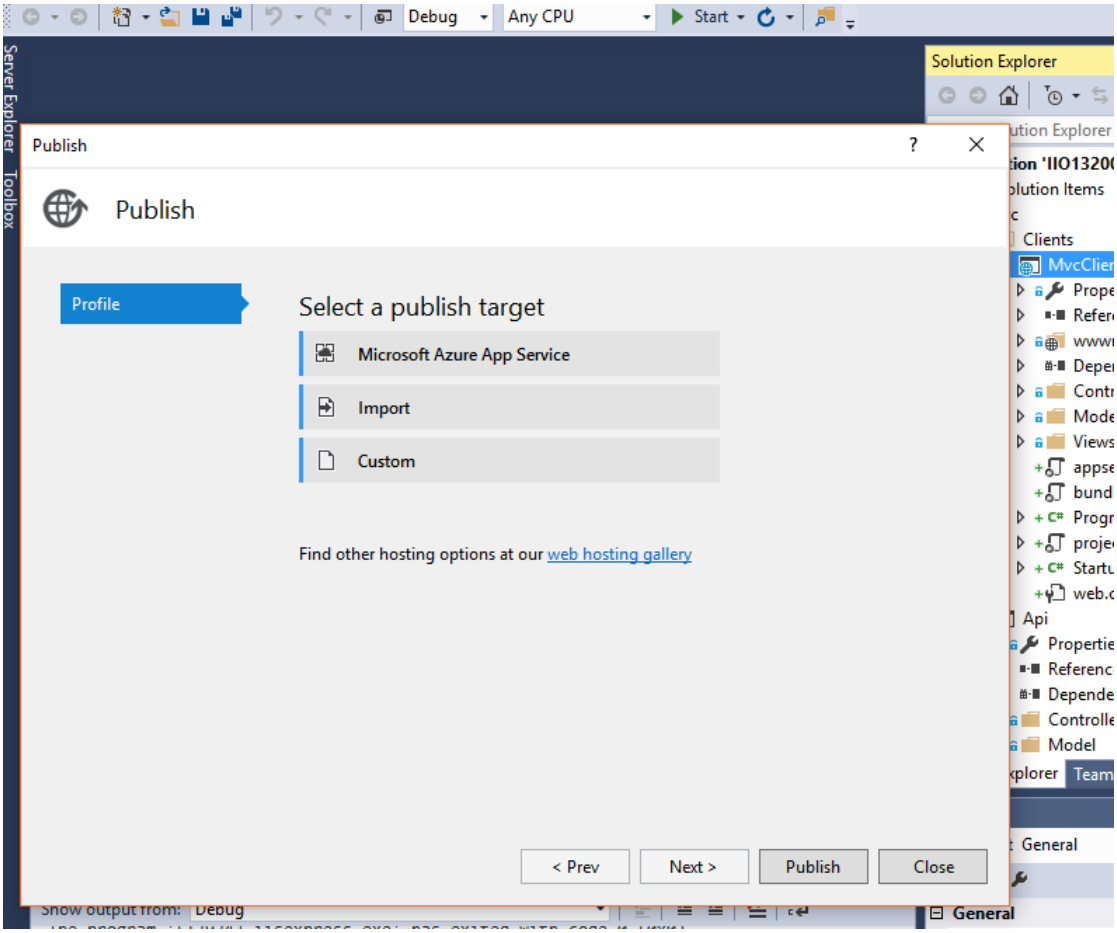
Pohdinta luvussa esitellään tekijät, vastuut ja työmäärän toteutuminen sekä ehdotus työn mahdolliseksi arvosanaksi.

2 Ohjelman asennus

Tässä luvussa on tarkoitus kuvata ohjelman asennus IIS:lle. Harjoitustyön määrittelyssä on kerrottu arkkitehtuurin kannalta tärkeimmät osa-alueet ja kirjastot (Reini 2016).

2.1 Asennuspaketin muodostaminen

Ohjelman asennus kannattaa aloittaa visual studiosta painamalla esimerkiksi web-projektin päällä hiiren oikeaa ja sieltä publish-toimintoa. Tämän jälkeen aukeaa kuvan yksi mukainen ikkuna.



Kuva1: Visual Studion Publish-toiminto

Edellä mainitusta toiminnosta voi valita, että asennetaanko ohjelma toimimaan esimerkiksi Microsoftin Azure App Service vai tehdäänkö custom-asennus esimerkiksi omalle palvelimelle. Tässä raportissa käsitellään vain custom-asennusta ja siihen liittyviä vaiheita.

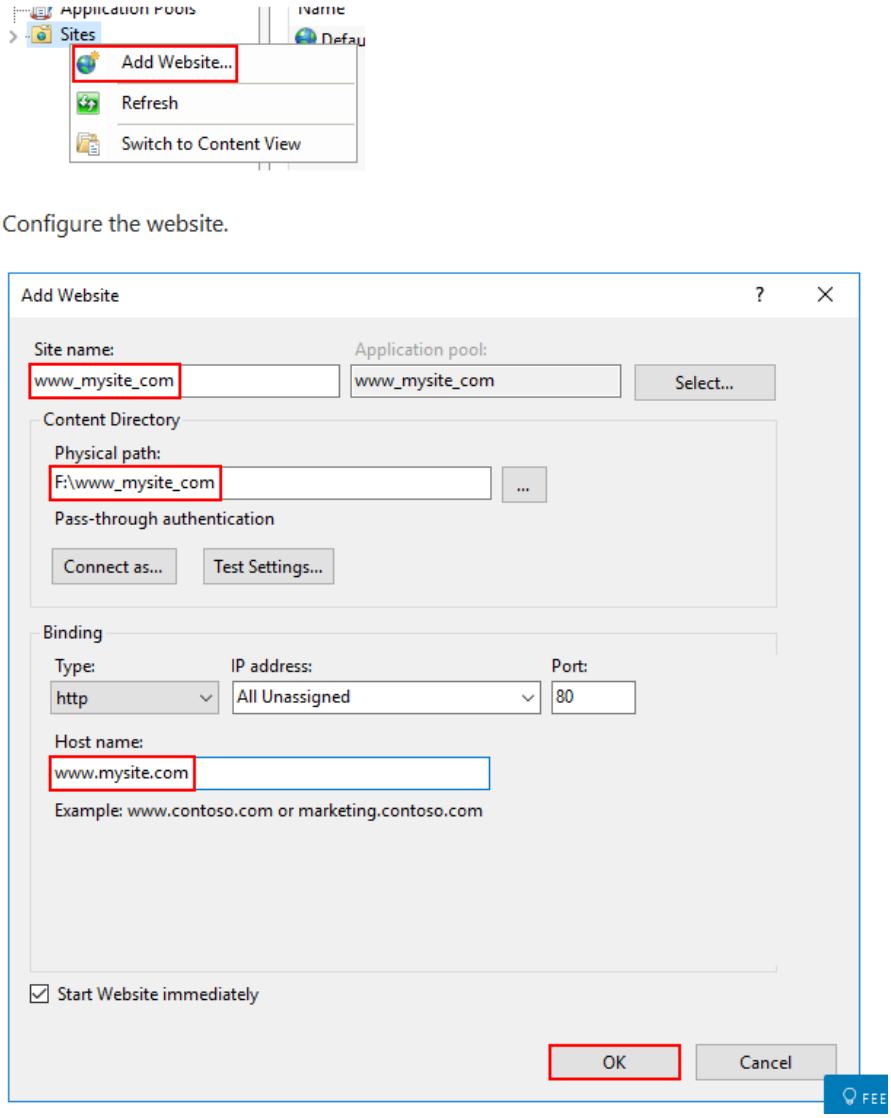
Valittaessa custom-asennus Visual Studio kysyy asennukselle profiilia ja siihen voi antaa esimerkiksi ohjelman nimen. Seuraavaksi publish-toiminto kysyy julkaisun tyyppiä ja siihen valitaan file system. Koodien kääntämiselle voi myös valita hakemiston, jonne esimerkiksi konfiguraatiotiedostot ja dll:t muodostetaan. Tässä työssä publish-toiminto on tehtävä mvc, api ja identityserver –projekteille. Kannattaa

tietenkin kääntää projektit omiin hakemistoihin, kun ne tullaan asentamaan iis:lle omina web-siteinaan. Hakemistot ja niiden sisältävät tiedostot voidaan viela zipata yhteen pakettiin, mikä siirretään palvelimelle ja puretaan sopivaan hakemistoon. Hakemistorakenteen voi tarkastaa Microsoftin tekemästä ohjeesta (Microsoft 2016b).

2.2 Asennus palvelimelle

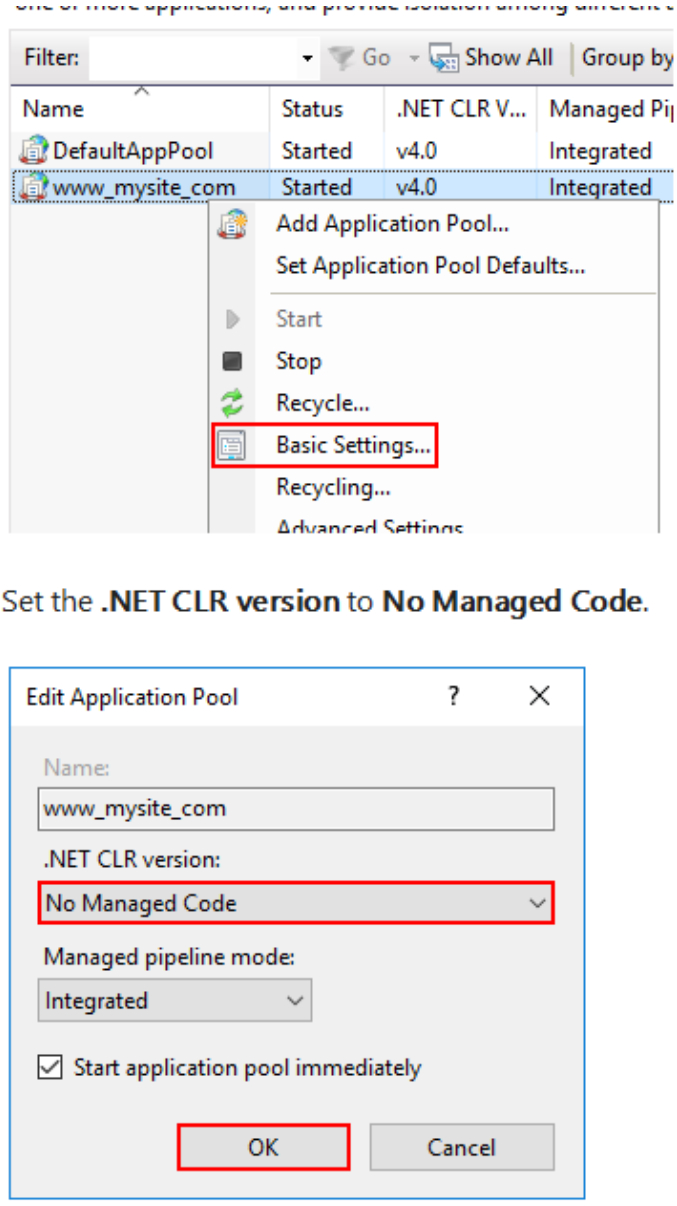
Raportissa oletetaan, että IIS on jo konfiguroitu valmiiksi asennettavaan ympäristöön. Mikäli näin ei ole, niin kannattaa lukea Microsoftin tekemä ohje, joka on esitelty tämän raportin lähdeluettelossa (Microsoft 2016c).

Seuraavaksi konfiguroidaan website toimimaan IIS:n päällä. Se tapahtuu siten, että IIS Managerissa luodaan uusi website, jolle annetaan nimi ja polku hakemistoon serverilla, jonne on laitettu Visual Studion Publish-toiminolla käännetyt tiedostot. Lisäksi määritellään alapuolella olean kuvan kaksi mukaisesti asetukset. (Microsoft 2016c.)



Kuva 2: Websiten asetukset

Sitten asetetaan websiten application pool kuntoon. Sen voi tehdä kuvan kolme mukaisesti, jonka jälkeen sovellusta voi savutestata, että lähteekö käyntiin (Microsoft 2016c).



Kuva 3: Application poolin määrittäminen

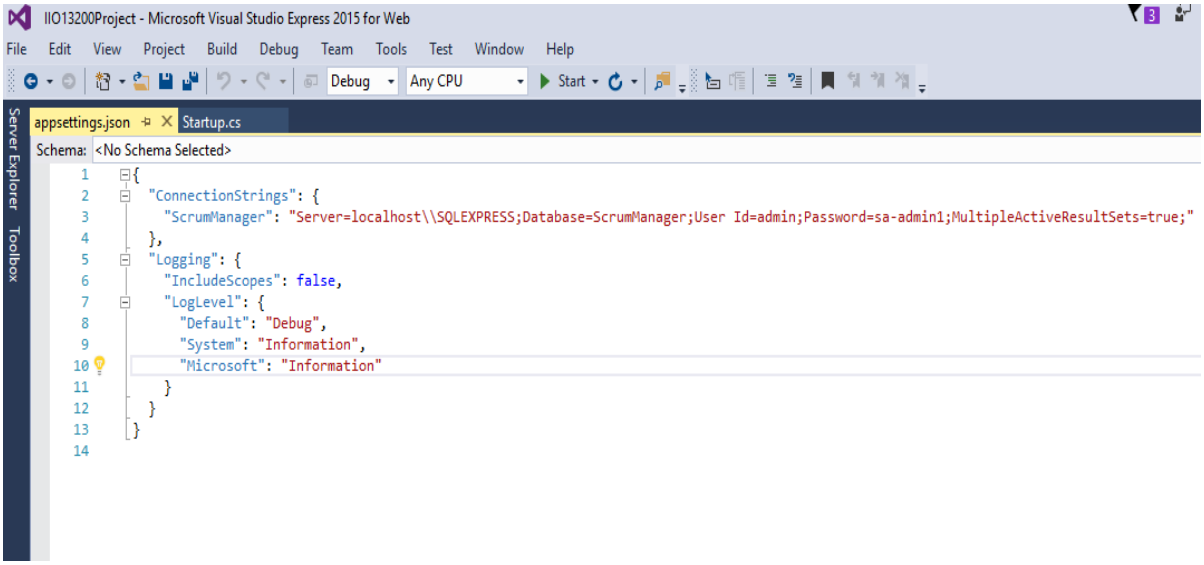
2.3 Connectionstrings

ASP.NET Coressa on esimerkiksi konfiguraatioasetuksien luku tiedostosta suunniteltu uudestaan ja uudessa tavalla konfiguraatiot luetaan json-tiedostosta. Vanhassa ASP.NET:ssa ne luettiin vielä web.configista, joka on xml-tiedosto. Core-projektissa on appsettings.json-tiedosto, jonka voi määritellä sovelluksen startup.cs-tiedostossa otettevan käyttöön. (Microsoft 2016a.)

Tämän raportin työssä se on otettu mm. käyttöön api-projektin startup.cs-tiedostossa. Kyseisen tiedoston ConfigureServices-metodissa on lisätty DataBaseContext käyttöön seuraavasti:

```
services.AddDbContext<ApiContext>(options =>
                        options.UseSqlServer(Configuration.GetConnecti-
onString("ScrumManager")));
```

Appsettings.json:ssa on em. contextille lisätty connectionstring seuraavasti:



Kuva 4: Connectionstringien määrittely

Kuvassa neljä on ConnectionStrings-lonkon sisällä määritelty ScrumManager-nimellä connectionstring, jossa määritellään serverin osoite ja tietokantainstanssin nimi. Seuraavaksi on määriteltynä tietokannan nimi ja sen käyttäjätunnus sekä salasana. Määrittele näihin oman tietokantapalvelimen osoite ja käyttäjätunnus sekä salasana.

3 Tietoa ohjelmasta

3.1 Toteutetut toiminnalliset vaatimukset

- Kehittäjänä haluan nähdä listan projektin tehtävistä.
- Kehittäjänä haluan nähdä listan minulle vastuutetuista kuluvan sprintin tehtävistä (osittain valmis).
- Kehittäjänä haluan lisätä uusia tehtäviä kuluvan sprintin käyttäjätarinoihin (osittain valmies).
- Kehittäjänä haluan ottaa kuluvan sprintin tehtäviä vastuulleni (osittain valmis).
- Scrum masterina haluan lisätä uusia projekteja.
- Scrum masterina haluan lisätä uusia sprintteja.
- Scrum masterina haluan nähdä listan oman tiimin henkilöistä.
- Scrum masterina haluan lisätä tiimiin henkilöitä.
- Scrum managerin käyttäjänä haluan kirjautua järjestelmään sisään.
- Scrum managerin käyttäjänä haluan muokata omia tietojani järjestelmässä.

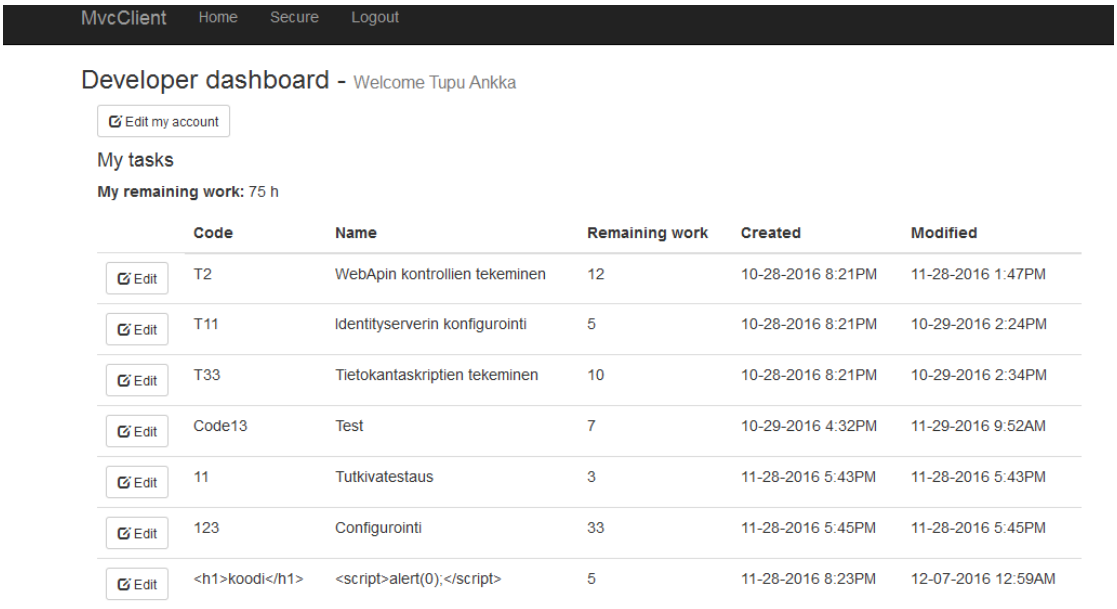
3.2 Toteuttamatta jääneet toiminnalliset vaatimukset

- Kehittäjänä haluan merkata minulle kuuluvan tehtävän valmiiksi.
- Scrum masterina haluan nähdä listan oman tiimin sprintin tehtävistä.
- Scrum masterina haluan nähdä listan projektin tehtävistä.
- Scrum masterina haluan nähdä kokonaiskuvan kuluvan sprintin tehtävien tilanteesta siten, että mitkä tehtävät ovat aloittamatta, kesken ja valmiit.
- Product ownerin vaatimukset jäivät kokonaan täyttämättä.

4 Käyttöliittymä

Käyttöliittymä on suunniteltu siten, että se on jaoteltu Scrumissa olevien roolien mukaan omiin työpöytiin. Eli Scrum masterilla, product ownerilla ja kehittäjällä on omat työpöydät, joissa on korostettu eri asioita. Määrittelydokumentissa oli kuvattu käyttöliittymän toimivan single page app -tyylisesti. Eli käyttöliitymä ei lataa server-iltä sivun päivityksen yhteydessä jatkuvasti sivuja uudestaan vaan täyttää sivun data sitä mukaa json:lla, kun käyttäjä on vuorovaikutuksessa ohjelman kanssa. (Reini 2016.)

Alla on esimerkkinäkymä kehittäjän työpöydästä:

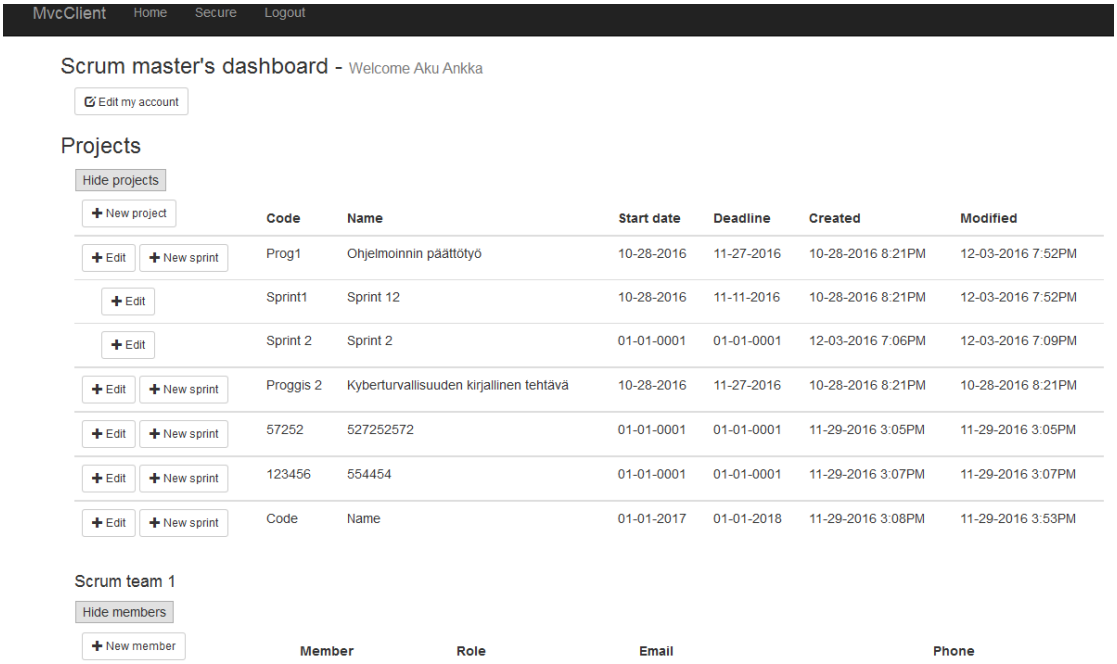


Kuva 5: Kehittäjän työpöytä

Kuvassa viisi näkyy kehittäjän työpöytä, johon sovellus ohjautuu, kun käyttäjä on kirjautunut sisälle ja painanut sen jälkeen secure-linkkiä (jatkokehityksessä ylimääräisen

linkin painaminen voitaisiin poistaa ja ohjata vaan työpöydälle). Sovellus katsoo käyttäjän roolista, että mikä työpöytä kuuluu avata. Työpöydällä käyttäjä näkee mitkä tehtävät ovat hänen vastuullaan ja paljonko on vielä työmäärää jäljellä. Lisäksi käyttäjä voi tehdä muokkauksia tehtäviin.

Seuraavaksi esimerkkinäkymä Scrum masterin työpöydästä:



Kuva 6: Scrum masterin työpöytä

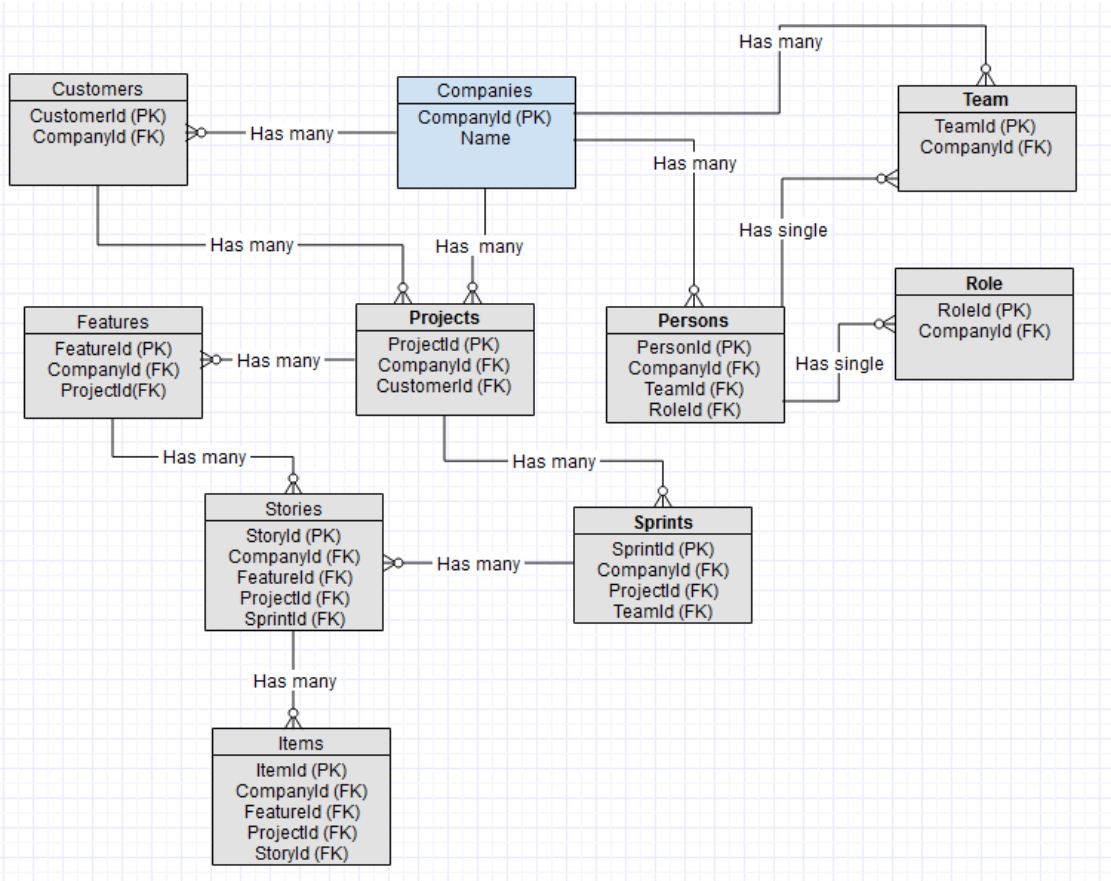
Kun Scrum master roolilla kirjautuu sisälle ohjelmaan, niin avautuu seuraavanlainen työpöytä, jossa voidaan perustaa uusia projekteja ja niiden alle sprintteja. Molempia voidaan muokata tässä näkymässä. Lisäksi Scrum master voi hallinnoida tiimin jäseniä ja lisätä uusia tarvittaessa.

5 Tietovarasto

Tietokannan on suunniteltu toimivan MS SQL Serverillä. Tietokanta luodaan Microsoft SQL Server Management Studion kautta käyttämällä New database-toimintoa, jossa tietokannalle määritellään mm. nimi ja owner. Ohjelman kehitys vaiheessa on käytetty admin-tunnusta, mutta järkevää on tehdä SQL Serverille uusi login ja käyttäjä sekä asettaa ne tietokannalle. Tämän jälkeen määritellä kyseinen käyttäjä ja salasana api-projektin appsettings.json-tiedostoon.

5.1 Tietokantakaavio

Tietokantakaavio ja taulujen viiteavaimet on suunniteltu sillä periaatteella, että jos lähestytään ohjelman rakentamista tietokanta edellä. Mikä tarkoittaa sitä, että asetetaan tietokantaan viiteavaimet kohdilleen taulujen välillä. Tämän jälkeen esimerkiksi voidaan Scaffold-toiminnolla generoida osan luokista automaattisesti, joissa on luokkaviittaukset kunnossa. Tästä on seuraavaa etua, että ei tarvitse käsin rakentaa luokkia ja kehitys tehostuu huomattavasti. Tässä projektissa en tosin lähtenyt scaffoldilla generoimaan koodia, mikä oli harmi sinänsä. (Miller 2016.)



Kuva 7: Tietokantakaavio

5.2 Tietokantaskriptit

Tietokantaskriptit on suunniteltu siten, että tietokanta voidaan luoda tarvittaessa kehitysvaiheessa helposti uudestaan poistamalla olemassa olevat taulut ja luomalla ne uudestaan, jos tarvitaan esim. uusia kenttiä tauluihin. Testidatan saaminen ei tuota ongelmia, koska tietokannan tauluihin lisätään testidataa automaattisesti siinä

vaiheessa, kun api käynnistyy. Api-projektin alla on Models-hakemistossa Seed-Data.cs-luokka, joka ajetaan apin startup.cs:ssa apin käynnistyessä.

Tietokantaskripti löytyy Githubista osoitteesta: https://github.com/K7696/IIO13200-harjoitustyo/blob/master/ScrumManager/RUN-SQL.sql

6 Tiedossa olevat ongelmat

Tiedossa olevat ongelmat ja bugit liittyvät puuttuviin toiminnallisuuksiin, jotka pitäisivät olla helposti ratkaistavissa pienellä lisätyöllä.

Lisäksi SSL-sertifikaatti olisi hyvä asentaa palvelimelle, että liikenne kulkisi selaimen ja palvelimen välissä salattuna. Tätä kuitenkaan ei ole järkevää tämän harjoitustyön ohessa pistää kuntoon.

Identityserver tunnistaa käyttäjät tällä hetkellä siten, että ne on InMemoryUsereina määritelty identityserverin koodiin. Identityserverin startup.cs-tiedostoon tulisi määritellä service ja koodata sen toteuttava luokka identityserver tai api –projektien alle, mikä lukee tietokannasta käyttäjän ja suorittaa sitä vasten tunnistautumisen. Tämäkin olisi pienellä lisätyöllä mahdollista pistää kuntoon.

7 Mitä ollaan opittu ja muuta pohdintaa

Arkkitehtuurin myötä on tutustuttu MVC:n, REST API:n, identityserverin ja AngularJS:n saloihin. Vaikka en kerennyt noin 70 tunnissa koodaamaan kaikkia ominaisuuksia kuntoon, niin kuitenkin voi olla tyytyväinen saavutettuun tulokseen. Jos saisi vielä neljä viikkoa panostaa kunnolla tähän työhön, niin voisi olla homma melko pitkällä. Pari kuukautta on kuitenkin melko lyhyt aika ohjelmistokehityksessä saada projekti valmiiksi siten, että se on tietoturvallinen ratkaisu.

Scaffoldia voisi jatkossa oikeasti testata ja hyödyntää enemmän saman tyyppisissä projekteissa. Sillä voisi generoida oikeasti paljon valmista koodia aikaan.

Haasteita oli monenlaisia saada, kuten identityserver toimimaan ja ei ollut tiedossa ASP.NET Coresta oikein mitään ennen projektia. Suurin haste oli löytää aikaa kun-

nolla tämän työn tekemiseen, kun olen nyt syksyllä aloittanut ylemmän amk-tutkinnon suorittamisen kyberturvallisuudesta ja sen tehtävät ovat todella laajoja. Tämän lisäksi käyn kokopäiväisesti töissä ja olen vaihtanut työpaikkaa syyskuun lopussa. Olen joutunut opiskelemaan uusia työvälineitä, prosesseja ja tuotteita ison kasan, joten todella hankalaa on ollut.

Tämän harjoitustyön tekijänä olen ollut yksin ja en ole saanut oikein apuja keneltäkään tehtävän suorittamiseen. Olen joutunut itse tutkimaan uusia aihealueita ja opettelemaan asioita erehdyksien kautta. Toivon opettajan huomioivan arvosanaa miettiessään positiivisesti, että olen tutkinut uusia tekniikoita ja kirjastoja sekä panostanut arkkitehtuurin miettimiseen vaikka kaikkia toimintoja en saanutkaan valmiiksi. Olen myös panostanut raporttiin ja määrittelyyn.

Lähteet

Microsoft. 2016a. Configuration. Referred 7.12.2016. https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration.

Microsoft. 2016b. Directory structure. Referred 7.12.2016. https://docs.microsoft.com/en-us/aspnet/core/hosting/directory-structure

Microsoft. 2016c. Publishing to IIS. Referred 7.12.2016. https://docs.microsoft.com/en-us/aspnet/core/publishing/iis

Miller, R. 2016. ASP.NET Core. Existing Database. Referred 7.12.2016. https://docs.microsoft.com/en-us/ef/core/get-started/aspnetcore/existing-db

Reini, P. 2016. Harjoitustyön määrittely. Viitattu 7.12.2016. https://github.com/K7696/II013200-job/blob/master/Readme.pdf