jamk.fi

Harjoitustyön määrittely

Scrum manager

Pasi Reini, YTC16S

Määrittely

IIO13200 .NET -ohjelmointi, Esa Salmikangas

27.10.2016 Information technology

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

Sisältö

1 Johdanto

Tämän dokumentin tarkoitus on esitellä IIO13200 .NET -ohjelmointikurssin harjoitustyön määrittelyt. Epäselvyyksien välttämiseksi tässä dokumentissa puhutaan harjoitustyön sijaan Scrum managerista, mikä on toteutettavan ohjelman nimi.

Ensimmäisessä luvussa esitellään toteutettavan ohjelman asiakas ja taustaa sille, miksi ohjelmistoprojekti on aloitettu. Samalla avataan myös asiakkaan omaa tuotekehitysprosessia ja siihen liittyviä haasteita.

Toisessa luvussa käydään läpi Scrum managerin tärkeimmät toiminnot. Tässä dokumentissa ei esitellä käyttötapauskaavioita vaan kuvataan käyttäjätarinoilla kukin käyttötapaus.

Kolmannessa luvussa esitellään Scrum managerissa käytettäviä teknologioita ja ohjelman arkkitehtuuri. Lisäksi tässä luvussa esitellään alustava lista tietokannan tauluista.

Viimeisessä luvussa arvioidaan Scrum managerin toteutukseen kuluva työaika ja pohditaan mahdollisia riskejä, joita saattaa ilmaantua projektissa.

2 Asiakas ja haasteet tuotekehityksessä

2.1 Asiakkaan tuotekehitysprosessi

Asiakkaan omassa tuotekehityksessä hyödynnetään Scrum-mallia, mikä tähtää tehokkaaseen tiimityöskentelyyn monimutkaisissa ohjelmistoprojekteissa. Kehitys tapahtuu sykleissä, joita kutsutaan sprinteiksi. Yksi sykli on ennalta määritelty minimissään viikon ja maksimissaan kuukauden kestävä ajanjakso. Sprintin aikana kehitystiimi suunnittelee, toteuttaa, testaa ja julkaisee periaattessa toimivan tuotteen. Tiimillä apuna on product backlog, johon on listattuna toteutettavat ominaisuudet. Niistä poimitaan sprinteille tehtävät, jolloin poimittavista tehtävistä muodostuu sprintin backlog. Kehittäjien lisäksi scrum-tiimissä toimivat scrum master ja product owner, joiden pitää päästä käsittelemään edellä mainittuja backlogeja. (Scrum.org.)

2.2 Haasteet tuotekehitysprosessissa

Asiakkaan tuotekehitystiimi toimii siis Scrum-mallin mukaan, ja he ovat adaptoineet sen ominaisuudet osaksi omaa tuotekehitystään varsin menestyksekkäästi. Asiakkaan ohjelmistoprojekteissa on kuitenkin haasteita. Nimittäin product backlogia ylläpidetään excelissä, jolloin sen organisointi ja läpikäynti on työlästä. Sprintin backlog on tiimin työtilan ilmoitustaululla, johon sprintin tehtävät on liimattu post-it-lapuilla. Tästä voi seurata se, että sprintin tehtäviä voi jäädä tekemättä, kun lappuja saattaa kadota taululta. Etätyötä tekevät henkilöt eivät pääse työpaikan ulkopuolelta käsiksi sprintin tehtäviin, ja tämän takia on vaikeaa seurata, mitä kenelläkin on työstettävänä. Edellä mainittuihin haasteisiin on tarkoitus vastata web-pohjaisella sovelluksella, mikä tukee tiimin työskentelyä ja mihin pääsevät myös etätyötä tekevät henkilöt selaimen välityksellä kiinni.

3 Scrum managerin käyttötapaukset

Tässä luvussa kuvataan käyttäjätarinoilla Scrum managerin keskeisimmät vaatimukset eri Scrum-roolien ja ohjelman käyttäjän (user) mukaan. Käyttäjätarinat (user stories) ovat kevyitä ja korkean tason määritelmiä järjestelmän vaatimuksille, mutta kuvattuna kuitenkin siten, että kehittäjät pystyvät antamaan järkevän työmääräarvion toteutettavista vaatimuksista. (Ambysoft Inc.)

3.1 Developer

- Kehittäjänä haluan nähdä listan projektin tehtävistä.
- Kehittäjänä haluan nähdä listan oman tiimin kuluvan sprintin tehtävistä.
- Kehittäjänä haluan nähdä listan minulle vastuutetuista kuluvan sprintin tehtävistä.
- Kehittäjänä haluan lisätä uusia tehtäviä kuluvan sprintin käyttäjätarinoille.
- Kehittäjänä haluan ottaa kuluvan sprintin tehtäviä vastuulleni.
- Kehittäjänä haluan merkata minulle vastuutetun tehtävän valmiiksi.

3.2 Product owner

- Tuoteomistajana haluan lisätä uusia tehtäviä projektille.
- Tuoteomistajana haluan muokata projektin tehtäviä.
- Tuoteomistajana haluan poistaa projektin tehtäviä.
- Tuoteomistajana haluan nähdä listan oman projektin tehtävistä.
- Tuoteomistajana haluan nähdä listan oman tiimin sprintin tehtävistä.

3.3 Scrum master

- Scrum masterina haluan nähdä listan oman tiimin henkilöistä.
- Scrum masterina haluan lisätä tiimiin uusia henkilöitä.
- Scrum masterina haluan nähdä listan oman tiimin sprintin tehtävistä.
- Scrum masterina haluan nähdä listan projektin tehtävistä.
- Scrum masterina haluan lisätä uusia projekteja.
- Scrum masterina haluan lisätä uusia sprintteja.
- Scrum masterina haluan nähdä kokonaiskuvan kuluvan sprintin tehtävien tilanteesta siten, että mitkä tehtävät ovat aloittamatta, kesken ja valmiit.

3.4 Scrum managerin käyttäjä

- Scrum managerin käyttäjänä haluan kirjautua järjestelmään sisään.
- Scrum managerin käyttäjänä haluan muokata omia tietojani järjestelmässä.

4 Käytettävät teknologiat ja arkkitehtuuri

4.1 ASP.NET Core

ASP.NET Core on uusi avoimen lähdekoodin ja usean alustan päällä toimiva framework, joita ovat esimerkiksi Windows, Mac ja Linux. Tämä framework on tarkoitettu modernien mm. pilvessä toimivien web-ohjelmien rakentamiseen. (Anderson, Luttin & Roth 2016.)

Ensimmäinen julkaisu ASP.NET:sta tuli ulos noin 15 vuotta sitten .NET Frameworkin osana. Sen jälkeen lukemattomat kehittäjät ovat hyödyntäneet sitä erilaisten web-

ohjelmien tekemisessä. ASP.NET Core sisältää monia arkkitehtuurisia muutoksia, joilla on tähdätty kevyempään ja modulaarisempaan frameworkiin. Core ei pohjaudu enää kuuluisaan System.Web.dll:aan, vaan se tukeutuu sarjaan erilaisia NuGet-paketteja. Tämä mahdollistaa sen, että voidaan ladata vain ne paketit, joita sovellus tarvitsee. Enää ei tarvitse ladata koko .NET frameworkia sovelluksen pohjalle, mikä tuo hyötyjä, sillä pienempi ohjelma pinta-ala antaa tiukemman turvallisuuden ja paremman suorituskyvyn. (Anderson, Luttin & Roth 2016.)

4.2 AngularJS

AngularJS ON moderni javascript framework, jonka on tehnyt Google. Angularia käytetään yleisesti SPA:n (Single page app) yhteydessä. ASP.NET Core MVC:lle Angularin voi ladata käyttöön siten, että Core-projektin hakemistorakenteessa olevaan bower.json-konfigurointitiedostoon lisätään viittaus dependencies-lonkoon esimerkiksi seuraavasti: “angular”: “1.5.7”. Tämän jälkeen kun bower.json-tiedosto tallennetaan, niin Visual studio lataa automaattisesti kyseisen paketin määritellyllä versiolla Core-projektille lib-hakemiston alle. Jotta AngularJS:n saa käyttöön web-projektin sivuilla, niin siihen pitää määritellä vielä viittaus _Layout.cshtml-tiedostoon, jota käytetään pohjana MVC-projektin näkymissä. (Addie & Venkata.)

AngularJS on siis tarkoitettu dynaamisten web-pohjaisten sovelluksien rakentamiseen. Se mahdollistaa HTML:n käyttämisen templaattikielenä ja sen laajentamisen. Angularin data bindaus ja depency injection eliminoivat suuren osan koodista, mitä muuten jouduttaisiin kirjottamaan. Tämä javascript-kirjasto on siis tarkoitettu käyttöliittymän rakentamista varten. Se osaa DOM-manipuloinnin ja AJAX-käsittelyt. Sillä voi rakentaa käyttöliittymätasolle kaikki CRUD-toiminnot (Create, Read, Update ja Delete), joita sovellus tarvitsee. Lisäksi sillä voi hoitaa data-bindauksen, perus templaatti directiivit, lomakevalidoinnit ja depency injectionin. (AngularJS.)

Depency injection esiintyy ASP.NET Coren ja AngularJS:n dokumenteissa varsin usein, joten on hyvä vähän aukaista sitä tässä määrittelyssä. Depency injection on siis tekniikka, jolla haetaan löyhää sidosta eri olioiden ja niiden riippuvuuksien välille. Kun luokat on suunniteltu depency injectionin näkökulmasta, ne ovat erittäin vähän

sidoksissa toisiinsa, koska niillä ei ole suoraan kova koodattua riippuvuutta niitä käyttävään instanssin välillä. (Addie & Smith.)

4.3 Entity Framework Core

Entity Framework Core on kevyempi ja laajennettava versio suositusta Entity Framework data access –teknologiasta. EF Core mahdollistaa kehittäjien työskentelyn tietokannan kanssa käyttämällä .NET olioita ja eliminoi suuren osaa siitä data-access koodista, mitä kehittäjien tavallisesti on täytynyt kirjoittaa. EF Core tukee monia tietokantamoottoreita. (Microsoft.)

Entity framework Coren voi asentaa esimerkiksi Web API:lle siten, että ladataan se Nuget-pakettina kirjoittamalla Visual studion Nuget package manager consoliin: “Install –Package Microsoft.EntityFrameworkCore.SqlServer.” (Microsoft.)

4.4 Identityserver4

Identityserver4 .net-pohjainen on OpenID Connect ja OAuth 2.0 framework ASP.NET Core:lle. Autentikointia tarvitaan silloin kun ohjelman on tiedettävä käyttäjän identiteetti. Yleisimpiä autentikointiprotokollia OpenID Connectin ja OAuth 2.0:n lisäksi, ovat SAML2p, WS-Federation. OpenID Connect on edellä mainituista protokollista uusin ja sen on ajateltu käyvän hyvin yhteen modernien ohjelmien kanssa. Se on rakennettu alusta lähtien mobiilisovellusympäristöä ajatellen ja se on suunniteltu API ystävälliseksi. (Allen & Baier.)

Ohjelmilla on muutamia perustavanlaatuisia tapoja kommunikoida API:n kanssa. Nimittäin käyttämällä sovelluksen omaa tai välittämällä käyttäjän identiteettiä ja joskus molempia tapoja on täytynyt yhdistellä. OAuth2 on protokolla, mikä sallii sovelluksien pyytää “access tokeneita” niitä jakavalta palvelulta (esimerkiksi Identityserver4), jotta voidaan kommunikoida turvallisesti API:n kanssa. Edellä mainitun pitäisi vähentää monimutkaista autentikointilogiikkaa sovelluksen ja API:n välillä. Lisäksi autentikointia voidaan tällä tavalla keskittää yhteen paikkaan. (Allen & Baier.)

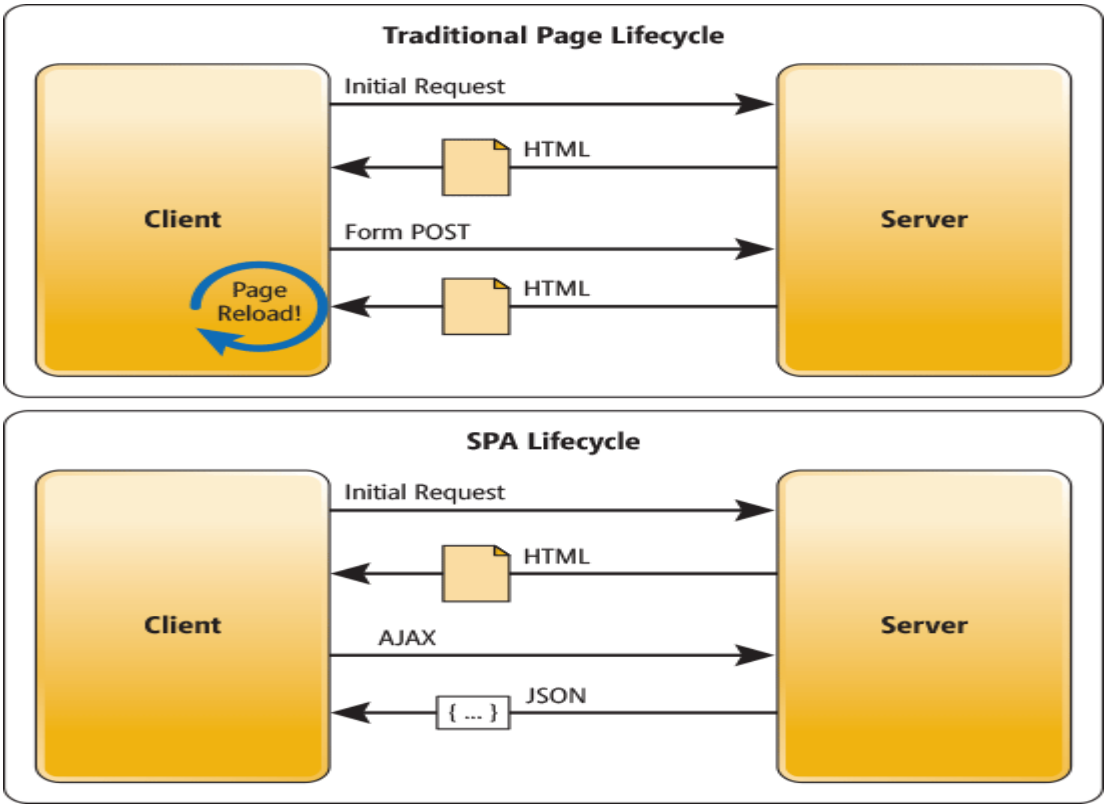
4.5 Arkkitehtuuri

4.5.1 Single page application

Käyttöliittymäarkkitehtuuri toteutetaan single page application -tyyliä hyödyntäen AngularJS:lla ja ASP.NET Core MVC:lla. Single page applicationilla (SPA) haetaan sitä, että web-sovellus lataa yhden HTML-sivun ja dynaamisesti päivittää sivun sisältöä sitä mukaa, kun käyttäjä on vuorovaikutuksessa sovelluksen kanssa. SPA käyttää ajaxia ja javascriptia responsiivisen web-sovelluksen luomiseen ilman, että tarvitsee jatkuvasti ladata sivuja. Tämä tarkoittaa sitä, että suurin osa tapahtumista tapahtuu käyttäjän selaimessa javascriptilla. Perinteisissä web-sovelluksissa joka kerta kun sovellus kutsuu palvelinta, niin renderöidään uusi HTML-sivu, mikä laukaisee sivun uudelleen latautumisen selaimessa. SPA:ssa tästä pyritään eroon lataamalla vain osia sivusta palvelimelta esimerkiksi JSON-datana, mikä mahdollistaa sovelluksien responsiivisemman käytön ilman ylimääräisiä sivun latauksia. (Wasson 2013.)

Tästä saadaan lisäksi arkkitehtuurin hyöty seuraavalla tavalla, että koko käyttöliittymän voi vaihtaa ilman, että tarvitsee palvelimen koodiin koskea, mikä implementoi sovelluksen business-logiikan. Käyttöliittymä ja palvelimen palvelut ovat siis itsenäisiä, jotka voidaan korvata ilman, että tarvitsee koko sovellusta hajottaa. (Wasson 2013.)

Seuraava kuva havainnollistaa SPA:n ja perinteisen web-sovelluksen eroja (Wasson 2013).



4.5.2 Web api

Scrum managerin tietojen jakajana toimii Web API, johon tehdään REST-tyyppisesti tiedon hakuja ja tallennuksia. Nykyajan web-sovelluksia rakennettaessa monesti kuulee puhuttavan REST:sta ja .net Web API:sta. Termillä API tarkoitetaan “Application Programming Interfacea”, mikä on synonyymi online web-palvelulle. Aikaisemmin SOAP saattoi olla suosittu web-palvujen yhteydessä, mutta nykyään REST on enemmän muodin huipulla. (Kearn 2015.)

REST, toisin sanoen “Representational State Transfer”, on arkkitehtuurinen malli API:lle, joka hyödyntää HTTP-protokollaa (Hypert Text Transfer Protocol) ja sen alle rakennettuja kommunikaatiometodeja. Melkein kaikki laitteet, jotka on yhdistetty internetiin käyttävät HTTP-protokollaa, mikä tekee sen hyväksi alustaksi API:lle. HTTP on pyyntö- ja vastaussysteemi, mikä tarkoittaa että client lähettää pyynnön “endpointille” ja se vastaa takaisin clientille. Kyseiset client ja “endpoint” voivat olla mitä vain, mutta yleensä kyseessä on esimerkiksi web-selaimen, palvelimen ja API:n välinen yhteys. (Kearn 2015.)

Web API:n yhteydessä törmää seuraaviin asioihin:

Request verbs: Verbit kuvaavat, mitä voi tehdä resurssin kanssa. Selain tyypillisesti käyttää GET-verbiä tietojen hakemiseen ”endpointista”. Muita verbejä ovat mm. POST, PUT ja DELETE. (Kearn 2015.)

Request headers: Otsikkotiedot lähetetään pyynnön yhteydessä. Näissä voidaan määritellä, minkä tyyppistä vastausta on pyydetty, ja lisäksi voidaan välittää muita, kuten autentikointiin liittyviä tietoja. (Kearn 2015.)

Request Body: Pyynnön yhteydessä voidaan bodyssa lähettää pyynnön data, kuten esimerkiksi POST-verbin yhteydessä, kun luodaan uutta dataa. Pyynnön mukana tyypillisesti välitetään JSON tai XML-dokumentti. (Kearn 2015.)

Response Body: Vastauksen body sisältää palvelimen vastauksen, mikä voi olla HTML-sivu tai API:n kanssa yleensä JSON tai XML-dokumentti (Kearn 2015).

Response Status codes: Palvelin palauttaa vastauksen mukana pyynnön tilan, mikä on esimerkiksi 200 OK (pyyntö onnistui) (w3.org).

4.6 Tietokanta

Scrum managerin tietovarastona käytetään tietokantaa, joka tulee toimimaan Microsoft SQL Serverin päällä. Tässä luvussa on listattuna Scrum managerissa käytettävät taulut, jotka luodaan SQL-skriptilla. Kyseinen skripti voidaan ajaa Microsoft SQL Server Management Studion kautta.

Lista tietokannan tauluista:

1. Companies-taulu sisältää Scrum managerin käyttäjäyritykseen liittyviä tietoja.
2. Customers-taulu sisältää projektin asiakkaan tietoja.
3. Features-taulu sisältää projektissa toteutettavat ominaisuudet.
4. Items-taulu sisältää käyttäjätarinoihin liittyvät tehtävät.
5. Persons-taulu sisältää kehitystiimin henkilöt.
6. Projects-taulu sisältää projekteihin liittyviä perustietoja.
7. Sprints-taulu sisältää sprinttien tietoja.
8. Stories-taulu sisältää käyttäjätarinoita.
9. Teams-taulu sisältää kehitystiimien tietoja.

5 Työajan arviointi

Työmäärät arvioidaan full stack –periaatteella, että käyttäjätarinan implementointi sisältää toteutuksen käyttöliittymälle, business logiikalle ja tiedon käsittelylle tietokantaan asti. Arvioin könttänä eri roolien mukaan työmäärät ja sen lisäksi selvitystöihin menevän ajan.

Developer -osio: 15 h

Product owner -osio: 15 h

Scrum master -osio: 15 h

Arkkitehtuuriin ja teknologioihin liittyvät tutkimukset: 30 h

Yhteensä koko homma noin: 75 h

Lähteet

Addie, S. & Smith, S. Depency Injection. Viitattu 24.10.2016. https://docs.asp.net/en/latest/fundamentals/dependency-injection.html

Addie, S. & Venkata, K. Using Angular for single page applications. Viitattu 24.10.2016. https://docs.asp.net/en/latest/client-side/angular.html

Allen, B. & Baier, D. Identityserver4. Viitattu 26.10.2106. http://docs.identityserver.io/en/dev/

Ambysoft Inc. User Stories: An Agile Introduction. Viitattu 26.10.2016. http://www.agilemodeling.com/artifacts/userStory.htm

AngularJS. What is Angular. Viitattu 24.10.2016. https://docs.angularjs.org/guide/introduction

Anderson, R., Luttin, S. & Roth, D. 2016. Introduction to ASP.NET Core. Viitattu 24.10.2016. https://docs.asp.net/en/latest/intro.html#what-is-asp-net-core

Kearn, M. 2015. Introduction to REST and .net Web API. Viitattu 24.10.2016. https://blogs.msdn.microsoft.com/martinkearn/2015/01/05/introduction-to-rest-and-net-web-api/

Microsoft. Entity Framework Core. Viitattu 26.10.2016. https://docs.efproject.net/en/latest/

Scrum.org. What Is Scrum. Viitattu 24.10.2016. https://www.scrum.org/Resources/What-is-Scrum

Wasson, M. 2013. ASP.NET – Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET. Viitattu 24.10.2016. https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

W3.org. Status Code Definitions. Viitattu 25.10.2016. https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html