SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

TYÖAJANHALLINTAJÄRJESTELMÄ

TEKIJÄ: Antti Lievonen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä Antti Lievonen	
Työn nimi Työajanhallintajärjestelmä	
Päiväys 9.11.2016	Sivumäärä/Liitteet 28/1
Ohjaajat Ohjelmistosuunnittelija Mikko Pääkkönen, lehtori Jussi Koistinen	
Toimeksiantaja/Yhteistyökumppani TCD Consulting and Research Oy	

Tiivistelmä

Opinnäytetyön aiheena oli kehittää TCD Consulting and Reseachin asiakkaalle soveltuva web-sovellus työajanhallintaan. Asiakasyritys oli koulu, jonka käytössä olleessa työajanhallintajärjestelmässä oli puutteita. Projektiin alkuun kuului ominaisuuksien suunnitelua ja eri .NET ja käyttöliittymän toteutustapojen vertailua. Sivuston tuli oli olla selkeä ja helppokäyttöinen ja sisältää työajanmerkintäsivu työntekijöille, työtuntien tarkastussivu tarkistajille, käyttäjienhallinta esimiehille ja raportin tallennus CSV-muodossa palkanlaskentahenkilöstölle. Muita vaadittuja ominaisuuksia olivat käyttäjien kirjautumistodennus Windows-tilien kautta, tarkastusoikeuksien hallinta, sähköposti-ilmoitus tarkistetuista töistä, virhetilanteiden kirjaaminen sekä responsiivisen käyttöliittymä. Lisäksi projektiin kuului työn tilaajalta saadun tietokannan muokkaaminen sovelluksen tarpeisiin sopivaksi.

Sovellus kehitettiin C#-ohjelmointikielellä, käyttäen MVC-arkkitehtuuria. Käyttöliittymän tekemiseen käytettiin HTML-, CSS- ja JavaScript-komponentteja.

Projektin lopputuloksena saatiin halutunlainen sovellus, joskin joitakin ominaisuuksia, kuten raportointia ja käyttöliittymän ulkoasua, joudutaan vielä jatkokehittämään asiakkaan palautteen perusteella.

Avainsanat
työajanhallinta, kalenteri, MVC, C#

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author Antti Lievonen			
Title of Thesis Working-Time Control System			
Date	9 November 2016	Pages/Appendices	28/1
Supervisors Mr. Mikko Pääkkönen, Software Developer, Mr. Jussi Koistinen, Lecturer			
Client Organisation /Partner TCD Consulting and Research Ltd			

Abstract

The subject of this thesis was to develop a working-time control system for TCD Consulting and Research's client. The web application needed to have work time submitting for employees, work evaluation for inspectors, user management for supervisors and a report generation feature for the payroll personnel.

Other required features for the application were Windows authentication for user logins, user access management, email notification for evaluated work events, error logging and responsive user interface. Ease of use and a good user interface were also required. Additionally, updating and managing the database were also a part of this project.

The application was created using the MVC-architecture and the C# programming language. The user interface uses HTML, CSS and JavaScript components.

The result of the project is an application that has all the necessary features, although some modifications are needed based on the client's feedback, mainly for the report creation and user interface.

Keywords
Control of Working Time, Calendar, MVC, C#

ESIPUHE

Haluan kiittää projektin tilaajaa TCD Consulting and Researchia mielenkiintoisesta opinnäytetyöaiheesta ja erityisesti ohjelmistosuunnittelija Arto Pärnästä työn ohjauksesta. Kiitokset myös Savonia-ammattikorkeakoulun ohjelmistosuunnittelijalle Mikko Pääkköselle ja lehtori Jussi Koistiselle työn ohjauksesta.

Kuopiossa 9.11.2016
Antti Lievonen

SISÄLTÖ

LYHENTEET JA MÄÄRITELMÄT

C# on Microsoftin kehittämä oliopohjainen ohjelmointikieli, jolla työn taustakoodit (back-end) toteutetaan.

HTML (eli HyperText Markup Language) on verkkosivujen tekemiseen kehitetty merkintäkieli, jolla sivuston perustoiminnot rakennetaan.

CSS (eli Cascading Style Sheets) on verkkosivujen ulkoasun muokkaamiseen tehty tyyliohjemuoto.

MVC (eli Model-View-Controller) on ohjelmistoarkkitehtuuri, jossa projekti jaetaan näkymiin, malleihin ja käsittelijöihin.

Visual Studio on kehitystyökalu, joka sisältää paljon työkaluja sovellusten kehittämiseen.

.NET framework on Microsoftin kehittämä ohjelmistokomponenttikirjasto Visual Studio-ohjelmistolle.

JavaScript on ohjelmointikieli, jolla verkkosivuihin saadaan lisättyä dynaamisuutta. HTML:n ja CSS:n kanssa nykyaikaisen verkkosivukehityksen perusta.

JavaScript-kirjasto on JavaScriptiin lisätoimintoja tuova koodipaketti.

jQuery on suosittu JavaScript-kirjasto, joka lisää siihen uusia ominaisuuksia ja mahdollistaa yksinkertaisen HTML-elementtien muokkaamisen.

Bootstrap on erityisesti mobiililaitteiden käyttöliittymän responsiivisuutta parantava web-sovelluskehys (framework).

Front-end on käyttäjän näkemä osuus verkkosivusta, siihen sisältyy HTML, CSS ja JavaScript.

Back-end on sovelluksen sisällä käyttäjältä piilossa tapahtuva tiedon käsittely.

Ajax (eli Asynchronous JavaScript and XML) mahdollistaa palvelinkutsut ilman, että JavaScript-koodin tarvitsee jäädä odottamaan vastausta palvelimelta, jolloin koko sivun uudelleenlataamista ei tarvita.

Postal on luokkakirjasto MVC-sovelluksiin. Sen avulla voi luoda sähköposteja käyttäen MVC-näkymiä pohjina.

CSV (eli Comma-Separated Values) on tekstipohjainen tiedostomuoto, jossa jokainen rivi esittää yhtä tietuetta. Tietueen kentät erotellaan toisistaan esimerkiksi pilkulla.

1 JOHDANTO

Opinnäytetyössä kehitetään yrityksen asiakkaalle työtuntisovellus. Asiakasyritys on koulu, joka tarvitsee uuden ja paremman työtuntijärjestelmän korvaamaan tähän asti käytössä olleen, jossa ei ollut kaikkia haluttuja ominaisuuksia tai ne olivat hankalia käyttää. Myös raporttien ulkoasussa on parantamisen varaa. Asiakasympäristössä käytetään vain Windows-koneita, joten sovelluksen monialusta tuesta ei tarvitse huolehtia.

Uuteen sovellukseen tulee kuulua tuntikirjaussivu, jossa voi pitää kirjaa työpäivien kulusta sekä hyväksymissivu, jossa tarkastaja näkee työntekijän lähettämät työtunnit ja voi kuitata ne joko hyväksytyiksi tai hylätyiksi. Sovellukseen kuuluu myös käyttäjähallinta esimiehille sekä työaikaraporttien tallentaminen Excel-tiedostoiksi, esimerkiksi palkanlaskentahenkilöstölle nähtäväksi. Sovelluksessa käytetään työn tilaajalta saatua SQL-tietokantaa.

Raportissa käydään läpi sovelluksen suunnittelu, toteutus ja käytetyt komponentit.

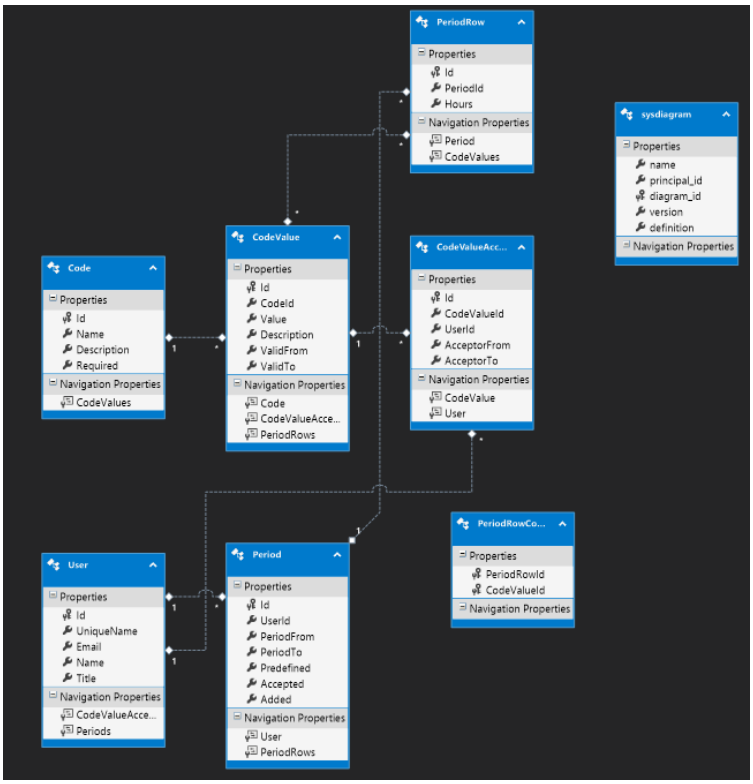
2 SUUNNITTELU

2.1 Projektin aloitus

Aivan projektin alussa asiakasyrityksen tarkoista tarpeista ei ollut vielä varmuutta, joten ensimmäiset kaksi viikkoa kuluivat suunnitteluun ja työn pohjustukseen. Työn tilaaja määritti, että projektin tuli olla C#-ohjelmointikielellä tehty ASP.NET MVC-sovellus. Lisätietoja ja ehdotuksia sain muutaman viikon välein järjestetyissä tapaamisissa työn tilaajan edustajalta.

2.2 Tietokanta

Työn tilaajalla oli jo projektin alussa valmiina SQL-tietokanta (kuva 1), jossa oli alustavat taulut kaikille tarvittaville ominaisuuksille. Työn edetessä tietokantaan tuli muutamia muokkauksia ja lisäyksiä.



KUVA 1. Tietokanta projektin alussa

Tietokanta pysyi projektin loppuun asti suurelta osin samana, muutamaa taulujen ja kenttien muokkausta ja lisäystä lukuun ottamatta. Lopullisen tietokannan taulut ovat Period, PeriodCodeValue, Code, CodeValue, CodeValueAcceptor, User, UserHours ja UserHoursCodeValue.

Period-tauluun tallennetaan kaikki käyttäjien luomat työtapahtumat. Pakollisia tietoja ovat jokaisen periodin alkamis- ja päättymisajankohdat, työtapahtuman luoneen käyttäjän tunnistetieto, hyväksymisaste ja lisäyspäivävämäärä. Valinnaisia kenttiä ovat muokkausajankohta, täyden hyväksymisen ajankohta ja luoneen käyttäjän kommentti. PeriodCodeValue on välitaulu, jonka avulla voidaan yhdistää yhteen Periodiin useita CodeValueita.

Code-taulu sisältää tiedot kaikista eri työtyypeistä, joita tapahtumaan voi sitä luotaessa lisätä. Pakollisia kenttiä ovat nimi ja pakollisuuden määritys, lisäksi valinnaisena kenttänä kuvaus. CodeValue-taulussa listataan kaikki työtapahtumaa luotaessa valittavissa olevat valinnat. Testidatana taulussa on esimerkiksi kustannuspaikka 1, kustannuspaikka 2, projekti 1, projekti 2. Pakollisina kenttiä ovat CodeId eli viittaus code-tauluun, jotta työtyypit ovat oikeiden valintojen alla, value-kenttä jossa on tapahtumatyypin numerokoodi sekä kuvaus eli käytännössä nimi. Lisäksi valinnaisina kenttinä ovat voimassaoloaikojen alkamis- ja loppumispäivämäärät, jotta työtyyppi voidaan rajata valittavaksi esimerkiksi pelkästään kesällä.

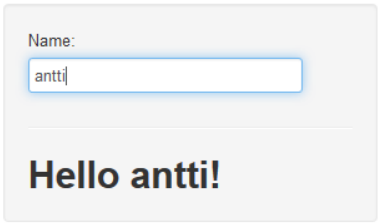
User-taulussa ovat käyttäjien perustiedot. Pakollisia tietoja ovat käyttäjänimi, sähköposti sekä nimi. Valinnaisia tietoja ovat titteli ja viittaus esimieheen. UserHours-taulua käytetään raporttien luonnissa tunnistamaan ja yhdistämään käyttäjien työtapahtumat. UserHoursCodeValues-taulu toimii välitauluna UserHoursin ja CodeValuen välillä.

2.3 Angular ja Knockout

Yksi projektin tavoitteista oli tutkia, olisiko järkevämpää käyttää jompaakumpaa sivuston ulkoasun asetteluun tarkoitetuista suosituista JavaScript-kirjastoista, AngularJS tai KnockoutJS. Molemmilla pystytään yhdistämään dynaamisesti datamalleja käyttöliittymäkomponentteihin sekä päivittämään sivun elementtejä ilman uudelleenlatausta.

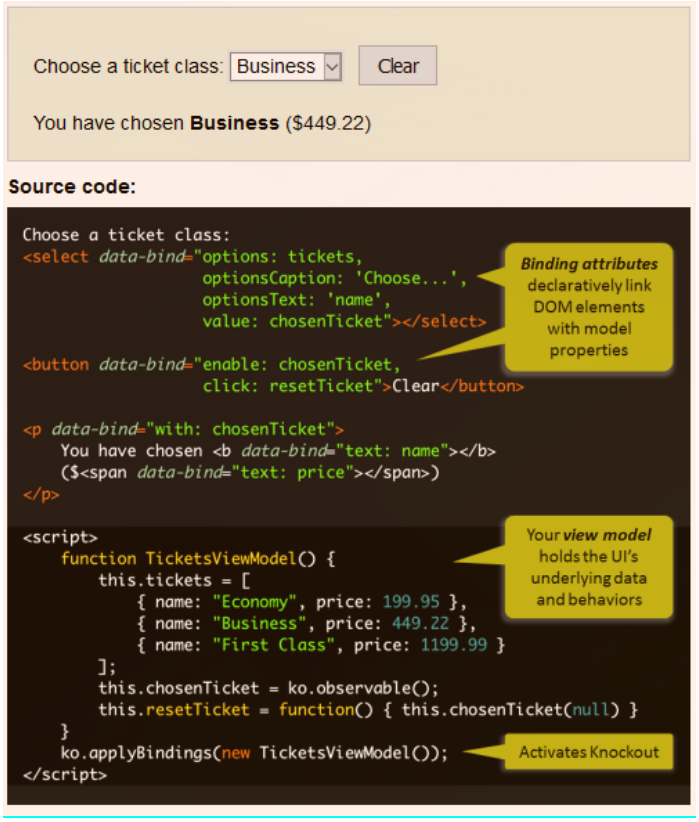
AngularJS on Googlen kehittämä JS-kirjasto, joka julkaistiin vuonna 2010 ja jolla pystytään muun muassa liittämään tietoa suoraan HTML-elementteihin (Google, 2016b). Alla olevassa esimerkissä (kuva 2) tekstikenttä ja otsikko on yhdistetty toisiinsa: heti kun käyttäjä alkaa kirjoittaa tekstikenttään, muuttuu myös otsikko. Tekstikentässä on määritetty angularin malli, ng-model=”yourName”. Samaa mallia kutsutaan tuplahakasulkujen sisällä otsikossa, jolloin tekstikentän mallin sisältämä teksti kopioituu myös otsikkoon.

```
<!doctype html>
<html ng-app>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/
angular.min.js"></script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here">
      <hr>
      <h1>Hello {{yourName}}!</h1>
    </div>
  </body>
</html>
```



KUVA 2. Esimerkki AngularJS:n toiminnasta (Google, 2016a.)

KnockoutJS on avoimen lähdekoodin JS-kirjasto, joka on myös julkaistu vuonna 2010 (Knockout, 2016a). Perustasolla Knockout toimii samankaltaisesti kuin Angular. Alasvetovalikko yhdistetään malliin, josta eri vaihtoehtojen nimet luetaan (kuva 3). Kun käyttäjä valitsee haluamansa vaihtoehdon, merkitään se valituksi ja luetaan sen tiedot mallista ja esitetään käyttäjälle.



KUVA 3. Esimerkki Knockoutin toiminnasta (Knockout, 2016b.)

Suurelta osin AngularJS sekä KnockoutJS sisältävät samankaltaiset ominaisuudet eri tavoilla toteutettuna, mutta erojakin on olemassa. Molemmat kirjastot tukevat esimerkiksi mallinteita (template), joilla koodi saadaan jaoteltua pienempiin, helpommin hallittaviin osiin. Angular on kuitenkin kokonaisuutena paljon Knockoutia laajempi paketti ja sisältää enemmän ominaisuuksia, kuten lomakkeiden validoinnin ja palveluita, joilla koodia voi jakaa useiden sivujen välillä, kun taas Knockout keskittyy vain tiedon esittämiseen.

2.4 ASP.NET

ASP.NET on .NET-kirjaston verkkosivujen luomiseen tarkoitettu osa, joka sisältää WebFormsin sekä MVC:n. WebForms tarjoaa työkalut helppoon sivun käyttöliittymän luomiseen, esimerkiksi tekstikenttiä ja linkkejä voidaan luoda yksinkertaisesti raahaamalla ne ”työkalupakista” haluttuun kohtaan sivulla. WebForms kääntää luodut elementit sivun latauksen yhteydessä HTML-elementeiksi. WebForms kuitenkin häviää MVC:lle koodin uudelleenkäytettävyydessä, suorituskyvyssä, projektin selkeydessä ja muokattavuudessa (Sukesh, 2016). Lisäksi MVC:n puhtaasta HTML:stä koostuvat näkymät mahdollistavat lisäkirjastojen kuten AngularJS:n helpon käytön.

2.5 MVC

MVC:n ideana on jakaa ohjelmisto kolmeen osaan: malleihin, näkymiin ja käsittelijöihin. MVC:n suurin etu on eri osien selkeä erottelu käyttötarkoituksen mukaan, mikä parantaa koodin rakennetta ja helpottaa ylläpidettävyyttä, testaamista sekä uudelleenkäytettävyyttä (Envato, 2016.)

Malleilla (Model) hoidetaan yleisesti kaikki tiedon käsittely ja bisneslogiikka. Malleista löydetään tavallisesti kaikki projektissa tarvittavat bisnesluokat ja metodit.

Näkymillä (View) esitetään käyttäjälle käyttöliittymä ja kaikki tarvittava tieto. Näkymä voi olla perus HTML-sivu, joka voi lähettää ja vastaanottaa tietoa käsittelijältä.

Käsittelijät (Controller) ottavat vastaan näkymiltä tulleita pyyntöjä ja palauttavat halutut tiedot, ne siis toimivat mallien ja näkymien välillä. Esimerkiksi käyttäjän painaessa nappia käyttöliittymässä lähtee kutsu ladata jotakin tietoa tietokannasta käsittelijälle, minkä jälkeen käsittelijä kutsuu mallia, joka palauttaa halutut tiedot takaisin käyttäjälle käsittelijän kautta.

2.6 Windows-autentikointi

Projektissa haluttiin käyttää Windows-autentikointia, jolloin käyttäjät voivat kirjautua palveluun samoilla tunnuksilla, joilla he kirjautuvat sisään toimialueeseen töissä tietokoneen avatessaan, eikä heidän tarvitse tehdä erillisiä tunnuksia tätä sovellusta varten. Windows-autentikoinnin hyötynä on myös käyttöoikeuksien hallinta aktiivihakemiston (Active Directory, AD) kautta. Luonnollisesti vain Windows-laitteet tukevat vakiona AD:ta (Microsoft, 2000). Esimerkiksi tarkastussivulle päästäkseen käyttäjältä vaaditaan, että hänellä on oikeus tarkastaa työtapahtumia eli hänen AD-roolinsa on asetettu tarkastajaksi asiakkaan palvelimella.

Autentikoinnin voi toteuttaa niin, että kuka tahansa pääsee osalle sivuista, mutta tietyt toiminnot vaativat kirjautumisen. Tässä projektissa kirjautuminen vaaditaan heti sivuille siirryttäessä, sillä sivuston kaikki ominaisuudet vaativat jotain tietoja käyttäjästä, esimerkiksi työaikakalenteri vaatii tiedon siitä, kuka käyttäjä on tallentanut mitäkin työtapahtumia. Sovelluksen tarvitsemat tiedot käyttäjästä haetaan kirjautumisen jälkeen yksilöllisen käyttäjänimen perusteella palvelimelta.

Käyttöoikeuksien hallinta oli tärkeä osa projektia, sillä vain kalenterinäkymän oli tarkoitus olla avoin kaikille käyttäjille. Oikeuksien tarkistus tapahtuu aina ennen käsittelijälle (controller) siirtymistä. Jos sovellus havaitsee käyttäjän yrittävän siirtyä sivulle, johon hänellä ei ole oikeuksia, näytetään virhesivu.

Käyttöoikeudet jakautuvat kolmeen ryhmään. Peruskäyttäjä pystyy käyttämään vain työtuntikalenteria töidensä merkitsemiseen. Esimies pääsee käyttäjienhallintaan muokkaamaan käyttäjätietoja kuten sähköpostiosoiteita, luomaan uusia käyttäjiä, hallitsemaan tarkastajien oikeuksia ja luomaan raportteja, joista näkee laajemmin henkilöiden tehdyt työtunnit ja niiden kohteet. Tarkastaja pääsee työtuntien tarkastussivulle, jossa voidaan hyväksyä tai hylätä työtapahtumia, joihin hänellä on esimiehen asettamat tarkastusoikeudet.

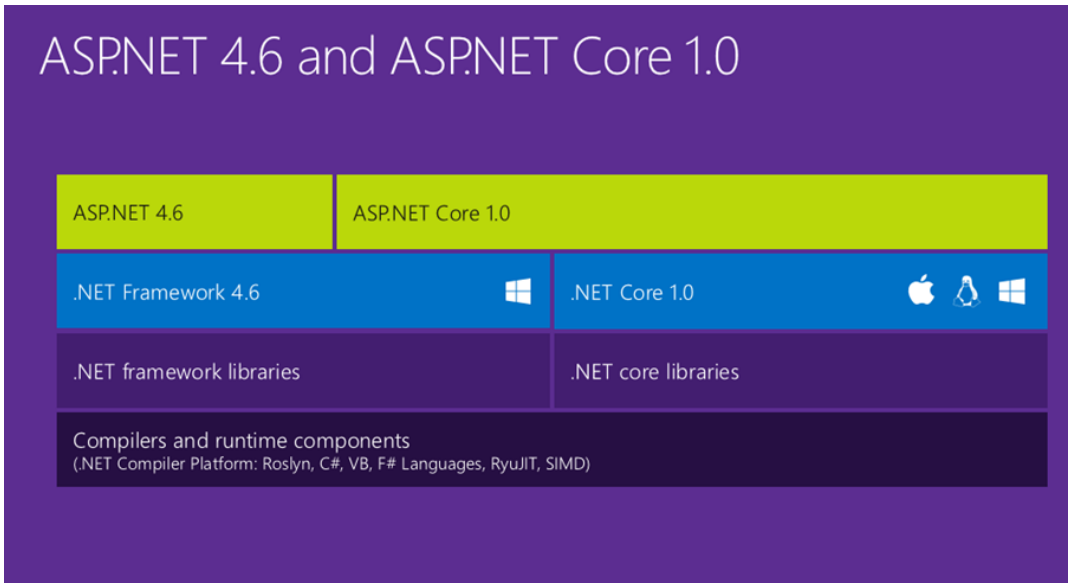
2.7 Responsiivisuus

Mobiililaitteiden yleisyyden takia kaikki nykyajan verkkosivut tulisi suunnitella niin, että ne näyttävät hyvältä kaikilla laitteilla. Vanhan tyylinen kovakoodattujen kokojen käyttäminen toimii vain sillä näyttökoolla, jolle se on suunniteltu, joten tarvitaan uusi lähestymistapa. Tähän pulmaan auttaa responsiivinen suunnittelu. Responsiivisuus tarkoittaa, että verkkosivu esittää sisältönsä selkeästi näytön koosta riippumatta, esimerkiksi järjestelemällä tai jättämällä joitain elementtejä näyttämättä pienemmillä näytöillä.

Responsiivisuutta voidaan luoda CSS3:n media queryillä, niiden avulla pystytään sivun tyylitiedostossa määrittämään HTML-elementeille eri tyylejä näytön koosta riippuen (Marcotte, 2010). Jos sivulla on esimerkiksi valikkopalkki, voidaan se näyttää tavallisesti tietokoneen näytöltä katsottuna, mutta älypuhelimella sivulle navigoitaessa media query tarkistaa näytön koon ja käyttää pienempää tyyliä valikkopalkille, jolloin sivun käytettävyys paranee, kun yläpalkki ei vie neljäsosaa näytöstä.

2.8 .NET Framework vai .NET Core

Ennen projektin aloittamista oli selvitettävä, kannattaisiko työ tehdä perinteisellä .NET Framework komponenttikirjastolla vai uudella .NET Corella (kuva 4).



KUVA 4. .NET Framework ja .NET Core (Hanselman, 2016.)

Framework (kuva 4) on alun perin 2002 julkaistu komponenttikirjasto, jonka uusin versio on 4.6.2. Frameworkin tavoitteina on tarjota sovelluskehittäjille johdonmukainen olioperustainen ohjelmointiympäristö, jossa ohjelmiston käyttöönotto on helppoa ja versioiden yhteensopivuusongelmat mahdollisimman vähäisiä. Lisäksi Framework kannustaa turvalliseen koodin suorittamiseen sekä pyrkii pääsemään eroon scriptien ja kääntäjien aiheuttamista suorituskykyongelmista. Myös johdonmukainen kehityskokemus erilaisten sovellusten välillä sekä alan standardien mukaan toimiminen ovat osa .NET Frameworkin tavoitteita (Microsoft, 2016b.)

Core on vuonna 2016 julkaistu modulaarinen komponenttikirjasto, jonka ei sinällään ole tarkoitus korvata Frameworkia vaan toimia sen rinnalla. Se tukee tällä hetkellä vain C#-ohjelmointikieltä, mutta tuettuja kieliä on tulossa lisää. Suurin ero Frameworkiin on mahdollisuus kehittää järjestelmäriippumatonta koodia, jolloin sovelluksia voidaan kehittää Windows-, macOS- sekä Linux-alustoille. Core on lisäksi avointa lähdekoodia, joten käyttäjillä on mahdollisuus muokata siitä tarpeidensa mukainen (Microsoft, 2016a.)

3 TOTEUTUS

3.1 Valitut tekniikat

Projektissa päädyttiin käyttämään .NET Frameworkia, koska asiakkaan ympäristössä on pelkkiä Windows-koneita ja koska Core ei vielä tue kaikkia kolmannen osapuolen .NET-kirjastoja. Myöskään Coren mahdollisesti tuoma lisäsuorityskyky ei olisi merkittävä asiakkaan pienessä järjestelmässä (CodeChannels, 2016).

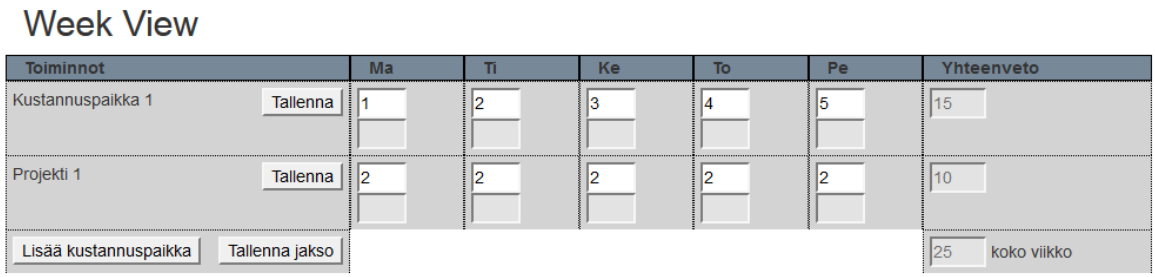
Angular vai Knockout selvityksen jälkeen ei projektissa loppujen lopuksi käyttetty kumpaakaan komponenteista, lukuun ottamatta pientä Angular-kokeilua töiden tarkistussivulla. MVC:n tukemalla Razor-merkintäkielellä sai aikaan hyvän lopputuloksen ilman lisäkirjastoja. Razorin avulla voidaan luoda palvelimen päässä C#- tai VisualBasic-koodista verkkoselainten ymmärtämiä HTML-elementtejä. Razor ei ole osa MVC:tä, vaan se toimii missä tahansa ympäristössä.

Projektin alussa oikeuksien tarkistus tapahtui ASP.NET:n valtuutuksilla, mutta projektin lopulla tehtiin oma valtuutusluokka, joka peri alkuperäisen Authorize-luokan ja kävi varmistamassa sovelluksen Web.config-tiedostosta, millä käyttäjillä ja käyttäjäryhmillä on oikeudet nähdä mitäkin sivuja.

Sivustolta vaadittuja ominaisuuksia esitellään lähemmin seuraavissa alaotsikoissa.

3.2 Työajankirjaussivun ensimmäinen versio

Projektin alussa sovellukseen tarvittavista ominaisuuksista ei ollut vielä täysin varmuutta. Aloin tehdä mielikuvani mukaista viikkotyötuntisivua (kuva 5), joka kuitenkin osoittautui lopulta hyödyttömäksi ja jouduin tekemään koko työtuntien tallennuksen uudelleen tavoitteiden selkeydyttyä.



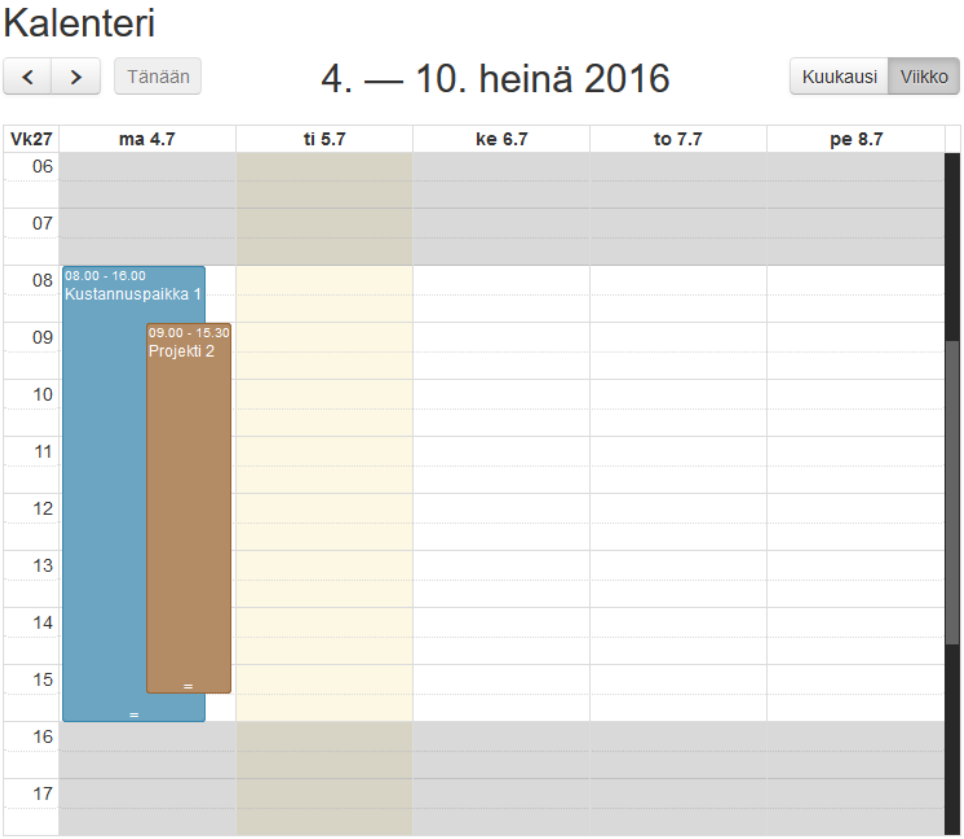
KUVA 5. Työajankirjauksen ensimmäisen versio

3.3 Työtapahtumien kirjaus

Työtapahtumien kirjaus oli sivuston pääominaisuus, jossa työntekijät voivat merkitä, missä kustannuspaikassa töitä tekevät. Aluksi suunnitelmissani oli tehdä yksinkertainen listanäkymä, johon käyttäjä voisi merkitä eri päivien tuntimäärät kustannuspaikoittain.

Lähestymistapa kuitenkin osoittautui ongelmalliseksi, koska työtapahtumiin piti saada merkittyä myös lisätietoja, kuten mahdolliset projektit. Jouduin etsimään paremman ratkaisun, jolloin löysin FullCalendar-nimisen JavaScript-lisäosan, joka näytti sopivan täydellisesti sovelluksen tarpeisiin. FullCalendarin helppokäyttöisyys, siisti ulkoasu ja lukuisat ominaisuuksia ja säätöjä säästivät usean viikon työn.

FullCalendar-toteutuksen työtapahtumien kirjauksessa käyttäjä voi merkitä työtuntinsa ja kustannuspaikkansa lukujärjestysmäisessä ympäristössä ja saada visuaalista palautetta töiden tilanteista (kuva 6).

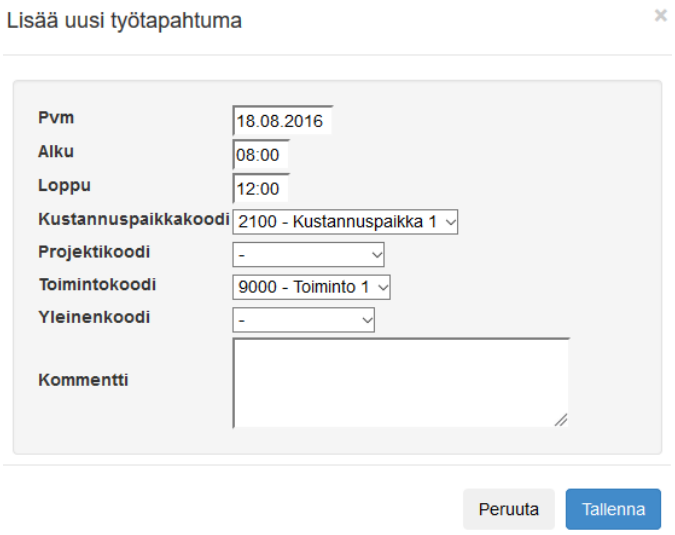


KUVA 6. FullCalendar-toteutuksen ensimmäinen versio

Kalenterissa käyttäjä näkee töiden tilanteen eri värein merkittynä (kuva 7). Sininen on lähettämätön, oranssi lähetetty, vihreä hyväksytty, punainen hylätty, keltainen osittain hyväksytty ja harmaa muusta järjestelmästä peräisin oleva työmerkintä. Kalenteria on mahdollista selata kuukausi-, viikko- tai päivänäkymässä.

KUVA 7. Erivaiheisten työtapahtumien värit esitettynä ohjesivulla

Aluksi lisävalinnat, kuten projektien lisääminen kustannuspaikkoihin, tehtiin luomalla useita päällekkäisiä työtapahtumia, joissa kussakin oli haluttu lisätieto. Tämä tapa osoittautui kuitenkin sekavaksi, kun lisätietoja annettiin enemmän kuin yksi. Myös työtuntien laskennassa tämä tapa olisi aiheuttanut ylimääräistä työtä.



KUVA 8. Työtapahtumien lisääminen

Uusien työtapahtumien lisääminen onnistuu klikkaamalla kalenteria halutusta kohtaa tai klikkaamalla ja raahaamalla haluttu alue, jolloin tapahtumalle voidaan asettaa sekä alku että loppu ilman, että tarvitsee näppäillä aikoja tekstikenttiin. Testitietokannasta löytyviä (kuva 8) tapahtumaominaisuuksia ovat kustannuspaikka, projekti, toiminto ja yleinen, joista muodostetaan lisäysikkunaan lista. Ainoastaan kustannuspaikka on merkitty tietokannassa pakolliseksi, joten siihen ei luoda tyhjää vaihtoehtoa. Kommenttikentän avulla voidaan välittää töiden tarkastajille ylimääräistä vapaamuotoista tietoa tai huomautuksia työpäivästä, esimerkiksi projektin etenemisestä.

Lopullisessa versiossa (kuva 9) työtapahtumien lähetysnappi avaa valintaikkunan, josta voi lähettää joko uudet eli lähettämättömät työt tai korjatut eli esimiehen hylkäämät ja käyttäjän korjaamat työt. Töiden keston muokkaaminen onnistuu vetämällä kursorilla tapahtuman alareunasta ja venyttämällä

tai kutistamalla työ haluttuun mittaansa. Töitä voi myös liikuttaa raahaamalla ne haluttuun sijantiinsa. Päällekkäiset tapahtumat on estetty niin tapahtumia luotaessa kuin muokatessakin.

KUVA 9. Lopullinen työkalenteri

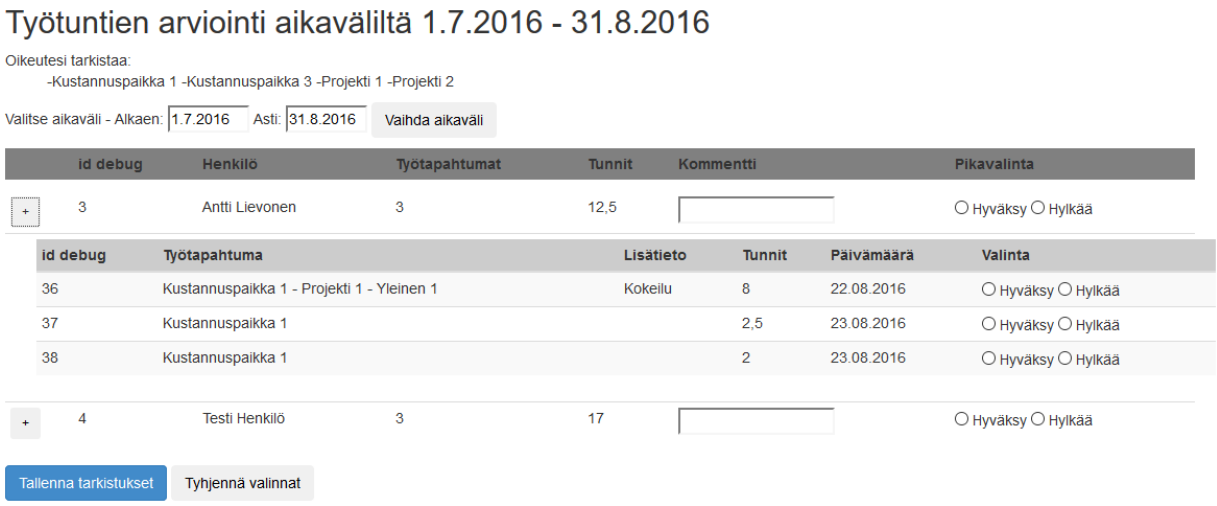
3.4 Työtapahtumien tarkastus

Sivuston toinen kriittinen osuus oli työtapahtumien tarkastussivu, jossa esimiehen oli tarkoitus päästä näkemään ja arvioimaan työntekijöiden lähettämät työmerkinnät joko hyväksytyiksi tai hylätyiksi.

Ensimmäinen versio arvioinnista oli käyttäjähallintasivulla, josta oikeutettu käyttäjä pääsi näkemään samantyylisen kalenterinäkymän kuin itse työntekijäkin, mutta ilman mahdollisuutta muokata, lisätä tai poistaa tapahtumia. Esimies pystyi joko hylkäämään tai hyväksymään kaikki näkyvissä olevat työtapahtumat, esimerkiksi viikon kerrallaan. Käytettävyydeltään tämä ratkaisu ei ollut erityisen hyvä, sillä yksittäisten työtapahtumien arviointi oli mahdotonta, jos päivässä oli enemmän kuin yksi tapahtuma. Tämä arviointisivuversio jätettiin esimiehille työntekijöiden tapahtumakalenterin näkemistä varten, mutta arviointiominaisuus poistettiin käytöstä.

Alkuperäiseltä tarkastussivulta puuttui myös asiakkaan tarvitsema ominaisuus, jossa jokaiselle kustannuspaikalle, projektille ja tapahtumalle piti pystyä määrittämään omat tarkastajat. Ilman arviointioikeuksien tarkistusta kuka tahansa esimiesoikeudet saanut voisi tarkastella ja arvioida kaikkia yrityksen työntekijöiden työtapahtumia. Lisäksi kaikki työt olivat aina joko hylättyjä tai hyväksyttyjä.

Uusi tarkastussivuversio (kuva 10) mahdollistaa työtapahtumien massahyväksynnän, mutta samalla myös yksittäisten tapahtumien arvioinnin. Sivu listaa tarkastajalle vain ne työtapahtumat joihin hänellä on oikeus ja joita ei ole vielä hänelle oikeutetuilta osin tarkastettu kenenkään toimesta. Osittain hyväksytyille tapahtumille kehitettiin väri erottamaan ne täysin tarkastamattomista tapahtumista. Työntekijän näkökulmasta työ on hyväksytty vasta kun kaikki työhön valitut osuudet on hyväksytty. Jos yksikin tarkastaja hylkää yhdenkään työosuuden, merkitään työ kokonaisuutena hylätyksi ja lähetetään takaisin työntekijälle korjattavaksi.



KUVA 40. Työtapahtumien tarkastussivu

Tarkastussivulla on käyttöä helpottavia ominaisuuksia, kuten listaus kaikista työtyypeistä, joita kirjautuneella käyttäjällä on oikeus tarkastaa sekä mahdollisuus valita aikavälin, jolta hyväksyttävissä olevat työtapahtumat näytetään. Jokaisella käyttäjärivillä on myös pikavalinta, jolla saa nopeasti asetettua käyttäjän kaikki valintapainikkeet hyväksytyiksi tai hylätyiksi.

Sähköposti-ilmoituksen lähetys työtuntien tarkastuksen yhteydessä oli yksi asiakkaan toiveista. Tarkastajan on mahdollista jättää arvioinnin yhteydessä kommentti lähetettäväksi kullekin työntekijälle. Sähköpostissa kerrotaan myös tarkastajan perustiedot sekä listataan kaikki tarkastetut työtapahtumat ja niiden tilanteet.

3.5 Käyttäjienhallinta

Käyttäjienhallinnan kautta pystyy luomaan ja poistamaan käyttäjiä sekä muokkaamaan käyttäjätietoja. Käyttäjienhallinnassa pystytään myös vaikuttamaan esimies-alaissuhteisiin. Sivulle mennessä tarkistetaan, kenen esimies kirjautunut käyttäjä on ja listataan vain heidät muokattavaksi. Käyttäjienhallinnan kautta esimies voi nähdä myös alaistensa kalenteritapahtumat samaan tapaan kuin käyttäjät itse, mutta ilman mahdollisuutta muokata tai lisätä työtapahtumia.

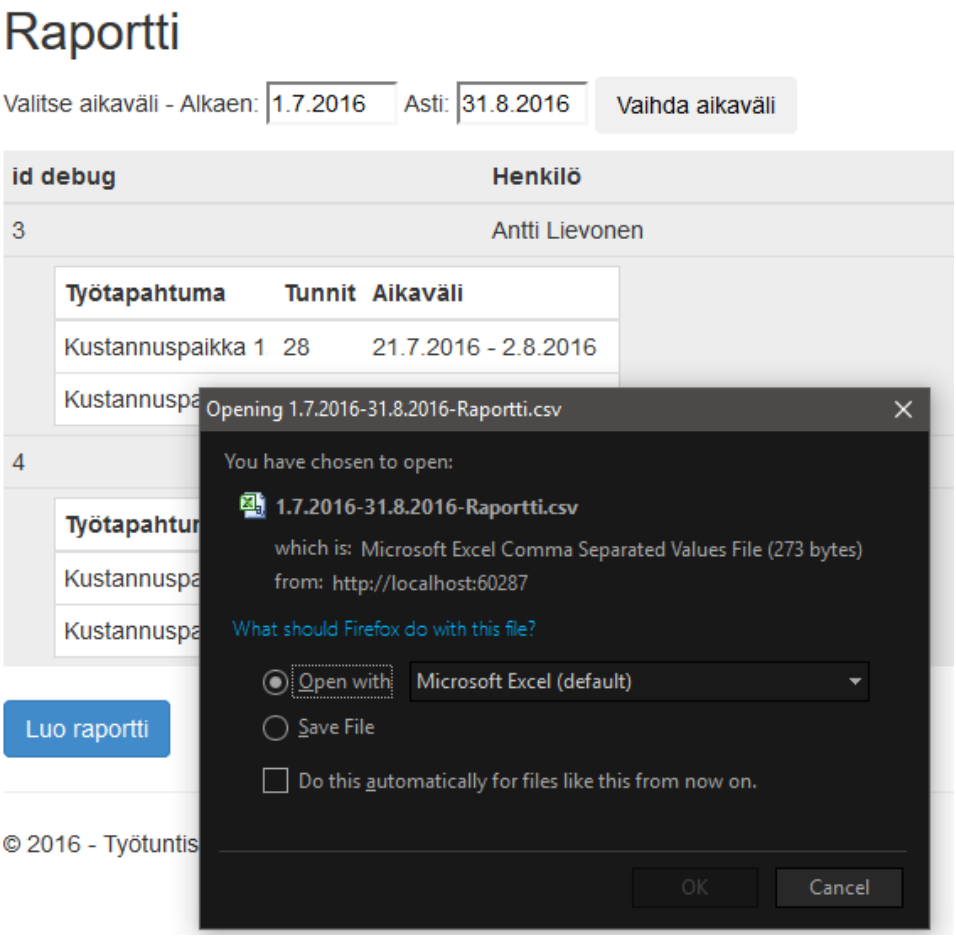
Käyttäjienhallinnan toinen osa on tarkastajien hallinta. Tarkastajienhallinassa voi asettaa jokaiselle kustannuspaikalle ja lisävalinnoille halutut henkilöt tarkastajiksi. Hyväksyjille voidaan antaa

voimassaoloaika, joka tarkistetaan aina tarkastussivulle edetessä. Tarkastaja näkee vain ne työtyypit, joissa hänen tarkastusoikeutensa alkamisaika on ennen nykyhetkeä ja päättymisaika on vielä tulevaisuudessa.

3.6 Raportointi

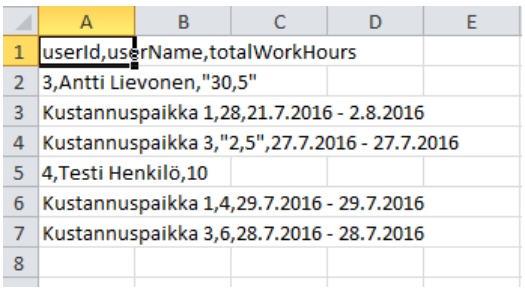
Viimeinen vaadituista ominaisuuksista oli raporttien luominen. Työntekijöiden työtapahtumista piti pystyä luomaan raportteja esimerkiksi palkanlaskentahenkilöstön käyttöön. Linkki raporttisivulle löytyy käyttäjienhallintasivulta. Raporttiin voidaan määritellä aikaväli, jolta työtapahtumat haetaan.

Raporttisivulla esitetään kaikki käyttäjien tapahtumakalentereihinsa merkitsemät työtapahtumayhdistelmät. Raportissa näytetään vain kokonaan hyväksytyt tapahtumat. Jos samaa työtapahtumayhdistelmää löytyy useita kappaleita, yhdistetään ne raportissa samalle riville ja niiden työtunnit lasketaan yhteen. Myös käyttäjän kokonaistunnit ilmoitetaan kaikkien aikavälillä tapahtuneiden työtapahtumien yhteenlaskettuna tuntimääränä. Raportti näyttää myös jokaiselle työtapahtumayhdistelmälle aikavälin, eli ensimmäisen ja viimeisen ajankohdan kyseiselle yhdistelmälle.



KUVA 5. Raportinluontisivu ja raportin tallennus

Kun luo raportti -painiketta painetaan, avautuu selaimen normaali tiedoston tallennusikkuna (kuva 11. Tiedosto on csv-muotoinen ja sisältää samat tiedot kuin esikatselusivu.



	A	B	C	D	E
1	userId,userName,totalWorkHours				
2	3,Antti Lievonen,"30,5"				
3	Kustannuspaikka 1,28,21.7.2016 - 2.8.2016				
4	Kustannuspaikka 3,"2,5",27.7.2016 - 27.7.2016				
5	4,Testi Henkilö,10				
6	Kustannuspaikka 1,4,29.7.2016 - 29.7.2016				
7	Kustannuspaikka 3,6,28.7.2016 - 28.7.2016				
8					

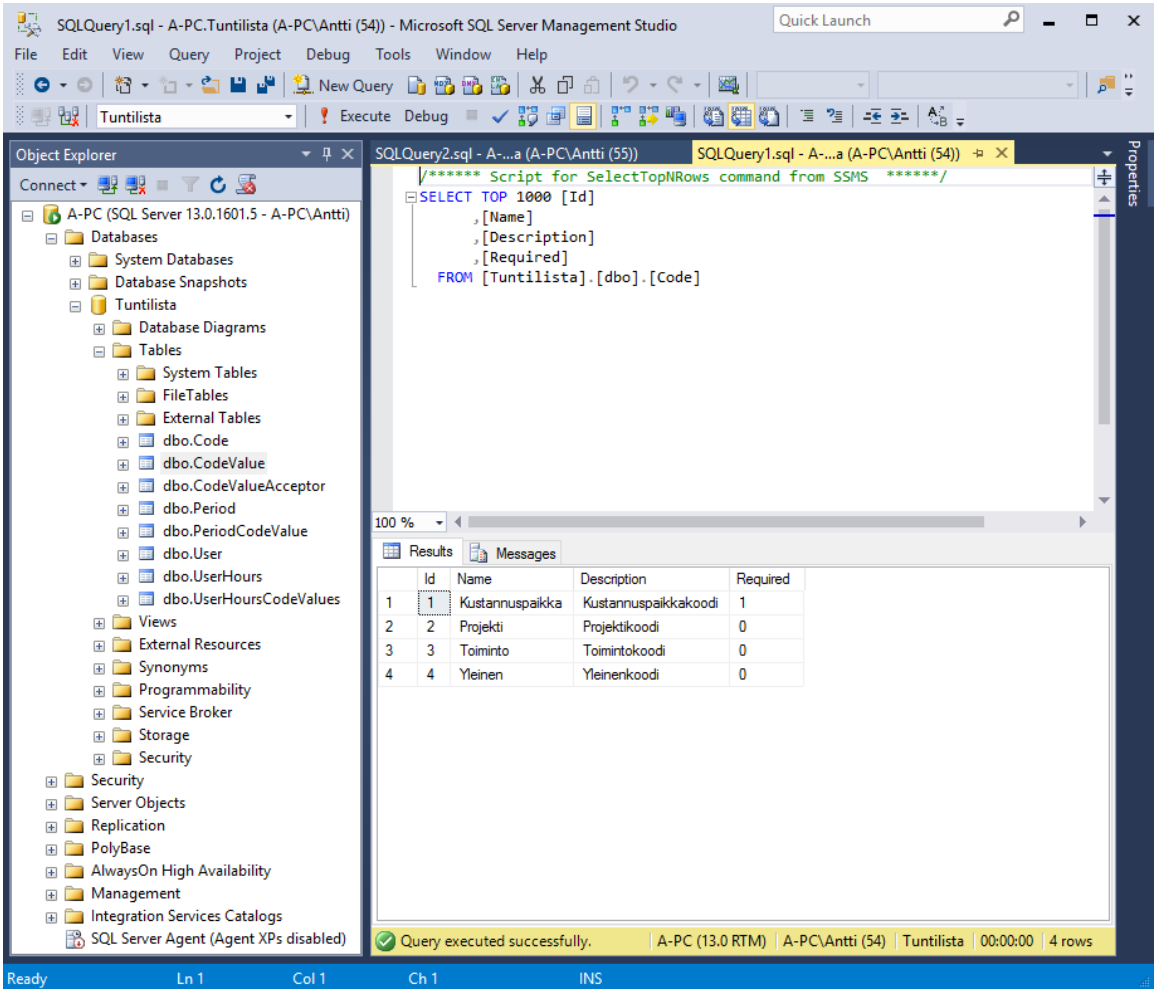
KUVA 6. Luotu raportti avattuna Excelissä

Raportin ulkoasu ja sisältö ovat mahdollisesti projektin keskeneräisin ominaisuus. Asiakkaan raporttiin tarvitsemista tiedoista ollut projektin aikana varmuutta, joten päädyttiin tallentamaan yleisimmät perustiedot (kuva 12).

4 KOMPONENTIT

4.1 SQL-server

Sovelluksessa käytettiin SQL-tietokantaa, jota hallittiin Microsoft SQL Server 2016 Manament Studion kautta (kuva 13). Management Studiolla on helppo suunnitella ja muokata tietokantoja ja tauluja sekä niiden yhteyksiä toisiinsa graafisen käyttöliittymän ansiosta.

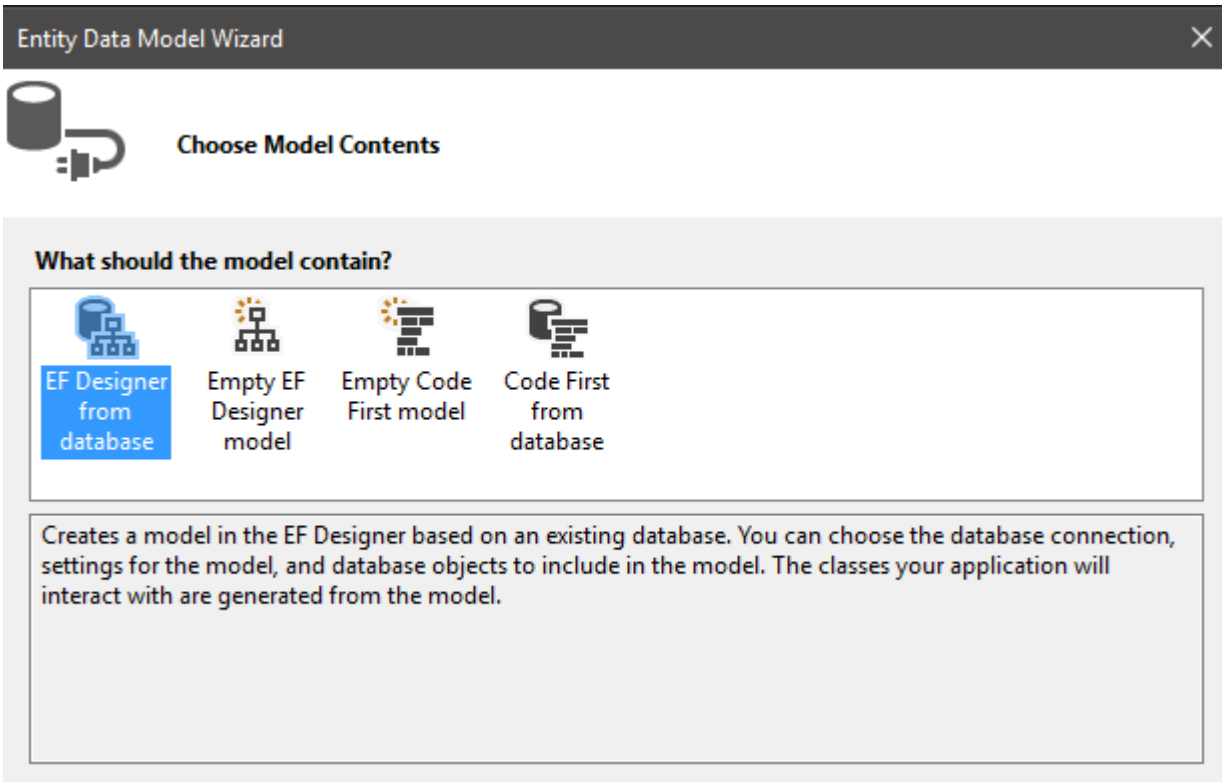


KUVA 7. SQL Server Management Studion käyttöliittymä

Tietokantaan kirjoittamiseen ja sieltä lukemiseen verkkosovelluksen kautta käytettiin Entity Frameworkia. Entity Framework (EF) on ORM-kirjasto (Object-Relational Mapping), joka hoitaa sovelluksen koodin ja tietokannan relaatiomallin yhteyden toisiinsa (Microsoft, 2015). ORM-kirjastojen tarkoituksena on minimoida tarvittava koodaaminen ja säätäminen tietokantayhteyksien luonnissa.

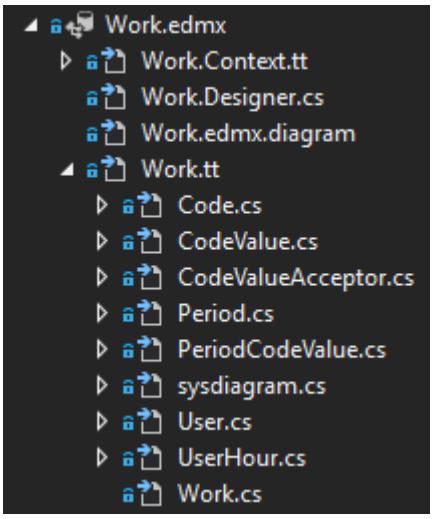
Entity Frameworkia voi käyttää kolmella tapaa. Ensimmäiseksi, jos tietokanta on jo olemassa, voidaan siitä luoda luokat sovellukseen. Toiseksi, jos luokat ovat jo olemassa sovelluksessa, voidaan niistä luoda tietokanta. Kolmanneksi, voidaan suunnitella tietokannan malli ja luoda siitä sekä tietokanta että sovelluksen luokat. (EntityFrameworkTutorial.net, 2016)

Sovellukseen luotiin luokat tietokannan pohjalta. Visual Studiolla saa luotua helposti tietokantayhteyden Entity Data Model Wizardilla (kuva 14). Yhteyden määrittämisessä luodaan yhteys SQL-serveriin ja valitaan haluttu tietokanta ja taulut.



KUVA 8. EF-yhteyden muodostaminen Visual Studiossa

Kun yhteys on muodostettu, luo Visual Studio automaattisesti luokat valituista tauluista (kuva 15). Luokkia voi käyttää tietokannasta lukemiseen ja sinne kirjoittamiseen. Esimerkiksi uuden rivin lisääminen Period-tauluun tapahtuu luomalla uusi Period-muuttuja ja lisäämällä siihen tarvittavat CodeValue-yhteydet ja muut tiedot, sitten Period lisätään EF-yhteyden muodostamisen aikana luotuun Periods-listaan ja tallennetaan muutokset.



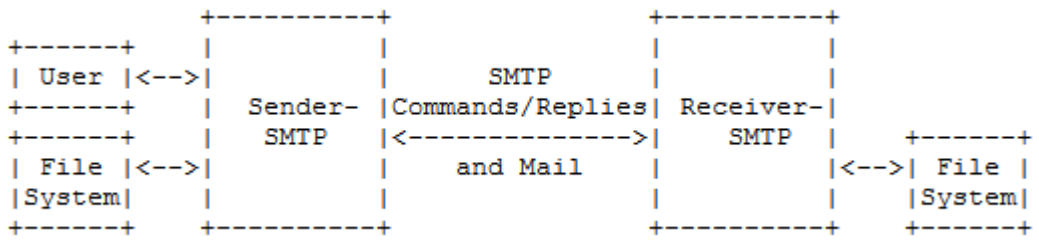
KUVA 95. Tietokannasta luodut luokat

4.2 FullCalendar

FullCalendar on MIT-lisensoitu avoimen lähdekoodin jQuery-lisäosa, joten sen käyttö on sallittua myös kaupallisissa ohjelmistoissa. Sen avulla saa tehtyä helposti tyylikkäitä kalenterikomponentteja. Lisäosa tarjoaa hyvät perusominaisuudet, joita tässäkin projektissa hyödynnettiin. Sivunlatauksen yhteydessä FullCalendar saa käsittelijältä JSON-muotoiset (JavaScript Object Notation) tapahtumatiedot, joista se tarvitsee pelkästään tapahtuman otsikon ja alkamisajankohdan. Lisätietoina voi antaa muun muassa loppumisajankohdan tai värin. Projektissa tapahtumille on lisäksi lisätty hyväksymispäivä, muokkauspäivä ja kommenttitieto.

4.3 Sähköpostin lähetys ja Postal

.NET Framework tarjoaa vakiona työkalut sähköpostin lähettämiseksi verkkosovelluksesta. Ohjelma lähettää viestin tiedot Simple Mail Transfer Protocol (SMTP) -palvelimelle, joka hoitaa viestin lähetyksen lopulliselle vastaanottajalle. SMTP määriteltiin vuonna 1982 julkaistussa RFC 821 internetstandardissa (ISOC, 1982). Sen tarkoituksena oli mahdollistaa sähköpostin lähetys luotettavasti ja tehokkaasti.



KUVA 10. SMTP:n toimintamalli alkuperäisestä määrittelydokumentista (ISOC, sivu 1, 1982.)

Viestin lähetyksessä on monta vaihetta (kuva 16). Ensin vastaanottajalle lähetetään viestin lähettäjän verkkotunnus, jonka vastaanottaja kuittaa. Seuraavaksi lähetetään lähettäjän sähköpostiosoite. Jos viesti voidaan vastaanottaa, lähetetään takaisin kuittaus. Sitten lähetetään yksi tai useampi sähköpostiosoite, johon viestiä ollaan lähettämässä. Vastaanottaja palauttaa tiedon siitä, voiko kyseisiin osoitteisiin lähettää viestiä. Lopuksi lähetetään viestin sisältö, jos vastaanottaja sai käsiteltyä sisällön, lähetetään kuittaus. Lopuksi yhteys katkaistaan. (Microsoft, 1998)

Postal on MVC-lisäosa, jolla voi lähettää sähköpostia käyttäen näkymää viestin pohjana. Postal on myös MIT-lisensoitu, eli se on kaikkien vapaasti käytettävissä. Jokaista tarkistettua käyttäjää kohden luodaan lista työtapahtumista, jotka on tarkistettu ja lisätään ne viestiin, yhdessä tarkistajan tietojen ja kommentin kanssa. (kuva 17). Työntekijä saa sähköpostiinsa kootut tiedot yksinkertaisena viestinä (kuva 18). Viestien lähetyksestä vastaavan sähköpostipalvelimen asetukset tallennetaan web.config-tiedostoon, josta Postal osaa käydä lukemassa ne tarvittaessa.

```
Hei @Html.Raw(Model.RecipientName as string)

@Html.Raw(Model.AcceptorName as string + "(" + Model.From as string + ") on tarkistanut työtapahtumiasi:")

@foreach(string str in Model.EvaluatedPeriods)
{
    @Html.Raw("- " + str + "\r\n");
}


Tarkistajan viesti: @Html.Raw(Model.AcceptorComment as string)
```

KUVA 11. Sähköpostiviestin pohjana toimiva MVC-näkymä

Työtuntisovellus

tuntikirjaus@tcdcon.com
ke 10.8.2016 12:49

Vastaanottaja: ☐ Antti H Lievonen;

Antti Lievonen (antti.h.lievonen@edu.savonia.fi) on tarkistanut työtapahtumiasi:

- 4.8.2016 6h 0min, Kustannuspaikka 1, Projekti 1 - hyväksytty
- 3.8.2016 5h 0min, Kustannuspaikka 1 - hyväksytty
- 9.8.2016 3h 0min, Kustannuspaikka 1 - hyväksytty

Tarkistajan viesti: Kaikki hyväksytty

KUVA 12. Tarkistettujen työtapahtumien ilmoitusviesti

4.4 Elmah

Projektiin haluttiin virheiden kirjaustoiminto, johon tarkoitukseen työn tilaaja suositteli ELMAH (Error Logging Modules and Handlers) ASP.NET-lisäosaa. ELMAH on helppo asentaa ja se toimii heti vakioasetuksilla (Mitchell & Aziz, 2012). Virheet tallennetaan XML-tiedostoiksi ErrorLogs-kansioon, josta niitä voi käydä selaamassa. Vaihtoehtoisesti virheet voisi laittaa tallentumaan tietokantaan. ELMAH:n mukana tulee myös virhesivu (kuva 19), jossa listataan kaikki tapahtuneet virheet ja niiden tiedot. Virhesivulta on mahdollista tarvittaessa suodattaa turhat virheet, kuten sivua ei löydy, eli virhe 404.

localhost:60287/elmah

Error Log for ROOT on A-PC

RSS FEED | RSS DIGEST | DOWNLOAD LOG | HELP | ABOUT

Errors 1 to 1 of total 1 (page 1 of 1). Start with 10, 15, 20, 25, 30, 50 or 100 errors per page.

Host	Code	Type	Error
A-PC	404	**Http**	The controller for path '/qwe' was not found or does not implement IController. Details...

KUVA 19. ELMAH-virhesivu

4.5 CsvHelper

Työntekijöiden tunneista generoidaan asiakkaalle raportti CSV-tiedostona (Comma-Separated Values). CSV-tiedostojen parsiminen onnistuu toki omatekoisellakin koodilla, mutta on otettava huomioon paljon pieniä ongelmatilanteita. Esimerkiksi kenttien erotinmerkin ollessa pilkku ja datan sisältäessä pilkkuja, kuten tietokannasta luetussa lauseessa voisi olla. Raportinluonnissa käytettiin siis CsvHelperiä työmäärän vähentämiseksi. CsvHelper on .NET-kirjasto, joka hoitaa CSV-tiedostojen lukemisen ja kirjoittamisen kätevästi sekä osaa pitää huolta, ettei virheellisiä kenttiä synny ja data pysyy halutussa muodossa (Close, 2016).

5 JATKOKEHITYS

Projektiin jäi muutamia keskeneräisiä tai parannuksia vaativia ominaisuuksia pääasiassa, koska asiakasyrityksen yhteyshenkilö oli kesälomalla juuri kun tein ohjelmistoa. Yksityiskohtien puuttuessa jouduin tekemään perustoiminnallisuuden ja jättämään joitain asioita tilaajan kehitettäväksi.

Raportin ulkoasua voisi kehittää ja muokata asiakkaan tarvitsemat käyttäjä- ja työtiedot nykyisten yksinkertaisten mallitietojen pohjalta. Kalenterin lukitsemista viimeisimpään lähetettyyn työtapahtumaan asti harkittiin, mutta se jätettiin toteuttamatta. Ominaisuus on helppo lisätä jälkeenpäin asiakkaan niin halutessa.

Kalenterinäkymän käyttöliittymä olisi käyttäjän kannalta varmasti informatiivisempi, jos tapahtumissa kerrottaisiin mitkä osat ovat hyväksttyjä, tarkastamatta tai hylättyjä. Peruskäyttäjiltä olisi myös luultavasti hyvä piilottaa ominaisuudet, joihin heillä ei ole oikeuksia. Tällainen on esimerkiksi työtuntien tarkastelu, siitäkin huolimatta, että käyttöoikeudet tarkistetaan eri sivuille siirryttäessä.

6 YHTEENVETO

Opinnäytetyöprojektin tavoitteena oli suunnitella ja kehittää työtuntijärjestelmä, joka soveltuisi tilaajayrityksen asiakkaan tarpeisiin. Sovelluksen tuli sisältää ominaisuuksia niin työntekijälle, kuin esimiehille, työtuntivastaaville sekä palkanlaskentahenkilöstölle.

Heti projektin alussa työn tekoa hidasti vähäiset tiedot asiakkaan tarpeista. Ensimmäinen testiversio työnkirjaussivusta jouduttiin hylkäämään, sillä se ei vastannut työn tilaajan näkemystä sivusta. Toiseen versioon löydettiin kuitenkin FullCalendar-lisäosa, joka sopi täydellisesti projektiin. Sen avulla oli helppo luoda visuaalisesti miellyttävä työajankirjausjärjestelmä, jonka ympärille muut ominaisuudet kehitettiin.

Projektin kankean alun jälkeen työ lähti etenemään hyvää vauhtia. Eri ominaisuuksien ja käytettyjen komponenttien suunnittelu työn alussa nopeutti työskentelyä. Työn vaatima MVC-arkkitehtuuri oli jo ennestään tuttu, mutta opin myös paljon uutta, varsinkin erilaisista lisäosista ja komponenttikirjastoista.

Työn tilaaja oli tyytyväinen projektin etenemiseen ja lopulliseen versioon. Kaikki vaaditut ominaisuudet saatiin pääosin tehtyä.

LÄHTEET JA VIITTAUKSET

Close, J. (2016). *CSV Helper*. Haettu 24. Lokakuuta 2016 osoitteesta GitHub: https://joshclose.github.io/CsvHelper/

CodeChannels. (2016). *When should I still use .NET Framework 4.x, instead of .NET Core?* Haettu 11. Syyskuuta 2016 osoitteesta CodeChannels: http://www.codechannels.com/article/microsoft/when-should-i-still-use-net-framework-4-x-instead-of-net-core/

EntityFrameworkTutorial.net. (2016). *What is Entity Framework?* Haettu 24. Lokakuuta 2016 osoitteesta Entity Framework Tutorial: http://www.entityframeworktutorial.net/what-is-entityframework.aspx

Envato. (2016). *MVC for Noobs*. (Envato) Haettu 7. Syyskuuta 2016 osoitteesta tutsplus: http://code.tutsplus.com/tutorials/mvc-for-noobs--net-10488

Google. (2016a). *AngularJS*. (Google) Haettu 5. Syyskuuta 2016 osoitteesta https://angularjs.org/

Google. (2016b). *Features & Benefits - Angular*. Haettu 23. Lokakuuta 2016 osoitteesta Angular: https://angular.io/features.html

Hanselman, S. (2016). *ASP.NET 5 is dead - Introducing ASP.NET Core 1.0 and .NET Core 1.0*. Haettu 27. Lokakuuta 2016 osoitteesta Hanselman: http://www.hanselman.com/blog/ASPNET5IsDeadIntroducingASPNETCore10AndNETCore10.aspx

ISOC. (Elokuuta 1982). *Simple Mail Transfer Protocol*. Haettu 24. Lokakuuta 2016 osoitteesta The Internet Engineering Task Force: https://www.ietf.org/rfc/rfc0821.txt

Knockout. (2016a). *GitHub*. Haettu 20. Lokakuuta 2016 osoitteesta Knockout versions: https://github.com/knockout/knockout/releases

Knockout. (2016b). *Knockout*. Haettu 5. Syyskuuta 2016 osoitteesta http://knockoutjs.com/

Marcotte, E. (2010). *Responsive web design*. Haettu 23. Lokakuuta 2016 osoitteesta A List Apart: http://alistapart.com/article/responsive-web-design

Microsoft. (1998). *Simple Mail Transfer Protocol*. (Wrox Press) Haettu 26. Lokakuuta 2016 osoitteesta Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/aa480435.aspx

Microsoft. (2000). *Active Directory Users, Computers, and Groups*. Haettu 27. Lokakuuta 2016 osoitteesta Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/bb727067.aspx

Microsoft. (2015). *Entity Framework*. Haettu 24. Lokakuuta 2016 osoitteesta Data Developer Center: https://msdn.microsoft.com/en-us/data/ef(v=vs.113).aspx

Microsoft. (2016a). *.NET Core Guide*. (Microsoft) Haettu 11. Syyskuuta 2016 osoitteesta Microsoft Documentation: https://docs.microsoft.com/en-us/dotnet/articles/core/index

Microsoft. (2016b). *Overview of the .NET Framework*. (Microsoft) Haettu 11. Syyskuuta 2016 osoitteesta Microsoft Documentation: https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx

Mitchell, S.;& Aziz, A. (2012). *ELMAH*. Haettu 24. Lokakuuta 2016 osoitteesta Google Developers: https://code.google.com/p/elmah/

Sukesh, M. (2016). *WebForms vs. MVC*. Haettu 20. Lokakuuta 2016 osoitteesta CodeProject: http://www.codeproject.com/Articles/528117/WebForms-vs-MVC

LIITE 1: SOVELLUKSEN SIVUKARTTA