Ajankäytön seurannan sovelluskehitysprojekti

Veikko Soosaar

Haaga-Helia ammattikorkeakoulu
Tradenomi (YAMK)
Liiketoiminnan teknologiat
Opinnäytetyö
2023

Tiivistelmä

Tekijä(t)
Veikko Soosaar.

Tutkinto
Tradenomi (YAMK).

Raportin/Opinnäytetyön nimi
Ajankäytön seurannan sovelluskehitysprojekti

Sivu- ja liitesivumäärä
54

Opinnäytetyössä tehdään sovelluskehitysprojekti, jonka tuloksena syntyy käyttökelpoinen sovelluksen prototyyppi. Kyseessä on ajanseurannan sovellus, jolla voidaan seurata ajan käyttöä erilaisten käyttäjän määräämien tehtävien parissa. Näin käyttäjä saa itselleen kerättyä tietoa omasta ajankäytöstään. Käyttäjä voi sovelluksesta saamansa tiedon perusteella tehdä analyysejä ja päätöksiä.

Opinnäytetyö etenee johdannon jälkeen perustamaan teoreettista tietopohjaa sovelluskehitysprojektille. Sovellus perustuu suurilta osin Microsoftin teknologioihin kuten .NET ja C#. Teoriapohjan jälkeen esitellään projektissa käytettäviä työkaluja, joilla sovellus tuotetaan. Seuraavaksi opinnäytetyössä siirrytään sovelluksen määrittelyihin. Sovellukseen tehdään käyttäjän vaatimusmäärittelyiden lisäksi myös toiminnallinen ja tekninen vaatimusmäärittely. Määrittelyiden jälkeen opinnäytetyössä tehdään sovelluksen eri osien suunnittelua, jonka jälkeen esitellään sovelluksen toteutusta. Opinnäytetyön lopuksi analysoidaan toteutunutta sovellusta ja pohditaan opinnäytetyön onnistumista.

Opinnäytetyössä saatiin toteutettua toimiva sovelluksen prototyyppi, jota ei kuitenkaan pidetä vielä valmiina julkiseen laajempaan käyttöön. Opinnäytetyön suunnittelu aloitettiin vuoden 2023 alussa ja opinnäytetyö, sekä siinä toteutettava sovellus valmistuivat toukokuussa 2023.

Asiasanat
ohjelmistokehitys, C#, SQL, .NET, Razor

Sisällys

1 Johdanto

Opinnäytetyössä rakennetaan tietopohja ja toteutetaan sovelluskehitysprojekti teoreettisesta tietopohjasta saatuja tietoja käyttäen. Sovellukselle tehdään määrittelyt eri näkökulmista ja näiden määrittelyjen pohjalta tehdään suunnittelutyötä. Suunnittelutyön valmistumisen jälkeen ryhdytään luomaan sovellusta määrittelyiden ja suunnittelun pohjalta. Määrittelyn, suunnittelun ja toteutuksen tukena toimii opinnäytetyön alussa tehty tietopohja.

Kehitettävällä sovelluksella halutaan antaa käyttäjälle mahdollisuus seurata ajan käyttöä eri tehtävien parissa. Sovelluksella käyttäjä pystyy kirjaamaan eri työhön tai vapaa-aikaan liittyviin tehtäviin ajankäyttöään. Näiden kirjausten perusteella sovelluksen käyttäjä pystyy hyödyntämään tietoa päätöksenteossa ja analysoimaan omaa ajankäyttöään laajemminkin.

1.1 Tavoitteet opinnäytetyölle ja kehittämiskysymys

Pääasiallinen tavoite tässä opinnäytetyössä on tuottaa toimiva prototyyppi tai konsepti mahdollisesta käytettävästä sovelluksesta ajankäytön hallintaan. Toisena tavoitteena on oppia ja kehittää omaa osaamista sovelluskehitysprojektin eri osa-alueilta. Oppimista kehitetään rakentamalla sovelluksen kehitykselle vahva tietoperusta. Lisäksi sovellukselle tehdään kattavat määrittelyt eri näkökulmista. Määrittelytyön lisäksi sovellukselle tehdään suunnittelutyötä. Tavoitteena on oppia näistä jokaisesta osa-alueesta ja hyödyntää näissä olevaa osaamista sovelluksen toteuttamisessa.

Opinnäytetyölle on asetettu kehittämiskysymykseksi: ”Miten ja millä välinein voidaan tehdä sovelluskehitysprojekti?”. Kysymykseen haetaan vastausta käymällä läpi eri sovelluskehityksen vaiheita, kuten määrittelyjen tekoa, suunnittelua ja toteutusta. Lisäksi käydään läpi tekniikoita ja työkaluja, joita käytetään tämän sovelluskehitysprojektin aikana. Lopullinen vastaus muodostuu projektikokonaisuudesta, joka tämän opinnäytetyön puitteissa luodaan.

1.2 Opinnäytetyön kulku

Tämän opinnäytetyö rakentuu kahdeksasta pääluvusta. Tämän ensimmäisen johdantoluvun jälkeen tuleva toinen luku sisältää teoreettisen pohjan sovelluskehitysprojektissa käytettävistä teknologioista. Kolmannessa luvussa käydään läpi projektissa hyödynnettävistä työkaluista, eli ohjelmista ja palveluista, joita sovelluksen toteuttamisessa käytetään. Neljäs luku sisältää sovelluksen määrittelyyn liittyvää dokumentaatiota useista eri näkökulmista ja viidennessä luvussa suunnitellaan kehitysprojektissa toteutettava sovellus. Tämän jälkeen tulee kuudes luku, jossa kerrotaan sovelluksen toteutuksesta ja näytetään esimerkkejä sovelluksessa käytetyistä ratkaisuista. Seitsemännessä luvussa analysoidaan toteutunutta sovellusta ja pohditaan mahdollisuuksia sovelluksen

kehittämiseen liittyen. Viimeinen, eli kahdeksas luku sisältää pohdinnan opinnäytetyön onnistumisesta.

1.3 Keskeiset käsitteet

Sovellus	Sovellus eli ohjelma on tietokonekoneohjelma, jolla voidaan tehdä yhtä tai useampia toimintoja. Tässä opinnäytetyössä tehdään verkkosovellus, eli verkossa toimiva sovellus.
Tietokanta	Tietotekniikassa tietokannalla viitataan kokoelmaan tietoa, mitä säilytetään elektronisissa tietosäilöissä.
Palvelin	Palvelimella tarkoitetaan niin fyysistä tietokonetta, kuin myös ohjelmistoa, jossa on käytössä ohjelmisto millä on tarkoitus tarjota toimintoja toisille ohjelmistoille.
Ohjelmointi	Tietokoneelle tai muulle laitteelle ohjelmointikielellä kirjoitetulla koodilla tehtävä prosessi, jonka lopputuloksena on sovellus.
Ohjelmointikieli	Ohjelmointikieli on kieli, jolla on formaali tyyli ja sillä toteutetaan ohjelmointia.
Koodi	Ohjelmointikielellä toteutettu koodi, ohjelmakoodi tai lähdekoodi, jonka avulla toteutetaan sovellus.
Versionhallinta	Tekniikka tai järjestelmä, jolla pidetään kirjaa tiedostoihin ja kansioihin syntyneistä muutoksista.
Autentikointi	Autentikointi eli todennus on tapa varmistaa, että käyttäjä on se, jona häntä pidetään.

2 Käytettävät tekniikat ja teknologiat

Tässä kohtaa opinnäytetyötä käymme läpi sovelluskehitysprojektissa käytettäviä tekniikoita ja teknologioita. Tekniikat ja teknologiat esitellään lyhyesti ja kerrotaan niiden käyttötarkoituksesta ja hyödyistä sovelluskehitysprojektiin liittyen. Tulen myös avaamaan syitä miksi nämä valinnat tehtiin käytettävien tekniikoiden ja teknologioiden osalta. Suurimman osan suhteen voidaan tiivistää syyksi se, että kyseiset tekniikat ja teknologiat ovat entuudestaan tuttuja ja toimivat hyvin yhdessä toistensa kanssa.

2.1 Projektin etenemisen hallinta

Sovelluskehityksen projektin etenemistä seurataan Microsoftin Azure DevOps palvelussa käyttämällä Kanban-taulua. Taulussa on kolme vaihetta, joiden välillä työtehtävät liikkuvat. Näistä vaiheista ensimmäinen on “To Do” eli siinä ovat tehtävät mitkä pitäisi tehdä, ja ne eivät ole vielä työn alla. “Doing” eli asiat mitkä ovat työn alla on keskimmäinen vaihe projektin tehtäville. Lopuksi viimeisenä vaiheena on “Done” eli tehtävät, jotka on saatettu loppuun ja niiden parissa ei tarvitse enää työskennellä. Projektin etenemisen hallintaan ei tarvita laajempaa tai monimutkaisempaa metodia tai järjestelmää, koska projekti on ulottuvuudeltaan kohtuullisen pieni ja tiimi koostuu tasan yhdestä jäsenestä, opinnäytetyön tekijästä. Liian monimutkainen projektinhallinta veisi ylimääräistä aikaa ja pahimmassa tapauksessa jopa hankaloittaisi projektin etenemistä.

2.1.1 Kanban

Kanban on laajasti käytössä oleva ketterän kehityksen viitekehys, jossa painotetaan ajantasaista kommunikointia ja täyttä läpinäkyvyyttä tehtävän työn osalta. Kanban muodostui alun perin 1940-luvun loppupuolella Toyotan tuotantoprosessien käytössä olevista tavoista. Tästä on on kehittynyt monipuolinen ja tehokas menetelmä monilla eri toimialoilla nykypäivänä, joista yksi suosittuja on ohjelmistokehitys. Kanbanin konsepti liittyy kanban-tauluun, minkä ympärillä Kanbania käyttävän tiimin tekeminen keskittyy. Tavallisen tyyppinen kanban-taulu koostuu kolmesta osiosta: tehtävät, jotka pitää tehdä, tehtävät, joita tehdään tällä hetkellä ja tehtävät mitkä on jo tehty. Riippuen kuitenkin työskentelevän tiimin koosta tätä taulua voidaan muokata esimerkiksi laajemmin kattavammaksi tiimin tarpeet huomioiden. Kanbanissa jokainen työtehtävä on eritelty omaksi kortikseen tällä kanban-taululla. Kanban voidaankin kääntää japanin kielestä tarkoittamaan visuaalista merkkiä. Nämä kortit sisältävät oleelliset tiedot tehtäväksi tarkoitetun työn suorittamiseen liittyen. Näin tiimin jäsenille välittyy tieto siitä mikä kyseisen työtehtävän tilanne on milläkin hetkellä. Yleensä tällaisella kortilla on tieto siitä, kuka on vastuussa kyseisen työn tekemisestä, pieni kuvaus tehtävästä työstä ja kuinka kauan työn tekemiseen arvioidaan menevän. Kortille voidaan tietenkin lisätä paljon

muutakin tietoa ja nykypäivänä onkin yleistä esimerkiksi lisätä sovelluskehityksen yhteydessä ruutukaappaus helpottamaan tehtävän määritystä. (Radigan s.a.)

Yhteenvetona voidaan todeta Kanbanin toimivan tehokkaana viitekehyksenä ketterälle kehittämiselle, tarjoten sitä käyttäville tiimeille joustavuutta, tehokkuutta ja läpinäkyvyyttä tekemisen suhteen. Visuaalinen tyyli ja työnkulun muokattavuus tarjoavat keinoja tiimin tuloksen parantamiseksi.

2.2 Sovellus

Sovellus tullaan kehittämään selainsovelluksena pääosin Microsoftin teknologioita hyödyntäen. Microsoftin tarjoama .NET 6 viitekehys ja sen päällä toimiva ASP.NET Core 6 viitekehys toimivat sovelluksen perustana. Ohjelman logiikka tehdään C#-ohjelmointikielellä palvelinpuolelle Razor Pages viitekehyksen avulla. Selainpuoli ohjelmasta kirjoitetaan suurimmaksi osaksi HTML merkintäkielellä, paikoin hyödyntäen Razorin syntaksia. Sivun ulkomuotoa muokataan CSS tyylisivuja hyödyntäen. Sovelluksen kirjautuminen ja autentikointi tehdään käyttämällä ASP.NET Core Identity rajapintaa. Pääasiallinen syy näiden teknologioiden valintaan sovelluksen kehittämiseksi on se, että käytän niitä itse päivittäisessä työssäni. Nämä teknologiat kuitenkin kehittyvät jatkuvasti ja uusimpien versioiden mukana tulee uusia ominaisuuksia, joita opetella. HTML ja CSS ovat selainsovellusten perustana ympäri verkkoa ja ne ovatkin laajassa käytössä joko suoraan tai välillisesti viitekehysten kautta luotuna. Viitekehysten, ohjelmointikielien ja teknologioiden osalta löytyy kuitenkin niin selainpuolen kuin palvelinpuolenkin puolelta paljon vaihtoehtoja. Tämän opinnäytetyön pääasiallinen tarkoitus on kuitenkin sovelluskehitysprojekti ja sen tuotos, joten uusien teknologioiden ja ohjelmointikielien opettelua pyrittiin välttämään sovelluksen kehityksen osalta.

2.2.1 .NET 6

Microsoft on luonut .NET alustan monenlaisten sovellusten kehittämiseksi eri käyttöalustoille. .NET tukee useita eri kieliä, editoreita ja kirjastoja. Kielet, joita .NET tukee ovat C#, F# ja Visual Basic. NuGet paketinhallinta tarjoaa yli 100 000 eri pakettia .NET kehittämisen tueksi. (Microsoft 2023a)

Microsoft kehitti .NET 6 version modernien, tehokkaiden, skaalautuvien sovellusten luomiseksi useille eri alustoille. Tämä versio julkaistiin marraskuussa 2021. Kyseessä on LTS eli pitkäaikaisen tuen versio, mikä tarkoittaa vähintään kolmen vuoden ajan päivityksiä ja tukea. Tässä uusimmassa pitkäaikaisen tuen versiossa on tullut uusia suoritustehon parannuksia ja hyötyjä, mitkä vähentävät palvelinten ylläpitokustannuksia. .NET 6 tukee myös entistä paremmin useammalle alustalle kohdistuvaa kehitystyötä tuki uusille versioille .NET tukemista kielistä ja viitekehyksistä kuten esimerkiksi C# versio 10 ja ASP.NET Core 6. Yksi uusi hyödyllinen ominaisuus on myös “Hot Reload”, mikä mahdollistaa sovelluksen muutosten näkymisen kehitysympäristössä ilman sovelluksen uudelleenrakentamista ja uudelleenkäynnistystä. (Lander 8.11.2021)

2.2.2 ASP.NET Core 6

ASP.NET Core on avoimen lähdekoodin versio suositusta verkkosovellusten kehitykseen käytetystä ASP.NET viitekehyksestä. ASP.NET Core julkaistiin ensimmäisen kerran 2016 ja se on luotu ajettavaksi Windows, Linux, macOS ja Docker ympäristöihin. Microsoft on kehittänyt ASP.NET Core viitekehystä pyrkien tekemään siitä nopea ja tehokkaan vaihtoehdon muihin verkkosovellusten kehitykseen käytettävien viitekehysten rinnalle. Tämän viitekehyksen voi asentaa sovelluksen mukana tai jopa yhteisesti palvelimelle sovellusten käytettäväksi. (Microsoft 2023b)

Microsoft julkaisi ASP.NET Core 6 version marraskuussa 2021 .NET 6 julkaisun yhteydessä. Tämä uusi versio tarjoaa .NET 6 kanssa samalla tavalla muun muassa uuden “Hot Reload” ominaisuuden, joka mahdollistaa sivuston muutosten näkemisen kehitysympäristössä ilman koko sovelluksen uudelleenrakentamista ja uudelleenkäynnistystä. Uudessa versiossa tulee myös mukana tuki uuteen mahdollisuuteen julkaista sovellus pienemmällä määrällä koodia. ASP.NET Core 6 versiossa tuli myös päivityksiä valmiina oleviin viitekehyksiin kuten esimerkiksi Bootstrap 5.1 versio. (Roth 8.11.2021)

2.2.3 C#

C#, englanniksi lausuttuna “see sharp”, on moderni olioperusteinen ohjelmointikieli. C# on myös vahvasti tyypitetty kieli mahdollistaen kehittäjille tyyppiturvallisen kehityskielen. C-perheen ohjelmointikielet kuten C, C++, Java ja JavaScript ovat samantyyppisiä kuin C# ja tästä johtuen kieli on helposti omaksuttavissa edellä mainittujen kielien osaajille. Komponenttien tekeminen C# ohjelmointikielellä on tyypillistä sen ollessa olioperusteisuuden lisäksi komponenttiperusteinen. Kuvassa 1 näytämme esimerkkiohjelmiston, joka on luotu C#-ohjelmointikielellä. Tyypillisessä C#-ohjelman koodissa löytyy aluksi “using”, jolla viitataan käytettävien kirjastojen tai luokkien nimiavaruuksiin. Tämän jälkeen “class” merkinnän jälkeen tulee luokan nimi, mikä on tässä esimerkkitapauksessa “Huomenta”. Luokan sisällä meillä on tässä tapauksessa yksi metodi “Main()”, joka alkaa kommenttirivillä. Kommenttirivi alkaa kahdella vinoviivalla “//”. Kommentin jälkeen meillä on metodikutsu “Console.WriteLine(..)”, joka kutsuu metodia, jolla kirjoitetaan konsoliin viesti. Metodin sisällä on sitten “Huomenta Suomi!” viesti, joka tulee konsoliin näkyviin, kun sovellus käynnistetään. (Kuva 1) Tämä ohjelmointikieli tarjoaa useita eri ominaisuuksia sisäänrakennettuna tukeakseen ohjelmistokehittäjän kehitystyötä. Tällaisia ominaisuuksia ovat esimerkiksi virheidenkäsittely, asynkroniset toimenpiteet ja LINQ-syntaksi. C# toimii .NET alustan päällä hyödyntäen siinä saatavilla olevia kirjastoja ja viitekehyksiä. (Microsoft 13.2.2023)

```
using System;

class Huomenta
{
    static void Main()
    {
        // Allaoleva rivi tulostaa konsoliin "Huomenta Suomi!"
        Console.WriteLine("Huomenta Suomi!");
    }
}
```

Kuva 1. Esimerkki C#-ohjelmasta (Veikko Soosaar)

Tässä sovelluskehitysprojektissa käytetään C# ohjelmointikieltä kaiken palvelinpuolella tapahtuvan toiminnan ohjelmointiin. Palvelinpuolella tapahtuvalla toiminnalla tarkoitetaan esimerkiksi tietokantaan kohdistuvia toimintoja, raporttien muodostamista ja tiedon käsittelyä. Tietokantaan kohdistuu tiedon hakua ja tiedon muutoksia kuten poistoja, päivityksiä ja lisäyksiä. Raporttien muodostamisen yhteydessä käsitellään tietoa ja muodostetaan siitä käyttäjälle halutussa muodossa oleva raportti.

2.2.4 HTML

HTML on internetin pääasiallinen merkintäkieli. Alun perin HTML suunniteltiin semanttista kuvailua varten tieteellisiin asiakirjoihin. HTML yleinen malli tosin on antanut sille mahdollisuuden tulla otetuksi käyttöön vuosien varrella monien erilaisten dokumenttien kuvailuun ja merkintään. Ensimmäisen viiden vuoden aikana 1990–1995 vuosina HTML koki useita erilaisia uudistettuja versioita, joita oli käytössä pääasiallisesti ensin CERNissä ja sitten myös IETFssä. W3C perustamisen jälkeen HTMLn kehitys muuttui ja 1995 yritettiin julkaista versio HTML 3.0, josta kuitenkin kehitettiin käytännönläheisempi versio 3.2 joka julkaistiin 1997. HTML4 tuli kuitenkin vuonna 1997 ja se oli pitkään käytössä, kunnes 2000- ja 2010-luvuilla alettiin kehittää HTML5 versiota ja standardia. HTML-dokumentti useimmiten koostuu elementeistä, jotka on voitu laittaa sisäkkäin ja nämä elementit sisältävät aloitusmerkinnän kuten esimerkiksi “<title>” ja lopetusmerkin kuten esimerkiksi “</title>” (kuva 2). Elementeissä voi olla myös sisällön lisäksi aloitusmerkinnän sisällä attribuutteja, jotka ilmaisevat lisätietoja kyseisestä elementistä. Esimerkkikuvassa näkyy “<a href=”lisatietoja.html”>” a-elementin aloitusmerkintä, jossa href-attribuutti esimerkiksi viittaa linkkiin, joka tässä tapauksessa johtaa “lisatietoja.html” verkkosivulle tämän kyseisen sivuston sisällä. Aloitus- ja lopetusmerkinnät tulee kirjoittaa oikein ja sisäkkäisyys tulee tehdä oikein, jotta käyttäjälle näkyvä verkkosivu näkyy oikeanlaisesti. (WHATWG 6.4.2023)

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Esimerkkisivu</title>
 </head>
 <body>
  <h1>Esimerkkiotsikko</h1>
  <p>Tämä on esimerkki ja <a href="lisatietoja.html">lisäietoja</a>-sivun esimerkkilinkki.</p>
  <!-- kommentti -->
 </body>
</html>
```

Kuva 2. Esimerkki HTML-dokumentista (Veikko Soosaar)

Opinnäytetyön sovelluskehitysprojektissa HTML merkintäkieltä verkkosivuston rakentamisessa. Käyttäjän näkemä verkkosivuston alla on HTML avulla rakennettu runko kuten lomakkeet ja painikkeet. Tätä HTML koodia sitten täydentää CSS avulla tehdyt tyylittelyt.

2.2.5 CSS

Cascading Style Sheets eli CSS on yksinkertainen mekaniikka tyylin ja muotoilun kuvailemiseksi verkkosivuilla (W3C s.a.). Bert Bos (17.12.2016) kirjoittaa CSS tarinan alkavan vuonna 1994 CERNissä Håkon Wium Lie huomatessa tyyliarkkien tarpeen verkkosivuille. Bos työsti itse Argo verkkoselainta ja päätyi tekemään yhteistyötä Lien kanssa tukemalla CSS käyttöönottoa. (Bos 17.12.2016)

CSS on sääntöpohjainen kieli ja se toimii kehittäjän määrittelemien sääntöjen mukaan. Kehittäjä määrittelee tyylisäännöt tietyille HTML-elementeille tai joukolle HTML-elementtejä. Esimerkiksi ensimmäisen tason otsikkoon käytettävälle elementille “h1” voidaan määritellä muun muassa väri ja fontin koko. Tämä tapahtuu kirjoittamalla CSS määrittelyille tarkoitetulla alueella kirjoittamalla kyseisen elementin jälkeen aaltosulut, joiden sisälle tyylimäärittelyt tulevat. Esimerkkikuvassa “h1”-elementille määritetään väriksi sininen ja fontin kooksi 4em. (Kuva 3) CSS on hyödyllinen ainoastaan, jos se on verkkoselaimet ovat ottaneet sen käyttöönsä. Verkkosivujen tekijän määrittelemät CSS-tyylittelysäännöt eivät toimi, ellei verkkoselain ole ottanut kyseisiä tyylittelyyn liittyviä ominaisuuksia käyttöönsä. (MDN 24.2.2023)

```
h1 {
  color: blue;
  font-size: 4em;
}
```

Kuva 3. Esimerkki CSS-syntaksista (Veikko Soosaar)

CSS tyylisääntöjä käytetään tässä opinnäytetyössä tehdyssä projektissa muodostamaan käyttäjälle miellyttävä ja käyttäjäystävällinen käyttökokemus. CSS-tyylittelyillä pyritään tässä sovelluksessa mahdollistamaan käyttäjälle helposti opittava käyttökokemus, jotta käyttäjälle muodostuu positiivinen mielikuva sovelluksen käytöstä ja hän kokee sovelluksen käytön helpoksi. Yksinkertaisimmillaan tällaiseen esimerkkinä voisi olla positiivisen toiminnon sisältävän painikkeen, kuten tallentamisen, merkitseminen vihreällä värillä. Vastavuoroisesti negatiivisen toiminnon, kuten poistamisen, sisältävä painike tulisi olla sitten punaisella värillä, jotta käyttäjä huomaa, että kyseessä voi olla jopa vaarallinen toiminto. Tässä opinnäytetyössä suoran CSS sijaan käytetään suurimmaksi osaksi Bootstrap nimistä viitekehystä. Bootstrap on CSS-viitekehys, joka antaa työkaluja verkkosivujen selainpuolen rakentamiseen nopeasti ja tarjoaa ison määrän erilaisia tyylejä käytettäväksi (Bootstrap s.a.).

2.2.6 Razor

Razor on ASP.NET verkkosovellusten viitekehykselle kehitetty merkintäsyntaksi. Razorilla on tarkoitus luoda dynaamisesti HTML-syntaksia käyttäen kehittäjille tuttuja kieliä kuten C# tai Visual Basic. Razorin käyttö ei edellytä mitään tiettyä editoria, vaan sitä voi käyttää millä tahansa editorilla ASP.NET viitekehystä käyttävien sovellusten kehityksessä. Kun Razoria kehitettiin, tarkoituksena oli kehittää siitä helposti opittava, nopea, sulava ja kompakti. Razorin tarkoitus on parantaa kehittäjän työtehokkuutta HTML merkintäkielen parissa. (Guthrie 3.7.2010)

Razorin syntaksissa yksi oleellisimpia asioita on tietää, että koodi alkaa sivulla, kun käytetään ät-merkkiä (@). Tämän jälkeen voidaan kirjoittaa esimerkiksi kaarisulkujen sisälle koodia, tai suoraan aiemmasta Razor syntaksilla kirjoitetusta koodista olevia muuttujia, jotka sitten tulostuvat sivulle. (FitzMacken 1.7.2022) Kuvassa 4 löytyy esimerkki Razorin syntaksista. Tässä esimerkissä kirjoitamme ensin Razorin koodilohkoon kaksi muuttujaa, viikonpaiva ja viikonpaivaViesti. Viikonpaiva muuttujaan haetaan DateTime rakenteesta viikonpäivä ja tämä sitten lisätään osittain valmiiksi kirjoitettuun viikonpaivaViesti-muuttujan viestiin. Lopuksi sitten tätä viikonpaivaViesti nimistä muuttujaa kutsutaan “<p>” elementin sisälle. (Kuva 4)

```
@{
    var viikonpaiva = DateTime.Now.DayOfWeek;
    var viikonpaivaViesti = "Tänään on: " + viikonpaiva;
}

<p>@viikonpaivaViesti</p>
```

Kuva 4. Esimerkki Razor syntaksista (Veikko Soosaar)

Sovellus, joka tässä opinnäytetyössä tehdään hyödyntää Razor syntaksia varsinkin tietokannasta haetun tiedon näyttämisessä. Razorin syntaksi mahdollistaa tiedon ympärille dynaamisesti rakennettavan HTML merkinnän ja näin saadaan tieto käyttäjälle näkyviin selkeästi.

2.2.7 Razor Pages

Microsoftin ASP.NET Core mukana esitelty Razor Pages on palvelinpuolella käytettävä verkkosivukeskeinen viitekehys. Razor Pages mahdollistaa sovelluksen rakentamisen vastuualueet puhtaasti eroteltuna. Razor Pages tukee Windows, Unix ja Mac ympäristöissä toimivia sovelluksia. Razor Pagesissa käytetään C# kieltä ja Razor syntaksia. Tämän viitekehyksen tarkoitus on tarjota kehittäjille verkkosivu-keskeinen toteuttaa sovelluksia. Razor Pagesin on tarkoitus olla helppo ottaa käyttöön ja sen skaalautuvuus sopii myös isommille ja kokeneemmille tiimeille. (Brind 18.2.2021)

Jotta voidaan hyödyntää Razor Pagesia niin se tulee ensin ottaa käyttöön ohjelman “Program.cs” tiedostossa. Tämän jälkeen Razor Pagesin selaimen puolelle näkyvän koodin tiedostot kuten esimerkiksi “Index.cshtml”, joka tässä tapauksessa esimerkkinä. Tälle tiedostolle luodaan “Index.cshtml.cs” niminen tiedosto, joka toimii tämän Index-sivun palvelinpuolen C# koodin lähteenä. Lisäksi tulee varmistaa pienet koodimuutokset, jotta tämä Razor Pages tekniikka toimii oikein ja tiedostot lukevat toisiaan. “Index.cshtml” tiedostoon tulee alkuun esimerkiksi “@model IndexModel” viittaus ja “Index.cshtml.cs” tiedostoon tulee luokka nimeltä “IndexModel” joka myös perii “PageModel” luokan. (Anderson, Brock & Larkin 25.3.2023)

Razor Pages on tämän opinnäytetyön sovelluskehityksessä käytössä. Iso osa sivuilla toimivasta logiikasta tapahtuu sivujen tiedostoihin liitetyllä kooditiedostolla, joka noudattaa Razor Pages toimintatapaa.

2.2.8 ASP.NET Core Identity

ASP.NET Core Identity on rajapinta, joka tarjoaa ominaisuuksia autentikointiin. Tällaisia ominaisuuksia ovat muun muassa käyttöliittymässä kirjautumisen, käyttäjien hallinnan, salasanat ja roolitukset. Rajapinta mahdollistaa käyttäjäkirjautumisen suoraan verkkosivulta uudella luodulla käyttäjätilillä tai vaihtoehtoisesti on myös mahdollisuus käyttää ulkopuolista kirjautumisen palveluntarjoajaa kuten Google, Microsoft tai Facebook. Tämä Identity rajapinta on useimmiten säädetty käyttämään SQL Server tietokantaa, mutta on myös olemassa mahdollisuus käyttää Azure Table Storage palvelua. (Anderson, R 1.12.2022)

Tämän opinnäytetyön puitteissa luodussa sovelluksessa käytetään tätä Identity rajapintaa käyttäjien autentikointiin ja kirjautumattomien käyttäjien pääsyn estäminen sovellukseen etusivua

pidemmälle. Sovelluksessa ei käytetä ulkopuolista kirjautumista, vaan käyttäjät luovat itse käyttäjätunnukset ja käyttäjätiedot tallennetaan SQL Server tietokantaan.

2.3 Tietokanta

Opinnäytetyössä tehtävälle sovellukselle tehdään myös tietokanta. Kyseessä on relaatiotietokanta, joka tukeutuu Microsoftin SQL Server relaatiotietokannan hallintajärjestelmään. Joitakin tietokantatoimintoja tehdään suorilla T-SQL kyselyillä, mutta iso osa tietokantaan kohdistuvasta toiminnasta tulee suoraan sovelluksen C# koodin kautta ilman itse kirjoitettuja T-SQL-lauseita. Seuraavaksi esitellään vähän taustatietoa aiheista SQL, T-SQL ja Microsoft SQL Server. Nämä ovat minulle päivätyöni ansiosta erittäin tuttuja teknologioita ja yksi iso syy näiden valitsemiselle. Valitsin nämä myös sen takia, että Microsoftin teknologioina ne toimivat hyvin muiden Microsoftin teknologioiden kanssa. Ei tietenkään olisi ongelmaa käyttää jotain muuta tietokantajärjestelmää SQL Serverin tilalla, kuten esimerkiksi MySQL tai MariaDb. Kuitenkin Microsoftin teknologioita käyttäessä koen hyötyväni siitä, että ne ovat samalta tekijältä.

2.3.1 SQL, T-SQL ja Microsoft SQL Server

SQL eli Structured Query Language kehitettiin IBM nimisen yrityksen tutkijoiden Raymond Boyce ja Donald Chamberlin toimesta. SQL oli aluksi yrityksen sisäiseen käyttöön, mutta useita vuosia myöhemmin se julkaistiin julkiseen käyttöön. Sittemmin muun muassa American National Standards Institute eli ANSI on todennut SQL olevan standardien mukainen kieli relaatiotietokantojen kanssa kommunikoimiseen. Eri yritysten tietokantajärjestelmien välillä voi olla eroavaisuuksia SQL kielen käytössä, mutta ne kaikki muistuttavat kuitenkin ANSI-hyväksyttyä versiota kielestä. SQL on kielenä todettu käyttäjäystävälliseksi ja sillä voidaan tehdä erittäin monipuolisia hakuja tietokantoihin. (Brooks 21.2.2023)

T-SQL on lyhennetty nimestä Transact-SQL, joka on SQL kielen laajennus ja sitä käytetään pääasiallisesti Microsoftin SQL Server relaatiotietokantojen kanssa. Kyseessä on siis SQL kielen kanssa hyvin läheinen kieli, mutta siinä on pieniä eroavaisuuksia ja lisäominaisuuksia, jotka tekevät siitä ainutlaatuisen. T-SQL ei ole avoimen lähdekoodin projekti toisin kuin SQL ja siinä on myös pieniä eroja syntaksissa. Yksi suurimpia eroja ovat kuitenkin lisäkomennot ja funktiot, joita T-SQL tarjoaa. Tällainen on esimerkiksi “ISNULL” niminen funktio, joka tarjoaa mahdollisuuden korvata NULL-arvon tietokannassa jollakin muulla arvolla. (Clark 4.3.2021)

Microsoft SQL Server on yksi pääasiallisista relaatiotietokantojen hallintajärjestelmistä, joita on markkinoilla saatavilla ja se palvelee monia erilaisia yrityksiä niin sovelluksien kuin analytiikankin yhteydessä. SQL Server on T-SQL kielelle pohjautuva ja sitä voidaan käyttää niin pilvipalveluna

kuin myös omalla palvelimella. Tarjolla on myös iso valikoima laajennuksia ja työkaluja tiedon hallintaan ja analytiikkaan. (Pérez 18.10.2021)

2.4 Versionhallinta

Sovelluksen kehityksessä käytetään versionhallintaa, jotta voidaan hallita sovellukseen tulevia muutoksia määrätietoisesti ja varmistaa mahdollisuus perua ei-haluttuja muutoksia. Versionhallinta myös mahdollistaa myöhemmin useamman ihmisen yhteistyön jouhevasti, kun on mahdollista käyttää versionhallintaa usean henkilön muutosten hallintaan. Tähän projektiin valittiin versionhallintaan Git, joka on minulle erittäin tuttu. Minulla on kokemusta myös muista versionhallintajärjestelmistä kuten esimerkiksi Subversion (svn), mutta Git on erittäin suosittu ja käytän sitä lähes päivittäin nykyisessä työssäni. Myös Gitin suosion takia se on hyvä valinta, koska näin voidaan toivoa sen olevan edelleen relevantti väline tulevaisuudessakin.

2.4.1 Git

Vuonna 2005 Linus Torvaldsin kehittämä Git on laajasti käytössä oleva moderni versionhallintajärjestelmä. Git on laajasti käytössä niin kaupallisilla toimijoilla, kuin myös avoimen lähdekoodin projekteilla. Hajautetun arkkitehtuurin ansiosta Git onkin hyvä esimerkki hajautetusta versionhallintajärjestelmästä. Tällä tarkoitetaan sitä, että toisin kuin joissakin muissa versionhallintajärjestelmissä, Gitissä jokaisella kehittäjällä, joka työskentelee säilötyn koodin parissa myös pitää hallussa koko koodisäilön versiohistoriaa. Tämä tarkoittaa käytännössä sitä, että esimerkiksi jos jokin kehittäjä kehittää ohjelman versiota 2.0, niin hän voi tekemiensä muutosten jälkeen siirtyä kehittämään vanhempaa versiota samasta sovelluksesta, kuten esimerkiksi versiota 1.3. Nämä erilliset versiot ovat omissa versionhallinnan haaroissa. Haarojen avulla voidaan työstää eri versiota tai ominaisuuksia eri kehittäjien toimesta samaan aikaan ilman sekaannuksia luotettavasti. Moni kokee Gitin oppimisen vaikeaksi, mutta tärkeimmät toiminnot oppiessaan kehittäjällä on käsissään erittäin tehokas työkalu. (Atlassian s.a.)

Tämän opinnäytetyön puitteissa Git pääasiallinen käyttö liittyy juuri nimenomaan sovelluksen versionhallintaan. Sovelluksesta tehdään vain yksi versio tätä opinnäytetyötä varten, mutta Gitin avulla voidaan esimerkiksi epämieluisien muutosten myötä palata vanhempaan, toimivaan versioon tai avata jopa uusi kehityshaara kokeellisten kehitysten merkeissä.

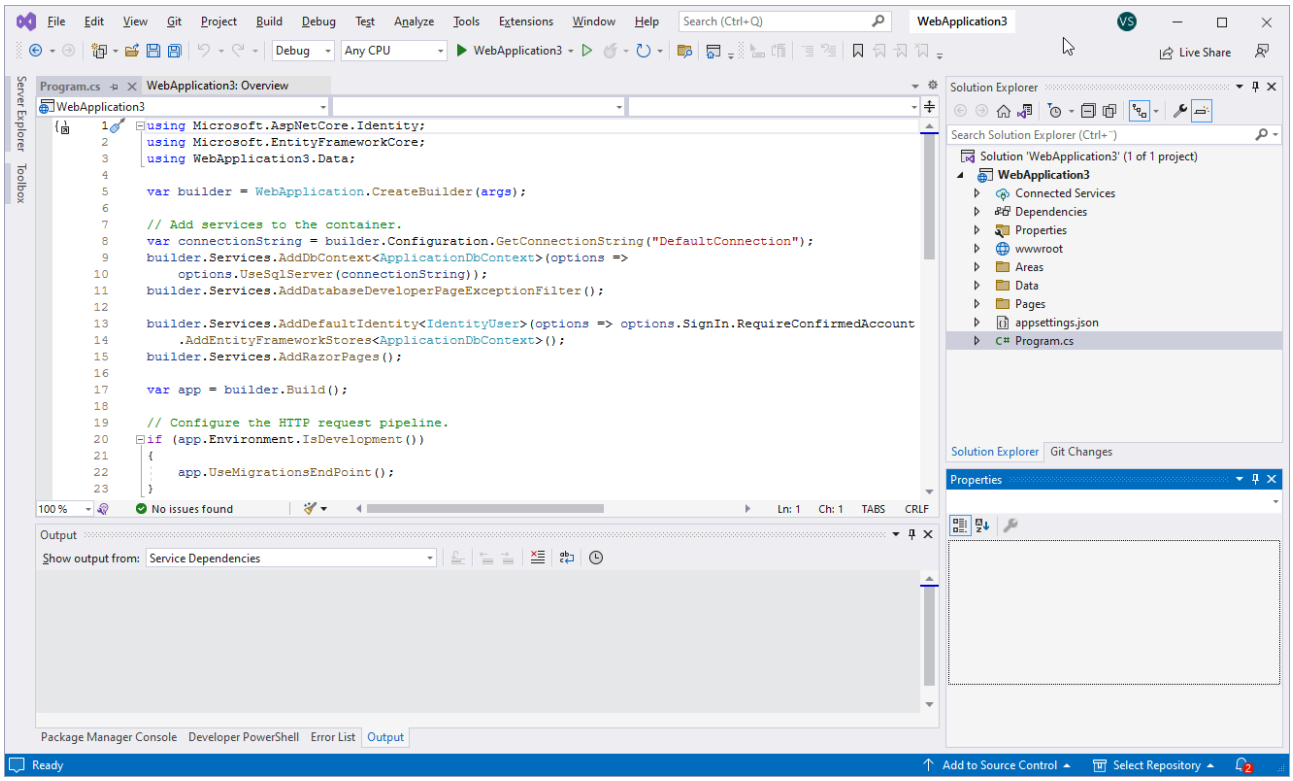
3 Käytettävät työkalut

Tässä opinnäytetyön osiossa esittelen tässä sovelluskehitysprojektissa käytettäviä työkaluja. Tarkoitus on esitellä lyhyesti käytettyjä työkaluja, kertoa niiden ominaisuuksista ja käyttötarkoituksista tämän sovelluskehitysprojektin yhteydessä. Suurin osa työkaluista on minulle vähintään hieman tuttuja ja se onkin yksi tärkeimpiä syitä kyseisten työkalujen valinnalle.

3.1 Microsoft Visual Studio 2022

Visual Studio 2022 on uusin versio Microsoftin kehittämästä ohjelmointiympäristöstä, joka sisältää kattavan valikoiman työkaluja ohjelmistojen luomiseen, levittämiseen ja virheiden paikantamiseen niiden sisällä. Tämä ohjelmointiympäristö tukee monia ohjelmointikieliä kuten esimerkiksi C#, C++ ja JavaScript. Visual Studio 2022 sisältää myös hyödyllisiä ominaisuuksia kuten IntelliCode, joka auttaa käyttäjää antamalla ehdotuksia ohjelmakoodin loppuun saattamiseksi. Tarjolla on myös työkaluja kirjoitetun koodin analysoimiseksi. Visual Studio 2022 yksi tärkeimpiä ominaisuuksia on tuki .NET viitekehykselle, mikä mahdollistaa kehityksen monille eri alustoille. (Microsoft, 2023c).

Tässä projektissa käyttöön valittu Visual Studio 2022 sisältää laajan tuen valitulle ohjelmointikielelle C# ja se on yksi merkitseviä syitä tämän työkalun valinnassa sen lisäksi, että se tukee projektiin valittua .NET viitekehystä. Suurin syy työkalun valinnalle on kuitenkin se, että Visual Studio 2022 on minulla käytössä päivittäin työssäni ja näin ollen saadaan sujuvoitettua opinnäytetyön etenemistä käyttämällä erittäin tuttua työkalua. Lisäksi tähän ohjelmointiympäristöön on integroitu muita hyödyllisiä työkaluja kuten versionhallintajärjestelmä, joka toimii tähän projektiin valitun Git kanssa. Lisäksi on myös käytettävissä projektinhallintaan liittyviä työkaluja, jotka toimivat Azure Devops kanssa yhdessä. Ohjelmointiympäristön virheiden paikannukseen tarjoamat työkalut ovat myös kattavat ja mahdollistavat projektin työstämisen tehokkaasti. Yksi erittäin hyödyllinen ominaisuus tavallisiin tekstieditoreihin verrattuna on koodikielen syntaksin väritys, eli syntaksin eri osat on värjätty eri värein niiden erottamiseksi toisistaan (kuva 5).



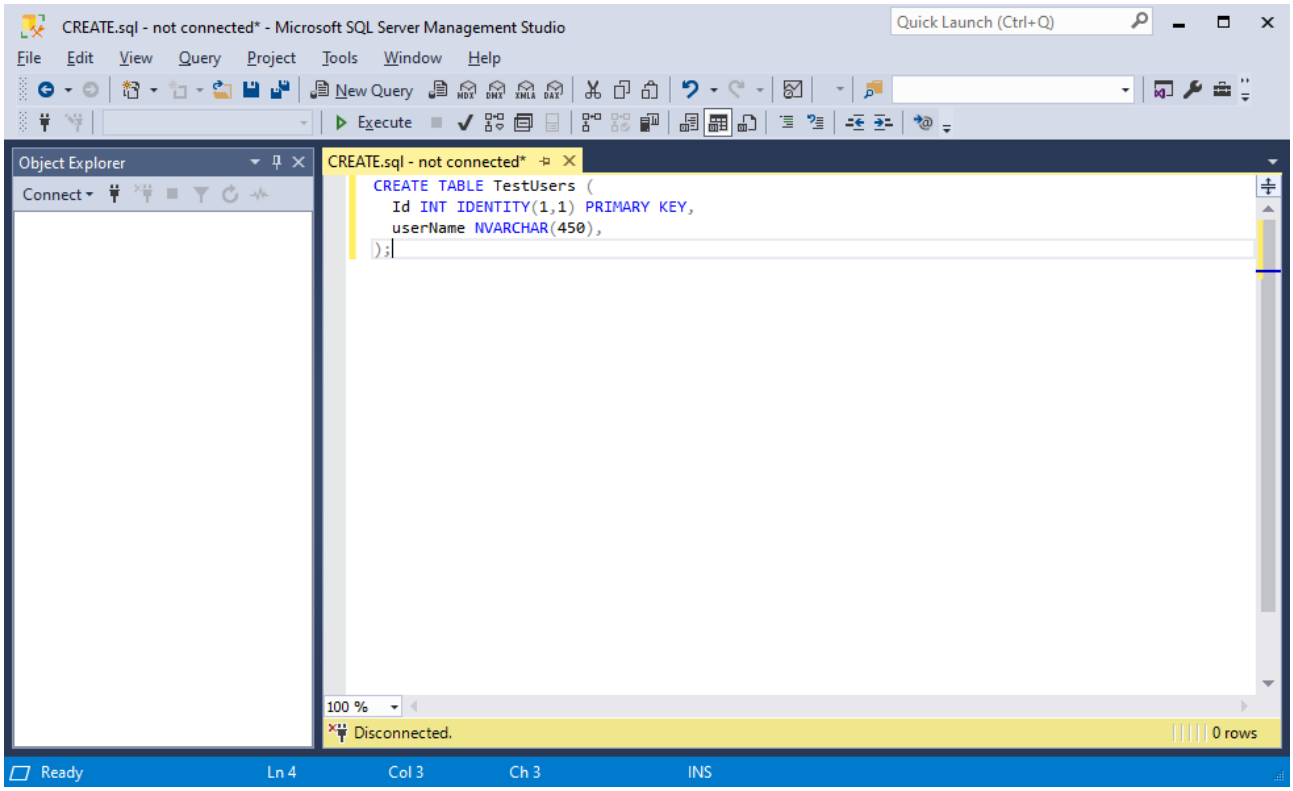
Kuva 5. Microsoft Visual Studio Community 2022 (Veikko Soosaar). Käytetty Microsoftin luvalla

3.2 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) on kattava työkalupakki Microsoftilta. SSMS tarjoaa mahdollisuuden hallita, konfiguroida, kehittää ja ylläpitää Microsoft SQL Server tietokantaan liittyviä instansseja ja komponentteja. Tämän työkalupakin avulla käyttäjä pystyy graafisen käyttöliittymän kautta luomaan, muuttamaan ja hallinnoimaan tietokannan olioita kuten tauluja, näkymiä tai tallennettuja menettelyjä. SSMS tarjoaa myös SQL lauseiden kirjoittamiseen tekstieditorin. Työkalupakista löytyy myös välineitä Analysis Servicen kautta olioiden hallintaan esimerkiksi prosessoinnin tai varmuuskopioiden luomiseksi. Lisäksi Integration Servicen kautta löytyy työkaluja pakettien hallintaan ja käynnistämiseen. Työkalupakista löytyy myös Reporting Service, minkä kautta pystyy tarkastelemaan raportteja. Lisäksi SSMS tarjoaa mahdollisuuden hallita roolitusta käyttöoikeuksien hallitsemiseksi. (Microsoft, 31.3.2023)

SQL Server Management Studio valittiin tähän opinnäytetyöprojektiin tietokannan suunnittelua, toteutusta ja hallinnointia varten. Yksi pääasiallisista syistä tämän tuotteen valitsemiseksi on se, että käytän kyseistä tuotetta lähes päivittäin työssäni. Koska tämä työkalupakki on minulle entuudestaan tuttu, säästää se paljon aikaa ja tehostaa ajankäyttöä antamalla mahdollisuuden keskittyä olennaisempiin asioihin opinnäytetyössä. SSMS mahdollistaa tietokannan helpon toteuttamisen ja SQL lauseiden ajamisen tietokannan testaamista ja tutkimista varten. Tietokantaa on myös erittäin

helppo hallinnoida tämän työkalupakin tarjoamien valmiiden apuvälineiden kautta. Kuten myös Visual Studio 2022, myös SSMS tarjoaa syntaksia värittämällä apua syntaksin lukemiseen tietokantalauseita tutkiessa tai luodessa (kuva 6).



Kuva 6. Microsoft SQL Server Management Studio (Veikko Soosaar). Käytetty Microsoftin luvalla

3.3 Microsoft Azure DevOps ja Azure Boards

Azure DevOps on Microsoftin luoma kokoelma palveluita ja prosesseja avustamaan ohjelmistokehityksen elinkaaren eri vaiheissa. Tähän sisältyy suunnittelua, ohjelmointia, ohjelman rakentamista, testausta, toimittamista ja seurantaa. Azure DevOps mahdollistaa organisaatioille luoda ja parantaa tuotteita nopeampaan tahtiin kuin perinteisillä lähestymistavoilla. Tämän kokoelman palveluita pääsee käyttämään internet-selaimen tai ohjelmointiympäristön kautta. Eri palveluita voi ottaa käyttöön tarpeen mukaan. (Microsoft, 11.10.2022)

Tässä opinnäytetyöprojektissa hyödynnämme Azure DevOpsin palveluista ainoastaan Azure Boards palvelua. Azure Boards on erillinen palvelu osana Azure DevOps kokoelmaa ja tällä palvelulla on tarkoitus auttaa tiimejä suunnittelemaan, seuraamaan ja keskustelemaan työstä koko ohjelmistokehityksen prosessin aikana. Palvelussa on mahdollista hallita tehtäviä kuten käyttäjätarinoita ja huomattuja ohjelmointivirheitä. Ketterille menetelmille kuten Scrum ja Kanban löytyy myös tuki.

Mahdollista on myös tehdä integraatio GitHub-palvelun kanssa tarvittaessa työn etenemisen seuraamiseksi. (Microsoft 22.3.2023)

Tämän opinnäytetyön osalta tärkein ominaisuus on Azure Boardsin tarjoama tuki Kanban-menetelmän tyylisille tauluille. Kanban-taulun avulla sovelluskehitystyötä pystyy seuraamaan tehokkaammin ja voidaan arvioida paremmin työskentelyn sujumista aikataulun mukaisesti. Kanban-taulut sopivat myös hyvin tämän projektin laajuuteen nähden. Ei ole tarvetta täysikokoiselle Azure DevOps ympäristölle ja esimerkiksi ketterälle Scrum-tyyppiselle kehittämiselle. Kanban-taulu ja siihen luetellut tehtävät sopivat hyvin tämän opinnäytetyön kaltaiseen projektiin, jossa on yksi tekijä kehitystyölle ja tarkoitus on saada vähimmäisvaatimukset täyttävä sovellus valmiiksi tietyn ajan puitteissa.

3.4 Microsoft Azure

Microsoft Azure, myös Azure nimellä tunnettu Microsoftin pilvipalvelualusta on tässä opinnäytetyössä käytössä sovelluksen ja sen tietokannan isäntäpalvelimena. Tämä tarkoittaa käytännössä sitä, että sovellus ja sen tietokanta eivät ole käynnissä omalla tietokoneellani tai omalla palvelimellani vaan ne toimivat Azuren pilvialustalla. Azure tarjoaa palveluita niin pienempään, kuin isompaankin käyttöön ja Azuren alustalla on tarjolla iso määrä erilaisia palveluita kehittäjien ja sovellusten ylläpitäjien käyttöön. Azure mahdollistaa myös eritasoisia ylläpitopalveluita. Käyttäjä voi ottaa käyttöön kokonaisen virtuaalikoneen tai vaihtoehtoisesti esimerkiksi hyödyntää palvelitonta viitekehystä pelkän koodin ajamiseen pilviympäristössä. Myös pelkän sovelluksen ylläpito on mahdollista. (Microsoft 14.12.2023)

Tässä opinnäytetyöprojektissa Azuren palveluista on käytössä Azure App Service ja Azure SQL Database. Sovelluksen ympäristöksi tulee Azure App Service ja sovelluksen käyttämä tietokanta tulee sijaitsemaan Azure SQL Database tietokantaympäristössä. Valitsin nämä palvelut, koska sovelluksen ollessa Microsoftin teknologioilla tehty, niin tuntui myös parhaalta vaihtoehdolta asentaa sovelluksen osat Microsoftin tarjoamiin pilvipalveluihin. Nämä palvelut ovat myös minulle ennestään tuttuja ja näin ollen olivat tämänkin takia luonteva valinta.

3.4.1 Azure App Service ja Azure SQL Database

Sovellusprojektissa käytetty Azure App Service on HTTP-protokollaan perustuva verkkopalvelu sovellusten, rajapintojen ja mobiilipalveluiden palvelinpuolen ylläpitoon. App Service tukee useita eri kieliä ja alustoja kuten .NET, .NET Core, Ruby, Node.js, PHP, Java ja Python. Sovellukset voivat olla käynnissä App Servicen kautta niin Windows kuin Linux pohjaisissa ympäristöissä. Sovellusten jako käyttöön ja ylläpito onnistuu monien eri suunnitelmien kautta monipuolisesti, jolloin on mahdollista esimerkiksi maksaa vain käytetystä kapasiteetista. (Microsoft 14.3.2023)

Azure SQL Database on Microsoft Azuren tarjoama täysin hallittu alusta palveluna. Kyseinen alusta käyttää uusinta versiota tietokantojen hallintajärjestelmästä SQL Server. Azure SQL Database tarjoaa hallitun ympäristön, jolloin Microsoft hoitaa kaiken palvelimeen liittyvän huolenpidon. Tämä tarkoittaa sitä, että palvelun käyttäjän ei tarvitse huolehtia päivityksistä, käyttöjärjestelmän tai palvelimen koodista, ylläpidosta, varmuuskopioista tai saatavuudesta. (Microsoft 3.3.2023a)

3.5 GitHub

GitHub on yritys, joka tarjoaa pilvipohjaista palvelua Git versionhallinnan käyttöön ja koodin säilöntään. Palvelun on tarkoitus helpottaa yksilöiden ja tiimien yhteistyötä ja Gitin käyttöä. Palvelu voi rekisteröityä ja säilöä koodia ilmaiseksi. Tästä syystä GitHub onkin suosittu erityisesti avoimen lähdekoodin projektien parissa. GitHub myy myös maksullisia yksityisiä koodisäilöjä ja yrityksille suunnattuja palveluita. (Kinsta 13.12.2022)

Opinnäytetyön projektissa GitHub palvelusta käytetään GitHub Repositories palvelua versiohallinnan hallinnoimiseen ja koodin säilömiseen. Lisäksi käytössä on GitHub Actions sovelluskoodin rakentamiseksi sovellukseksi ja sen toimittamiseksi Azure App Servicen ylläpidettäväksi. Olen aikaisemmin käyttänyt GitHub Repositoriesia koodien säilömiseksi, ja tämän takia se olikin ensimmäinen valintani tätä käyttötarkoitusta varten. GitHub Actions on itselleni kuitenkin tullut vasta tämän opinnäytetyöprojektin yhteydessä tutuksi. Sitä käytettiin tämän projektin sovelluksen rakentamisen ja toimittamisen prosesseihin tekemällä siihen työnkulku eli Workflow.

3.5.1 GitHub Repositories ja GitHub Actions

GitHub Repositories mahdollistaa koodisäilöjen ylläpidon ja hallinnan. Nämä säilöt sisältävät kaikki projektiin liittyvät tiedostot ja niiden versiohistorian. Koodisäilöt voivat olla yksittäisen ihmisen tai organisaation omistuksessa. Vaikka säilö olisi yhden ihmisen omistuksessa, niin siinä olevaa koodia voidaan antaa myös muiden ihmisten muokata. Koodiin voi myös tehdä muutosehdotuksia, joita sitten esimerkiksi säilön omistaja voi hyväksyä tai hylätä. Näkyvyysasetuksilla säilöjä voidaan asettaa julkisesti näkyviksi tai yksityiseksi, jolloin voidaan rajata sitä ketkä pääsevät tarkastelemaan säilön sisältöä. (GitHub s.a.a)

GitHub Actions on jatkuvan integraation ja toimittamisen alusta, jossa on mahdollista automatisoida sovelluksen rakentaminen, testaus ja toimittamisen prosessit. Alustan pääkomponentit ovat Workflows eli työnkulku, Events eli tapahtumat, Jobs eli työt, Actions eli toiminnot ja Runners eli työnkulun toimeenpanijat. (GitHub s.a.b)

4 Sovelluksen määrittely

Tässä vaiheessa opinnäytetyötä esitellään sovellus ja siihen liittyvät tavoitteet eri näkökulmista. Käydään läpi itse sovellusta, sovelluksen käyttötarkoitusta, kohderyhmää ja sovelluksen erilaisia vaatimusmäärittelyitä. Vaatimusmäärittelyitä löytyy käyttäjän, toiminnallisen ja teknisen näkökulman pohjalta. Vaatimusmäärittelyiden pohjalta sovellus rakennetaan haluttuun muotoon ja niiden perusteella voidaan myös arvioida sovelluksen toteutuksen onnistuneisuutta. Jos vaatimusmäärittelyt täyttyvät, voidaan nähdä sovelluksen täyttävän vähimmäisvaatimukset ja tarpeet.

4.1 Kuvaus

Tämä uusi sovellus kehitettiin, kun löytyi tarve seurata ajankäyttöä ja saada raportteja siitä, miten käytän aikaani. Tarkoitus on antaa tietoa avustamaan päätöksentekoa ja tehostamaan työskentelyä. Tämän käyttötarkoituksen innoittamana sovellukselle tuli nimeksi AikaHalli. AikaHalli kuvastaa hallia, johon säilötään erilaisiin tehtäviin liittyvää tietoa ajankäytöstä. Sovellus tarjoaa intuitiivisen ja käyttäjäystävällisen käyttöliittymän ajankäytön seuraamiseen ja tietojen lisäämiseen järjestelmään. Käyttämällä tämän hetken uusimpia versioita teknologioista kuten .NET ja SQL Server, saadaan rakennettua sovellus, joka on kestävä ja luotettava kaikenlaisille käyttäjille.

Sovelluksessa tulee olemaan useita eri toimintoja, kuten seurattavien tehtävien räätälöinti, aikaleimojen manuaalinen lisäys, kerätyn datan esittely ja tarkastelu. Sovellukselle on tehty vaatimusmäärittelyt käyttäjän, toiminnallisesta ja teknisestä näkökulmasta. Näiden vaatimusmäärittelyjen avulla varmistetaan sovelluksen toteutuksen oikea suunta. Vaatimusmäärittelyt toimivat myös dokumentaationa sovelluksen käytöstä ja ominaisuuksista.

4.2 Käyttötarkoitus

Sovelluksen pääasiallinen käyttötarkoitus on seurata ajankäyttöä erinäisten tehtävien parissa. Tämän avulla toivotaan käyttäjän pystyvän tarkastelemaan esimerkiksi tuottavuuttaan erinäisten tehtävien osalta. Ajankäytön raporttien pohjalta voidaan myös tunnistaa ajankäytön kaavoja, jolloin voidaan tehdä päätöksiä ja muutoksia näiden tietojen analysoinnin perusteella.

4.3 Kohderyhmät

Tässä osiossa pyritään tunnistamaan sovelluksen kohderyhmät, eli käyttäjäryhmät, jotka hyötyisivät sovelluksen käytöstä. Käydään läpi eri kohderyhmiä ja heidän mahdollisia käyttötapauksiansa.

4.3.1 Opiskelijat

Opiskelijoilla on mahdollisuus käyttää sovellusta eri kursseille ja opiskeluihin käyttämänsä ajan seuraamiseen. Tätä ajankäyttöä seuraamalla opiskelija voi optimoida ajankäyttöään eri aiheiden opiskeluun optimaalisemmin ja näin voi parantaa koulumenestystään haluamillaan kursseilla käyttämällä niihin aiempaa enemmän aikaa.

Esimerkiksi voisi olla kuvitteellinen tapaus, jossa opiskelija haluaa parantaa arvosanojaan niiden aineiden osalta missä menestys on huonompi. Opiskelija ryhtyy seuraamaan ajankäyttöä sovelluksen avulla ja kuukauden päästä voi katsoa miten hän on aikaansa käyttänyt eri aineiden opiskeluun. Opiskelija huomaa, että aineet missä arvosana on huonompi ovat myös samalla aineita mihin hän käyttää paljon vähemmän aikaa. Opiskelija ryhtyykin käyttämään enemmän aikaa kyseisten aineiden opiskeluun ja ryhtyy seuraamaan tiheämpään ajankäyttöä eri aineiden välillä.

4.3.2 Työntekijät

Työntekijöille avautuu mahdollisuus hyödyntää sovellusta työtehtäviin kuluvan ajan seuraamiseksi. Tämä mahdollistaa tuottavuuden parantamisen ja työtehtävien tasapainottamisen. Moni saattaa kokea käyttävänsä liikaa aikaa tietynlaisiin työtehtäviin ja tämän sovelluksen avulla pystyykin seuraamaan ajankäyttöä eri tehtävien välillä. Varsinkin etätyöntekijöille sovellus voi olla hyödyllinen auttamaan työn ja vapaa-ajan tasapainottamisessa. Etätyöskentelyn lomassa voi olla vaikea seurata ajankäyttöä työtehtävien parissa normaalin arjen lomassa. Jos työntekijällä on velvollisuus kirjata ylös käyttämiään työtunteja eri projektien tai työtehtävien parissa, on tästä sovelluksesta siihenkin hyötyä. Moni tekee töitä laskuttamalla asiakasta ja tehtyjen työtuntien seuranta voi olla työlästä ilman kunnollista työkalua ajankäytön seurantaan.

Esimerkiksi jokin työntekijä saattaa tehdä konsulttiyrityksessä työtä asiakasorganisaatiolle. Asiakasorganisaatio vaatii konsultilta kirjanpitoa käytetyistä työtunneista laskutukseen liittyen. Nämä tunnit tulee palauttaa tavallisessa Excel-tiedostossa taulukkomuodossa. Taulukossa riittää, että löytyy projekti ja siihen käytetyt tunnit, eikä tarvitse tarkempaa erittelyä. Aiemmin konsulteille on ohjeistettu, että ihan itse kirjanpito esimerkiksi muistiossa on toimiva ratkaisu. Nyt kuitenkin tämän uuden sovelluksen myötä on mahdollista pitää kirjaa asiakasprojekteihin käytetyistä tunneista ja saada projektikohtaiset erittelyt. Konsultilta säästyy paljon vaivaa, kun ei tarvitse alkaa itse laskemaan käytettyjen tuntien määrää käsin tehtyjen kirjausten perusteella.

4.3.3 Esihenkilöstö ja tiiminvetäjät

Esihenkilöillä ja tiiminvetäjille on moniulotteisempia käyttötarkoituksia sovellukselle. Samalla tavalla kuin työntekijät, myös esihenkilöt ja tiiminvetäjät voivat seurata omaa työajan käyttöä ja samat

hyödyt ovat tältä osin samankaltaiset. Toisin kuin työntekijöillä, niin esihenkilöstö ja tiiminvetäjät voivat kuitenkin hyödyntää myös heidän allaan työskentelevien työntekijöiden tietoa ajankäytöstä. Kun alaiset jakavat tietoa ylöspäin käytetyistä työtunneista, voidaan tunnistaa pullonkauloja ja paljon aikaa vieviä tehtäviä. Näihin voidaan sitten vaikuttaa uusilla päätöksillä ja ratkaisuilla. Nämä ratkaisut ja päätökset voivat tehostaa työntekoa ja kohentaa työhyvinvointia, kun voidaan auttaa työntekijöitä toimimaan järkevämmin.

Kuvitellaan esimerkiksi tilanne, jossa tiiminvetäjä huomaa jonkin projektin etenevän hyvin hitaasti verrattuna muihin. Tässä tilanteessa tietenkin luonnollista olisi keskustella asiasta projektissa työskentelevän tai työskentelevien henkilöiden kanssa. Mutta tämän sovelluksen ollessa käytössä tiiminvetäjä voikin tarkastella tiimin jäsenten työpanoksia eri projektien välillä. Jos näyttää siltä, että jokin projekti etenee hitaasti ja sille ei kohdennu juuri ollenkaan tunteja, niin voidaan siirtää työn painopistettä enemmän kohti projektia tiimin jäsenten osalta. Jos taas näyttää siltä, että projektiin menee paljon aikaa ja esimerkiksi siinä työskentelee vain yksi tai kaksi jäsentä, niin ratkaisu voikin löytyä apujoukkojen lisäämisestä tiimin muista jäsenistä. Sovellus siis toimii tiiminvetäjälle tukena keskusteluissa ja päätöksenteossa, auttaen ratkaisemaan ongelmia lisätyllä tiedolla.

4.3.4 Itseään kehittävät yksilöt

Sovellusta käyttävät yksilöt, jotka eivät aio käyttää sovellusta työhön liittyen voivat silti saada sovelluksesta paljon hyötyjä. Yksilöt voivat sovelluksen avulla seurata ajankäyttöä esimerkiksi harrastuksiin, kuntoiluun ja muihin itseään kehittäviin aktiviteetteihin liittyen. Näitä aktiviteetteja seuraamalla käyttäjä kykenee hahmottamaan omaa ajankäyttöään paremmin ja sen myötä asettamaan itselleen realistisempia tavoitteita ja tasapainoilemaan eri aktiviteettien välillä.

Esimerkkitilanteessa käyttäjä tallentaa vapaa-ajan käyttöään sovellukseen ja katsoo viikon välein omaa ajankäyttöään. Tästä raportista saattaa tehdä huomioita kuten esimerkiksi sen, että aikaa tulee käytettyä aivan eri tavalla eri viikonpäivinä. Esimerkiksi voi olla mahdollista, että viikon alkupuolella liikuntaan menee paljon enemmän aikaa kuin viikon loppupuolella. Käyttäjä pystyykin tämän ajankäytön seurannan seurauksena muuttamaan tapojaan ja tekeminen tasapainottuu tämän seurauksena. Raportista voidaan myös pidemmällä aikavälillä huomata, jos joinakin tiettyinä vuodenaikoina liikunnan harrastaminen muuttuu radikaalisti, kuten vaikka esimerkiksi syksyllä säiden viilentyessä.

4.4 Käyttäjän vaatimusmäärittely

Tässä kohdassa käymme läpi vaatimusmäärittelyä tälle sovellukselle. Tässä tullaan kuvaamaan tarpeita, odotuksia ja toiveita lopputuloksesta käyttäjän ja sidosryhmien näkökulmasta (Altexsoft 2021). Nämä tiedot auttavat ohjaamaan sovelluksen kehitystä suuntaan, jossa toteutuu kaikki

käyttäjän näkökulmasta tarpeelliset vaatimukset. Esittelemme myös käyttäjätarinoita kuvaamaan tyypillisiä sovelluksen käyttötilanteita saadaksemme dokumentaatiota halutuista tavoista käyttää sovellusta.

4.4.1 Yleiset käyttäjän vaatimukset

Sovelluksen tule olla käytettävyydeltään intuitiivinen ja käyttäjäystävällinen. Käyttökokemuksen tulisi olla niin helppo, että käyttäjä ei tarvitse erillistä koulutusta tai teknistä osaamista sovelluksen käyttämiseen ja hyödyntämiseen.

Käyttöliittymän tulee olla yksinkertainen ja puhdas, ilman mitään ylimääräistä häiriötä käyttäjälle. Sovelluksen tulee reagoida herkästi käyttäjän toimiin ja tarjota virtaviivainen nopea käyttökokemus.

Sovelluksen käyttöönoton yhteydessä käyttäjän tulee päästä rekisteröitymään valitsemallaan sähköpostiosoitteella ja salasanalla. Käyttäjän pitää myös pystyä kirjautumaan sisälle ja ulos halutessaan sovelluksesta.

Käyttäessä sovellusta, tulee käyttäjän kyetä luomaan tehtäviä. Näitä tehtäviä tulee pystyä aktivoimaan käyntiin ja myös lopettamaan niiden aktiivinen tila, jotta saadaan tallennettua tieto ajankäytöstä tehtävään liittyen.

Käyttäjän tulee myös päästä muokkaamaan sovellukseen tallennettuja tietoja omien käyttötarpeiden mukaiseksi niin, että käyttäjä ei ole sovelluksen omien ennalta asetettujen vaihtoehtojen varassa ja mahdollisesti virheellisesti tallentunut tieto voidaan korjata oikeanlaiseksi tarvittaessa. Tieto, joka tallennetaan sovelluksen tietokantoihin, tulee olla muokattavissa niin, että käyttäjä pääsee lisäämään, muokkaamaan ja poistamaan halutessaan tarvittavia tietoja. Tämä koskee niin tehtäviä, kuin myös tehtäviin liittyviä tallennettuja aikaleimoja.

Tallennettua tietoa ajankäytöstä tulee voida näyttää käyttäjällä helposti ymmärrettävässä muodossa. Käyttäjän pitää pystyä halutessaan näkemään tallennettu tieto monessa eri muodossa. Tällä mahdollistetaan käyttäjälle tiedon analysointi halutulla tavalla ja sen perusteella voidaan sitten tehdä päätöksiä.

Käyttäjän pitää pystyä käyttämään sovellusta usealta eri laitteelta ja alustalta. Sovelluksen pitää olla käytettävissä niin pöytätietokoneella, kuin myös muulla laitteella kuten esimerkiksi tabletilla tai älypuhelimella. Laitteiden lisäksi käyttäjän pitää pystyä käyttämään sovellusta myös haluamallaan selaimella.

Sovelluksen tallentaman tiedon pitää olla turvallisesti tallennettu säilöön niin, että kuka tahansa ei pääse tarkastelemaan tietoja ilman lupaa. Käyttäjän pitää myös päästä näitä tallennettuja tietoja tarkastelemaan ja käyttämään sovellusta usealta eri laitteelta samalla käyttäjätunnuksella.

4.4.2 Käyttäjätarinat

Käyttäjätarinat ovat epävirallinen yleinen selitys sovelluksen ominaisuudesta käyttäjän näkökulmasta. Sen tarkoitus on kertoa, miten sovelluksen ominaisuus tarjoaa arvoa asiakkaalle. Käyttäjätarinat tuovat käyttäjät keskusteluun tärkeään asemaan. Käyttäjätarinoissa ei ole tarkoitus käyttää teknistä kieltä. Näillä käyttäjätarinoilla pyritään tuomaan syy ja hyöty ominaisuuden kehittämisestä kehittäjätiimin tietouteen. (Rehkopf s.a.)

Käyttäjätarina 1: Käyttäjä haluaa rekisteröidä käyttäjätilin ja päästä kirjautumaan käyttäjätilillä sisään sovellukseen, sekä ulos.

Käyttäjätarina 2: Käyttäjä haluaa luoda tai valita tehtävän kuten työprojektin, siivoamisen tai lenkkeilyn, jonka jälkeen hän haluaa päästä aktivoimaan.

Käyttäjätarina 3: Käyttäjä haluaa päästä luomaan, muokkaamaan ja poistamaan tehtäviä omalla sivullaan.

Käyttäjätarina 4: Käyttäjä haluaa päästä tarkastelemaan ajankäyttöään eri tehtävien parissa, sovelluksen tulee näyttää käyttäjälle eri tehtävien parissa laskettu aika.

Käyttäjätarina 5: Käyttäjä haluaa päästä lisäämään, muokkaamaan ja poistamaan tallennettuja aikaleimoja tehtäviin liittyen omalla sivullaan.

4.5 Toiminnallinen vaatimusmäärittely

Toiminnallisen vaatimusmäärittelyn tarkoitus tässä opinnäytetyössä on kertoa sovelluksessa olevista toiminnoista, ominaisuuksista ja odotetuista käyttötapahtumista, jotka ovat sovellukselta vaadittu käyttäjävaatimusten täyttämiseksi. (Altexsoft 2021)

4.5.1 Käyttäjätilin hallinta

Järjestelmän tulee mahdollistaa käyttäjälle rekisteröityminen ainutlaatuisella sähköpostiosoitteella ja salasanalla.

Järjestelmän tulee sallia käyttäjän kirjautuminen omalla rekisteröimällään sähköpostiosoitteella ja salasanalla.

4.5.2 Ajanseuranta

Järjestelmän tulee sallia käyttäjälle ajankäytön seurannan tehtävien luonti-, muokkaus- ja poistotoiminto.

Järjestelmän tulee antaa käyttäjälle mahdollisuus käyttää ajankohdan leimaamisen valitsemalleen tehtävälle. Käyttäjän pitää päästä valitsemaan leimasta aloittamisen ja lopettamisen ajankohta halutessaan.

Järjestelmän käyttäjän pitää päästä syöttämään manuaalisesti aikaleimoja omille tehtävilleen. Näiden manuaalisten aikaleimojen tulee voida sisältää samat tiedot kuin myös sovelluksen kautta automaattisesti tulleiden leimojen.

Järjestelmässä pitää olla mahdollisuus lisätä muistiinpanoja tai huomautuksia jokaiseen aikaleimaan halutessaan.

4.5.3 Raportit

Järjestelmän pitää sallia käyttäjälle mahdollisuus selata aikaleimoista muodostuvia raportteja valitsemilleen päivämäärille ja tehtäville. Päivämäärien ja tehtävien valintojen tulee toimia niin yhdessä kuin erikseenkin.

Järjestelmän tulee antaa käyttäjälle mahdollisuus nähdä ajankäytön erittely taulukkomuodossa. Tähän sisältyy joko kaikki tiedot tai vain osa tiedoista riippuen valinnoista.

4.5.4 Tiedon säilytys ja tietoturva

Järjestelmässä tulee olla turvallinen käyttäjätietojen säilytys. Tietoihin ei saa päästä käsiksi kukaan ulkopuolinen kenellä ei ole oikeutta nähdä kyseisen käyttäjän tietoja.

4.5.5 Järjestelmään sisään syötettävä tieto

Järjestelmän tulee ottaa vastaan käyttäjän kirjautumistiedot. Tämän sovelluksen tapauksessa se tarkoittaa sähköpostiosoitetta ja salasanaa.

Järjestelmän pitää ottaa vastaan käyttäjän tehtävien ja aikaleimojen tiedot. Nämä tiedot voivat siis tulla niin aikaleimojen kirjauksen ominaisuudesta tai käyttäjän itse manuaalisesti syöttämänä lomakkeen kautta.

Järjestelmän kuuluu ottaa vastaan tieto raportin rajaamiseen liittyviä kriteerejä. Tässä sovelluksessa se tarkoittaa päivämäärää ja tehtävää, joiden perusteella näytetään aikaleimat käyttäjälle.

4.5.6 Järjestelmästä ulos saatava tieto

Järjestelmä antaa ulospäin käyttäjälle raportteja ajankäytöstään. Nämä voivat olla taulukkomuotoisia tai listauksena.

Järjestelmä antaa ulospäin käyttäjälle tiedostomuodossa raportteja ajankäytöstään. Tällaisia tiedostomuotoja on esimerkiksi CSV-tiedosto.

4.6 Tekninen vaatimusmäärittely

Tekninen vaatimusmäärittely kertoo mitä sovelluskehittäjien pitää tehdä ratkaistakseen tekniset ongelmat ja haasteet, jotta sovellus toimii kuten on haluttu. Tekniset vaatimusmäärittelyt ovat tärkeitä, koska niiden avulla sovelluksen käytöstä tulee paras mahdollinen ymmärrettävyyden näkökulmasta, niin kehittäjien kuin käyttäjienkin näkökulmasta. (Indeed 2023)

Varmistamalla tämän teknisen vaatimusmäärittelyn toteutumisen, voimme varmistua sovelluksen kykenevän myös täyttämään toiminnalliset käyttövaatimukset ja toiminnallisten käyttövaatimusten mukana myös käyttäjän vaatimukset tulevat täyttymään.

4.6.1 Teknologiat

Selainpuoli ohjelmasta tullaan toteuttamaan Razor Pages viitekehystä käyttäen. Tämän lisäksi hyödynnetään HTML, CSS ja JavaScript kieliä.

Palvelinpuoli ohjelmasta toteutetaan .NET 6 viitekehyksen pohjalle ASP.NET Core 6 viitekehyksen avulla. Ohjelmointi itsessään toteutetaan C#-kielellä.

Tietokannan toteutus tullaan tekemään Microsoft SQL Server tietokantojen hallintajärjestelmällä, jossa on Microsoft Azure SQL tietokanta.

Sovelluksen sivut tehdään Razor PageModel suunnittelumallin pohjalta.

4.6.2 Suorituskyky

Sovelluksen tulee pystyä vastaamaan käyttäjän toimintoihin nopeasti ja tarjoamaan virtaviivaisen käyttökokemuksen. Tavoitteena on pitää suurimmassa osassa käyttäjän tekemien toimintojen ja sovelluksessa sen seurauksena tulevien tapahtumien välisenä aikana korkeintaan kaksi sekuntia. Yli kahden sekunnin vastausajat ovat hyväksyttyjä vaativissa toiminnoissa, kuten esimerkiksi suurien tietokantahakujen ja raporttien muodostamisen yhteydessä.

Sovelluksen tulee toimia käyttäjien ja tiedon määrän kasvaessa ilman, että sovelluksen teho ja nopeus kärsivät. Sovelluksesta ei saa tulla hitaampaa, vaikka käyttäjien määrän kasvamisen lisäksi samanaikaisten käyttäjien määrä kasvaisi myös.

4.6.3 Skaalautuvuus

Sovelluksen arkkitehtuuri on suunniteltava niin, että se tukee skaalautumista laajempaa käyttöä varten. Skaalautuvuutta pitäisi pystyä lisäämään niin palvelinten lukumäärän, kuin myös palvelinten tehojen ja resurssien suhteen.

Skaalautuvuuden tulee olla helposti toteutettavissa, jotta voidaan nopeasti mukautua kasvaviin käyttäjämääriin ja tiedon määrän lisääntymiseen.

4.6.4 Tietoturva

Sovelluksen on noudatettava tietoturvan suhteen toimialalla yleisiä standardeja. Tiedon tulee kulkea käyttäjän, sovelluksen ja tietokannan välillä turvallisesti. Lisäksi tulee olla varotoimet tehtynä yleisimpiä tietoturvariskejä vastaan.

Käyttäjän autentikointi tulee tehdä turvallisin keinoin. Sovelluksessa käytettävä vähintään ASP.NET Core Identityn tarjoamaa kirjautumisjärjestelmä.

4.6.5 Kehitys ja jakelu

Sovelluksen käyttämä versionhallintajärjestelmä on Git. Tämän avulla seurataan muutoksia lähdekoodiin ja mahdollistetaan tulevaisuudessa joustava yhteistyö muiden mahdollisten kehittäjien kanssa.

Sovelluksen tulee käyttää jatkuvaa integrointia ja julkaisua ohjelman jatkuvan kehityksen tukena. Tämä tehdään GitHub Actions työkalulla Azure AppServices palveluun.

Sovellus tulee jakaa luotettavalle ja kestävälle alustalle, joka tulee olemaan tässä tapauksessa Microsoft Azure ja siellä oleva Azure AppService palvelu.

4.6.6 Standardit

Sovelluksen tulee noudattaa toimialan standardeja koodin ja arkkitehtuurin osalta.

Sovelluksen on kyettävä reagoimaan muuttuvaan käyttöympäristöön, antamalla käyttäjälle yhtä hyvä kokemus sovelluksen käyttöympäristöstä riippumatta. Käytännössä käyttäjän pitää pystyä käyttämään sovellusta yhtä tehokkaasti riippumatta siitä onko käyttöympäristönä esimerkiksi älypuhelin, tabletti tai pöytätietokone.

4.6.7 Dokumentaatio

Sovelluksesta on tehtävä kattavat dokumentaatiot niin sen toiminnasta, kuin myös käyttäjälle näkyvistä ominaisuuksista. Tämän opinnäytetyön sovellusmäärityksetovat iso osa tätä dokumentaatiota.

Sovellus tulee myös testata useissa eri käyttöympäristöissä. Tämä tarkoittaa käytännössä sitä, että sovellus tulee testata esimerkiksi paitsi eri selaimilla, niin myös erilaisilla laitteilla kuten esimerkiksi pöytätietokoneen lisäksi älypuhelimilla.

5 Sovelluksen suunnittelu

Opinnäytetyön tässä kohdassa ryhdymme suunnittelemaan kehitettävää sovellusta. Käymme läpi käyttötapauksia käyttötapauskaavioiden avulla ja sovelluksen ulkonäköä suunnitellaan rautalankamalleja hyväksi käyttäen. Tiedon kulkua kuvaamaan tehdään tiedonkulkukaaviot ja niiden jälkeen esitellään myös tietokantamalli. Järjestelmäarkkitehtuuri ja sovelluksen levittäminen esitellään myös kaavioiden avulla. Nämä suunnitelmat visuaalisten apujen kanssa auttavat sovelluksen toteutuksessa. Jos kyseessä olisi projekti missä sidosryhmiä olisi enemmän, suunnitelmat olisivat myös mahdollisesti suurena apuna sidosryhmille projektin etenemisen seurannan ymmärtämisessä. Kaavioiden tekemiseen hyödynnettiin avoimen lähdekoodin verkkosovellusta diagrams.net, joka mahdollistaa kaavioiden luomisen ilmaiseksi (diagrams.net s.a.).

Sovelluksen kehityksessä pyritään täyttämään määritykset täyttävät vähimmäisvaatimukset. Tällä tarkoitetaan tässä tapauksessa sitä, että sovellus on enemmänkin prototyyppi tai näyte sovelluskonseptin toimivuudesta, eikä lopullinen käyttövalmis tuote. Sovellukseen ilmeneviä parannuksia, kehitysideoita ja puutteita tullaan kuitenkin kirjaamaan ylös mahdollista jatkokehitystä varten.

5.1 Käyttötapaukset

Sovellukselle on tunnistettu luvussa 5.4.2 käyttäjätarinoiden ja muiden määritysten pohjalta kuusi olennaista käyttötapausta. Näistä käyttötapauksista tehtiin käyttötapauskaavio (kuva 7), josta näkyy käyttäjälle mahdollistetut toimenpiteet. Näitä ovat rekisteröinti, kirjautuminen, tehtävien hallinta, aikaleimojen hallinta, ajanseuranta ja raportit. Rekisteröinti ja kirjautuminen ovat tärkeä osa sovellusta, koska niiden avulla tunnistetaan käyttäjä ja mahdollistetaan oikeanlaisen tiedon tallentaminen ja esittäminen. Rekisteröinti tapahtuu kirjautumattoman käyttäjän siirtyessä “Rekisteröidy”-sivulle. Tälle sivulle päästään “Kirjaudu sisään”-sivun kautta tai vaihtoehtoisesti valitsemalla “Rekisteröidy” etusivulta. Käyttäjä rekisteröityy antaen sähköpostin, salasanan ja vielä salasanan uudelleen oikeanlaisen kirjoitusasun varmistamiseksi. Tämän jälkeen käyttäjä kirjautuu automaattisesti sisään ja pääsee käyttämään sovellusta oikeuksiensa puitteissa.

Kirjautuminen sisään tapahtuu käyttäjän siirtyessä “Kirjaudu sisään”-sivulle. Tälle sivulle pääsee joko siihen viittaavasta linkistä tai vaihtoehtoisesti käyttäjän yrittäessä siirtyä sivulle, joka vaatii kirjautuneen käyttäjän. Kirjautuminen vaatii sähköpostin ja salasanan toimiakseen. Tämän jälkeen käyttäjä pääsee käyttämään sovelluksen ominaisuuksia vapaasti.

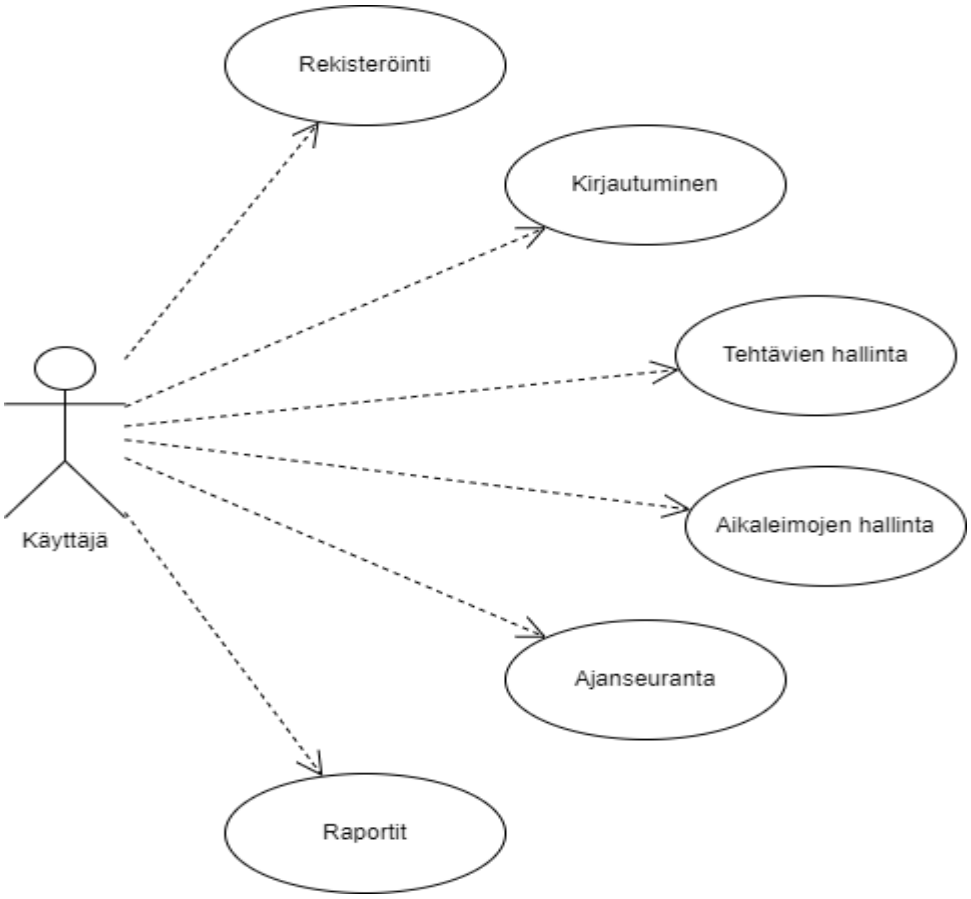
Käyttäjälle alkuun tärkeimpiä ominaisuuksia on tehtävien hallinta. Tältä sivulta käyttäjä pääsee lisäämään, muokkaamaan ja poistamaan tehtäviä. Tehtävät ovat olennainen osa ajanseurantaa, joten käyttäjän tulee tällainen luoda ennen sovelluksen muiden ominaisuuksien hyödyntämistä.

Tehtävien hallinnassa käyttäjä pääsee valitsemaan tehtävälle nimen ja antamaan tälle kuvauksen ennen tallentamista. Kun käyttäjä on luonut tehtävän, pääsee hän hyödyntämään ajanseurantaa ja aikaleimojen hallintaa.

Aikaleimojen hallinta ei ole sovelluksen käyttämisen kannalta välttämätön. Tämä sivu kuitenkin mahdollistaa käyttäjälle aikaleimojen lisäämisen, muokkaamisen ja poistamisen. Tällainen toiminta voi olla tarpeen käyttäjän esimerkiksi unohtaessa aikaleiman aloittamisen tai lopettamisen oikealla ajalla. Aikaleimoja hallitaan valitsemalla aikaleimalle tehtävä ja tämän jälkeen voidaan syöttää alkamisaika, loppumisaika ja lisätietoja.

Ajanseuranta-sivulla tapahtuu käyttäjän pääasiallinen sovelluksen käyttö. Tällä sivulla käyttäjä pääsee valitsemaan tehtävän ja käynnistämää tai lopettamaan sen. Käyttäjä voi lisätä aloitetulle tehtävälle lisätietoja. Käyttäjän poistuessa sivulta tehtävä pysyy edelleen käynnissä, kunnes se lopetetaan tai sille asetetaan loppumisaika aikaleimojen hallinnan kautta.

Käyttäjä haluaa jossain vaiheessa myös seurata ajankäyttöään. Tämä tapahtuu raportit-sivulla, jossa käyttäjä pääsee tarkastelemaan raportteja eri tehtäviin käytetystä ajasta. Tämän lisäksi käyttäjällä on mahdollisuus ladata raportteja tiedostoina omalle tietokoneelle.



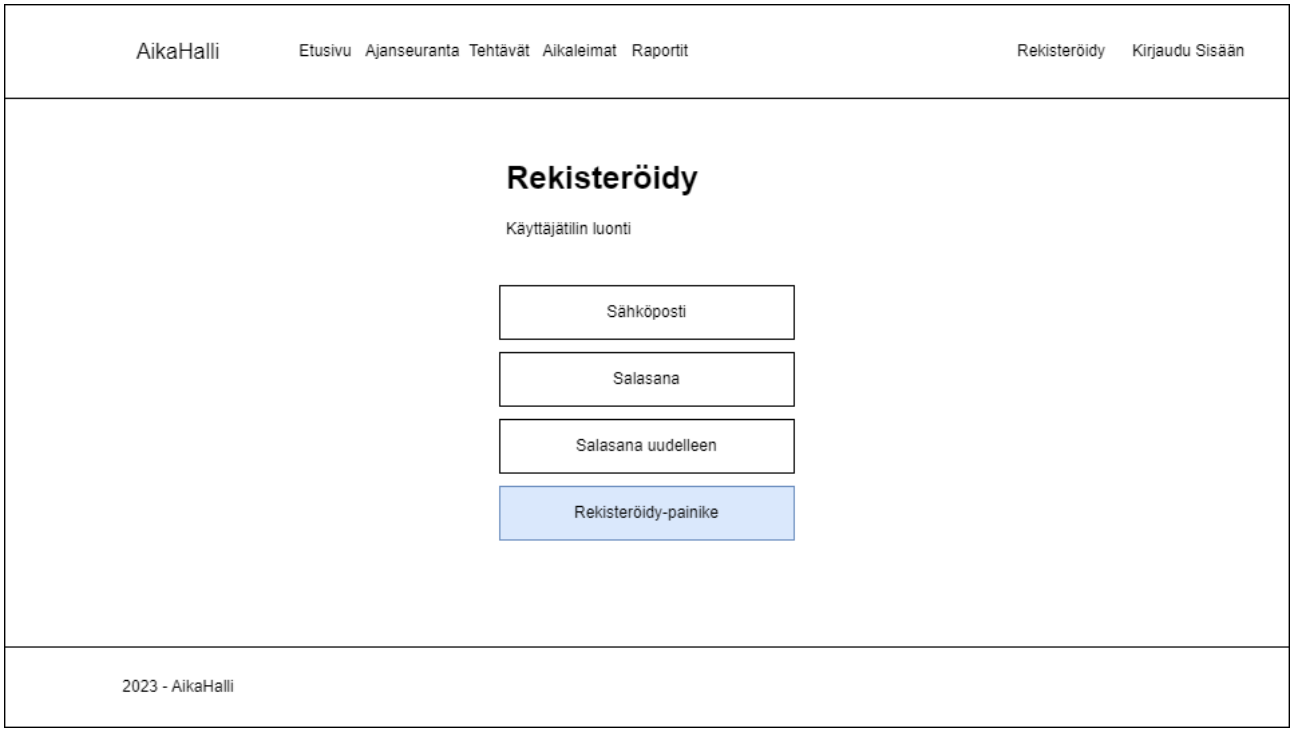
Kuva 7. Käyttötapauskaavio (Veikko Soosaar)

5.2 Sovelluksen ulkonäkö ja rakenne

Sovelluksen ulkonäköä ja rakennetta on helppo suunnitella rautalankamallin mukaan. Rautalankamallilla tarkoitan piirrettyä mallia, joka muistuttaa kuin se olisi tehty rautalangoista. Rautalankamalli on siis karkea versio käyttöliittymästä ja sen ei tarvitse vastata lopullista tulosta (Lynch & Horton, s.a.). Tämän sovelluksen rautalankamalleissa yhteistä kaikilla sivuilla on sivuston yläpalkki ja alapalkki. Yläpalkista löytyy kaikille yhteisesti vasemmalla reunalla sovelluksen nimi ja keskempänä navigaatiovalikko, josta pääsee siirtymään linkkien kautta sivuston eri osa-alueille. Oikealla reunassa on sitten “Rekisteröidy” ja “Kirjaudu Sisään” linkit jos käyttäjä ei ole kirjautunut sisään vielä kuten näkyy kuvissa 8, 9 ja 10. Käyttäjän ollessa sisään kirjautunut oikeassa yläreunassa näkyy käyttäjänimi ja mahdollisuus kirjautua ulos, mikä näkyy kuvissa 11, 12 ja 13. Sivuston alapalkista löytyy vuosi ja sovelluksen nimi.

Kuva 8. Etusivun rautalankamalli (Veikko Soosaar)

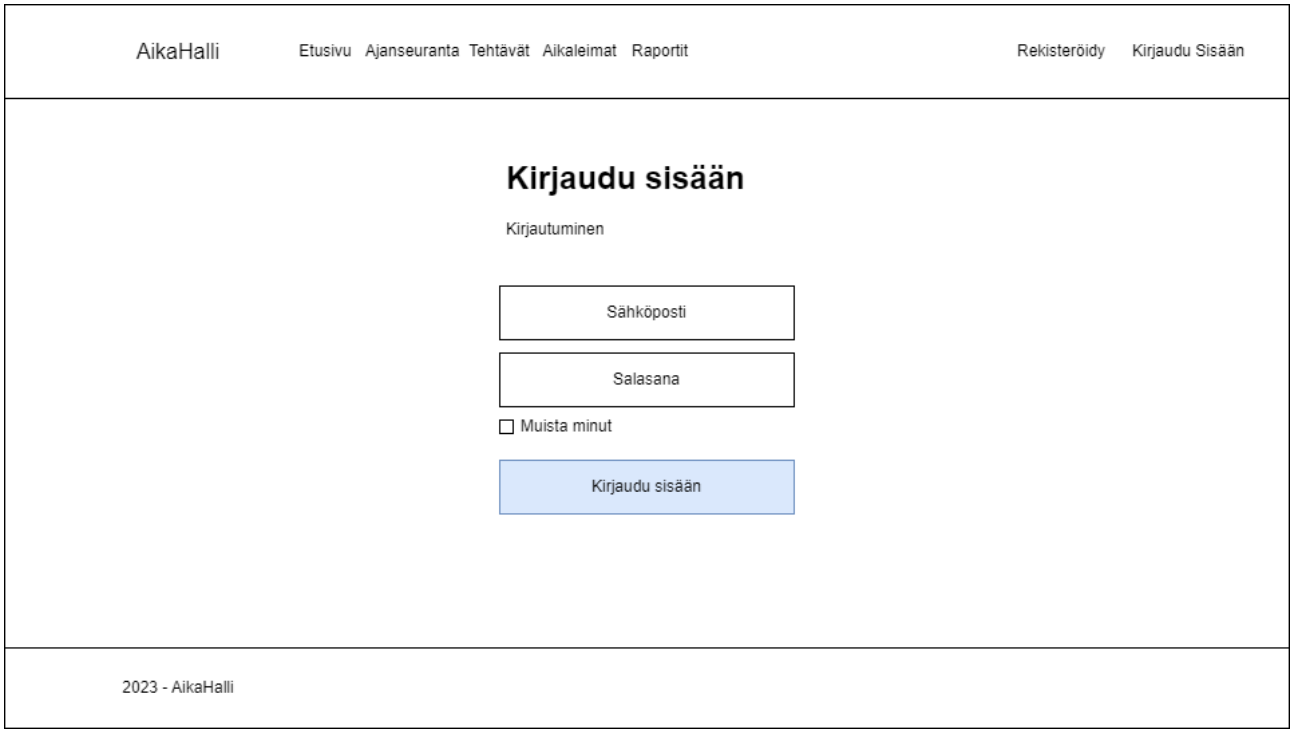
Etusivulta löytyy tervetuliaisteksti, jossa kerrotaan lyhyesti sovelluksesta. Tämän tekstin alapuolelle tulee linkit ajanseuranta, tehtävät, aikaleimat ja raportit sivuille. Näiden linkkien yhteydessä myös kerrotaan lyhyesti sivuista. (Kuva 8)



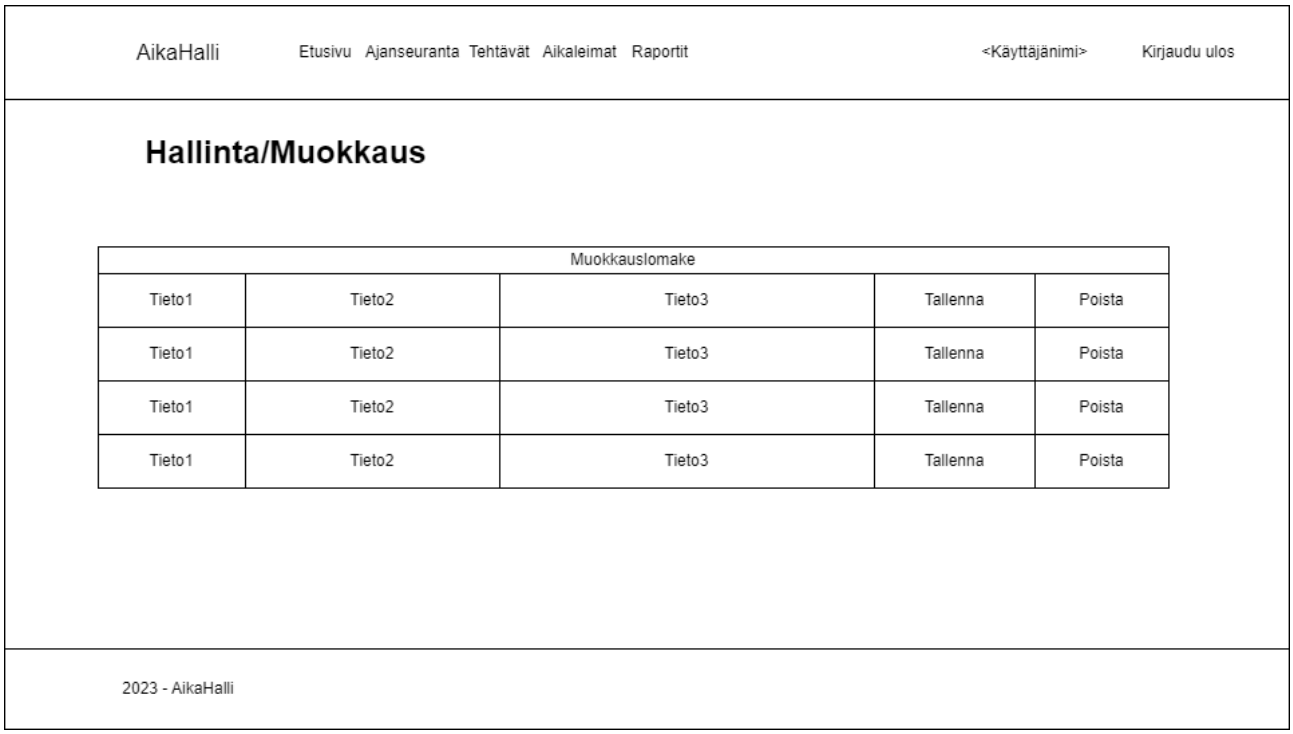
Kuva 9. Rekisteröinti-sivun rautalankamalli (Veikko Soosaar)

Käyttäjän tulee ennen sovelluksen käyttöä rekisteröityä ja se tapahtuu rekisteröintisivun kautta. Tältä sivulta löytyy lomake, johon täydennetään käyttäjän sähköpostiosoite, salasana ja vielä salasana myös uudelleen kirjoitusasun oikeellisuuden tarkistamiseksi. Lomakkeen täytettyään käyttäjä pääsee painamaan “Rekisteröidy”-painiketta. (Kuva 9)

Jos käyttäjä on jo rekisteröitynyt, pääsee hän kirjautumaan itse sisäänkirjautumissivun kautta. Sisäänkirjautumissivu muistuttaa paljon rekisteröitymissivua. Sivulta löytyy lomake, johon täydennetään sähköposti ja salasana. Lisäksi löytyy valintalaatikko, josta käyttäjä voi valita “Muista minut”, jotta sovellus muistaa käyttäjän ja pysyy sisään kirjautuneena. Lomakkeen täytettyään ja halutessaan valinnan tehneenä käyttäjä voi painaa “Kirjaudu sisään”-painiketta. Jos tiedot ovat oikein, pääsee käyttäjä kirjautumaan tämän seurauksena sovellukseen. (Kuva 10)

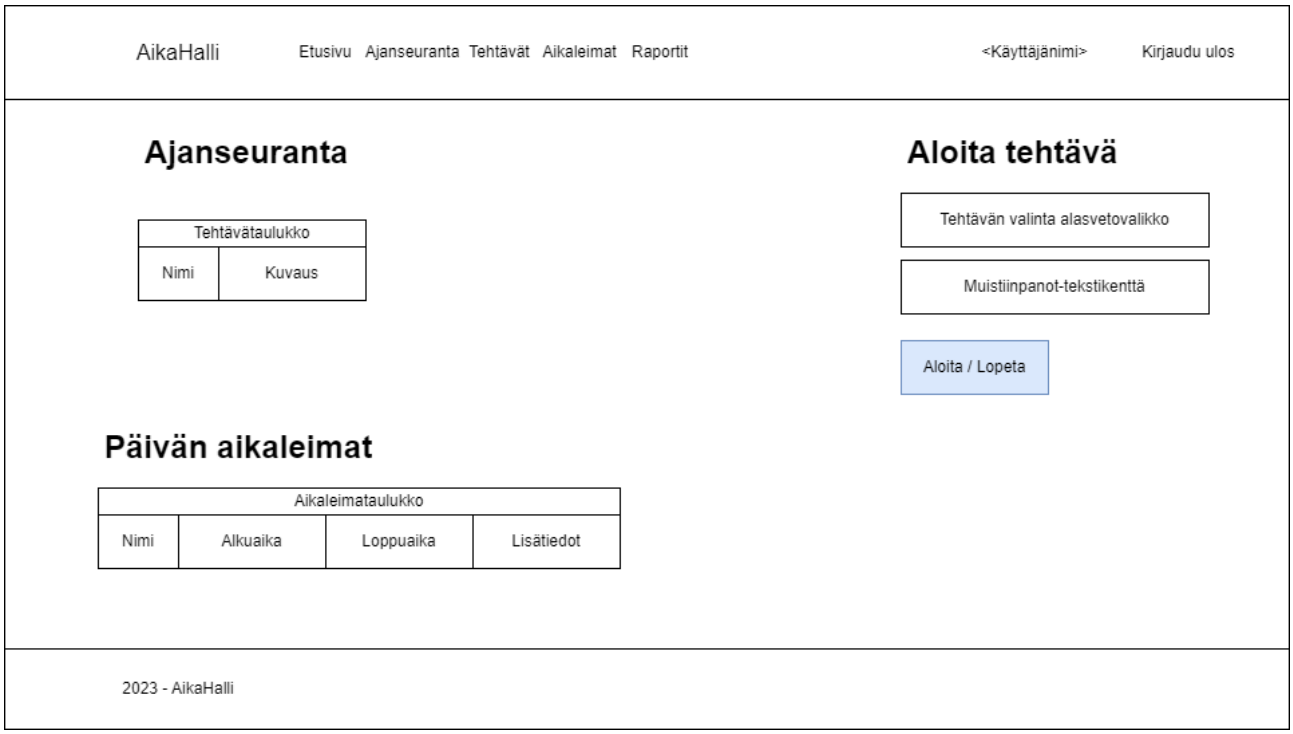


Kuva 10. Kirjautumis-sivun rautalankamalli (Veikko Soosaar)



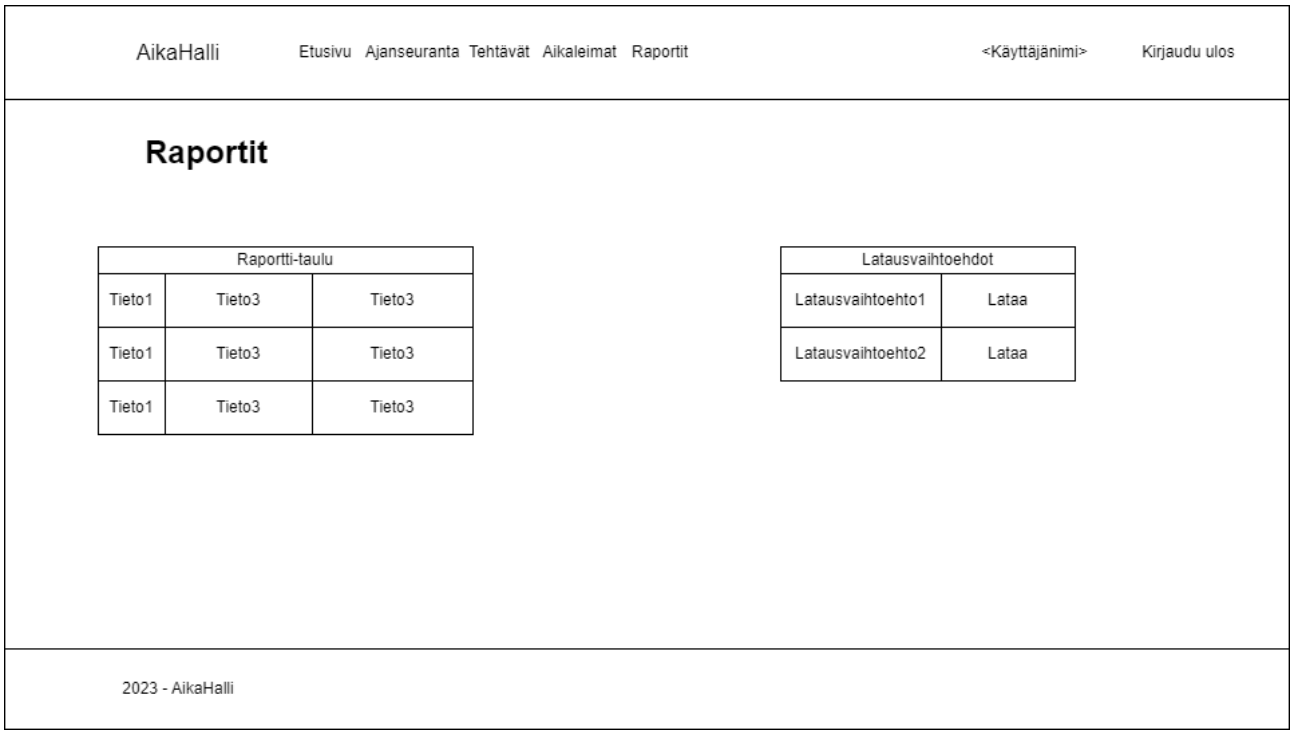
Kuva 11. Hallinta-sivun rautalankamalli (Veikko Soosaar)

Sovellukseen tulee tehtäville ja aikaleimoille omat hallintasivut. Nämä tulevat olemaan kuitenkin hyvin samanlaisia, joten ne saavat yhteisen rautalankamallin. Vaikka kenttien määrä eroaa näiden sivujen välillä, niin periaate pysyy samana ja mallia voidaan soveltaa molempiin. Hallinta-sivulta löytyy lomake, jossa voi luoda uuden tai muokata olemassa olevaa tehtävää tai aikaleimaa. Lomake tehdään taulukko-muotoisesti, jossa jokaiselle riville löytyy oma “Tallenna” ja “Poista” painike. Lomakkeessa tulee olemaan kentät eri määrälle tietoja riippuen siitä, onko kyseessä tehtävä vai aikaleima. (Kuva 11)



Kuva 12. Ajanseuranta-sivun rautalankamalli (Veikko Soosaar)

Ajanseuranta-sivu koostuu useasta eri osasta. Vasemmalla yläreunassa näkyy “Tehtävätaulukko” johon tulee taulukko käyttäjän tehtävistä ja niiden kuvauksista. Tämän tarkoitus on auttaa käyttäjää hahmottamaan mitä tehtäviä hänellä on käytössään ennen kuin hän valitsee tehtävän, jonka aloittaa. Oikeassa yläreunassa näkyy “Aloita tehtävä” otsikolla oleva osio. Tästä osiosta löytyy alasvetovalikko tehtävän valitsemiselle, tekstikenttä muistiinpanoille ja painike tehtävän aloittamiseksi tai lopettamiseksi. Sivun alareunasta löytyy “Päivän aikaleimat”. Tässä osiossa näytetään käyttäjälle käyttäjän tämän päivän aikaleimat. Tällä pyritään muodostamaan käyttäjälle sovellusta käyttäessään näkemys hänen tämän päivän ajankäytöstään ja hän voi käyttää tätä tietoa hyödykseen päättäessään tekemisestään. (Kuva 12)



Kuva 13. Raportit-sivun rautalankamalli (Veikko Soosaar)

Viimeisenä löytyy rautalankamalli raporteille tarkoitetulle sivulle. Tällä sivulla on vain kaksi osiota. Vasemmanpuoleisesta osiosta löytyy “Raportti-taulu” jossa näytetään käyttäjälle raportti ajankäytöstään. Oikeanpuoleisesta osiosta löytyy “Latausvaihtoehdot”-taulukko, josta käyttäjä pääsee lataamaan tietokoneelleen tai muulle laitteelle tiedostomuodossa raportteja ajankäytöstään. (Kuva 13)

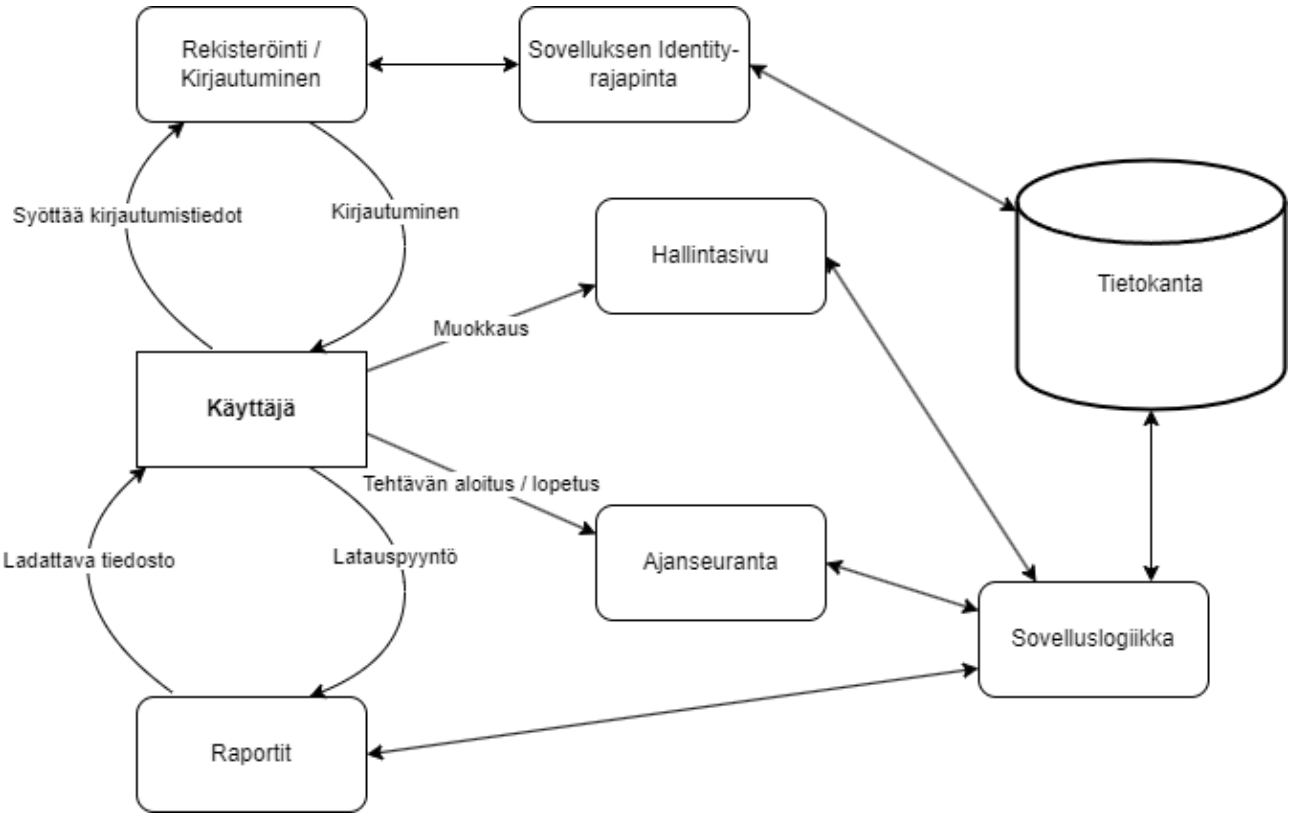
5.3 Tiedon kulku sovelluksessa

Tieto kulkee sovelluksessa käyttäjän ja tietokannan välillä montaa eri reittiä. Rekisteröinnin ja kirjautumisen yhteydessä käyttäjä syöttää kirjautumistiedot lomakkeelle ja nämä tiedot sitten välitetään sovelluksen Identity-rajapinnalle. Rajapinta joko rekisteröitymisen yhteydessä tallentaa tiedot tietokantaan, tai kirjautumisen yhteydessä tarkistaa tietojen oikeellisuuden. Kun rajapinta on saanut varmistuksen tietokannasta kirjautumistietojen oikeellisuudesta, palauttaa se vastauksen sivulle. Sivu sitten joko evää käyttäjän kirjautumisen tai antaa luvan kirjautumiselle. (Kuva 14)

Käyttäjän tehdessä muutoksia tehtävien tai aikaleimojen hallintasivulla, menee tieto tästä sovelluslogiikan puolelle. Sovelluslogiikka katsoo tietojen muodon olevan oikea ja tallentaa muutokset tietokantaan, mikäli tiedot ovat hyväksyttävässä muodossa. Tämä muutos sitten välittyy hallintasivulle käyttäjän näkyville. Tietokannasta myös haetaan olemassa olevat tiedot tehtävistä ja aikaleimoista hallintasivulle näkyviin. (Kuva 14)

Ajanseuranta-sivun logiikka toimii samantyyppisesti kuin hallintasivun. Käyttäjä valitsee täyttää lomakkeen, eli valitsee työtehtävän ja mahdollisesti kirjoittaa lisätietoja. Tämän jälkeen hän painaa Aloita, joka vie pyynnön eteenpäin sovelluslogiikan puolelle. Sovelluslogiikan puolella katsotaan ovatko tiedot hyvälaatuisia ja jos ne ovat, niin lähdetään viemään tiedot tietokantaan. Tietokannasta palautetaan sitten myös esimerkiksi sivun uudelleenlatauksen myötä tieto, että on olemassa kesken oleva tehtävä, jos sellainen aloitettiin. Lisäksi tietokannasta palautetaan sivulla näkyviä muita tietoja, kuten tehtävälistaus ja tämän päivän aikaleimat. Jos on olemassa aloitettu tehtävä, niin lomakkeella näkyy aloittamispainikkeen sijaan lopetuspainike. Tätä painamalla käyttäjä viestii sovelluslogiikalle haluavansa lopettaa kyseisen tehtävän ja sovellus vie tietokantaan tästä tiedon, eli tehtävän lopetusajankohdan. (Kuva 14)

Käyttäjän mennessä “Raportit”-sivulle, haetaan sivulle näkyviin sovelluslogiikan kautta tietokannasta olemassa olevat tiedot näkyviä raportteja varten. Lisäksi käyttäjä voi tehdä pyynnön tiedostolataukselle sivulla, jolloin ohjelma muodostaa sovelluslogiikan puolella tiedoston tietokantojen tietojen pohjalta. Tämä tiedosto sitten annetaan käyttäjälle ladattavaksi tietokoneelle. (Kuva 14)



Kuva 14. Tiedonkulkukaavio (Veikko Soosaar)

5.4 Tietokannan suunnittelu

Tietokanta on tärkeä osa sovellusta. Tietokantaa tarvitaan käyttäjän tietojen tallentamiseen, säilömiseen ja noutamiseen. Tämän projektin puitteissa tietokantaan tehdään vain sovelluksen toiminnan kannalta välttämättömiksi nähdyt osat. Tietokanta tulee olemaan relaatiotietokanta, joka toimii Microsoft SQL Server tietokantahallintajärjestelmässä. Tämä SQL Server järjestelmä tulee toimimaan Microsoft Azure pilvipalvelualustalla.

Osa tietokannan tauluista tulee ASP.NET Core Identity rajapinnan tarjoamista valmiista tauluista. Näitä ovat autorisointiin ja käyttäjäprofiileihin liittyvät taulut. Aikaleimoihin ja tehtäviin liittyvät taulut tehdään itse ja niissä hyödynnetään yhteyttä Identity rajapinnan tarjoamaa käyttäjätietoa tietojen yhdistämiseen käyttäjään. Kuvassa 15 on näkyvillä luontilauseet, joilla luotiin tehtäville taulu UserTasks ja aikaleimoille taulu TimeEntries.

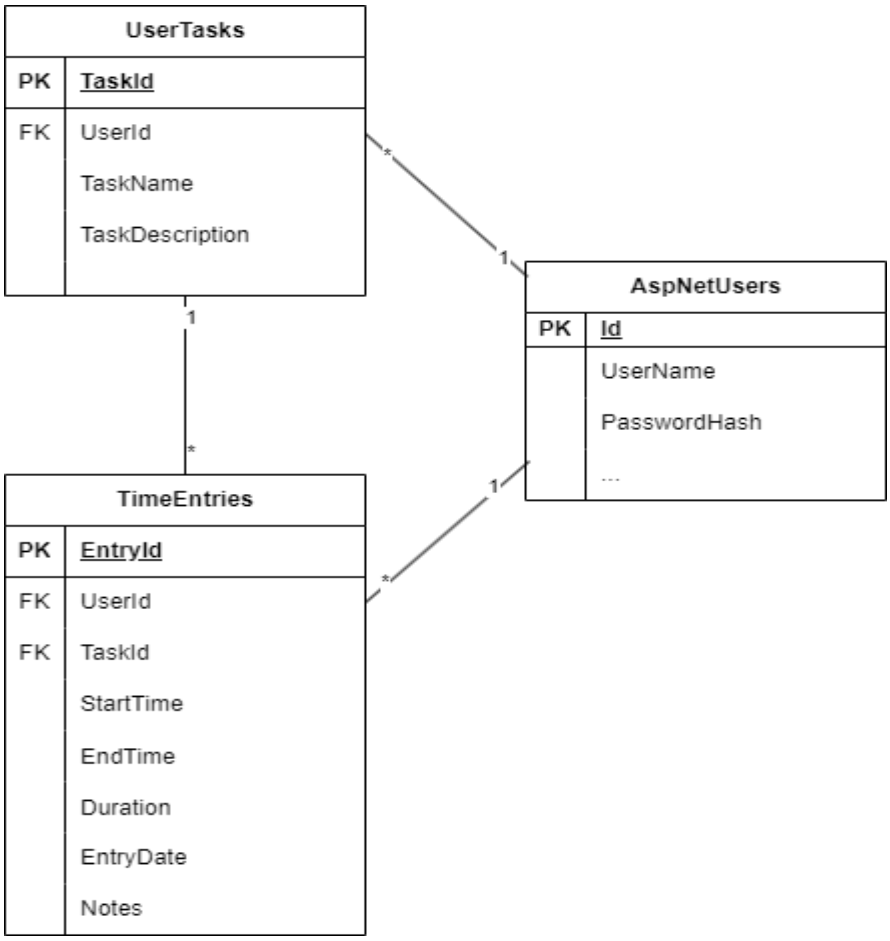
```
CREATE TABLE UserTasks (
  TaskId INT IDENTITY(1,1) PRIMARY KEY,
  UserId NVARCHAR(450),
  TaskName NVARCHAR(255),
  TaskDescription NVARCHAR(255),
  CONSTRAINT FK_AspNetUsers_UserTasks FOREIGN KEY (UserId)
    REFERENCES AspNetUsers(Id)
);

CREATE TABLE TimeEntries (
  EntryId INT IDENTITY(1,1) PRIMARY KEY,
  UserId NVARCHAR(450),
  TaskId INT,
  StartTime DATETIME,
  EndTime DATETIME,
  Duration INT,
  EntryDate DATE,
  Notes NVARCHAR(255),
  CONSTRAINT FK_AspNetUsers_TimeEntries FOREIGN KEY (UserId)
    REFERENCES AspNetUsers(Id),
  CONSTRAINT FK_UserTasks_TimeEntries FOREIGN KEY (TaskId)
    REFERENCES UserTasks(TaskId)
);
```

Kuva 15. UserTasks ja TimeEntries taulujen T-SQL luontilauseet (Veikko Soosaar)

Luontilauseista nähdään kenttien nimien lisäksi myös kenttien tietotyyppi ja kentän enimmäispituus sulkujen sisällä. Esimerkiksi TaskName kenttä viittaa tehtävän nimeen ja se on NVARCHAR-tekstimuodossa, jonka enimmäispituus on 255 merkkiä. INT tyyppi viittaa kokonaislukuun. DATETIME tietotyyppi viittaa tietotyyppiin, johon merkitään päivämäärä ja aika. DATE tietotyyppi viittaa

pelkästään päivämäärään. IDENTITY merkintä viittaa yksilöivään kenttään. PRIMARY KEY ja FOREIGN KEY viittaavat pääavaimeen ja viiteavaimeen. (Microsoft 3.3.2023b)

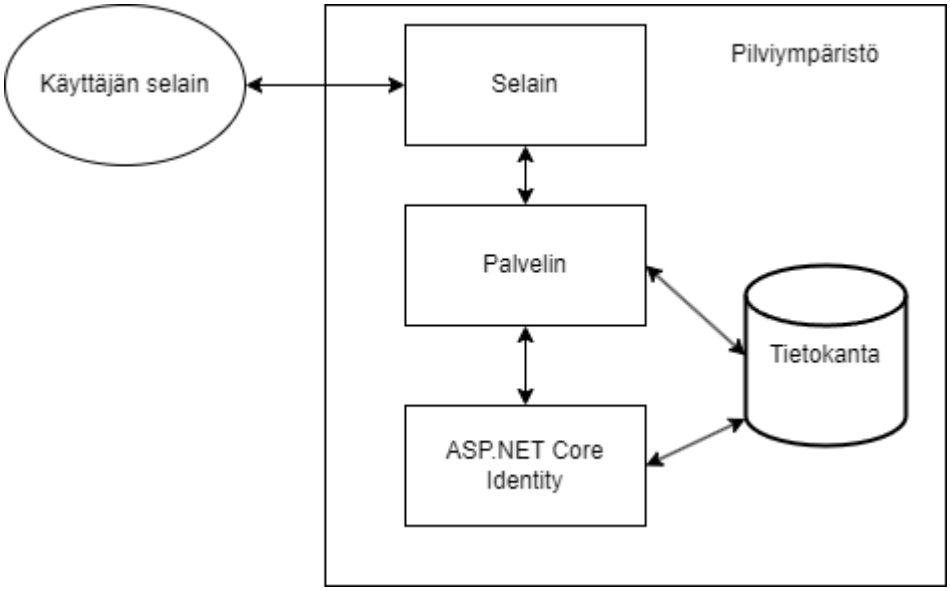
Automaattisesti luoduista tauluista hyödynnämme AspNetUsers nimistä taulua. Tähän tallennetaan käyttäjän tiedot ja meitä kiinnostaa muihin tauluihin liittyen käyttäjän yksilöivä Id tieto. Tällä tiedolla yhdistämme aikaleimat taulusta TimeEntries ja tehtävät taulusta UserTasks. Emme välitä tässä suunnittelussa muista automaattisesti luoduista tauluista, emmekä muista tiedoista kuin Id-tieto, joka tulee tuosta “AspNetUsers” taulusta. Tämä Id niminen tieto toimii pääavaimena, johon viitataan UserTasks taulun UserId kentästä ja TimeEntries taulun UserId kentästä vierasavaimena. UserTasks taulussa on yksilöivä kenttä TaskId, johon löytyy viittaus TimeEntries taulun TaskId kentästä vierasavaimen muodossa. UserTasks taulusta löytyy myös tehtävän nimelle kenttä TaskName ja tehtävän kuvaukselle kenttä TaskDescription. TimeEntries taulusta löytyy myös yksilöivä kenttä, jonka nimi on EntryId. Aiemmin mainittujen UserId ja TaskId kenttien lisäksi löytyy aikaleiman alkuajasta kertova StartTime ja tehtävän lopusta kertova EndTime. Lisäksi löytyy aikaleiman kestosta kertova Duration, aikaleiman viimeisimmästä päivitysajasta kertova EntryDate kenttä ja Notes kenttä kertoo aikaleimaan liittyvistä lisätiedoista. (Kuva 16)



Kuva 16. Tietokantakaavio (Veikko Soosaar)

5.5 Järjestelmäarkkitehtuuri

Järjestelmäarkkitehtuurissa katsotaan järjestelmää korkeammalta tasolta. Tarkoitus on käydä läpi isosta näkökulmasta järjestelmän eri osiot. Näitä ovat selainpuolen toiminta, palvelinpuolen toiminta ja infrastruktuuri. Sovellus toimii Microsoft Azure alustan Azure App Service pilvipalveluympäristössä. Käyttäjä käyttää verkkoselaintaan sovelluksen käyttämiseksi. Sovellus näyttää selainpuolen ratkaisun käyttäjälle. Tämä selainpuolen ratkaisu kommunikoi palvelinpuolen logiikan kanssa. ASP.NET Core Identity toimii autorisoinnin rajapintana käyttäjälle. Palvelinlogiikka ja autorisoinnin rajapinta molemmat keskustelevat tietokannan kanssa. (Kuva 17)



Kuva 17. Järjestelmäarkkitehtuuri (Veikko Soosaar)

5.6 Sovelluksen rakentaminen ja toimittaminen

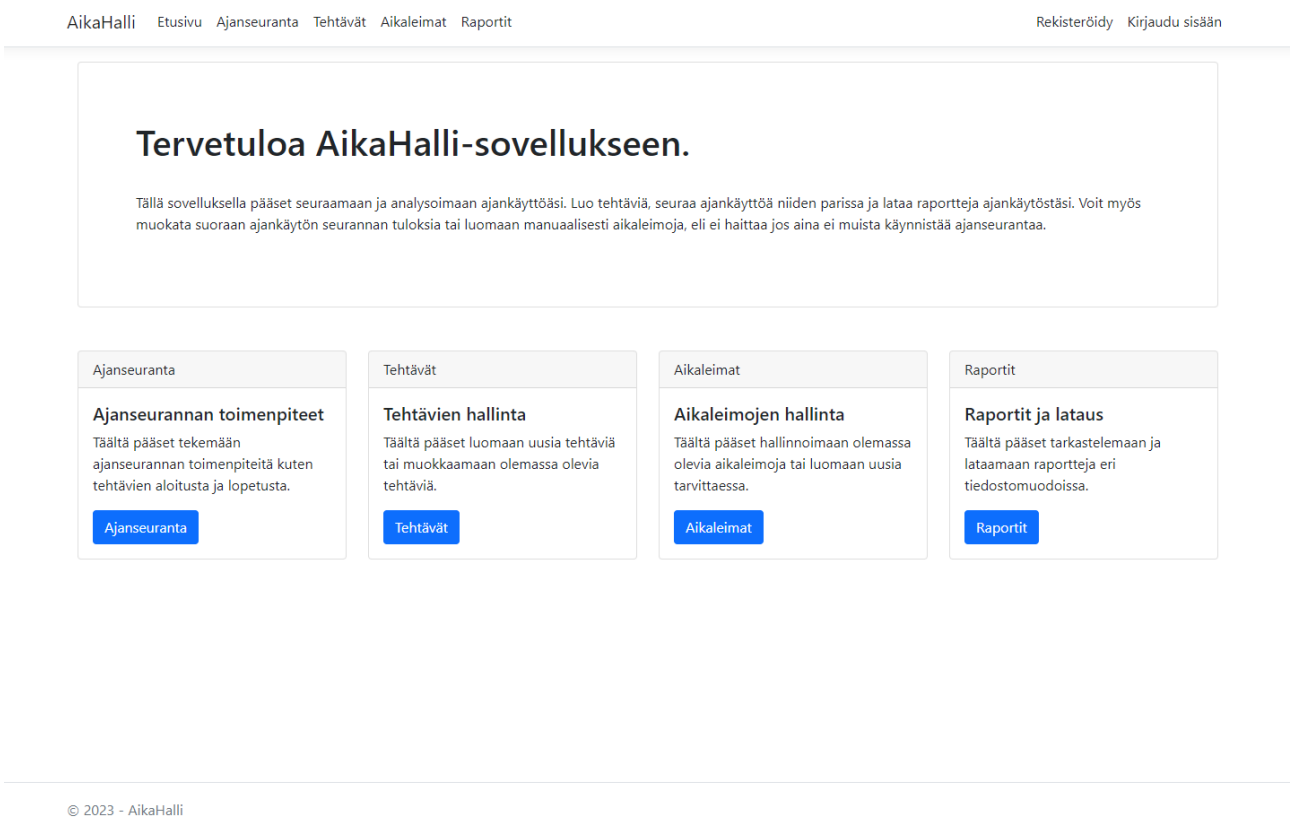
Sovelluksen lähdekoodin siirtyessä versiohallinnan kautta Git-versionhallintajärjestelmän kautta GitHub palvelussa sijaitsevaan koodisäilöön GitHub Actions palvelussa käynnistyy prosessi. Tämä prosessi rakentaa ja toimittaa sovelluksen Microsoft Azure App Services palveluun käynnistettäväksi.

6 Sovelluksen toteutus

Tässä osassa opinnäytetyötä esitellään sovelluksen eri osioiden toteutusta. Ensin käydään läpi selainpuolen toteutusta, mikä on merkittävä käyttäjän käyttökokemuksen kannalta. Sen jälkeen käymme palvelinpuolen toteutusta läpi. Palvelinpuoli on se osio sovelluksesta, jossa kaikki logiikka tapahtuu ja kaikki viestintä selainpuolen ja tietokannan välillä tapahtuu palvelinpuolen kautta. Tietokannan toteutuksesta kertovassa osiossa esitellään tietokantataulujen luomista ja käyttöä. Tietokannan toteutuksen esittelyn jälkeen esitellään ympäristöön ja sovelluksen käyttöön vientiin liittyviä toteutuksen osia. Näihin kuuluvat versionhallinta, sovelluksen vientiprosessi koodista käytettäväksi sovellukseksi ja ympäristö, jossa sovellus toimii käytännössä.

6.1 Selainpuolen toteutus

Selainpuolen toteutuksessa hyödynnettiin Razor syntaksia HTML merkintäkielen lisäksi. Tyylittelyissä hyödynnettiin Bootstrap viitekehyksen antamia työkaluja. Sovellus toteutettiin suunnitelmien ja vaatimusten mukaisesti. Kuvassa 18 näkyy sovelluksen etusivu.



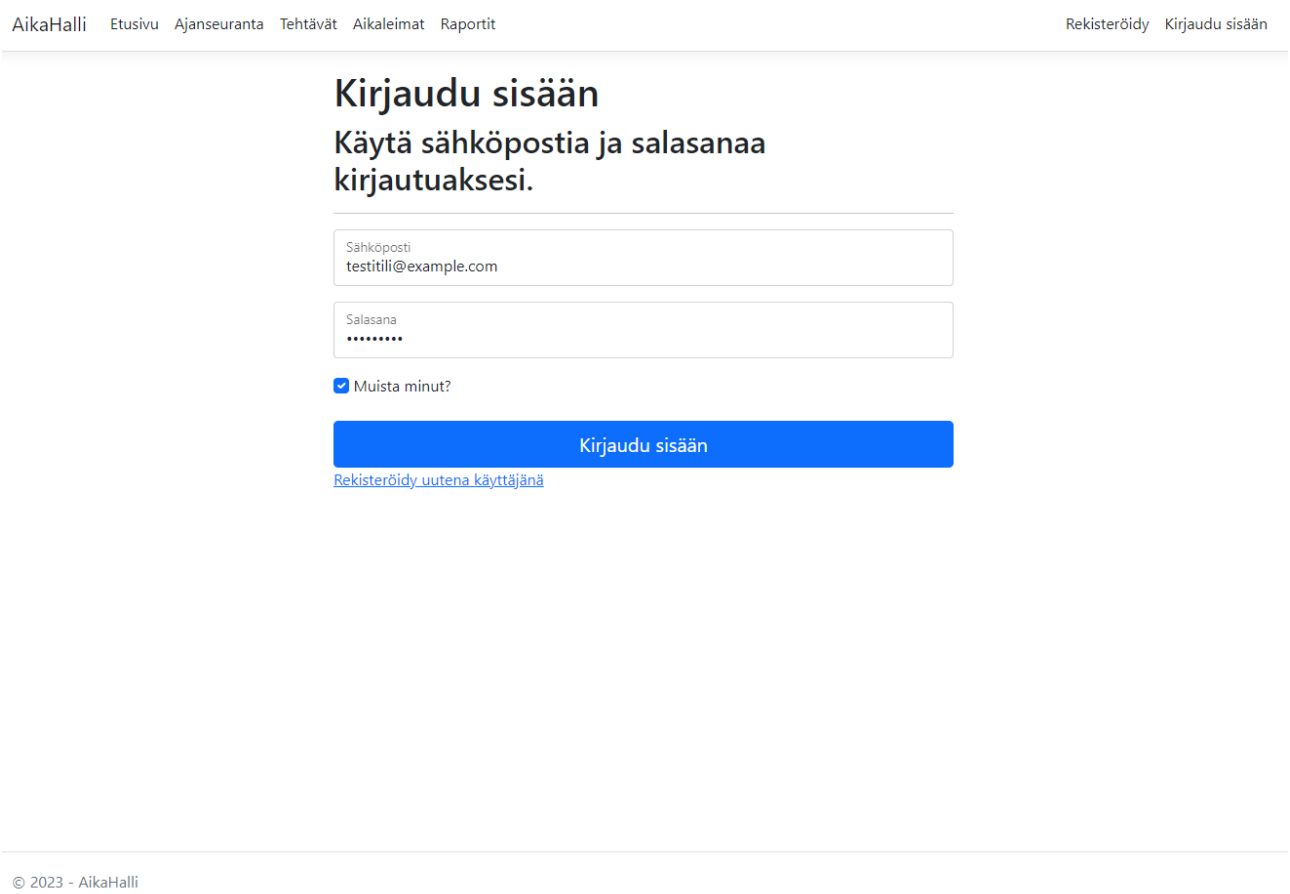
Kuva 18. Sovelluksen etusivu (Veikko Soosaar)

Etusivulla käytettiin sisältöä varten HTML div-elementtejä, joita muotoiltiin Bootstrapin luokkien avulla. Tässä tapauksessa käytettiin Bootstrapin korttielementtejä, minkä rakenteesta ja toteutuksesta esimerkki kuvassa 19.

```
<div class="card">
    <div class="card-header">
        Ajanseuranta
    </div>
    <div class="card-body">
        <h5 class="card-title">Ajanseurannan toimenpiteet</h5>
        <p class="card-text">Täältä pääset tekemään ajanseurannan toimenpiteitä kuten tehtävien aloitusta ja lopetusta.</p>
        <a class="btn btn-primary" asp-page="/TimeTracker">Ajanseuranta</a>
    </div>
</div>
```

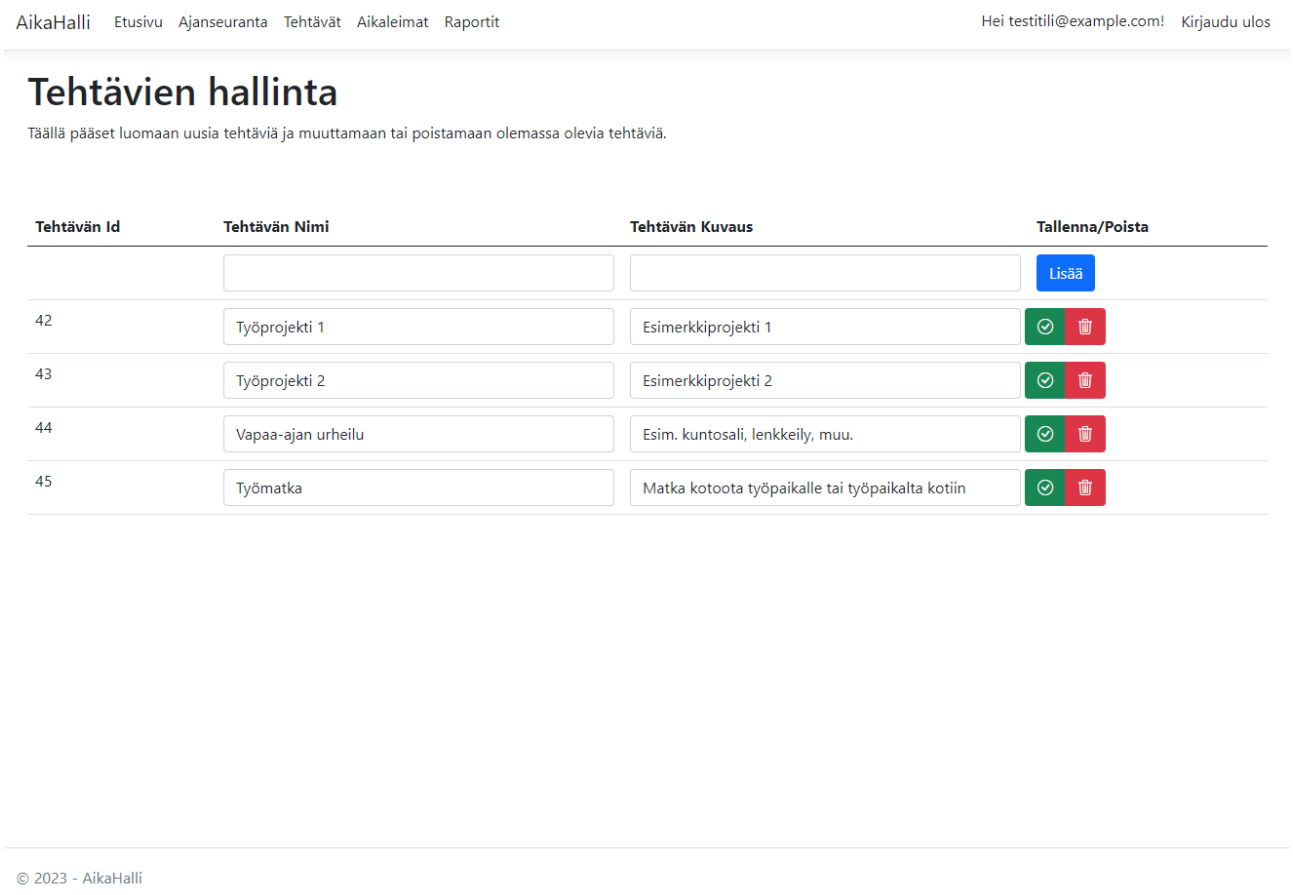
Kuva 19. Etusivun HTML-koodi Ajanseuranta-osiolle tehty Bootstrap korttitoteutuksella (Veikko Soosaar)

Sivun oikealla ylälaidalla on toteutettuna linkit Rekisteröidy ja Kirjaudu sisään (kuva 18). Nämä vievät sovelluksen rekisteröitymisen ja kirjautumisen sivuille, jotka ovat hyvin samankaltaiset. Kuvassa 20 näkyy esimerkkinä kirjautumissivu, jossa täytetään lomake sähköpostilla ja salasanalla kirjautumista varten.



Kuva 20. Sovelluksen kirjautumissivu (Veikko Soosaar)

Sovelluksen rekisteröitymisen tai sisäänkirjautumisen jälkeen voidaan siirtyä esimerkiksi uutena käyttäjänä siirtyä tehtävien hallintasivulle. Tällä sivulla käyttäjä pääsee luomaan, muokkaamaan tai poistamaan tehtäviä. Uuden tehtävän lisäämisestä kertoo sininen “Lisää”-painike. Olemassa olevien rivien päivittämiseksi voidaan painaa rivin lopussa olevaa vihreätä hyväksymispainiketta tai poistamista halutessa punaista painiketta, jossa on roskakorin kuva. (Kuva 21)



Kuva 21. Sovelluksen Tehtävien hallinta sivu (Veikko Soosaar)

Kuvassa 21 näkyvillä olevat tehtävät on toteutettu HTML taulukkona, joka toimii lomakkeen sisällä. Taulukon rivit on luotu Razor syntaksia hyödyntäen dynaamisesti. Myös muokattavien kenttien tiedot on kiinnitetty palvelinpuolen koodissa oleviin muuttujiin Razor-syntaksin mukaisesti “asp-for”-merkinnän avulla. Tauluun ei luoda uuden tehtävän luomisrivin jälkeen rivejä, jos niitä ei löydy tietokannasta. Tämä tieto tarkistetaan tämän Razor syntaksin alussa katsomalla onko “Model.Items” muuttuja arvoltaan null, eli tyhjä. (Kuva 22)

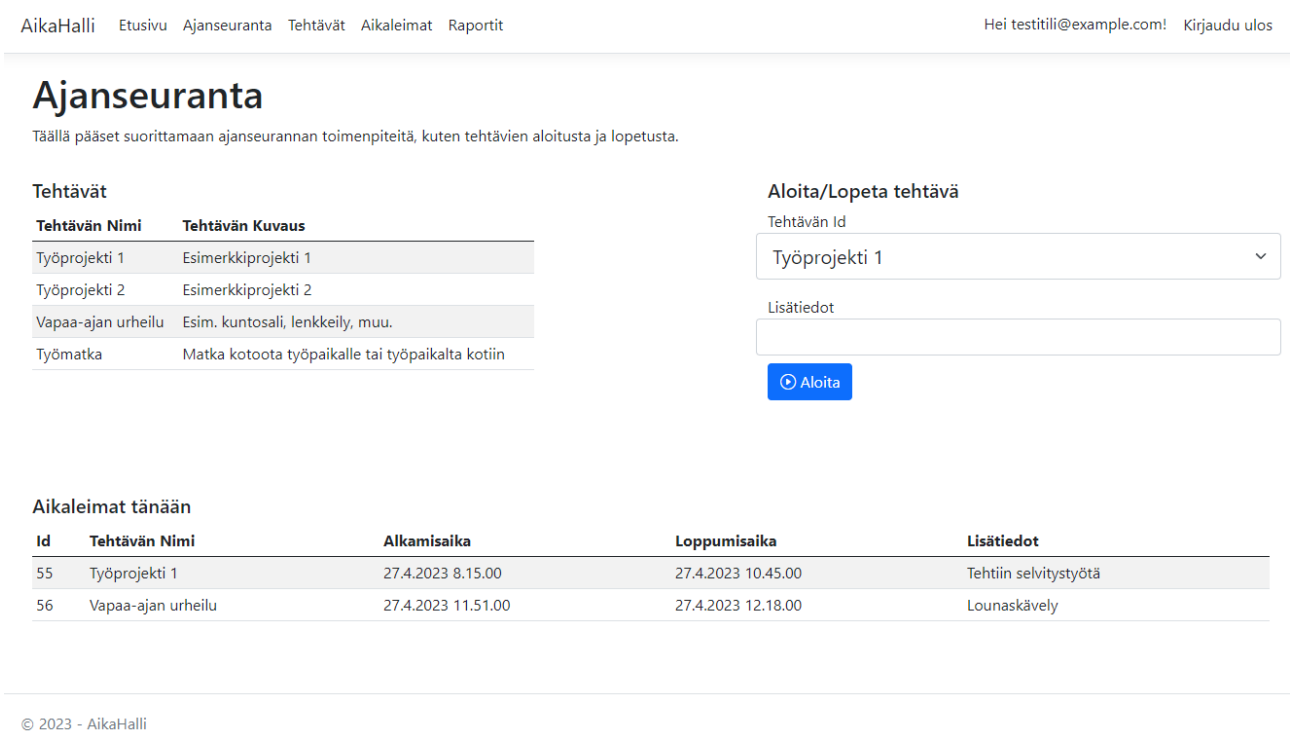
```
<form id="task" asp-page="Tasks" method="post">
    <table class="table">
        <thead>
            <tr>
                <th>Tehtävän Id</th>
                <th>Tehtävän Nimi</th>
                <th>Tehtävän Kuvaus</th>
                <th>Tallenna/Poista</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td></td>
                <td><input type="text" asp-for="NewItem.TaskName" class="form-control"  maxlength="255"/></td>
                <td><input type="text" asp-for="NewItem.TaskDescription" class="form-control"  maxlength="255"/></td>
                <td><button type="submit" asp-page-handler="AddNewUserTask" class="btn btn-primary">Lisää</button></td>
            </tr>
            @if (Model.Items is not null)
            {
                for (int i = 0; i < Model.Items.Count; i++)
                {
                    <tr>
                        <td>@Model.Items[i].TaskId</td>
                        <td><input asp-for="@Model.Items[i].TaskName" type="text" value="@Model.Items[i].TaskName"
                            class="form-control" name="Items[@Model.Items[i].TaskId].TaskName"  maxlength="255"/></td>
                        <td><input asp-for="@Model.Items[i].TaskDescription" type="text" value="@Model.Items[i].TaskDescription"
                            class="form-control" name="Items[@Model.Items[i].TaskId].TaskDescription"  maxlength="255"/></td>
                        <td>
                            <div class="row btn-group flex-nowrap">
                                <button type="submit" asp-page-handler="UpdateUserTask" asp-route-id="@Model.Items[i].TaskId"
                                class="btn btn-success"><i class="bi-check-circle"></i></button>
                                <button type="submit" asp-page-handler="DeleteUserTask" asp-route-id="@Model.Items[i].TaskId"
                                class="btn btn-danger"><i class="bi-trash3"></i></button>
                            </div>
                        </td>
                    </tr>
                }
            }
        </tbody>
    </table>
</form>
```

Kuva 22. Tehtävien hallintasivun taulukon rivien ohjelmakoodi (Veikko Soosaar)

Aikaleimojen hallintasivu toimii hyvin samankaltaisesti kuin tehtävien hallintasivu. Erona näiden kahden hallintasivun välillä on kenttien määrä ja tyypit. Aikaleimojen hallinnan sivulla on enemmän muokattavia kenttiä ja erityisesti päivämäärän ja ajan kenttä on hyvin erilainen. Lisäksi jokaisen rivin alussa on pudotusvalikko, josta voi valita tehtävän aikaleimalle. (Kuva 23)

Kuva 23. Sovelluksen Aikaleimojen hallinta sivulla oleva aikaleimojen muokkaustaulukko (Veikko Soosaar)

Sovelluksen käyttäjän käytön kannalta eniten käytettyjä osioita on luultavasti Ajanseuranta. Sivulla aloitetaan ja lopetetaan tehtävien aikaleimoja. Sivulla on myös listaus olemassa olevista käyttäjän tehtävistä ja tämän päivän aikaleimoista. Tehtävien ja päivän aikaleimojen listaus on toteutettu samankaltaisesti kuin kuvassa 22, mutta ilman että kentät ovat muokattavissa. Aloita/Lopeta tehtävä osiossa käytetään lomaketta, jossa valitaan tehtävä ja annetaan mahdolliset lisätiedot tehtävästä. Nämä tiedot sitten siirtyvät palvelinpuolelle käsiteltäväksi. (Kuva 24)

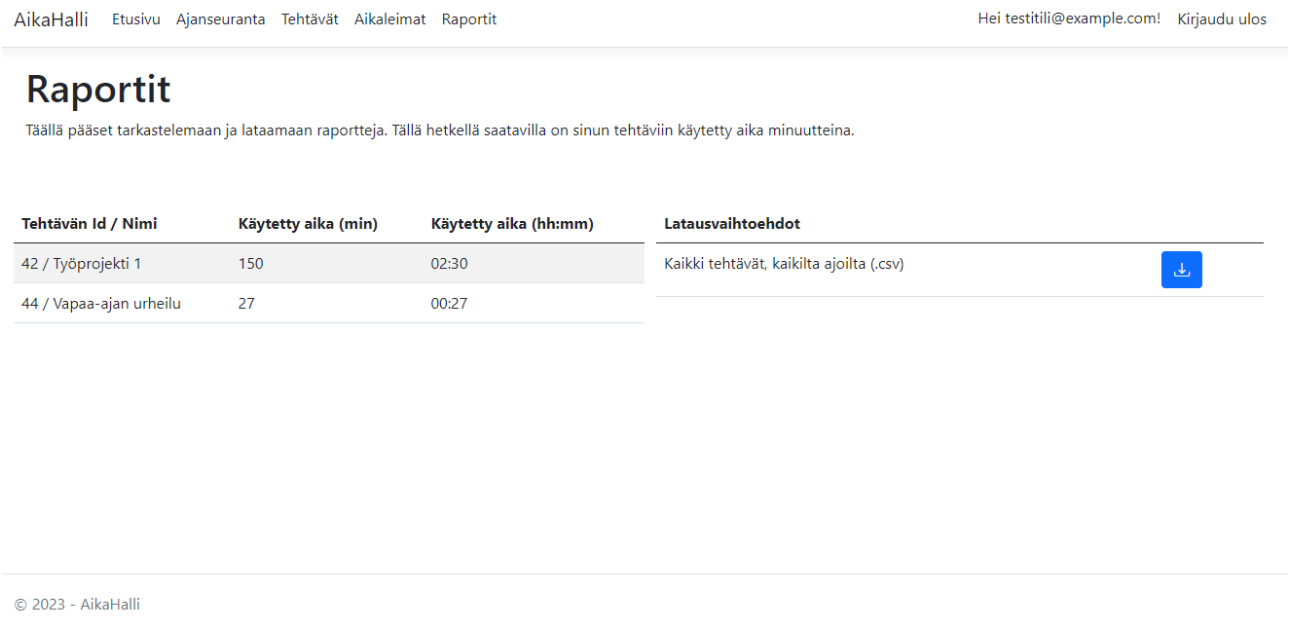


Kuva 24. Sovelluksen Ajanseuranta osion sivu (Veikko Soosaar)

Käyttäjän painaessa ajanseurannan sivulla Aloita, ladataan sivu uudelleen ja katsotaan, tuleeko palvelinpuolelta tieto kesken olevasta tehtävästä. Jos tulee niin “Aloita/Lopeta tehtävä”-osio muuttuu ja Aloita painikkeen tilalle tulee painike Lopeta (kuva 25).

Kuva 25. Sovelluksen Ajanseuranta sivun Aloita/Lopeta tehtävä osio (Veikko Soosaar)

Käyttäjä haluaa jossain vaiheessa sovelluksen käyttöään tarkastella raportteja ajankäytöstään. Raportit-sivulle toteutettiin taulukko, jossa näkyy ajankäytön raportti tehtäväkohtaisesta ajankäytöstä. Toinen osio sivussa sisältää taulukon, jossa näky raporttien latausvaihtoehdot, joista tällä hetkellä toteutettiin pelkästään vaihtoehto “Kaikki tehtävät, kaikilta ajoilta”. Latausta varten tehtiin sinisen värinen painike, jossa on latausikoni. (Kuva 26)

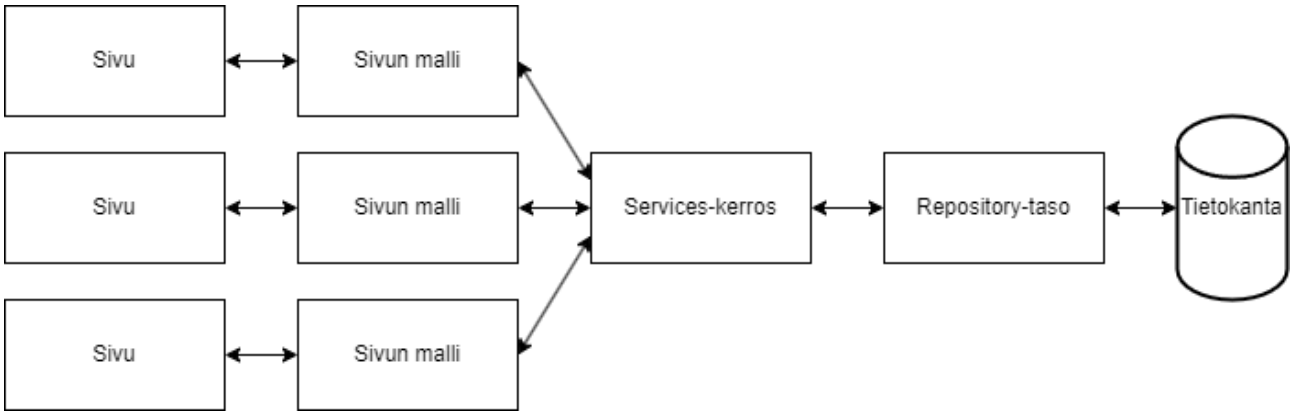


Kuva 26. Sovelluksen Raportit-sivu (Veikko Soosaar)

6.2 Palvelinpuolen ja tietokannan toteutus

Sovelluksen selainpuolelta, eli käyttäjän selaamilta sivuilta tapahtuu kommunikaatiota palvelinpuolelle. Tässä opinnäytetyöprojektissa käytetään palvelinpuolen arkkitehtuuriin “Service / Repository Pattern” mallia. Tämän mallin ajatus on, että sivu ja sivun mallin logiikka ovat täysin erillään syvemmästä sovelluslogiikasta ja tietokantaan kohdistuvista toimenpiteistä. Sovelluslogiikka sijaitsee Services-tasolla ja tietokantaan kohdistuvat toimenpiteet Repository-tasolla. (Michaels 21.1.2023)

Tämän projektin sovelluksessa esimerkiksi tieto siirtyy selainpuolen sivuilta sivun mallille ja sivun mallilta selaimelle. Sivun mallissa tapahtuu sivuun liittyvät toiminnot, kuten esimerkiksi muuttujien määritys sivulla olevia elementtejä varten tai kutsujen tekeminen Services-tasolle. Services-tasolla sijaitsee sovelluksen logiikka ja sieltä voidaan tehdä esimerkiksi kutsuja Repository-tasolle. Repository-tasolla tehdään kaikki tietokantaan kohdistuvat toiminnot ja kutsut. (Kuva 27)



Kuva 27. Sovelluksen rakenne (Veikko Soosaar)

Sovelluksen sivun avautuessa sivu mallissa tapahtuu kutsu OnGetAsync nimiseen metodiin. Esimerkiksi Ajanseuranta-sivulla OnGetAsync metodissa alustetaan lista Items, joka on lista TimeEntry olioista. TimeEntry on aikaleiman tiedot sisältävä olio ja tähän listaan halutaan lista näistä olioista käyttäjän aikaleimojen tiedoilla. Metodissa lähdetään liikkeelle listan alustamisen jälkeen hakemalla UserId-muuttujaan käyttäjän yksilöivä id-tieto GetUserId-metodilla. Tätä UserId tietoa tarvitaan, kun halutaan seuraavaksi hakea käyttäjälle tämän päivän aikaleimat Items listaan. Tämä listan täyttäminen tapahtuu kutsumalla IAikaHalliService ilmentymän _aikaHalliService metodissa GetAllUserTimeEntriesToday. Tähän metodiin asetetaan parametriksi UserId, jotta tiedetään, kenen käyttäjän tietoja halutaan hakea. Samalla periaatteella haetaan myös tiedot TaskList listaan ja CurrenItem muuttujaan. (Kuva 28)

```
public async Task<IActionResult> OnGetAsync()
{
    Items = new List<TimeEntry>();
    try
    {
        UserId = GetUserId();
        Items = await _aikaHalliService.GetAllUserTimeEntriesToday(UserId);
        TaskList = await _aikaHalliService.GetAllUserTasks(UserId);
        CurrentItem = await _aikaHalliService.GetCurrentTimeEntry(UserId);
    }
    catch (Exception)
    {

        throw;
    }
    return Page();
}
```

Kuva 28. Ajanseuranta sivumallin OnGetAsync metodi (Veikko Soosaar)

Seuraavaksi AikaHalliService luokan GetAllUserTimeEntriesToday metodissa hyödynnetään IAikaHalliRepositoryn ilmentymää _aikaHalliRepository ja kutsutaan sieltä luokkaa GetAllUserTimeEntriesToday UserId-parametrin kanssa (kuva 29). Tässä tapauksessa ei Service-tasolla tehty muuta loogista päättelyä kuin kutsu Repository-tasolle. Tällä tasolla kuitenkin voidaan tehdä esimerkiksi tiedon käsittelyä ennen olion lähettämistä Repository-tasolle (kuva 30).

```
public async Task<List<TimeEntry>> GetAllUserTimeEntriesToday(string userId)
{
    return await _aikaHalliRepository.GetAllUserTimeEntriesToday(userId);
}
```

Kuva 29. AikaHalliService luokan metodi GetAllUserTimeEntriesToday (Veikko Soosaar)

```
3 references
public async Task AddTimeEntry(TimeEntry timeEntry)
{
    HandleTimeEntry(timeEntry);
    await _aikaHalliRepository.AddTimeEntry(timeEntry);
}
```

Kuva 30. AikaHalliService luokan metodi AddTimeEntry (Veikko Soosaar)

Repository-luokassa GetAllUserTimeEntriesToday metodi luo ja käyttää context-tietokantakontekstia, joka on tyypiltään ApplicationDbContext. Kyseessä on tietokantakonteksti, jonka avulla voidaan tehdä toimenpiteitä esimerkiksi tietokannan tauluihin. Tämä luodaan CreateDbContext-metodilla, joka saadaan IDbContextFactoryn ilmentymästä _appContextFactory. Seuraavaksi luodaan timeEntries muuttuja, johon haetaan tietoja context-tietokantakontekstista TimeEntries taulun viittauksesta tekemällä Where-metodilla haku tietojen rajaamiseksi. Haetaan täsmääviä UserId tietoja, eli käyttäjän yksilöivien tietojen perusteella tehdään rajaus. Lisäksi rajataan Where-haulla myös aikaleiman alkuajan mukaan katsomalla, että StartTime tiedon päivämäärä on sama kuin tämän päivän päivämäärä. Lopuksi palautetaan lista timeEntries. (Kuva 31)

```
public async Task<List<TimeEntry>> GetAllUserTimeEntriesToday(string userId)
{
    using var context = _appContextFactory.CreateDbContext();
    var timeEntries = await context.TimeEntries
                            .Where(x => x.UserId == userId
                                    && (x.StartTime ?? DateTime.Today).Date == DateTime.Today)
                            .ToListAsync();

    return timeEntries;
}
```

Kuva 31. AikaHalliRepository luokan metodi GetAllUserTimeEntries (Veikko Soosaar)

Services ja Repository tasoilla on useita eri metodeja eri käyttötarkoituksiin, mutta pohjimmiltaan prosessi ja rakenne ovat saman kaltaisia kuin edellä esitellyssä tämän päivän aikaleimojen haussa.

Repository tason käyttämä tietokanta on toteutettu kuvan 15 mukaisilla T-SQL luontilauseilla. Nämä lauseet on ajettu SQL Server Management Studio 19 ohjelmistolla SQL Server tietokantapalvelimelle Samasta ohjelmistosta päästään ajamaan myös T-SQL lauseita esimerkiksi tiedon hakua varten. Esimerkiksi seuraavalla lauseella voidaan hakea kaikkien käyttäjien luomat tehtävät:

SELECT * FROM dbo.UserTasks

Tällä lauseella suoritetun haun tuloksena saadaan tulosjoukko UserTasks taulusta, joka näkyy kuvassa 32. Tulosjoukossa TaskId tieto alkaa numerosta 42, koska aiemmin luodut rivit ovat poistettu testauksen yhteydessä.

	TaskId	UserId	TaskName	TaskDescription
1	42	bbe24c54-6c22-4926-8d2d-5a8e33200eef	Työprojekti 1	Esimerkkiprojekti 1
2	43	bbe24c54-6c22-4926-8d2d-5a8e33200eef	Työprojekti 2	Esimerkkiprojekti 2
3	44	bbe24c54-6c22-4926-8d2d-5a8e33200eef	Vapaa-ajan urheilu	Esim. kuntosali, lenkkeily, muu.
4	45	bbe24c54-6c22-4926-8d2d-5a8e33200eef	Työmatka	Matka kotoota työpaikalle tai työpaikalta kotiin

Kuva 32. Tietokannan UserTasks taulun sisältö (Veikko Soosaar)

6.3 Sovelluksen vienti käyttöympäristöön

Sovellus viedään käyttöympäristöönsä Microsoftin Azure App Services palveluun GitHubin Actions palvelun kautta. Actions palveluun on määritelty toiminto, joka reagoi uusiin muutoksiin tämän sovelluksen koodisäilössä. Kun tehdään koodisäilöön uusi päivitys versiohallinnan kautta, aktivoituu tämä toiminto Actions palvelussa. Tämä toiminto oli valittu valmiiksi tarjotuista vaihtoehdoista, jolloin ei ollut tarve itse muunnella toiminnon osia itse. Toiminto rakentaa koodisäilössä olevasta koodista toimivan ohjelmapaketin ja vie sen sitten Azure App Services palveluun käynnistettäväksi.

Azure-palvelussa luotiin ryhmä sovelluksen resursseille nimeltä AikaHalli_group. Tämän resurssiryhmän sisälle luotiin AikaHalli niminen App Service, aikahallisqlserver niminen SQL Server tietokantapalvelin ja aikahallidatabase niminen tietokanta. AikaHalli App Servicelle asetettiin asetuksissa sovelluksen konfiguraatiotiedoston tietokantayhteysosoitteen korvaava tieto. Tällöin ei tarvitse huolehtia tietokantayhteysosoitteen päätymisestä vahingossa versionhallintaan ja voidaan pitää kyseinen tieto App Service palvelussa salassa. Muita muutoksia ei palveluun tehty ja koko prosessi on erittäin automatisoitu.

7 Toteutetun sovelluksen analysointi

Tässä kohdassa opinnäytetyötä analysoidaan toteutettua sovellusta. Ensin arvioidaan onnistumista, eli onnistuiko sovelluksen toteutus ja kuinka hyvin se täyttää vaatimukset. Pohditaan myös sovelluksen tilannetta ja sen jatkokehitysmahdollisuuksia. On hyvä tiedostaa sovelluksen kohdat, jotka saattavat vaatia lisää työstöä tai uusia ominaisuuksia.

7.1 Sovelluksen toteutuminen

Sovellus täyttää käyttäjätarinoiden vaatimukset ja sitä pystyy käyttämään tarkoituksenmukaisesti. Tarkoitus oli luoda toimiva sovellus, eikä lopullista tuotetta. Eli opinnäytetyöprojektin vaatimassa laajuudessaan tuotos on onnistunut. Sovelluksessa on kuitenkin vielä paljon parannettavaa ja jatkokehittämisen potentiaali on iso.

7.2 Sovelluksen parantamis- ja jatkokehitysmahdollisuudet

Sovellukseen tehtiin erittäin yksinkertaistettu autorisointi. Käyttäjältä vaaditaan vain sähköposti ja salasana, eikä sähköpostia tarvitse edes varmistaa tai salasanaa palauttaa sähköpostin kautta. Autorisoinnin parannukset ovatkin yksi suurimman prioriteetin parannuksia mitä sovellukselle voidaan tehdä. Lisäämällä mahdollisuus jo sähköpostin varmistamiselle ja salasanan palauttamiselle voidaan lisätä paljon enemmän turvaa sovellukselle ja sen käyttäjille. Lisäksi voitaisiin lisätä mahdollisuus kirjautua kolmannen osapuolen palvelun, kuten esimerkiksi Google-tilin kautta. Tämä lisäisi käyttömukavuutta ja mahdollistaisi käyttäjille helpon tavan ryhtyä sovelluksen käyttäjäksi.

Aikaleimojen hallinnan sivulla yksi käyttökokemuksen kannalta haastava osa on aloitusajan ja lopetusajan kenttien käyttäminen. Nämä kentät voitaisiin jakaa erillisiin päivämäärä- ja aikakenttiin, minkä lisäksi voitaisiin hyödyntää olemassa olevia kirjastoja, jotka tarjoavat paremman käyttökokemuksen tämäntyyppisten kenttien käyttämiselle. Sovelluksesta puuttuu myös tarkistus sille, että jos aikaleiman loppumisaika on ennen aloitusaikaa. Tällainen tapaus johtaisi virheellisen raporttiin, koska aika laskettaisiin negatiivisesti. Yksinkertainen korjaus tähän voisi olla esimerkiksi asettamalla tehtävän lopettamisaika samaksi kuin aloitusaika, mikäli se on asetettu liian pieneksi käyttäjän toimesta. Käyttäjää varoittamalla ja validointia käyttämällä voidaan myös saada käyttökokemuksen kannalta paljon parempia tuloksia. Sovelluksessa validointi on tällä hetkellä olematonta tai vähintään puutteellista eri lomakkeilla. Käyttäjälle ei anneta tarpeeksi tietoa hänen syöttäessään vääränlaista tietoa kenttiin ja kenttien sisältöä ei tarkisteta tarpeeksi hyvin hyvän käyttökokemuksen takaamiseksi.

Raportit-sivulla on tässä toteutuksessa hyvin rajattu määrä sisältöä. Tällä hetkellä on mahdollista nähdä tai ladata raportti pelkästään kaikista tehtävistä kaikilta ajoilta. Olisi hyvä tarjota käyttäjälle

mahdollisuus rajata esimerkiksi raportin aikaväliä käyttäjän asettamilla päivämäärillä tai valmiilla vaihtoehdoilla kuten “tämä kuukausi” tai “tämä viikko”. Niin näkyvien, kuin ladattavienkin raporttien osalta olisi hyvä mahdollistaa käyttäjälle valita sarakkeet, joita käyttäjä haluaa raportilleen. Ladattavien raporttien osalta myös tiedostotyyppien suhteen olisi hyvä olla laajempi valikoima. Tämänhetkisen CSV-tiedostotyypin lisäksi olisi hyvä tarjota esimerkiksi Excel-tiedosto tai tavallinen tekstitiedosto. Raportoinnin tukena voisi olla myös hyvä näyttää kuvaajia raporttitaulukkoihin liittyen. Kuvaajien avulla voidaan auttaa käyttäjää ymmärtämään paremmin ajankäyttöään ja se tuo sovellukseen lisää monipuolisuutta tarjottuun sisältöön.

Sovelluksessa on tällä hetkellä vahva oletus siitä, että käyttäjä ymmärtää kentät ja sovelluksen toimintaperiaatteen. Sovelluksen käyttökokemuksen vahvistamiseksi olisi kuitenkin hyvä lisätä lomakkeiden otsikoiden tai kenttien lähelle ikoni, josta aukeaisi pieni vihjeteksti kyseiseen elementtiin liittyen. Haluttaessa voitaisiin jopa tehdä täysin uusi sivu sovellukselle, jossa olisi kootusti ohjeita ja esimerkiksi “Usein Kysytyt Kysymykset”-osio neuvomassa käyttäjää sovelluksen käytössä.

Tällä hetkellä sovelluksessa näkyy etusivulla linkit ja kortit eri sovelluksen osioihin. Olisi hyvä tutkia olisiko parempi peittää nämä linkit ja kortit, jos käyttäjä ei ole kirjautunut sisälle. Tällä hetkellä voi olla, että käyttäjälle tulee mielikuva näille sivuille pääsemisestä jo kirjautumattomana. Kortit etusivulla voisi peittää esimerkiksi kirjautumissivun linkillä, jolloin käyttäjälle selviää jo etusivulle päästessään kirjautumisen olevan pakollista sovelluksen käyttämiseksi.

Sovellusta ei ole testattu optimoinnin tai nopeuden kannalta ollenkaan tässä opinnäytetyössä. Sovellus hakee ja vie tällä hetkellä kaikille sivuille päästessään ja toimintoja tehdessään tiedot suoraan tietokantaan. Välimuistitus voisi olla hyvä tietokantakuorman keventämisen väline. Välimuistituksella voidaan pitää jotkin tietokannan tiedot välimuistissa, jolloin niitä ei tarvitse hakea jokaisella sivun avauksella uudelleen tietokannasta. Tällainen tapaus voisi olla vaikka, jos käyttäjä käy luomassa ensin tehtäviä ja tämän jälkeen siirtyy Ajanseurannan sivulle. Molempien sivujen latauksen yhteydessä haetaan käyttäjän luomat tehtävät tietokannasta. Näitä tehtäviä ei kuitenkaan muuteta ollenkaan Ajanseurannan sivulla, joten tiedot voisi hyvinkin hakea välimuistista suoran tietokantahaun sijaan. Tietokannan käyttöä olisi muutenkin hyvä optimoida. Taulurakenne saattaa kaivata uudistusta ja lisää sarakkeita. Käyttäjille voisi antaa lisää mahdollisuuksia erilaisten tietojen tallentamiseen.

Tietokannassa ja sovelluksessa voisi myös lokittaa käyttäjien tapahtumia, jotta voidaan selvittää esimerkiksi väärinkäytöksiä tai virhetilanteiden syntyhetkien tietoja mahdollisesti. Vaikka sovelluksessa ei ole ainakaan nykyisessä versiossa henkilötietoja, niin olisi silti hyvä varmuuden vuoksi lokittaa sovelluksen käyttöä. Käyttäjä saattaa vahingossa tallentaa arkaluontoista tietoa esimerkiksi aikaleimojen lisätiedot-kenttään. Käyttäjämäärän kasvaessa myös tietokannan koko kasvaa ja voi

tulla eteen tilanteita, joissa huomataan tietokannan toimivan hitaammin kuin olisi toivottu. Paitsi suunnittelemalla tietokantarakennetta uudelleen, niin tietokantaan olisi myös hyvä harkita indeksointia tauluille tietojen nopeampaa hakemista varten.

Sovelluksen selainpuolen käyttöliittymässä voitaisiin parantaa käyttökokemusta käyttämällä enemmän värejä ja elementtejä erottelemaan eri sivun osiot toisistaan. Selainpuolella olisi myös hyvä katsoa voisiko siellä asetella joitakin elementtejä paremmin. Vaikka sovellus toimiikin erinomaisesti erikokoisilla näytöillä, niin koot on silti suunniteltu tällä hetkellä tietokoneen näytön kokoa mielessä pitäen. Älypuhelimella sovelluksen käyttö onnistuu, mutta käyttökokemus voi tuntua hieman kömpelöltä.

Palvelinpuolen ohjelmakoodissa on ainakin vähäsen paranneltavaa. Ohjelmakoodissa voisi uudelleen järjestää esimerkiksi vähentämällä samojen metodien toistamista. Tällä hetkellä esimerkiksi GetUserId-metodi, joka etsii käyttäjän yksilöivän tunnisteen, toistetaan usealla eri sivulla. Tämän metodin voisi viedä omaan luokkaansa ja sitten kutsua sitä kyseisestä luokasta käsin. Tällöin metodin tarvittaessa muutoksia ei ole tarvetta päivittää metodia kuin vain yhdestä paikasta ja näin ollen ylläpidettävyys paranee huomattavasti. Olisi myös hyvä pohtia ovatko kaikki tietokantakutsut tarpeellisia, vai voisiko jonkun laajemman kutsun tietoja hyväksikäyttää jossakin metodissa ilman erillistä kapeampaa tietokantakutsua.

8 Pohdinta opinnäytetyön onnistumisesta

Tässä luvussa pohdimme opinnäytetyön onnistumista. Käydään läpi tavoitteiden onnistumista ja sitä, mitä tämän opinnäytetyön tekemisestä opittiin. Opinnäytetyö toteutettiin suunnitellun aikataulun mukaisesti ja siinä saatiin luotua toimiva sovellus. Opinnäytetyön suunnittelu ja valmistelu aloitettiin alkuvuodesta 2023 ja se valmistui toukokuun 2023 alussa.

8.1 Opinnäytetyön tavoitteiden toteutuminen

Opinnäytetyössä oli tavoitteena luoda sovelluskehitysprojekti teoreettisen osion tietojen pohjalta. Määritykset, suunnitelma ja toteutus onnistuivat mielestäni hyvin tässä opinnäytetyössä. Teoreettinen pohja antoi hyvät raamit sovelluksen kehitykselle ja sovelluksesta saatiin toimiva tuotos. Erityisesti määrittelyt antoivat paljon apua lopullisen tuotteen toteuttamisessa. Tavoitteena oli myös vahvistaa omaa osaamista koko sovelluskehityksen elinkaaren osalta ja sen tavoitteen saavuttaminen onnistui erinomaisesti. En ollut aiemmin tehnyt näin kattavaa määrittelyä tai suunnittelua sovelluksesta ja koen, että tulevaisuudessa kykenen tekemään vieläkin parempaa määrittelyä ja suunnittelua.

8.2 Oma oppiminen ja tietopohjan vahvistuminen

Käytin opinnäytetyössä paljon jo tuntemaani ja osaamaani teknologiaa. Halusin kuitenkin syventyä näihin paremmin ja kehittää osaamistani kokonaisvaltaisemmin sovelluskehitysprojektin eri osa-alueilla. Aiemmasta osaamisestani suurin osa kosketti juuri sovelluksen ohjelmointia, mutta sain tämän opinnäytetyöprojektin kautta syvennettyä myös ohjelmointiin liittyvää osaamistani. Lisäksi sain paljon kokemusta ja tietoa määrittelystä ja suunnittelusta. Koen määrittelytyön olevan erittäin tärkeä osa sovelluskehitysprojektia. Ilman kattavaa määrittelyä sovelluksen kehittäminen voi olla hidasta ja vaatia useamman läpikäynnin samoille ominaisuuksille, jotta asiakas on tyytyväinen. Kun määrittelyt on tehty hyvin, voidaan toteuttaa juuri asiakkaan vaatima toive ja varmistaa näin tyytyväisyys. Tässä opinnäytetyössä olevat määrittelyt eivät kuitenkaan ole täydellisiä. Varsinkin sovelluksen kestävyyden ja virheensietokyvyn suhteen olisi ollut hyvä tehdä tarkennuksia määrittelyihin. Opinnäytetyössä kuitenkin oli pääasiallisena tarkoituksena luoda toimiva sovellus, joka olisi tasoltaan prototyyppi tai konsepti mahdollisesta tuotteesta, jota voidaan jatkokehittää paremmaksi tulevaisuudessa. Opin myös hyvin kritisoimaan omaa kehitystäni ja kirjaamaan ylös puutteita, sekä jatkokehitysehdotuksia.

8.3 Opinnäytetyön tuotoksen tulevaisuus

Opinnäytetyön tuotoksena toteutunut sovellus ei mielestäni ole läheskään valmis julkiseen käyttöön. Sovelluksessa on vielä paljon parannettavaa ja kehitettävää, ennen kuin sitä voisi antaa

avoimeen käyttöön. Sovelluksen ylläpidosta voisia kertyä kuitenkin tällöin jopa kohtuuttomia kustannuksia, joten tulisi harkita tarkkaan, miten tämä toteutettaisiin. On mahdollista, että jatkan jossakin vaiheessa sovelluksen kehittämistä vieläkin pidemmälle ja tuon sen yleiseen käyttöön. Mielestäni tässä toteutuneessa sovelluksessa on paljon potentiaalia hyödyllisenä sovelluksena itselleni ja mahdollisesti monelle muullekin. Sovelluksen laajentamisen yhteydessä pääsen mahdollisesti kehittämään osaamistani vielä syvemmälle. Olisi myös mielenkiintoista päästä tutkimaan sovelluksen mahdollisesti laajemman käytön vaikutuksia sovelluksen arkkitehtuurin ja tietokannan kestävyyden näkökulmasta. Tämän opinnäytetyön kirjoitushetkellä tämä tuotettu sovellus ei tule olemaan laajassa julkisessa käytössä.

Lähteet

Altexsoft, 23.7.2021. Functional and Nonfunctional Requirements: Specification and Types. Luettavissa: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/. Luettu: 15.4.2023.

Anderson, R. 1.12.2022. Introduction to Identity on ASP.NET Core. Microsoft. Luettavissa: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio. Luettu: 16.4.2023.

Anderson, R., Brock, D. & Larkin, K. 25.3.2023. Introduction to Razor Pages in ASP.NET Core. Microsoft. Luettavissa: https://learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-6.0&tabs=visual-studio. Luettu: 16.4.2023.

Atlassian s.a. What is Git? Luettavissa: https://www.atlassian.com/git/tutorials/what-is-git. Luettu: 18.4.2023.

Bootstrap, s.a. Build fast, responsive sites with Bootstrap. Luettavissa: https://getbootstrap.com/ Luettu: 27.4.2023.

Bos, B. 17.12.2016. A brief history of CSS until 2016. W3C. Luettavissa: https://www.w3.org/Style/CSS20/history.html. Luettu: 16.4.2023.

Brind, M. 18.2.2021. Welcome To Learn Razor Pages. Luettavissa: https://www.learnrazorpages.com/. Luettu: 16.4.2023.

Brooks, C. 21.2.2023. What is SQL? Business News Daily. Luettavissa: https://www.businessnewsdaily.com/5804-what-is-sql.html. Luettu: 17.4.2023.

Clark, D. 4.3.2021. SQL vs T-SQL: Understanding the Differences. Dataquest. Luettavissa: https://www.dataquest.io/blog/sql-vs-t-sql/. Luettu: 17.4.2023.

Diagrams.net, s.a. About diagrams.net. Luettavissa: https://www.diagrams.net/about. Luettu: 26.4.2023.

FitzMacken, T. 1.7.2022. Introduction to ASP.NET Web Programming Using the Razor Syntax (C#). Microsoft. Luettavissa: https://learn.microsoft.com/en-us/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-c. Luettu: 16.4.2023.

GitHub s.a.a. About repositories. Luettavissa: https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories. Luettu: 19.4.2023.

GitHub s.a.b. Understanding GitHub Actions. Luettavissa: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions. Luettu 19.4.2023.

Guthrie, S. 3.7.2010. Introducing “Razor” – a new view engine for ASP.NET. Microsoft. Luettavissa: https://weblogs.asp.net/scottgu/introducing-razor. Luettu: 16.4.2023.

Indeed, 11.3.2023. Technical Requirements (With Definition and List of Examples). Luettavissa: https://www.indeed.com/career-advice/finding-a-job/technical-requirements. Luettu: 15.4.2023.

Kinsta, 13.12.2022. What Is GitHub? A Beginner’s Introduction to GitHub. Luettavissa: https://kinsta.com/knowledgebase/what-is-github/. Luettu: 18.4.2023.

Lander, R. 8.11.2021. Announcing .NET 6 – The Fastest .NET Yet. Microsoft .NET Blog. Luettavissa: https://devblogs.microsoft.com/dotnet/announcing-net-6/. Luettu: 16.4.2023.

Lynch, P. & Horton, S. s.a. Web Style Guide 3rd Edition: Presenting Information Architecture. Web Style Guide. Luettavissa: https://webstyleguide.com/wsg3/3-information-architecture/4-presenting-information.html. Luettu 26.4.2023.

MDN, 24.2.2023. What is CSS? Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS. Luettu: 16.4.2023.

Michaels, P. 21.1.2023. The Service / Repository Pattern. The Long Walk. Luettavissa: https://pmichaels.net/service-repository-pattern/. Luettu: 27.4.2023.

Microsoft 2023a. What is .NET? Luettavissa: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet. Luettu: 16.4.2023.

Microsoft 2023b. What is ASP.NET Core? Luettavissa: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core. Luettu: 16.4.2023.

Microsoft 2023c. Visual Studio 2022 Preview. Luettavissa: https://visualstudio.microsoft.com/vs/preview/. Luettu: 4.4.2023.

Microsoft 14.12.2022. Get started guide for Azure developers. Luettavissa: https://learn.microsoft.com/en-us/azure/guides/developer/azure-developer-guide. Luettu: 18.4.2023.

Microsoft 13.2.2023. A tour of the C# language. Luettavissa: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. Luettu: 16.4.2023.

Microsoft 3.3.2023a. What is Azure SQL Database? Luettavissa: https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview?view=azuresql. Luettu: 18.4.2023.

Microsoft 3.3.2023b. CREATE TABLE (Transact-SQL). Luettavissa: https://learn.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql?view=sql-server-ver16. Luettu: 27.4.2023.

Microsoft 14.3.2023. App Service overview. Luettavissa: https://learn.microsoft.com/en-us/azure/app-service/overview. Luettu: 18.4.2023.

Microsoft 22.3.2023. What is Azure Boards. Luettavissa: https://learn.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops. Luettu: 17.4.2023.

Microsoft 31.3.2023. What is SQL Server Management Studio (SSMS)? Luettavissa: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16. Luettu: 4.4.2023.

Microsoft 11.10.2023. What is Azure DevOps? Luettavissa: https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops. Luettu: 5.4.2023.

Pérez, S. 18.10.2021. What is Microsoft SQL Server and what is it for? Intelequia. Luettavissa: https://intelequia.com/en/blog/post/what-is-microsoft-sql-server-and-what-is-it-for. Luettu: 17.4.2023.

Radigan, D. s.a. What is Kanban? Atlassian. Luettavissa: https://www.atlassian.com/agile/kanban. Luettu: 16.4.2023.

Rehkopf, M. s.a. User stories with examples and a template. Atlassian. Luettavissa: https://www.atlassian.com/agile/project-management/user-stories. Luettu 13.4.2023.

Roth, D. 8.11.2021. Announcing ASP.NET Core in .NET 6. Microsoft .NET Blog. Luettavissa: https://devblogs.microsoft.com/dotnet/announcing-asp-net-core-in-net-6/. Luettu: 16.4.2023.

W3C, s.a. WHAT IS CSS? Luettavissa: https://www.w3.org/Style/CSS/. Luettu: 16.4.2023.

WHATWG, 6.4.2023. HTML Living Standard. Luettavissa: https://html.spec.whatwg.org/multipage/introduction.html. Luettu: 16.4.2023.