Henri Tapio Helin

Asiakastukijärjestelmän rakentaminen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka / Ohjelmistotekniikka

Insinöörityö

03.12.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Henri Helin Asiakastukijärjestelmän rakentaminen 30 sivua 03.12.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Yliopettaja Erja Nikunen Osastopäällikkö Mikko Ristikangas

Tämän insinöörityön tavoitteena oli oppia, kuinka luoda verkkopohjainen asiakastukijärjestelmä, joka on yhteydessä toiminnanohjausjärjestelmä Jiraan. Asiakastukijärjestelmän tarkoituksena on virtaviivaistaa nykyistä tuen toimintamallia, jossa tukihenkilöiden tulee manuaalisesti kirjata tukipyynnöt Jiraan.

Insinöörityön tuloksena on web-sovellus, joka toimii asiakastukijärjestelmänä Siton asiakkaille. Asiakastukijärjestelmä kattaa tukipyyntölomakkeen, chat-palvelun, uutiskanavan, koulutusten tilauksen sekä usein kysytyt kysymykset -osion.

Sovellus koostuu kahdesta osasta, jotka ovat back-end ja front-end. Back-endinä toimii ASP.NET-sovellus, ja front-endinä Angular 4 -sovellus. ASP.NET-sovellus on yhteydessä Siton käyttämään asiakastukijärjestelmä Jiraan, johon asiakkaiden tekemät tukipyynnöt kirjoitetaan.

Kirjallinen osuus jakautuu front-endin ja back-endin tarkasteluihin. Front-end osuus keskittyy Angular 4 -sovelluksen tekoon. Back-end -osuus käsittää ASP.NET-osuuden, JIRA-integraation sekä autentikaation.

Avainsanat	Angular 4, TypeScript, Web API, Identity Framework, Jira

Author Title Number of Pages Date	Henri Helin Building a Customer Support System 30 pages 03.12.2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructors	Erja Nikunen, Principal Lecturer Mikko Ristikangas, Head of Department

The goal of this bachelor's thesis was to learn how to create a web- based customer support system that is connected to a issue tracking product called Jira. The use case for the application is to replace the current fully manual customer support system.

The outcome of this bachelor's thesis is a web application that will act as a customer support system for Sito's customers. The customer service system includes support request form, chat service, news channel, training form and frequently asked questions- page.

The application consists of two parts: the back-end and the front-end. Back-end uses the ASP.NET framework, and is connected to the issue tracking product Jira, where the support request data is sent. The front-end part uses Google's Angular framework.

In the paper, front-end part focuses on the development of the Angular 4 application, while the back-end part focuses on the ASP.NET, JIRA integration and ASP.NET Identity authentication.

Keywords	Angular 4, TypeScript, Web API, Identity Framework, Jira

Sisällys

Lyhenteet ja käsitteet

API	Application Programming Interface, eli ohjelmointirajapinta. Vastaanottaa http-pyyntöjä. Toimii välikappaleena, jonka kautta sovellukset voivat kommunikoida keskenään.
CSS	Cascading Style Sheets. Verkkodokumenteille kehitetty tyyliohje, joka määrittelee sivuston ulkonäön.
HTTP	HyperText Transfer Protocol. Asiakaspalvelinprotokolla, jolla voidaan siirtää dataa.
JSON	JavaScriptObjectNotation. Avain-arvotyyppinen tiedosto-muoto, joka on standardi tiedonsiirrossa.
JWT	JsonWebToken. JSON-tyyppinen käyttöoikeustietue, joka toimii varmenteena autentikaation yhteydessä.
LINQ	Language Integrated Query. .NET sovelluskehyksen komponentti, jolla voidaan tehdä tietokantahakuja
REST	Representational State Transfer. Arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen
SDK	Source Development Kit. Sovelluksen valmistajan tekemä kokoelma työkaluja, jolla sovelluskehittäjä pystyy luomaan sovelluksia valmistajan sovelluspaketille.
Yhden sivun web-sovellus	Yhden sivun web-sovelluksella tarkoitetaan web-sovellusta, jonka eri sivut ladataan saman sivun sisällä olevaan kehykseen sen sijaan, että sivu ladattaisiin kokonaan uudelleen.
SQL	Structured Query Language. Kyselykieli, jolla voi tehdä tietokantahakuja.

URL	Uniform Resource Locator. Osoite, jota käytetään osoittamaan websivua

1 Johdanto

Tässä insinöörityössä keskitytään Tukiportaali-nimisen websovelluksen tekoon käyttäen ASP.NET-sovellusta back-endinä, ja Angular 2 sovelluskehystä front-endinä. Ohjelmointiympäristönä käytetään Microsoft Visual Studio 2017: aa. Lukijalta odotetaan, että hän osaa käyttää Visual Studio 2017:ää ja että hän osaa C#:n, TypeScriptin sekä JavaScriptin perusteet.

Insinöörityö tehtiin Sito Oy:lle. Sito on suomalainen infrastruktuuriin erikoistunut yritys, joka tarjoaa asiakkaillensa suunnittelija- ja asiantuntijapalvelut sekä digitaaliset palvelut koko infrastruktuurin elinkaarelle. Siton suurin digitaalinen tuote, SpatialWeb, on karttapalvelu, joka on käytössä useissa Suomen kunnissa. Sitolla on lukuisia sovelluksia myytynä Suomen eri kunnille, valtionhallinnolle sekä yrityksille.

Tukiportaalin ideana on olla keskitetty paikka, josta Siton asiakkaat saavat apua ja tukea ongelmissa, sekä toimia informaatiokanavana asiakkaille. Tukiportaali tehtiin yrityksen sisäisesti, sillä se pienensi huomattavasti kustannuksia verrattuna siihen, jos Sito olisi hankkinut tukipalvelun joltain muulta yritykseltä. Esimerkiksi sovellus nimeltä Jira Service Desk olisi täyttänyt Siton asiakastuen vaatimukset, mutta sitä ei otettu käyttöön kustannuksellisista syistä. Kustannussyiden lisäksi ohjelmiston sisäinen kehitys mahdollistaa sen räätälöinnin juuri sellaiseksi, kun halutaan ilman kompromisseja.

Omana tavoitteenani oli kehittyä ohjelmoijana luoden alusta loppuun sekä ASP.NET- että Angular-sovelluksen, jotka kommunikoivat keskenään. Angular- ja ASP.NET-teknologiat valittiin Siton uusien linjausten takia, joiden mukaan kaikki uudet sovellukset tullaan rakentamaan kyseisillä teknologioilla. Näin sovellus on yhtenäinen Siton muiden sovellusten kanssa.

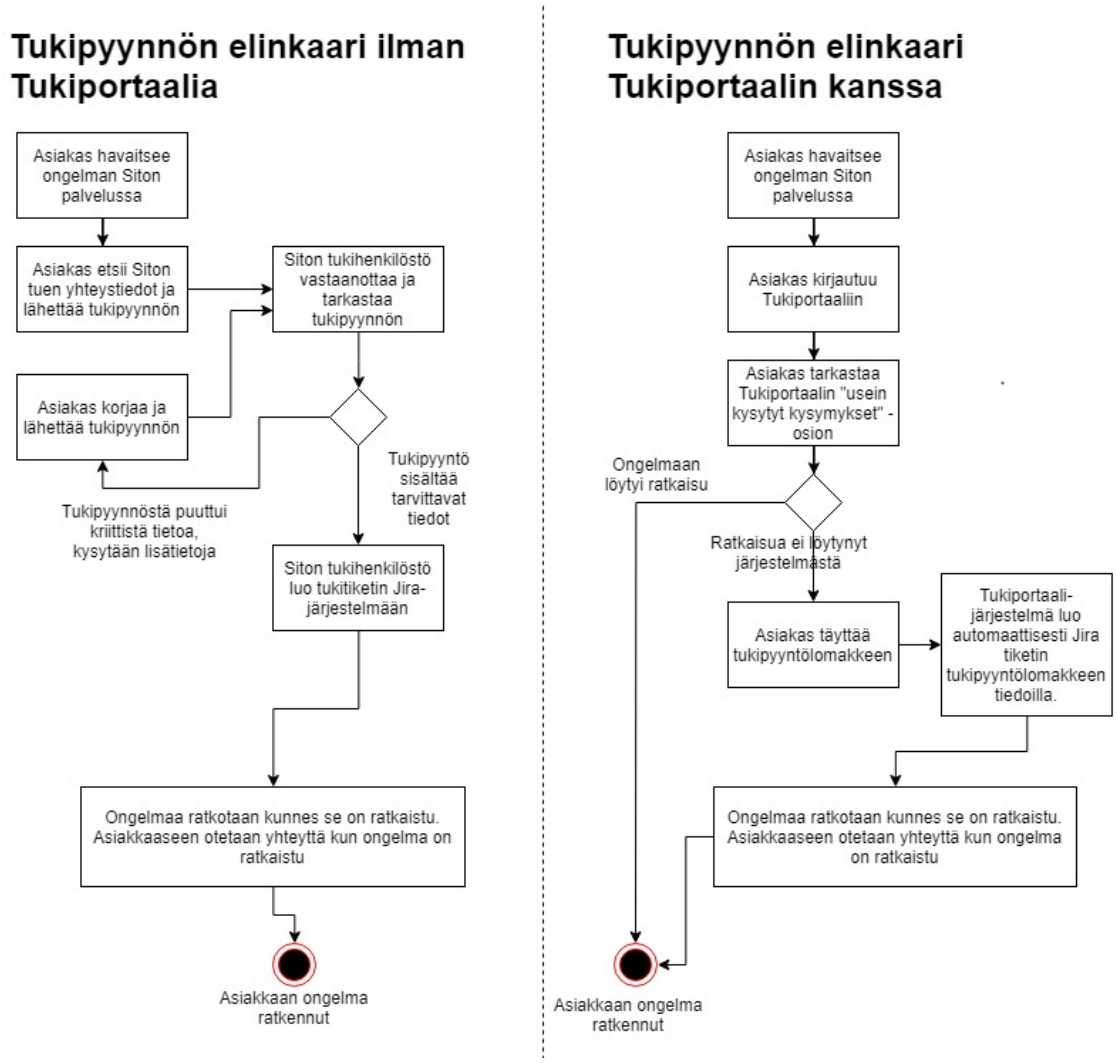
Siton puolelta halutuin ominaisuus Tukiportaalia varten oli mahdollisuus lähettää toiminnanohjausjärjestelmä Jiran tikettejä Jira-järjestelmään suoraan sovelluksen kautta.

2 Sovelluksen käyttötarkoitukset

Ennen Tukiportaalin suunnittelua Sito tarkasteli muita vaihtoehtoisia tukijärjestelmiä, joita olisi voitu käyttää Tukiportaalin tekemisen sijasta. Tukijärjestelmän tuli sisältää tapa, jolla asiakkaan tehdyt tukiviestit saataisiin suoraan Jiraan. Näin säästetään tukihenkilöstön käyttämää aikaa nykyiseen järjestelmään verrattuna.

Siton asiakastukijärjestelmä ennen insinöörityössä tehtyä Tukiportaalia oli seuraavanlainen: Asiakas löytää Siton asiakaspalvelun sähköpostiosoitteen, ja lähettää sähköpostiviestin ongelmastaan asiakastuelle. Keskustelua käydään asiakkaan kanssa niin kauan, kunnes Siton tukitiimillä on tarpeeksi informaatiota kirjoittaa tukitiketti Siton käyttämään toiminnanohjausjärjestelmä Jiraan. Tukitiimi tekee tukitiketin pohjalta tarvittavat korjaukset ja parannukset, jonka jälkeen varmistetaan asiakkaalta, että asiat ovat kunnossa, ja tiketti suljetaan tehdyksi.

Tukiportaalin myötä asiakastukiprosessi tehostuu (kuva 1). Sen sijaan, että asiakas lähettäisi tukipyynnön sähköpostitse, hän käyttää Tukiportaali-sovelluksen tukipyyntölomaketta. Lomake on rakennettu niin, että se sisältäisi kaikki tarvittavat tiedot ongelman selvittämiseksi, kuten selaintiedot ja käytetyn sovelluksen. Tiketti tallennetaan suoraan Jiraan sovelluksen toimesta, mikä säästää tukihenkilökunnan aikaa niiden tekemiseltä.



Kuva 1. Aktiviteettikaavio tukipyynnön etenemisestä ennen ja jälkeen Tukiportaalia.

Tukipyyntöjen lisäksi sivustolta löytyy myös lomake koulutuksen tilaukseen. Tukiportaali toimii myös Siton tiedotuskanavana ja tietolähteenä asiakkaille. Se sisältää sovellusten käyttöohjeita ja usein kysyttyjä kysymyksiä. Tukiportaalia käytetään siis tehostamaan asiakastukiprosessia Sitossa.

3 Tavoitteet

3.1 Henkilökohtaiset tavoitteet

Insinöörityön tavoitteena oli suunnitella ja toteuttaa Sitolle selaimella käytettävä asiakastukijärjestelmä, Tukiportaali, jonka kautta Siton asiakkaat pystyvät hakemaan apua ja tukea nopeasti ja tehokkaasti. Tukiportaali auttaa asiakkaita tarjoamalla chat-palvelun, usein kysyttyjä kysymyksiä, käyttöohjeita, tukipyyntölomakkeen sekä koulutuksen tilauslomakkeen.

Henkilökohtaisena tavoitteenani oli kehittää osaamistani ja luoda verkkosovellus käyttöliittymää, rajapintaa ja tietokantaa myöten. Haastavimpana tavoitteena asetin itselleni autentikaation luomisen, sillä en ollut aiemmin tekemisissä ASP.NET-autentikaation kanssa.

3.2 Tukiportaalijärjestelmän tavoitteet

Tukipyyntölomake tulisi olla sellainen, että siihen täytetyt tiedot menisivät suoraan Siton toiminnanohjausjärjestelmä Jiraan niin kutsutuksi ”taskiksi”. Lomakkeesta asiakas voi valita sen prioriteetin, sitä koskevan sovelluksen sekä muita hyödyllisiä tietoja, jotka auttavat tuen antamisessa.

Sivuston chat-palvelu hankittiin ulkoisena palveluna. Se on helppokäyttöinen Chat- palvelu, joka sisältää sekä chat-ikkunan asiakkaalle, että ylläpitosovelluksen tukihenkilöstölle. Asiakas voi käyttää chat-ominaisuutta kun hän haluaa vastauksia nopeasti.

Sivulla tulisi olla autentikaatiojärjestelmä. jonka avulla vain Siton asiakkaat pääsisivät sivustolle. Autentikaatio tulisi toteuttaa niin, että se olisi mahdollisimman käyttäjäystävällinen ja helppo huoltaa ja laajentaa tarvittaessa.

Tukiportaalissa tulee olla usein kysytyt kysymykset -osio, joka on tarkoitus kattaa jo kysyttyjä kysymyksiä ja vastauksia. Täältä asiakas saattaa löytää kysymykseensä vastauksen ilman, että hänen tarvitsee kysyä sitä tukihenkilöstöltä. Tukihenkilöt ylläpitävät usein kysyttyjä kysymyksiä, ja listaavat niitä sitä mukaa, kun niitä ilmaantuu. Asiakas pystyy

rajaamaan kysymyksiä sovelluksen mukaan sekä järjestämään kysymykset päivämäärän, nimen tai sovelluksen perusteella.

Tukiportaalin tulee myös sisältää lomakkeet avun pyyntöön, kun chat-henkilöstö ei ole paikalla, sekä koulutuksen tilauslomakkeen, josta asiakas pystyy tilaamaan itselleen tai yritykselleen koulutuksen liittyen joihinkin Siton palveluista

4 Käytetyt teknologiat

Tässä luvussa selitetään lyhyesti sovelluksessa käytetyt teknologiat asiakassovelluksen osalta. Luvussa 5 kerrotaan näiden teknologioiden implementaatio sovelluksen käyttöliittymässä. Luvussa 6 käydään sovelluksen rajapinnan teknologiat ja toteutukset.

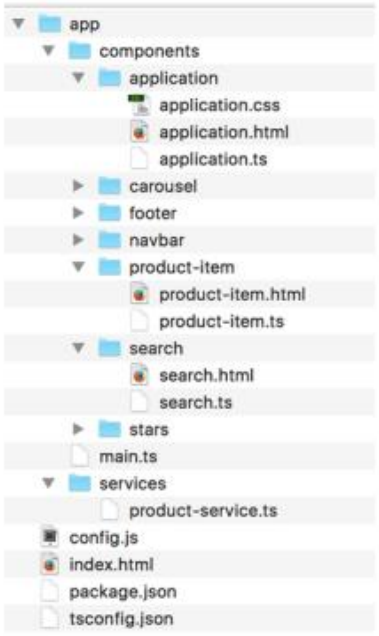
Sovelluksessa käytettävät teknologiat ovat pääasiassa Siton nykyisten käytäntöjen mukaisten määräysten mukaiset. Kaikissa Siton uusissa verkkosovelluksissa tullaan käyttämään Angularia ja Bootstrapia käyttöliittymän toteutuksessa, ja ASP.NET:iä sovellusrajapinnan toteutuksessa. Käytännöistä huolimatta olisin päätynyt näihin teknologioihin, sillä minulla on sekä Angularista että ASP.NET:stä riittävän paljon aiempaa kokemusta todetakseni ne toimiviksi ja luotettaviksi teknologioiksi.

4.1 Angular-sovelluskehys

Angular on Googlen kehittämä JavaScript-sovelluskehys, jota yleisimmin käytetään responsiivisien yhden sivun web-sovellusten (single-page application) tekemiseen. Angular käyttää ohjelmointikielenä JavaScriptin supersettiä, Typescriptiä.

Angular-sovelluskehystä käytetään asiakaspuoleisten (client-side) web-applikaatioiden tekemiseen HTML-kielellä ja joko JavaScriptiä tai TypeScriptiä käyttäen. Typescript käännetään sovelluksen käännösvaiheessa JavaScriptiksi. Kuten useat muutkin JavaScript-sovelluskehykset, myös Angular-sovelluskehys koostuu useista eri kirjastoista. [1.]

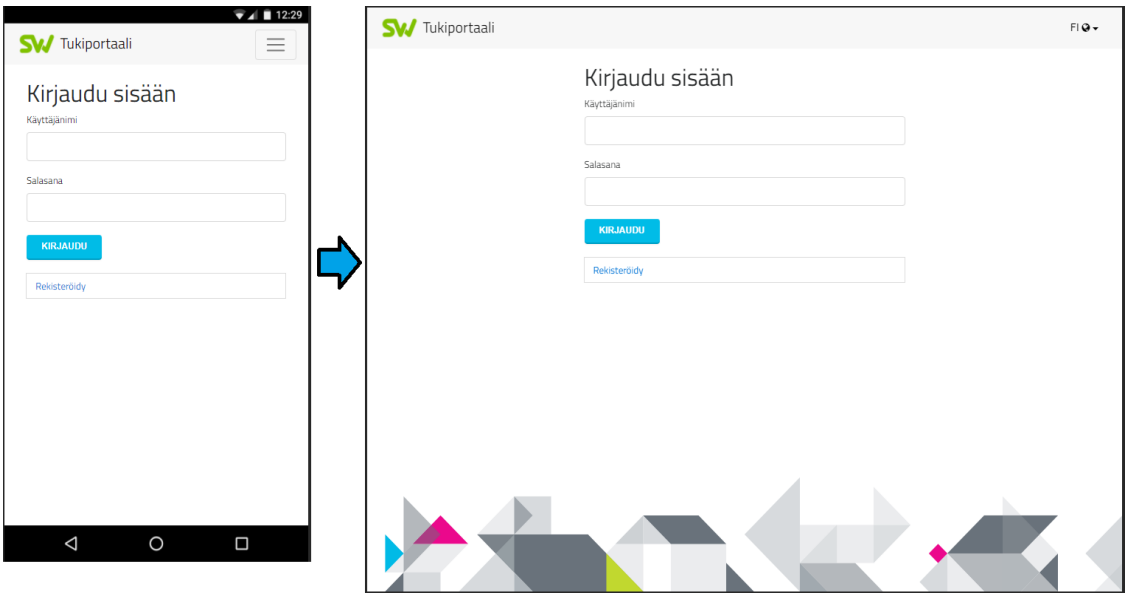
Kuten kuvasta 2 nähdään, Angular-sovellus muodostuu templateista (templates), komponenteista (components) ja palveluista (services). Yksi komponentti kattaa HTML-templaten (.html), Typescript-kontrollerin (.ts) sekä mahdollisen CSS-tyylitiedoston (.css). Palvelut ovat riippuvuusinjektion avulla injektoituja luokkia, joilla tyypillisesti haetaan dataa ulkoisista palveluista asynkronisesti. [2.]



Kuva 2. Angular-sovelluksen rakenne [2.]

4.2 Responsiivinen käyttöliittymä

Bootstrap on suosittu HTML-, CSS- ja JavaScript-sovelluskehys, jota käyttämällä pystytään luoda responsiivisia web-applikaatioita, jotka on ensisijaisesti suunniteltu mobiililaitteille (mobile-first- periaate) [3.]. Sovelluksessa on käytetty Bootstrap sovelluskehyksen Angular toteutusta, Ng-bootstrapia, yhdessä Bootstrap 4 tyylitie doston kanssa. Kuvasta (kuva 3) näkyy selvästi, kuinka Bootstrapillä tehty sovellus skaalautuu laitteen mukaan.



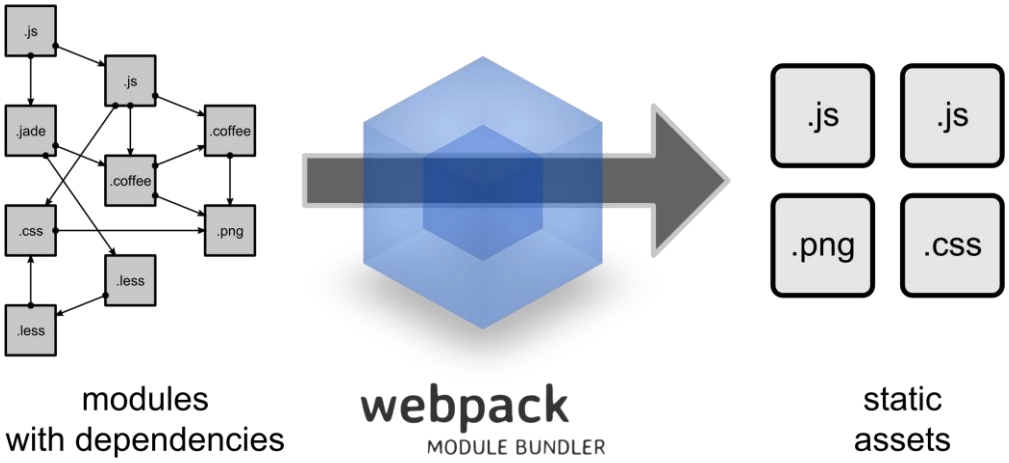
Kuva 3. Bootstrapin responsiivisuus Nexus 5X (vas) puhelimella sekä iPad (oik) -tabletilla

4.3 Moduulipaketoija

Kun sovellus viedään palvelimelle, halutaan, että se on kevyt ja optimoitu. Webpack on moduulipaketoija (module bundler), jota käytetään Angular 2-sovelluksen pakkaamiseen. Pakatessaan Webpack pakkaa sovelluksen tiedostokooltaan pienemmäksi sekä optimoi sovelluksen siten, että se on kevyempi ja lähettää vähemmän http-palvelupyyntöjä palvelimelle [1.].

Webpackin avulla Angular-sovelluksen monimutkainen modulaarinen rakenne paketoidaan pienempään kokoon käyttämällä erilaisia tekniikoita. Koodi niin sanotusti ”rumennetaan” (uglify), jolloin siitä tulee vaikeammin luettavaa ihmiselle, mutta rakenteeltaan se on selkeämpi ja suoraviivaisempi tietokoneelle.

Webpackia havainnollistava kuva 4 kuvastaa, kuinka Angular-sovelluksen monimutkainen modulaarinen rakenne muunnetaan Webpackin avulla staattisiksi tiedostoiksi.



Kuva 4. Havainnollistava kuva Webpackin toiminnasta [1]

5 Asiakasohjelmisto

Asiakasohjelmistolla tarkoitetaan sitä osaa sovelluksesta, jonka käyttöliittymään käyttäjällä on suora pääsy omalta päätelaitteelta. Asiakassovellus hakee dataa sovellusrajapinnasta ja pystyy näyttämään sen käyttäjälle käyttöliittymässään. Tässä luvussa esitellään asiakassovelluksen käyttöliittymä ja selitettään, kuinka se on toteutettu.

5.1 Määrittely

Tukiportaali on Siton asiakkaiden käyttöön tarkoitettu verkkosivu, jonka kautta käyttäjät voivat antaa palautetta, tehdä kehitysehdotuksia sekä saada apua liittyen Siton sovelluksiin ja palveluihin. Tukiportaali haluttiin tehdä mahdollisimman käyttäjäystävälliseksi ja nopeaksi käyttää. Sen tulee toimia uusimmilla selaimilla, suosituimmilla käyttöjärjestelmillä sekä tietokoneella, tabletilla ja älypuhelimella.

Sivu on suljettu palvelu, johon pääsee vain kirjautumalla sisään. Sovelluksessa tulee olemaan chat-ominaisuus, jonka avulla käyttäjä voi pyytää apua nopeasti Siton käyttäjätuesta. Jos käyttäjän apupyynnöllä ei ole kiire, tai chat-palvelussa ei ole työntekijää päivystämässä, käyttäjä voi tehdä apupyynnön myös lomakkeella.

Käyttäjällä on mahdollisuus tehdä tukipyyntöjä, joista luodaan automaattisesti tiketit Siton käyttämään Atlassian Jira-järjestelmään. Omia tikettejä voi seurata sivulla olevasta ”Omat tiketit”-osiosta, josta näkee tikettien tilan, kuvauksen sekä kommentit.

Käyttäjällä on myös pääsy sovellusten ja palveluiden käyttöohjeisiin Tukiportaalin kautta. Käyttöohjeet ovat linkkejä ulkoisiin sivuihin, eikä niiden toteutuksiin oteta tässä insinöörityössä kantaa. Lisäksi sivulta löytyy asiakaspalvelun yhteystiedot, kuten puhelinnumero, lähiosoite ja sähköpostiosoite.

5.2 Angular 2:n toteutus

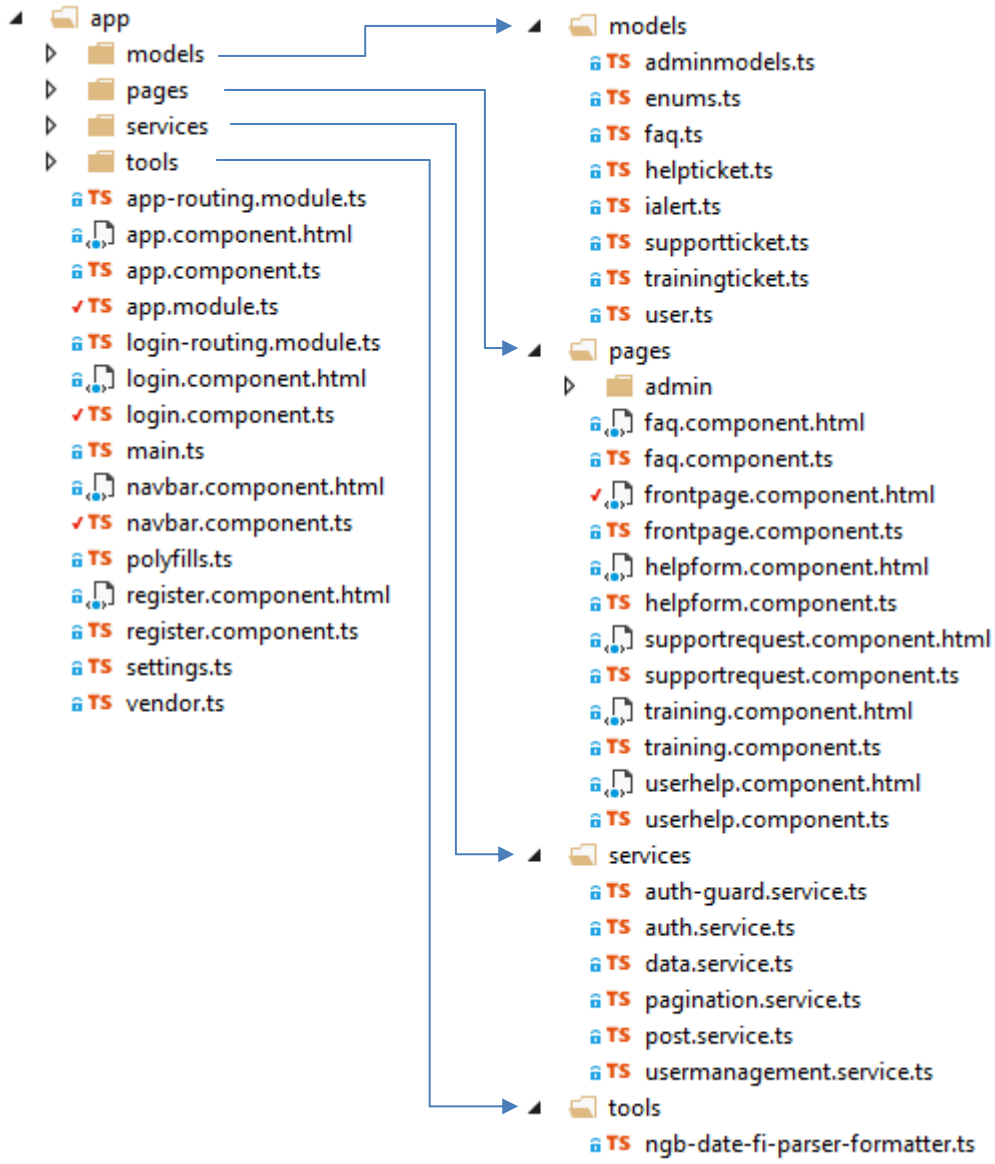
Tukiportaali on Angular-sovellus, jolla on sille tyypillinen komponenttipohjainen rakenne. Sovelluksen ”app” kansio sisältää Tukiportaalin Angular-osuuden. Sovelluksen eri osa-alueet on sijoitettu niitä vastaaviin kansioihin models, pages, services ja tools.

Kuvasta 5 näkyy tukiportaalin Angular-toteutuksen kansiorakenne. Models-kansio sisältää datan käsittelyä helpottavia model-luokkia, joihin dataa säilötään ajon aikana. Pages sisältää sivuston eri sivut, lukuun ottamatta kirjautumis- ja rekisteröitymissivuja. Yksi sivu kattaa sen Typescript-osuuden (.ts) ja HTML-osuuden (.html). Pages-kansion alakansio admin sisältää ylläpidon kannalta tärkeät sivut.

Services-kansio sisältää Angularissa sovelluksen toiminnallisuuden sisältävät palveluluokat eli servicet. Niistä auth-guard ja auth-palvelut hoitavat autentikaatioon liittyvät asiat, kuten sisään- ja uloskirjautumisen, rekisteröitymisen ja automaattisen kirjautumisen. Data- ja post-palvelu hoitavat vastaavasti datan tuomisen ja lähettämisen sovellusrajapinnasta. Pagination-palvelu hoitaa sivutuksen. Usermanagement on vain ylläpidolle saatavilla oleva palvelu, jolla voi hallita rekisteröityneitä käyttäjätunnuksia.

Tools-kansio sisältää parserin ng-bootstrapin date picker -komponenttiin. Se muovaa ng-bootstrapin datepickerin ajan suomalaiseen dd-mm-yyyy-muotoon.

App-kansion juuressa sijaitsevat rekisteröinti- ja kirjautumissivut, navigointipalkki sekä Angularin toiminnan kannalta tärkeät komponentit, kuten reititykset, moduulit, main-, vendor- ja polyfills- tiedostot.



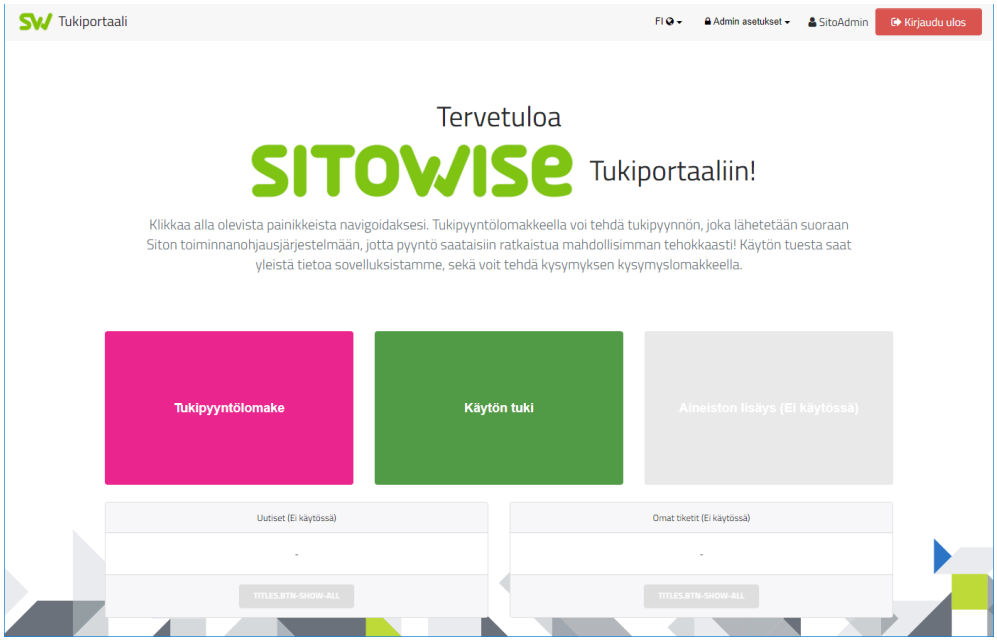
Kuva 5. Tukiportaali sovelluksen Angular osuuden kansiorakenne

Angular-toteutuksen kaikkia koodeja ei käydä läpi. Seuraavissa alaluvuissa tarkastellaan muutamaa tiedostoa, joista selviää Angular-sovelluksen yleinen rakenne, jolla tavoin muutkin sivut on rakennettu. Osa sivuista esitellään vain pintapuolisesti.

5.2.1 Etusivun esittely

Etusivu on yksinkertainen Bootstrap-sivu ilman normaalista poikkeavaa toiminnallisuutta (kuva 6). Sivulle kirjautuessaan käyttäjälle avautuu näkymä, jossa on kolme isoa nappia,

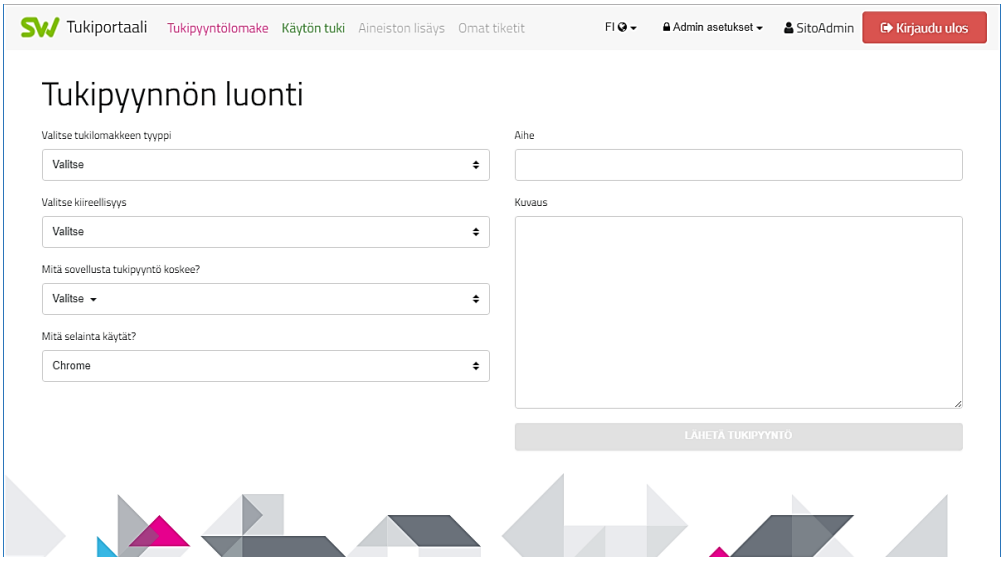
joilla voi navigoida tukipyyntölomakkeeseen, käytön tukeen tai aineiston lisäykseen. Lisäksi etusivulta näkyy omat avoimet tukitiketit sekä Siton julkaisemat uutiset.



Kuva 6. Ruutukaappaus tukiportaalin etusivusta

5.2.2 Tukipyyntölomake

Tukipyyntölomakkeelle pääsee painamalla etusivun vastaavaa painiketta. Lomake koostuu viidestä pudotusvalikosta ja kahdesta tekstikentästä (kuva 7). Jotta lomake voidaan lähettää, kenttien täytyy olla oikein täytettyjä. Tästä huolehtii Angularin validointi, joka varoittaa käyttäjää, jos jokin kentistä ei ole täytetty oikein.



Kuva 7. Ruutukaappaus tukipyyntölomakkeesta. Tyyppi ja kiireellisyys kentät vastaavat Jiran Type ja Priority-kenttiä.

Tukipyyntölomakkeen koodista 2 näkyy tyypillinen Angular-komponentin rakenne; Komponentin selector asetetaan halutuksi ja templateUrl:n sijainti asetetaan vastaamaan sivun HTML-toteutusta. Komponentti esitellään ”export class”-määreellä, jonka jälkeen määritellään muuttujat, mukaan lukien SupportTicket-malliluokkaa vastaava ticket.

Ticket olion (koodi 1) jokainen muuttuja on käytössä HTML-lomakkeessa. HTML-lomakkeen kentät ja ticket-olion muuttujat sidotaan toisiinsa Angularin ns. kaksisuuntaisen sidonnan (two-way binding) avulla, jolloin jos muuttujan arvoa muutetaan, se päivittyy sekä HTML-näkymään että JavaScript muuttujaan [4.]. HTML-kenttään lisätään [(ngModel)]-muuttujan arvoksi sitä vastaava ticket-olion arvo. Esimerkin (koodi 1) tapauksessa HTML-lomakkeen kuvaus kenttään sidotaan ticket-olion description-muuttuja.

```
export class SupportTicket {
    constructor(
        public type: number,
        public priority: number,
        public application: number,
        public browser: string,
        public subject: string,
        public description: string
    ) { }
```

```
<div class="form-group">
    <label for="text-description">{{'supportform.description' | translate:param}}</label>
    <textarea class="form-control" id="text-description" style="height:230px"
              [(ngModel)]="ticket.description" name="text-description"></textarea>
</div>
```

Koodi 1. Tukitiketin "two way binding": vasemmalla tukitiketin malliluokka, oikealla sitä käyttävä textarea komponentti

Tarkastellaan koodi 2:ta. Constructor-funktio suoritetaan, kun tukipyyntölomake avataan. Siinä alustetaan service-luokat, joita tullaan käyttämään tukipyyntölomakkeen käsittelyssä ja lähettämisessä. ngOnInit suoritetaan constructor-funktion jälkeen. Siinä alustetaan selainvalinta vastaamaan käytössä olevaa selainta, sekä haetaan lista sovelluksista DataServicen funktiolla getApplicationList.

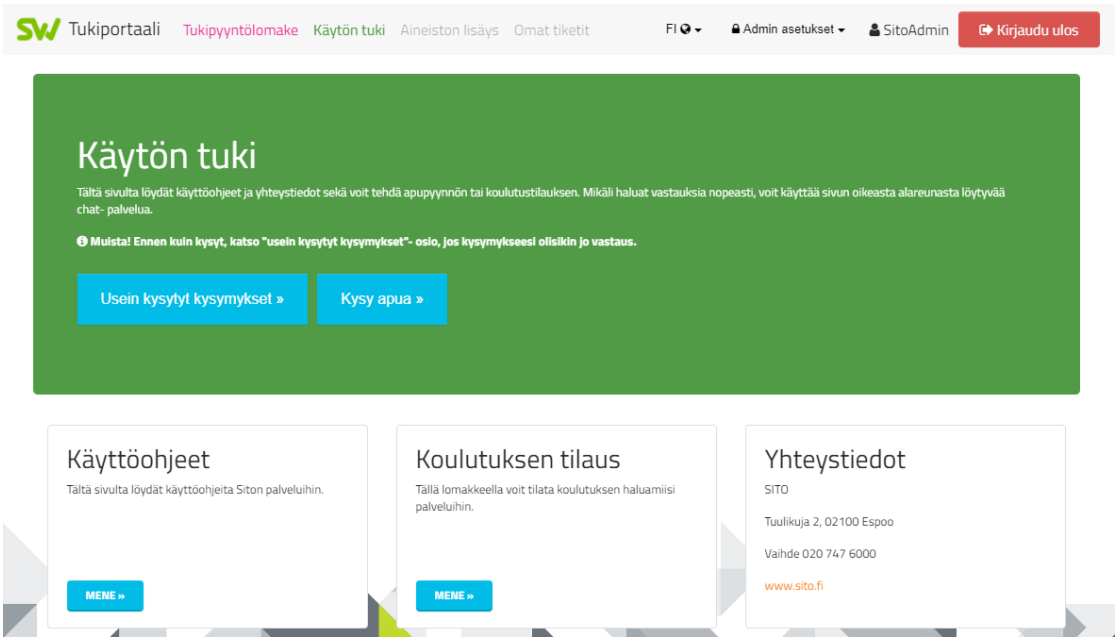
OnSubmit-funktiota kutsutaan, kun käyttäjä painaa ”lähetä tukipyyntö” -näppäintä. Aluksi submitInProgress boolean arvo muutetaan todeksi, jolloin käyttöliittymä reagoi näyttämällä latausanimaation. Tämän jälkeen täytetystä ticket-oliosta luodaan kopio JavaScriptin Object.assign-komennolla, jotta sitä voidaan käsitellä niin, että lomakkeen tiedot eivät muuttuisi. Lopuksi lomakkeen tiedot lähetetään PostServicen postSupportTicket-funktiolla, joka onnistuessaan lähettää tukipyynnön Jiraan. Onnistuessaan kutsutaan AfterSubmitSuccess-funktiota, joka näyttää käyttäjälle viestin onnistuneesta lähetyksestä modaalissa ja asettaa submitInProgress-muuttujan arvon epätodeksi. Epäonnistuessaan tehdään samat toimenpiteet funktiolla afterSubmitFail, mutta modaalin viesti viittaa epäonnistuneeseen lähetykseen.

```
@Component({
    selector: 'frontpage',
    templateUrl: './supportrequest.component.html'
})
export class SupportrequestComponent {

    ...

    private ticket: SupportTicket = new SupportTicket(0, 0, null, "", "", "");

    ...

    constructor(private postService: PostService, private modalService: NgbModal, private
    router: Router, private dataService: DataService) { }

    ...

    ngOnInit() {
        this.setUserBrowser();
        this.dataService.getApplicationList().then(list => this.applications = list);
    }

    ...

    private onSubmit() {
        this.submitInProgress = true;
        let tempTicket: SupportTicket = new SupportTicket(0, 0, null, "", "", "");
        Object.assign(tempTicket, this.ticket);

        ...

        this.postService.postSupportTicket(tempTicket).then(() =>
        this.AfterSubmitSuccess()).catch((e) => this.AfterSubmitFail(e));
    }

    ...
}
```

Koodi 2. Ote tukipyyntölomakkeen koodista

5.2.3 Käytön tuki

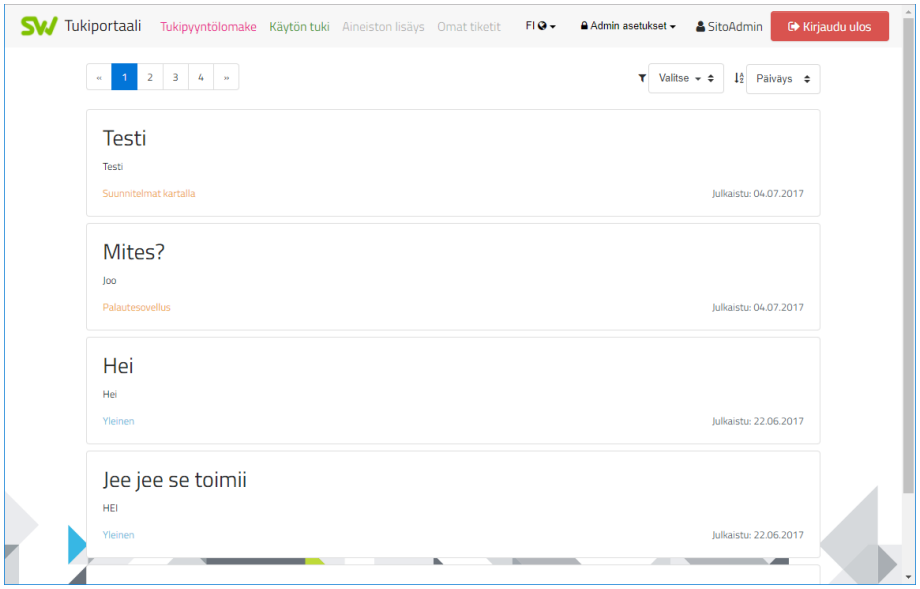
Etusivun tapaan käytön tuki toimii ohjaavana portaalina muille sivuille. Tämän sivun kautta käyttäjä pystyy navigoimaan usein kysyttyihin kysymyksiin, apulomakkeeseen, käyttöohjeisiin sekä koulutuksen tilaukseen (kuva 8). Sen lisäksi tältä sivulta löytää Siton yhteystiedot.



Kuva 8. Ruutukaappaus käytön tukisivusta

5.2.4 Usein kysytyt kysymykset

Usein kysytyt kysymykset-sivulla käyttäjä pystyy selaamaan sivun ylläpitäjien päivittämiä kysymyksiä ja niiden vastauksia. Kysymykset voidaan järjestää päivämäärän, nimen tai kategorian perusteella, jonka lisäksi ne voidaan myös suodattaa kategorian perusteella (kuva 9).



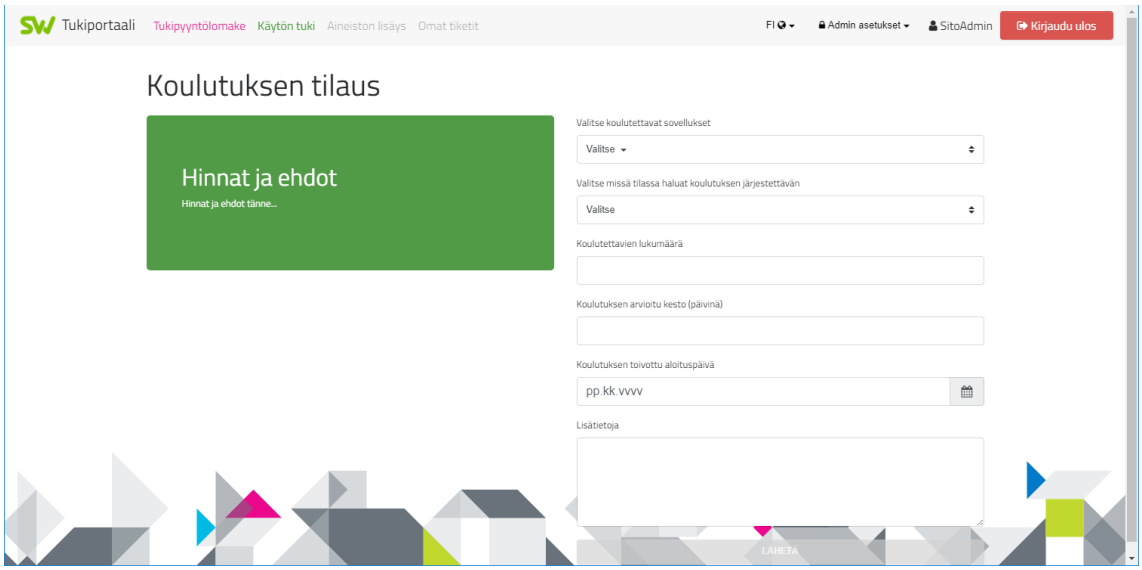
Kuva 9. Ruutukaappaus usein kysytyt kysymykset sivusta

Usein kysytyt kysymykset sijaitsevat SQL server -tietokannassa, ja ne saadaan API-kyselynä backendiltä. Usein kysytyille kysymyksille on lisäksi olemassa hallintapaneeli, jolla pystyy luoda, muokata ja poistaa kysymyksiä. Kysymyksiä voi hallita vain käyttäjät, joilla on admin-oikeudet.

Usein kysytyt kysymykset jaetaan usealle sivulle, eli sivutetaan [5.]. Sivutuksesta huolehtii PagerService-niminen palveluluokka, joka ottaa listan usein kysytyistä kysymyksistä, ja jakaa sen viiden olion mittaisiin taulukoihin.

5.2.5 Kysy apua ja koulutuksen tilauslomakkeet

Näillä lomakkeilla käyttäjä pystyy kysymään apua tuelta tai lähettämään koulutuksen tilauksen (kuva 10).



Kuva 10. Koulutuksen tilauslomake

5.2.6 Autentikaatio Angular-sovelluksessa

Luku 5.2.6 poistettu luottamuksellisena toimeksiantajan pyynnöstä.

5.2.7 Sovellusrajapinnan datan käsittely

Service eli ”palvelu”-luokkia käytetään tukiportaalin tapauksessa pääosin datan tuomiseen ja lähettämiseen sovellusrajapinnasta. Sovellusrajapinnan käsittelyn lisäksi tukiportaali sisältää servicejä, jotka huolehtivat autentikaatiosta.

Koodi 7 esittelee tyypillisen tavan, jolla dataa haetaan sovellusrajapinnasta. GetApplicationList-funktiossa tehdään yksinkertainen http:n get-metodi, ja tuloksesta riippuen palautettu JSON-arvo muunnetaan olioksi tai annetaan virheilmoitus. Aluksi asetetaan ”headers”:n arvot mieleisiksi, tässä tapauksessa ”Content-Type”:ltä odotetaan JSON-tyyppistä arvoa, sekä authorisaatiota varten odotetaan validia JWT. Lopuksi funktio palauttaa lupauksen (Promise), jonka onnistuttua pyritään muuntamaan saatu JSON, tässä tapauksessa ”IMultiSelectOption”-tyyppiseksi, olioksi. Angularissa lupaus on asynkroninen toimenpide, joka ”lupaa” palauttavansa jotain. Lupauksen onnistumisen jälkeen tehdään then-määreeseen asetettu toimenpide. virheen sattuessa tehdään catch-määreessä määritellyt toimenpiteet.

```
Koodi poistettu luottamuksellisena toimeksiantajan pyynnöstä
```

Koodi 3. Ote data.service.ts-koodista.

Koodi 8 esittelee tavan, jolla sovellusrajapintaan voidaan lähettää data post-metodilla. Funktio on muuten täysin sama, mutta get:n sijaan käytetään post-metodia tukitiketin lähetykseen. Tukitiketti tulee kuitenkin ensin muuntaa JSON-muotoiseksi merkkijonoksi JSON.stringify-komennolla.

```
Koodi poistettu luottamuksellisena toimeksiantajan pyynnöstä
```

Koodi 4. Ote post.service.ts koodista.

6 Sovellusrajapinta (API)

Sovellusrajapinnan avulla asiakasohjelmisto pystyy pyytämään ja lähettämään dataa tietokannasta. Se toimii ikään kuin viestinvälittäjänä käyttöliittymän ja tietokannan välillä saamalla jotain dataa http-pyyntönä ja lähettämällä jotain dataa käsittelyn jälkeen takaisin http-vastauksena.

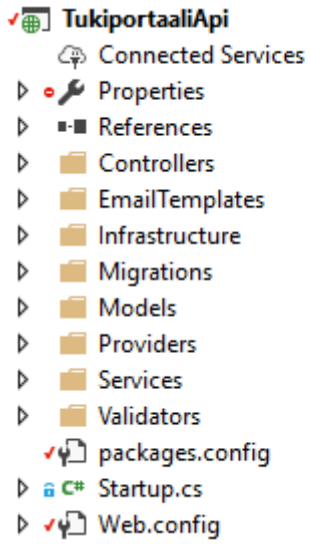
Tukiportaalin sovellusrajapinta on REST-rajapinta, joka tarkoittaa sitä, että sovellus käyttää GET-, PUT-, POST- ja DELETE http-kutsuja datan lähetykseen ja saamiseen. GET-kutsulla pyydetään, PUT- ja POST -kutsuilla lähetetään ja DELETE-kutsulla poistetaan dataa. Esimerkiksi Tukiportaali käyttää POST-kutsua lähettääkseen tukipyynnön sovellusrajapinnan kautta Jiraan.

6.1 ASP.NET Web API

Web API eli verkkopohjainen ohjelmointirajapinta (Application Programming Interface) on sovellus, joka käsittelee esim. HTTP Post -metodilla lähetettyä dataa. Ohjelmointirajapinnoilla saadaan eri sovellukset kommunikoimaan keskenään turvallisesti. Tukiportaalin ASP.NET Web API -rajapinta käsittää autentikaation, tietokannan datan tuomisen ja lähetyksen, Jiraan lähetettävän datan käsittelyn sekä sähköpostiviestien lähetyksen.

6.2 Tukiportaalin Web API:n rakenne

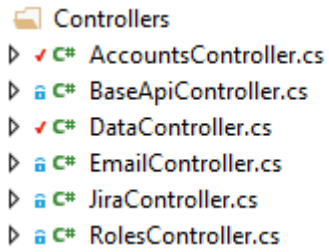
Tukiportaalin sovellusrajapinta koostuu kahdeksasta kansiosta (kuva 11). Tukiportaalin datan lähetyksen kannalta tärkeimmät ovat Controllers ja Models. Loput kuusi kansiota liittyvät autentikaation toteutukseen.



Kuva 11. Ruutukaappaus Web API:n kansiorakenteesta

6.3 Web API-kontrollerit

Kontrolleri on ASP.NET-luokka, jolla pystytään reitittämään html-metodit ASP.NET-funktioihin. Tukiportaalin Web API:ssa on kontrollerit tunnusten, tietokantojen datan, sähköpostien, Jiran datan ja käyttäjäroolien hallintaan (kuva 12).



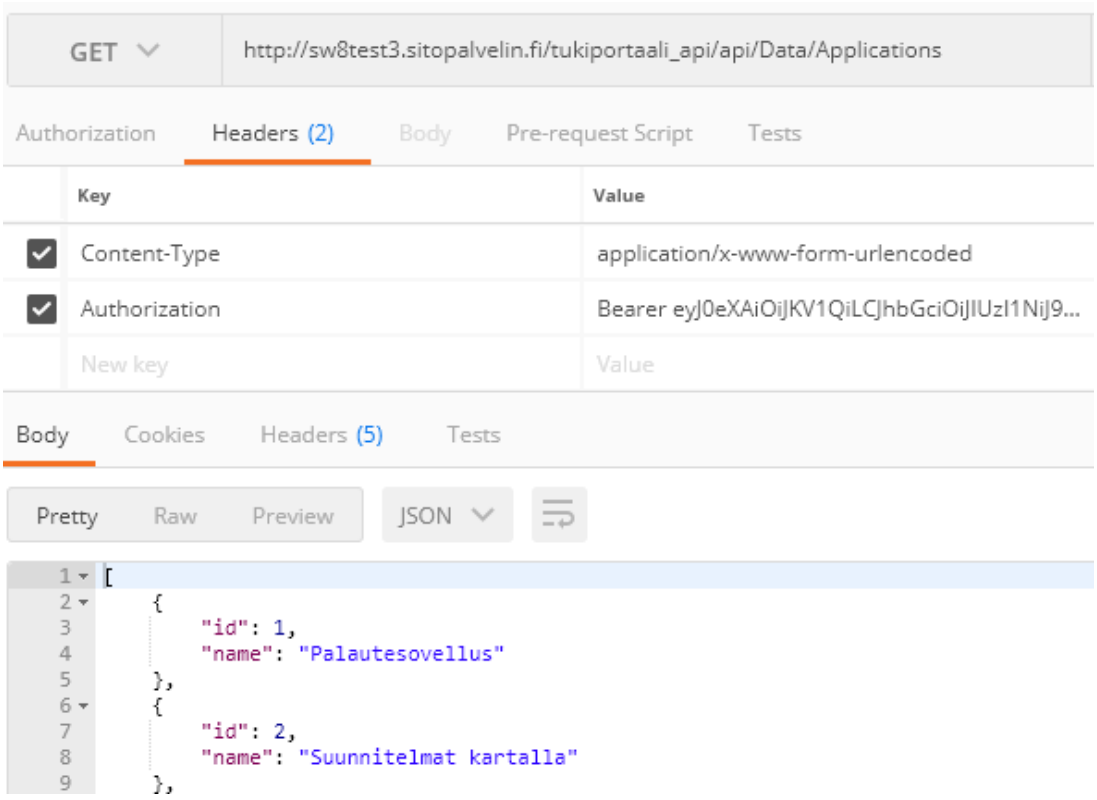
Kuva 12. Ruutukaappaus kontrollereista

Esimerkkinä tarkastellaan DataController-luokan koodia (Koodi 9), jossa haetaan lista sovelluksista ja palautetaan se. Sovelluslista sijaitsee SQL Server-tietokannassa. Datakontrollerin url-loppuosa on api/Data, joka määritetään luokan määrittelyn yhteydessä ”RoutePrefix”-määreellä. Kaikki kontrollerit implementoivat ApiController-rajapintaa, mutta Tukiportaalin tapauksessa kontrollerit implementoivat BaseApiControlleria, joka puolestaan implementoi ApiControlleria.

Tarkastellaan GetApplications-funktiota. Funktiota ennen on määritelty, että sen kutsuminen vaatii käyttöoikeuksia (Authorize), että sitä kutsutaan http GET-metodilla (HttpGet), ja että sen sitä kutsutaan reitistä api/Data/Applications (Route). Saatu sovelluslista palautetaan JSON-muotoisena takaisin (Kuva 13).

```
[RoutePrefix("api/Data")]
public class DataController : BaseApiController
{
    // GET: api/Data/Applications
    [Authorize]
    [HttpGet]
    [Route("Applications")]
    public IList<ApplicationsHelper> GetApplications()
    {
        using (var db = new SupportPortalEntities())
        {
            List<ApplicationsHelper> apps = db.Applications.Select(x =>
                new ApplicationsHelper() { ID = x.ID, Name = x.Name }).ToList();
            return apps;
        }
    }
```

Koodi 5. Ote DataController.cs-koodista. GetApplications-funktiolla saadaan lista kannassa olevista applikaatioista.



Kuva 13. Kuvakaappaus Postman-sovelluksesta. API palauttaa datan JSON-muotoisena.

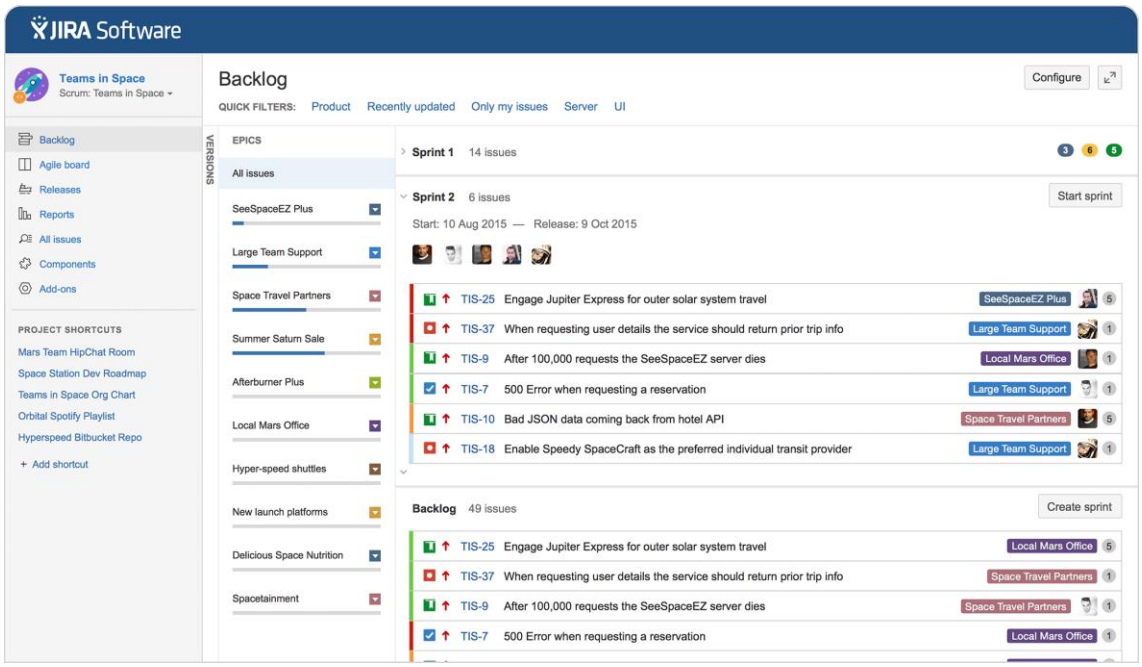
6.4 Jiran datan käsittely

Yksi tärkeimmistä Tukiportaalin tehtävistä on tukipyyntöjen lähettäminen suoraan sovelluksesta JIRA:aan ilman, että tukihenkilöstö joutuu niitä sinne itse kirjoittamaan. Atlassian on tehnyt oman API-rajapinnan, jota kautta voidaan lähettää, muokata, listata ja poistaa dataa JIRA-toiminnanohjausjärjestelmästä.

6.4.1 Atlassian JIRA

JIRA on Atlassianin kehittämä ja ylläpitämä kaupallinen toiminnanohjausjärjestelmä [6.]. Sen tehtävänä on auttaa projektin hallinnassa antamalla projektissa työskenteleville työkalut projektin seurantaan.

JIRA-järjestelmän keskeisessä osassa ovat tiketit. Ne ovat yksittäisiä tehtäviä, joita JIRA ohjelmistolla voidaan hallita. Yksi projekti koostuu monista pienistä tiketeistä, joita seuraamalla projektin eri vaiheista on helppo saada yleiskuva, mikä auttaa sekä ohjelmoijia, että johtoa seuraamaan projektin etenemistä (kuva 14).



Kuva 14. Atlassian Jira järjestelmän backlog näkymä, josta näkee sprintin tehtävät

6.4.2 JIRA:n integraatio ASP.NET sovellukseen

JIRA API:n käsittelyyn on projektissa otettu käyttöön avoimen lähdekoodin Atlassian.NET SDK, jota kehittää Federico Silva Armas [7.]. Atlassian.NET SDK sisältää työkaluja, jotka helpottavat JIRA Web API:n tarjoamien toimintojen käyttöä ASP.NET-ympäristössä.

Jiraan täytyy ensin saada yhteys, joka tapahtuu CreateRestClient-funktiota käyttämällä (Koodi 10). url, käyttäjätunnus ja salasana sijaitsevat web.config-tiedostossa, josta ne haetaan AppSettings.Get funktiolla. Tämän jälkeen API:a voidaan käyttää.

```
jira = Jira.CreateRestClient(
   ConfigurationManager.AppSettings.Get("JIRA_url"),
   ConfigurationManager.AppSettings.Get("JIRA_username"),
   ConfigurationManager.AppSettings.Get("JIRA_password")
);
```

Koodi 6. Ote JiraController.cs-tiedostosta. JIRA REST yhteyden luominen.

Tämän jälkeen pystytään tekemään kutsuja JIRA-järjestelmään Rest clientin kautta. JIRA:sta pystytään esimerkiksi hakemaan lista kaikista tukiportaali- projektin tiketeistä yksinkertaisen LINQ-kyselyn avulla (Koodi 11).

```
public IQueryable<Issue> Get()
{
    var issues = from i in jira.Issues.Queryable where i.Project == "Support Portal"
        orderby i.Created select i;

    return issues.AsQueryable();
}
```

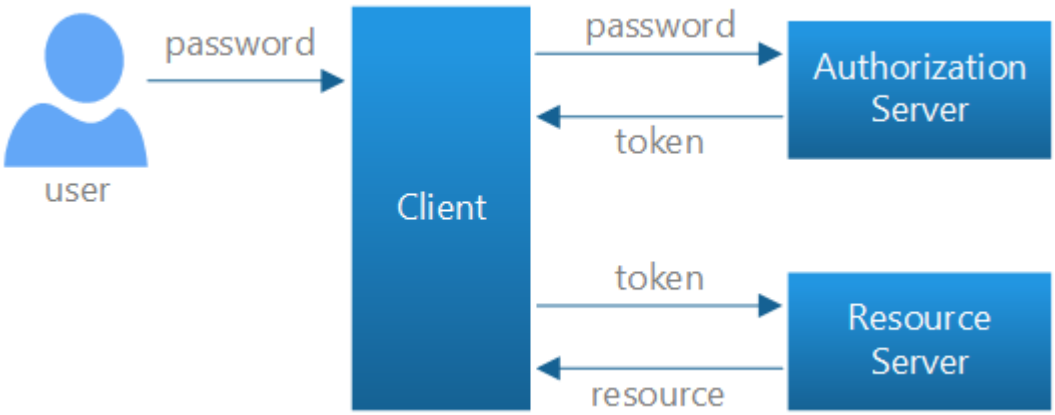
Koodi 7. Ote JiraController.cs-tiedostosta. Tikettien hakeminen JIRA:sta LINQ-kyselyllä.

6.5 Autentikaatio

Tukiportaali on suljettu palvelu, joka tarkoittaa sitä, että sivustolle ei pääse kuka tahansa käyttäjä. Sivustoa käyttääkseen käyttäjä tarvitsee yksilöllisen käyttäjätunnus-salasana-parin. Koska sivusto halutaan olla suljettu palvelu, täytyy autentikaatiojärjestelmän olla toimiva ja luotettava, jotta ulkopuoliset henkilöt eivät pääse katsomaan arkaluonteisia tietoja tai muuten käyttämään palvelua vääriin tai hämäräperäisiin tarkoituksiin.

Tukiportaalin autentikaatio on tehty Taiseer Joudehin artikkelin [8.] pohjalta. Autentikaatiojärjestelmänä käytetään ASP.NET Identity Frameworkia [9.]. Kuvassa 15 on esitetty yleiskuva siitä, miten autentikointi toimii Tukiportaalissa. Käyttäjä syöttää kirjautumislomakkeeseen tunnuksensa ja salasanansa, jotka lähetetään turvallisesti palvelimelle. Autentikaatiopalvelin huolehtii tunnus-salasanaparin paikkansapitävyydestä vertaamalla niitä tietokantaan tallennettuihin tietoihin. Jos kirjautumistiedot täsmäävät, autentikaatiopalvelin lähettää web-sovellukselle paluuviestinä yksilöllisen JSON Web Tokenin (JWT). JWT:tä voidaan jatkossa käyttää autentikoitumiseen, kun halutaan kutsua Web Api:n kontrollereita. JWT:lle on asetettu aika, jolloin se umpeutuu. Tällöin käyttäjän on kirjauduttava palveluun uudelleen, jolloin hän saa uuden JWT:n.

Käyttäjät rekisteröityvät palveluun omalla sähköpostillaan. Sama sähköpostiosoite toimii myös käyttäjätunnuksena sivulle kirjauduttaessa. Sivulle rekisteröityäkseen käyttäjän sähköpostiosoitteen domain-nimi tulee olla sivulle sallittujen domain nimien listalla, jotka määritellään tietokannassa. Tämä on siksi, että Siton asiakkaat ovat yleisesti Suomen kuntien ja kaupunkien työntekijöitä, joten heillä on omat sähköposti domainit. Esimerkiksi esimerkki@gmail.com osoitteilla ei palveluun pystytä kirjautua, mutta esimerkki@Sito.fi pystyttäisiin.



Kuva 15. ASP.NET-autentikaation yleiskuva

7 Yhteenveto

Insinöörityön tarkoituksena oli rakentaa Sitolle ASP.NET backendiä käyttävä Angular-sovellus, jolla pystyttäisiin lähettämään tukitikettejä Atlassian Jira-järjestelmään. Työ onnistui mielestäni hyvin, ja kaikki asetetut tavoitteet saavutettiin.

Työn suurimpana haasteena pidin projektin teknistä suunnittelua. Koska tein projektia yksin, sain joitakin vapauksia sen suhteen, miten projekti etenee ja mitä työkaluja käytän. Toki minulla oli projektipäällikkö vahtimassa, että oikeat hommat tulee tehtyä, mutta oli minusta itsestäni kiinni, miten ne tehtiin ja aikataulutettiin. Vapauden myötä minulla oli myös paljon vastuuta projektin onnistumisen suhteen. Tukiportaalin ohessa osallistuin muihin projekteihin, joten minun täytyi miettiä, miten priorisoida eri työtehtäviä, jotta hommat tulisi tehtyä.

Projektin aikana uusina asioina itselleni oli autentikaation tekeminen ja Jira- API:in integroiminen. Opin, miten autentikaatio implementoidaan ASP.NET-sovellukseen ja miten sitä käytetään asiakassovelluksessa. Jira-API:n integroiminen onnistui hyvin helposti, Atlassian SDK nuget-paketin avulla.

Sovellukseen ei tehty automatisoituja yksikkötestejä. Sen sijaan testasin jokaisen ominaisuuden sekä rajapinnassa että käyttöliittymässä huolellisesti ennen kuin siirryin seuraavan ominaisuuden tekoon. Jatkokehitystä varten yksikkötestejä olisi hyvä lisätä tulevaisuudessa.

Jatkokehityksen kannalta sovelluksen toteutus tehtiin helposti muokattavaksi ja ymmärrettäväksi. Sovelluksen koodiin on kirjoitettu paljon ohjaavia kommentteja, minkä lisäksi sovellusta varten kirjoitetaan dokumentaatio. Sovellusta on helppo laajentaa ja siihen on helppo lisätä uusia ominaisuuksia Angularin selkeän rakenteen ja modulaarisuuden vuoksi.

Sovellus tullaan ottamaan testikäyttöön vuoden 2017 syksyllä muutaman asiakkaan kanssa, jotka ovat suostuneet testaamaan sovellusta. Muille asiakkaille Tukiportaali tulee käyttöön vuoden loppuun mennessä.

Lähteet

1 Angular 2 Architecture Overview. 2017. Verkkodokumentti. Google. https://angular.io/guide/architecture.

2 Yakov Fain, Anton Moiseev. 2017. Angular Development with Typescript.

3 Bootstrap. 2017. Verkkodokumentti. Bootstrap Core Team http://getbootstrap.com/.

4 Rangle.IO Using Two-Way Data Binding. Verkkodokumentti. Range.IO team https://angular-2-training-book.rangle.io/handout/components/app_structure/two_way_data_binding.html.

5 Angular 2 - Pagination Example with Logic like Google. 2017. Verkkodokumentti. Jason Watmore. http://jasonwatmore.com/post/2016/08/23/angular-2-pagination-example-with-logic-like-google.

6 Jira. 2017. Verkkodokumentti. Atlassian https://www.atlassian.com/software/jira.

7 Atlassian.NET SDK. 2017. Verkkodokumentti. Federico Silva Armas. https://bitbucket.org/farmas/atlassian.net-sdk/wiki/Home.

8 Secure a Web API with Individual Accounts and Local Login in ASP.NET Web API 2.2. 2014. Verkkodokumentti. Mike Wasson https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/individual-accounts-in-web-api.

9 ASP.NET Identity 2.1 with ASP.NET Web API 2.2. 2015. Verkkodokumentti. Taiseer Joudeh http://bitoftech.net/2015/01/21/asp-net-identity-2-with-asp-net-web-api-2-accounts-management/.