VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Juha Visuri

LAATUPOIKKEAMIEN KÄSITTELYJÄRJESTELMÄN RAKENTAMINEN

Case VAMM Steel Oy

Liiketalous
2020

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Juha Visuri
Opinnäytetyön nimi	Laatupoikkeamien käsittelyjärjestelmän rakentaminen
Vuosi	2020
Kieli	suomi
Sivumäärä	29
Ohjaaja	Raija Tuomaala

Tämä opinnäytetyö käsittelee laatupoikkeamien käsittelyjärjestelmän toteutusta moderneja teknologioita käyttäen. Työn tavoitteena oli luoda järjestelmä poikkeamien kirjaukseen ja käsittelyyn. Työn toimeksiantaja VAMM Steel Oy.

Järjestelmä tulisi korvaamaan tällä hetkellä Excelissä tehtävän reklamaatioiden tilastoinnin ja käsittelyn. Järjestelmässä yrityksen toimihenkilöt voisivat kirjata ja käsitellä asiakas-, sisäisiä- sekä toimittajareklamaatioita. Työssä käydään läpi toteutuksessa käytettävät teknologiat ja ohjelmistokehykset, joita työt toteuttamiseen käytettiin.

Työn tuloksena syntyi toimiva ohjelmistokokonaisuus, jonka toimeksiantaja ottaa käyttöön kesäkuun 2020 aikana. Projekti oli osana toimeksiantajan tuotannon kehitysprojektia, johon Business Finland on myöntänyt 100 000 euron tuen.

Avainsanat Laadunhallinta, ASP.NET Core, Angular 9

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Data Processing

ABSTRACT

Author	Juha Visuri
Title	Creating a Complaint Management System
Year	2020
Language	Finnish
Pages	29
Name of Supervisor	Raija Tuomaala

This thesis studies a complaint management system solution implementation using modern technologies. The aim for this project was to build a system documenting and handling complaints. The study was commissioned by VAMM Steel Oy.

The system would replace Excel where all complaint handling and statistics are currently managed. In this system, the supervisors of every team could document and handle customer, supplier and internal complaints. In this thesis, technologies and frameworks used in implementation of the software were examined.

The result was a working complaint management system, which the client will put into use in June 2020. The project was part of the client's production development project, which was funded by Business Finland for 100,000€

Keywords Quality management, ASP.NET Core, Angular 9

SISÄLLYS

KÄSITTEET

NPM

Node Package Manager on verkossa toimiva alusta, johon kehittäjät voivat julkaista omia avoimen lähdekoodin Node.js -kirjastoja.

Ohjelmistokehys

Sovelluskehys muodostaa ohjelmiston rungon, jonka päälle ohjelmisto voidaan rakentaa.

REST

Representational State Transfer on HTTP-protokollaan perustuva ohjelmistoarkkitehtuurimalli, jota käytetään ohjelmistorajapintojen toteutuksessa.

SPA

Single Page Application on verkossa toimiva sovellus, jossa näkymiä vaihdetaan dynaamisesti ilman selainikkunan uudelleenlatausta.

SOAP

Simple Object Access Protocol on HTTP-protokollaan perustuva strukturoitu tiedonvälitysmalli.

KUVIO- JA TAULUKKOLUETTELO

1 JOHDANTO

Opinnäytetyössä toteutetaan laatupoikkeamien hallinta-, ja käsittelyjärjestelmä VAMM Steel Oy:lle. Järjestelmän rakentamisessa hyödynnetään moderneja verkko-ohjelmoinnin tekniikoita ja työkaluja, joita ovat esimerkiksi Angular ja ASP.NET Core. Työssä käydään läpi järjestelmässä käytettävät teknologiat ja niiden keskeisimmät ominaisuudet teoriassa ja varsinaisessa toteutuksessa. Työssä käydään läpi myös ohjelmiston rakennetta sekä poikkeamien käsittelyprosessia peilaten toimeksiantajan tarpeisiin.

1.1 Toimeksiantaja

Työn toimeksiantaja VAMM Steel Oy on vaasalainen ohutlevyteollisuuden yritys. Yritys valmistaa pääasiassa tuotteita asiakkaidensa moottoreihin ja koneisiin ohutlevystä tai metallia työstämällä. Yritys työllistää tällä hetkellä Vaasan tehtaalla Strömberg Parkin alueella noin 65 työntekijää, joista 11 on toimihenkilöitä. (VAMM Steel Oy 2020.)

1.2 Työn tavoitteet

Työn tavoitteena on toteuttaa yrityksen sisäverkossa toimiva verkkoalusta, jossa laatupoikkeamien käsittely olisi mahdollisimman yksinkertaista ja nopeaa. Ohjelmisto tulee pääasiassa yrityksen laatutiimin ja työnjohtajien käyttöön. Se korvaa tällä hetkellä käytössä olevan Excel-taulukon, johon työntekijöillä on pääsy verkkoaseman kautta. Ohjelmisto on myös mahdollista liittää VAMM Steelillä käytössä olevaan Lemonsoft-toiminnanohjausjärjestelmään, josta poikkeamien asiakkaat ja tuotenimikkeet ovat päivitettävissä automaattisesti. Tätä kautta laaturaporteille on mahdollista saada laadunhallinnan standardimääreitä kuten PPM-luku ja reklamaatioprosentti. Myös poikkeamien juurisyiden selvittämisen prosessia pyritään yhtenäistämään, jotta toistuvat poikkeamat voidaan löytää. Järjestelmä kattaa sisäisten poikkeamien sekä asiakas- ja toimittajapoikkeamien kirjauksen ja käsittelyn.

2 PALVELINOHJELMISTOT JA TEKNIIKAT

Kappaleessa esitellään ohjelmiston palvelinpuolen toteutuksessa käytetty ASP.NET Core -ohjelmistokehys ja sen edeltäjänä toiminut ASP.NET. Työssä käydään läpi myös ASP.NET-ohjelmistokehityksessä käytettävät kielet, arkkitehtuuri ja muut työssä käytetyt NuGet-paketit.

2.1 ASP.NET ja ASP.NET Core

ASP.NET on osa Microsoftin vuonna 2002 julkaisemaa .NET-ohjelmistokehystä. ASP.NET on maaliskuussa 2020 maailman toiseksi käytetyin verkkosivustojen toteutuksessa käytetty ohjelmistokehys. Ensimmäisessä versiossaan ASP.NET mahdollisti dynaamisten verkkosivustojen lisäksi myös XML (SOAP)-palveluiden toteutuksen. Datalähtöisen palveluiden rakentaminen työpöytätyyppisillä komponenteilla ja tapahtumilla oli nopeaa ja tehokasta Microsoftin Visual Studio -ohjelmistokehitysympäristössä. Versioissa 2, 3.5, 4 ja 4.5 ASP.NET sai pieniä lisäyksiä, mutta suuri muutos tuli Microsoftin julkaistessa sen avoimeen lähdekoodiin. Versioiden 3.5 ja 4 välissä Microsoft julkaisi ASP.NET:in rinnalle täysin uuden ohjelmistokehyksen, joka perustui MVC-arkkitehtuuriin ja oli edeltäjänsä mukaan avoimen lähdekoodin kirjasto. Vaikka ASP.NET MVC olikin rakennettu edeltäjänsä päälle, oli sillä täysin uusi rakenne. Se perustuu ajatusmalliin, jossa sovellus jaetaan kolmeen pääosaan; malli (model), näkymä (view), ja ohjain (controller). Mallissa käyttäjän pyynnöt ohjataan ohjaimelle, joka toteuttaa käyttäjän pyytämän toiminnon tai noutaa pyydetyt tiedot. Ohjain valitsee oikean näkymän käyttäjän pyytämän tiedon näyttämiseen ja täyttää sen mallista saadulla tiedolla. (Peres 2017.)

ASP.NET on Windows-alustalle tehty .NET-pohjainen ohjelmistokehys, kun taas ASP.NET Core on alustariippumaton. ASP.NET Coren Microsoft julkaisi vuonna 2016, ja sen tarkoituksena oli rikkoa rajat eri alustojen välillä. Koska ASP.NET on .NET-pohjainen ohjelmistokehys, oli Microsoftin kirjoitettava koko ASP.NET Core -kirjasto uudelleen saadakseen siitä alustariippumattoman. Työn laajuus oli

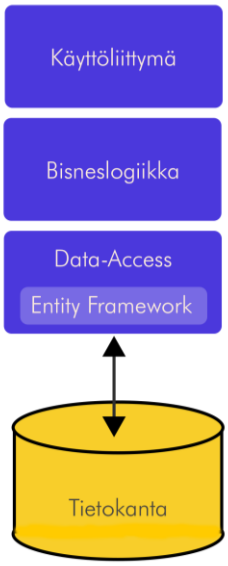
kuitenkin niin suuri, että Microsoftin oli priorisoitava tärkeimpiä ominaisuuksia ensimmäiseksi. Heinäkuussa 2016 julkaistiin versio 1.0, joka sisälsi MVC-mallin jättäen Web Formsin taakseen. ASP.NET Core sisälsi myös Web API:n, joka alkuperäisessä ASP.NET:issä oli erillisenä kirjastona, vaikka se piti sisällään paljon MVC:n luokkia samoilla nimillä. (Peres 2017.)

2.2 C Sharp

C Sharp (C#) on Microsoftin kehittämä ja 2001 julkaisema ohjelmointikieli. C# on vahvasti tyypitetty ja oliopohjainen kieli, joka mahdollistaa alustariippumattomien ohjelmistojen kehityksen. Vaikka C Sharpin kehitysestä ja ylläpidosta vastaakin Microsoft, on se silti avoimen lähdekoodin kieli. (Chand 2020.)

2.3 Entity Framework

Entity Framework (EF) on Microsoftin tarjoama ohjelmistokehys, joka poistaa tietolähtöisten sovellusten Data Access -osioiden manuaalisen kirjoituksen .NET-sovelluksissa. Data Access -osioihin EF automatisoi tietokantayhteyden avaamiset, tiedon noutamiset sekä tietoaineistojen muuntamiset .NET-objekteiksi ja takaisin. Automaattinen prosessi estää inhimilliset virheet ja nopeuttaa palvelinpuolen ohjelmiston kehitystä. (Entity Framework Tutorial 2020.)



Kuvio 1. Entity Framework.

2.4 JSON Web Token

JSON Web Token (JWT) on turvallinen tapa välittää tietoa osapuolten välillä JSON-objektin muodossa. Siirrettävän tiedon luotettavuus voidaan varmistaa sen digitaalisen allekirjoituksen ansiosta. JWT voidaan allekirjoittaa kahdella tavalla. Ensimmäinen tapa on salainen avain, joka allekirjoitetaan käyttäen HMAC-algoritmia. Toinen tapa on käyttää julkista ja salaista avainta ja salakirjoittaa nämä RSA- tai ECDSA -algoritmeja käyttäen. Mikäli autentikointimerkki (token) on allekirjoitettu käyttäen julkisen ja salaisen avaimen yhdistelmää, voi myös tokenin allekirjoittajan identiteetistä myös varmistua. (JWT 2020.)

JWT:n yleisin käyttötarkoitus on käyttäjän autentikointi. Yleisimmässä tilanteessä käyttäjä kirjautuu palveluun, jolloin palvelin todentaa käyttäjän ja palauttaa tälle JWT-tokenin. Saadulla tokenilla käyttäjä voi päästä käsiksi palveluihin tai resursseihin, jotka saatu token mahdollistaa. Token siis lähetetään jokaisen palvelinkutsun mukana, jolloin palvelin tarkistaa sen alkuperäisyyden ja voimassaolon, minkä jälkeen se palauttaa käyttäjälle pyydetyt tiedot. (JWT 2020.)

3 TOTEUTETUN KÄYTTÖLIITTYMÄT TEKNIIKAT

Tähän mennessä on esitelty projektin palvelinpuolen toteutuksessa käytetyt kielet ja teknologiat. Tässä kappaleessa käydään läpi ohjelmiston käyttöliittymän toteutuksessa käytetyn Angularin tärkeimmät ominaisuudet, keskeisimmät npm-paketit ja kehityksessä käytetyt TypeScript -ohjelmointikielen.

3.1 TypeScript

TypeScript on Microsoftin 2012 julkaisema avoimen lähdekoodin ohjelmointikieli. TypeScript-koodia suoritettaessa koodi käännetään ensin JavaScript-ohjelmointikieleen ennen sen suorittamista selaimessa tai erillisessä JavaScript-moottorissa. (Fain & Moiseev 2020)

3.2 Angular

Angular on Googlen kehittämä avoimen lähdekoodin JavaScript-ohjelmistokehys. Angular julkaistiin vuonna 2016 nimellä Angular 2, joka pohjautui edeltäjäänsä AngularJS-ohjelmistokehykseen. Pian julkaisun jälkeen nimi muutettiin pelkäksi Angulariksi. Googlen tiimi julkaisee ohjelmistokehyksestä uuden version kaksi kertaa vuodessa. Niissä päivitetään uusia toimintoja parantaen myös tehokuutta. (Moiseev & Fain 2018a.)

Angular on komponenttiperustainen ohjelmistokehys, jossa näkymä koostuu yhdestä pääkomponentista ja sen sisään ladattavista alikomponenteista. Alikomponenteilla on mahdollista myös olla omia alikomponentteja ja niin edelleen, jolloin sivuston rakennetta voidaan kuvata puumallisesti. Emokomponentti voi välittää tietoa alikomponenteille muuttujien kautta tietoa ilman, että lapsikomponentti tietää mistä tieto tuli. Lapsikomponentti voi välittää tietoa myös rakenteessa ylöspäin tietämättä, kenelle tieto lähtee. Tämä mahdollistaa samojen komponenttien uudelleenkäytön monissa eri näkymissä ilman että niitä täytyisi kirjoittaa uudelleen. (Moiseev & Fain 2018a.)

3.2.1 Komponentit

Komponentti on Angularin käyttöliittymän takana toimiva luokka, jota voidaan käyttää lisäämällä siihen @Component-decorator. Decorator on funktio, joka mahdollistaa koodin laajennuksen sekä metatiedon lisäämisen. Komponentti voi koostua .html, .css ja spec.ts -tiedostoista, joihin voidaan kirjoittaa komponentin ulkoasu, tyylit ja automaattisesti ajettava testaus. Komponenttia ei välttämättä kuitenkaan ole pakko jakaa edellä mainittuihin osiin, jolloin tyylit ja ulkoasu määritellään komponentin sisällä. Komponentti voidaan ottaa käyttöön lisäämällä se sovelluksen @ngModule()-decoraattoriin. Komponentti voi ottaa tietoa vastaan hierarkiassa sen yläpuolella olevalta komponenteilta Input-parametrien avulla ja vastaavasi välittämään sitä takaisin EventEmitter-luokan avulla. (Moiseev & Fain 2018b.)

3.2.2 Palvelut

Arkkitehtuurin ylläpitämiseksi bisneslogiikan toiminnot kirjoitetaan omiin palveluihin. Palveluissa suoritetaan kaikki tiedon haku ja päivitys palvelimelta kuin myös tiedon käsittely ja muokkaus. Yhtä tai useampaa palvelua voidaan kutsua komponenttien sisältä. (Moiseev & Fain 2018b.)

3.2.3 Moduulit

Angularin arkkitehtuurissa sovelluksen osa-alueet voidaan jakaa pienenpiin kokonaisuuksiin, joita kutsutaan moduuleiksi. Moduuli sisältää aina yhden kokonaisuuden mm. kaikki komponentit, palvelut, direktiivit. Isossa sovelluksessa moduulit voidaan jakaa esimerkiksi liiketointa-alueittain, jolloin laskutus tai kirjanpito voivat toimia omina moduuleinaan. Pienissä sovelluksissa ei kuitenkaan moduuleihin jakaminen ole välttämätöntä, vaan kaikki sisältö voidaan liittää suoraa juurimoduuliin. Moduuleihin on myös mahdollista lisätä omia alimoduuleita, jolloin suuret sovellukset voidaan helposti jakaa pieniin kokonaisuuksiin. (Moiseev & Fain 2018b.)

3.2.4 Reititys

Angularin on Single Page Application (SPA) -perusteinen ohjelmistokehys. Tällaisissa sovelluksissa ei käyttäjän selainikkunaa päivitetä uudelleen sivun vaihdon yhteydessä. Tähän tarpeeseen Angular sisältää Router-elementin, joka mahdollistaa näkymien välillä liikkumisen ilman selainikkunan päivitystä. Näkymien väliset navigoinnin perustuvat selaimen osoitteen muutoksiin, joiden perusteella Router osaa muuttaa näkymää. (Angular 2020.)

Router voi kuljettaa osoitteessa myös parametreja, joiden perustella uusi näyttö ladataan. (Angular 2020.) Alla olevassa kuvassa on havainnollistettuna projektissa käytetyn asiakaspoikkeamat-näytön reititys, jossa reitin parametrinä välitetään poikkeaman yksilöllinen numero.

Kuvio 2. Reititys.

3.3 Bootstrap

Bootstrap on maailman suosituin avoimen lähdekoodin ohjelmistokehys responsiivisten verkkosivujen luontiin. Bootstrap julkaistiin vuonna 2011 ja sitä ylläpitää 12 hengen GitHub-tiimi. Bootstrap on vuosien varrella saanut kolme suurta päivitystä, joissa elementtien responsiivisuutta on parannettu ja uusia komponentteja julkaistu. (Bootstrap 2020.)

3.4 Node Package Manager

Node Package Manager (npm) on verkossa toimiva alusta, jonne kehittäjät voivat julkaista omia avoimen lähdekoodin Node.js -paketteja. Paketteja voidaan ladata komentokehotteen kautta joko projektikohtaisesti tai globaalisti. Paketin lataus projektikohtaisesti lisää halun paketin Node.js -projektin `/.node_modules/`-kansioon, jolloin siihen voidaan viitata projektin sisältä. Paketteja voidaan ladata myös työasemalle globaalisti lisäämällä asennuskomennon perään -g, jolloin paketti asentuu työasemakäyttäjän omaan polkuun /bin. (Node.js 2011.)

Npm-pakettien pääasiallisena tarkoituksena on ratkaista tai helpottaa tietyn aihealueen käsittelyä. Hyvänä esimerkkinä tästä on Moment.js -kirjasto, jonka tarkoituksena on helpottaa kelloaikojen käsittelyä. Se mahdollistaa aikamääreiden uudelleenmuotoilun sekä muuntamisen toiselle aikavyöhykkeelle. Toinen esimerkki npm-paketista on Ngx-charts, jolla tiedon visualisointi käyttäjälle onnistuu valmiiden kaavioiden avulla.

3.5 Angular Material

Angular Material on Googlen julkaisema avoimen lähdekoodi komponenttikirjasto, joka tarjoaa koodit Angularin käyttöliittymän luontia varten. Angular Material pohjautuu Google Material Desing -ohjeistukseen, joka tarjoaa yleisimmät komponentit responsiivisen käyttöliittymän luontiin. (Material Design 2020.)

4 TOTEUTUS

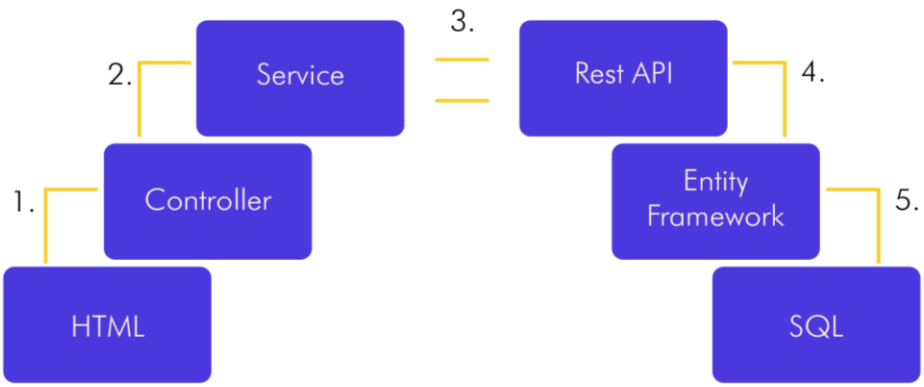
Teoriaosuudessa on tähän mennessä esitelty järjestelmän toteutuksessa käytettyjä työkaluja, tekniikoita ja niiden keskeisimpiä toiminnallisuuksia. Tässä luvussa tullaan kuvaamaan järjestelmän käyttötarkoitusta yrityksessä, ohjelmiston rakennetta sekä sen toteutusta.

4.1 Tarpeiden kartoitus

VAMM Steel Oy on ohutlevytuotteita valmistava yritys, joka pääasiallisesti tuottaa osia asiakkaidensa koneistoihin. Yrityksen toimihenkilöt vastaavat tilausten käsittelystä, ajoittamisesta tuotantoon, mahdollisten aliosien ostamisesta sekä osien piirustuksista. Tämän prosessin tukena yrityksessä on käytössä Lemonsoft-toiminnanohjausjärjestelmä, joka mahdollistaa koko ketjun aina tilauksen saapumisesta kuljetukseen saakka. Poikkeamien käsittelyyn ei yrityksessä kuitenkaan ole omaa järjestelmää tai muutakaan selvää mallia, jolla poikkeamatilanteen saataisiin ratkaistua. Saapuvat poikkeamat usein hoidetaan sähköpostitse suoraa asiakkaan kanssa, jonka jälkeen poikkeama tallennetaan verkkolevylle. Poikkeamia on satunnaisesti tilastoitu Excel-työkirjaan, mutta tämän päivittäminen on ollut manuaalista ja jälkikäteen työlästä. Poikkeamien kokonaiskustannuksista tai syistä ei tilastoa ole erikseen tehty, vaan poikkeamaa käsitellyt henkilö on vain saattanut hyvittää tuotteet asiakkaalle. (Visuri 2020.)

4.2 Poikkeamakäsittelyjärjestelmän rakenne

Ohjelmisto kokonaisuudessaan rakentuu käyttäjälle näkyvästä HTML-sivusta, Angularin ohjaimesta ja palvelusta, ASP.NET Core Web API:sta, EntityFramework -ohjelmistokehyksestä sekä SQL-tietokannasta. Tiedon siirto käyttöliittymästä tietokantaan tapahtuu alla esitetyn kuvan mukaisesti.



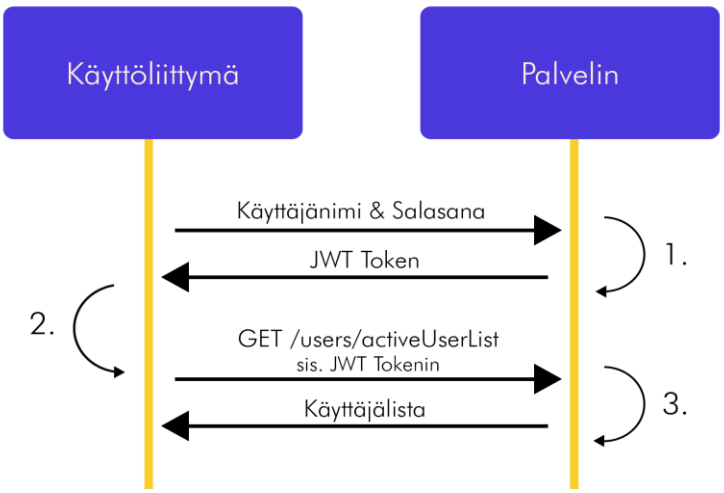
Kuvio 3. Teknologiakaavio.

1. Prosessi alkaa, kun käyttäjä suorittaa toiminnon HTML-sivulla. Esimerkkinä tällaisesta toiminnasta on painikkeen klikkaus.
2. Näkymää hallitseva Angular-ohjain (Controller) kutsuu palvelua (Service) ja välittää sille tarvittavat tiedot.
3. Palvelu ottaa ohjaimelta tulevan tiedon vastaa ja luo siitä palvelinkutsun. Tietoa tallennettaessa kutsun luontiin käytetään http.post-metodia.
4. ASP.NET Core Web API ottaa tulevan pyynnön vastaan, jolloin sen sisältämä koodi suoritetaan.
5. Entity Framwork kääntää koodin SQL-lauseeksi ja suorittaa tietueen tallennuksen tietokantaan.

4.3 Kirjautuminen

Ohjelman käyttö alkaa siihen tunnistautumisella. Kirjautumisnäytöllä käyttäjä voi kirjautua hänelle luodulla käyttäjätunnus ja salasanaparilla sisään, jolloin palvelin luo istunnon (session) ohjelmaan. Kun käyttäjätunnus ja salasana ovat oikeat, palauttaa palvelin käyttäjälle JWT-tokenin, joka tallennetaan selaimen välimuistiin. Jokaisen palvelinkutsun yhteydessä käyttöliittymästä lähtevään kutsuun lisätään token, jolloin palvelin voi varmistua käyttäjän identiteetistä sekä session voimassaolosta. Käyttöliittymä tarkkailee myös palvelimelta palaavia vastauksia vanhentuneiden istuntojen varalta, jotta käyttäjä osataan ohjata uudelleenkirjatumaan, mikäli istunto on vanhentunut. Alla olevassa kaaviossa

palvelimelta noudetaan lista aktiivisista käyttäjistä mallintaen samalla autentikoinnin toimintaa.



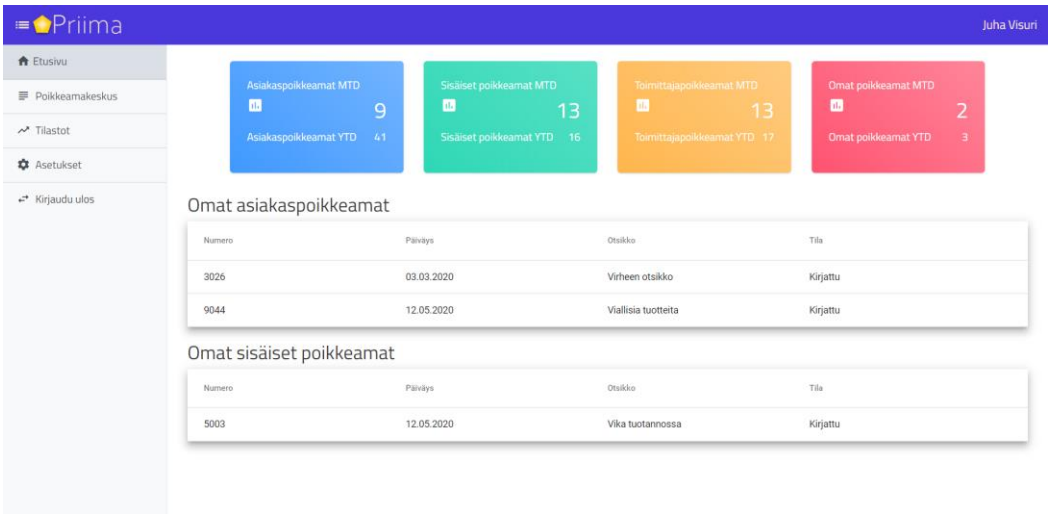
Kuvio 4. Autentikointi.

1. Käyttöliittymästä tulevat käyttäjätunnukset tarkistetaan ja käyttäjän kirjautumisesta luodaan istunto. Istunnon tiedot sisältävä JWT-token palautetaan käyttäjälle.
2. Palvelimelta saatu JWT-token tallennetaan käyttäjän selaimen välimuistiin, josta se voidaan lisätä palvelinkutsun otsikkoon. Käyttöliittymästä lähetetään palvelimelle nyt pyyntö listata kaikki aktiiviset käyttäjät sisältäen JWT-tokenin.
3. Palvelimelle saapuvasta kutsusta luetaan JWT-token ja sen kelpoisuus tarkastetaan. Mikäli JWT-token on kelvollinen palauttaa palvelin pyydetyn käyttäjälistan.

4.4 Etusivu

Etusivu toimii käyttäjän omana työnäkymänä. Näytön yläosan korteissa esitetään poikkeamien määristä yhteenvedon. Se sisältää kaikkien poikkeamatyyppien määriä vuoden sekä kuukauden alusta. Korttien alla listataan kirjautunutta

käyttäjää koskevat asiakas- ja sisäiset poikkeamat, joihin käyttäjä on merkitty käsittelijäksi. Poikkeamalistoilta käyttäjä voi navigoitua suoraa poikkeamalle, jonka käsittelyn jälkeen se poistuu listauksesta.



Kuvio 5. Etusivu.

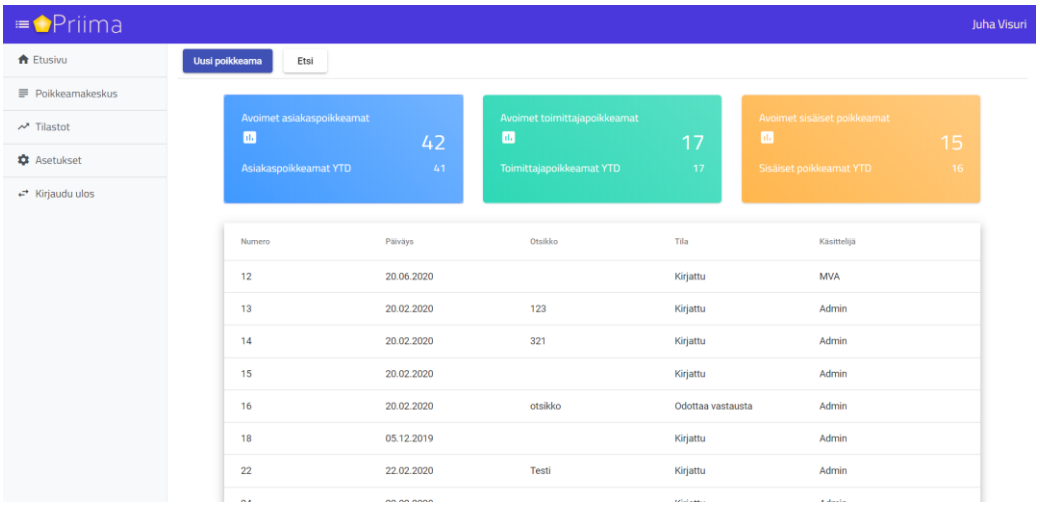
Käyttäjän käsittelyä odottavat pyynnöt saadaan haettua lukemalla käyttäjän yksilöivä numero JWT-tokenin rungosta. Alla olevassa funktiossa käyttäjän numero luetaan JWT-tokenin sisältä.

```
private int getUserFromJwt()
{

    string stream = Request.Headers["Authorization"];
    stream = stream.Substring("Bearer ".Length).Trim();
    var handler = new JwtSecurityTokenHandler();
    var jsonToken = handler.ReadToken(stream);
    var tokenS = handler.ReadToken(stream) as JwtSecurityToken;
    var userId = tokenS.Claims.First(claim => claim.Type == "sub").Value;
    int iUserId = Int32.Parse(userId);

    return iUserId;
}
```

Kuvio 6. Tokenin käyttäjä.

4.5 Poikkeamakeskus

Poikkeamakeskus-näyttö toimii avoimien poikkeamien yleisnäkymänä. Näytön tarkoituksena on esitellä yleisimpiä lukuja poikkeamista sekä esittää avoimet poikkeamat.



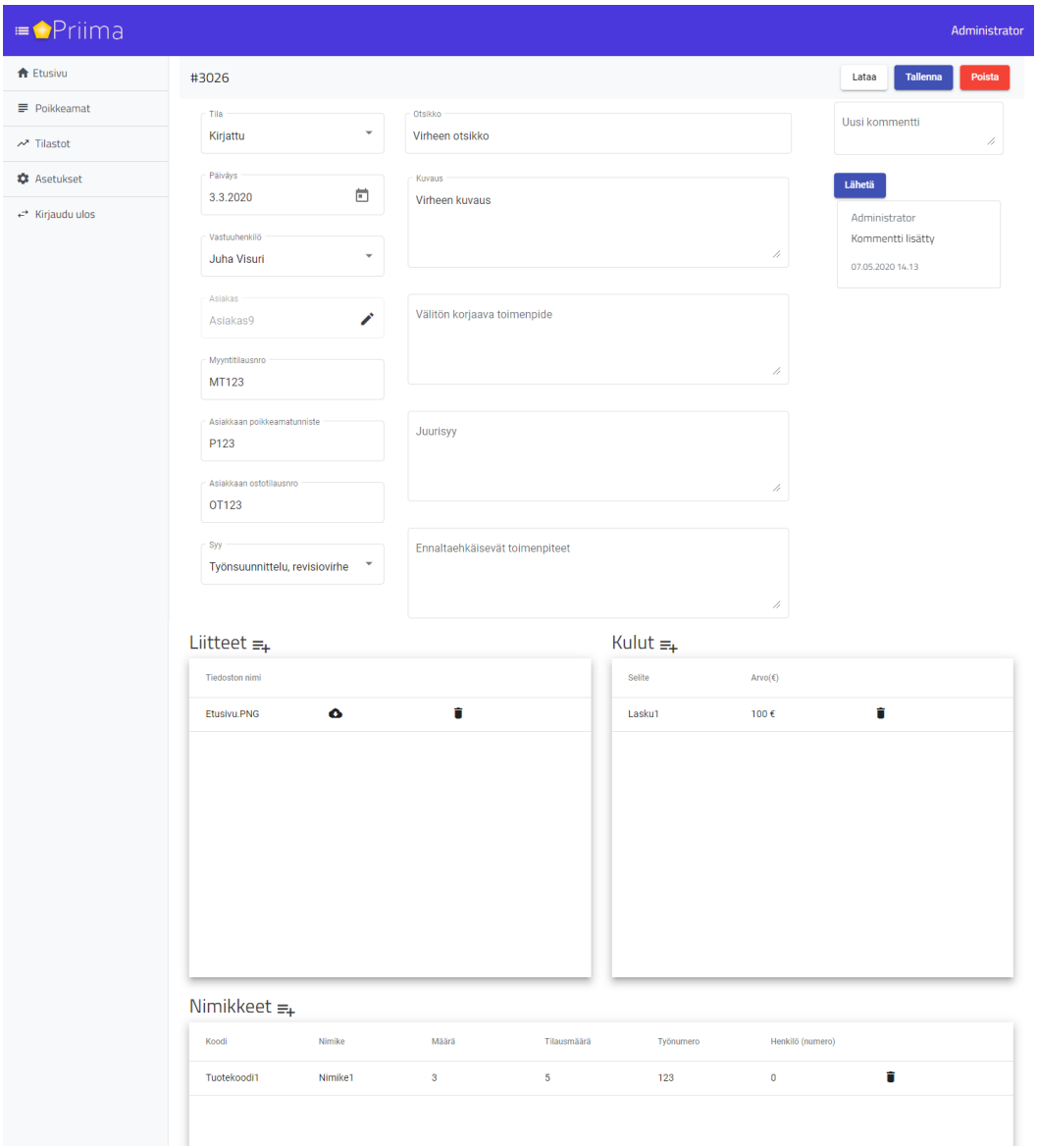
Kuvio 7. Poikkeamat.

Näytön yläosassa käyttäjä voi luoda uuden poikkeaman painikkeesta avautuvan dialogin kautta, sekä hakea vanhoja poikkeamia erilaisin parametrein. Ylärivillä poikkeamista koostetaan avainlukuja kuten YTD (Year-To-Date) ja MTD (Month-To-Date), joilla mitataan poikkeamien määriä vuoden ja kuukauden aluista. Korttia klikkaamalla taulukkoon ladataan valitus poikkeamatyypin avoimet poikkeamat. Poikkeamalistalla esitellään kaikki avoimet poikkeamat sivutetusti.

4.6 Laatupoikkeama

Laatupoikkeamanäytöllä esitetään poikkeaman perustietojen lisäksi liitteet, kulut, nimikkeet sekä sisäiset kommentit. Perustiedossa poikkeaman käsittely voidaan ohjata esimerkiksi osaston työnjohtajalle, jolloin poikkeaman näkyy hänellä

etusivun työlistalla. Syykodit kentässä käyttäjä voi pudotusvalikosta valita jonkin järjestelmään perustetuista syykoodeista, joka on poikkeamien tilastoinnin kannalta avainasemassa.



Kuvio 8. Laatupoikkeama.

Poikkeamanäytön oikeaan reunaan on lisätty myös kommenttikenttä, johon käyttäjät voivat kirjata sisäisiä kommentteja. Kommentin lisäys päivittää uuden kommentin listan ylimmäksi tiedoksi sisältäen kommentin ajan sekä kirjoittajan nimen.

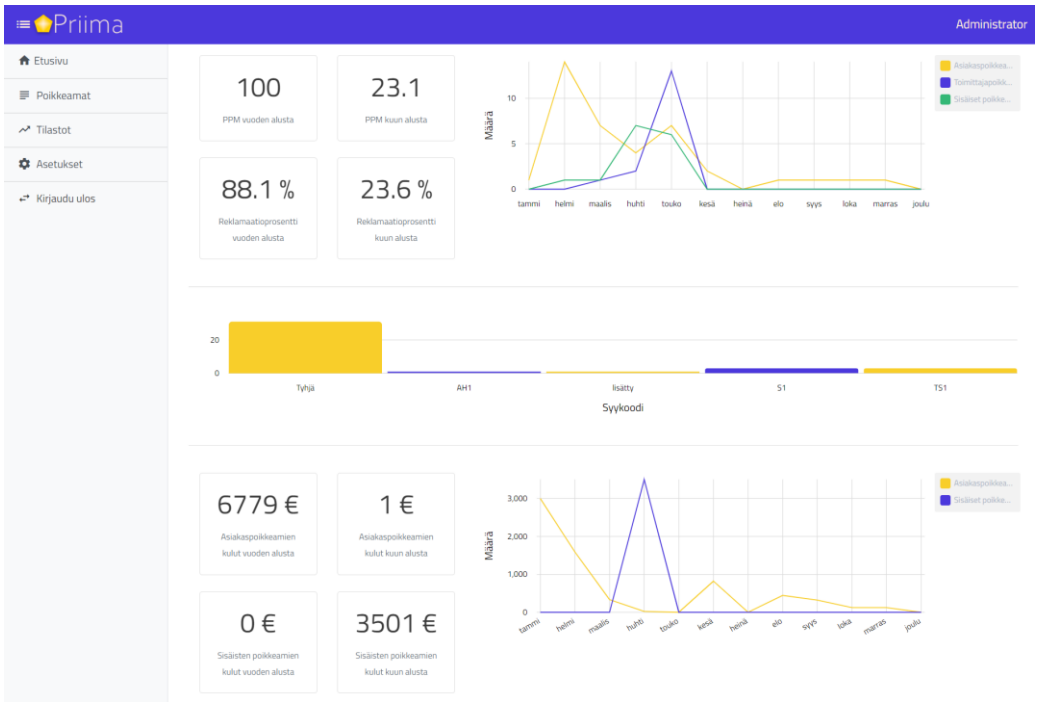
Poikkeamalle voidaan tallentaa myös liitteitä. Uuden liitteen lisäyksessä käyttäjän lataama tiedosto muunnetaan merkkijonoksi, jolloin sen tallennus suoraa titokataan onnistuu. Tällä toiminnolla vältetään erillisen tietostohakemiston luonti palvelimelle, jolloin polkujen käyttöoikeudet täytyisi ottaa huomioon. Poikkeamalle tallennettu liite voidaan riviltä myös ladata omalle työasemalle, jolloin se käännetään merkkijonosta takaisin alkuperäiseksi tiedostoksi.

Poikkeamalle voidaan lisätä myös siitä aiheutuneita kuluja. Kulurivien lisäys tapahtuu dialogin kautta, jossa käyttäjä voi antaa kululle myös selitteen. Poikkeamien kulurivejä käytetään kulujen tilastoinnissa tilastot-näytöllä.

Näytön viimeiseen osaan poikkeamalle voidaan merkitä poikkeamaan liittyviä tuotenimikkeitä. Tuotenimikkeen lisäys tapahtuu oma dialogin kautta, jossa riville voidaan syöttää tuotanimikkeen lisäksi kappalemäärä sekä tilattu määrä.

4.7 Tilastot

Tilastonäytöllä koostetaan poikkeamien tiedot tilastoihin. Näytöllä käyttäjälle esitetään poikkeamin määriä, kuluja sekä syitä kuluvan vuoden ja kuukauden aluista laskettuina. Näyttö jakautuu kolmeen osaan, joissa käyttäjälle näytetään kootusti poikkeamien määriä, poikkeamine yleisimpiä syitä sekä laatupoikkeamien aiheuttamia kuiluja.



Kuvio 9. Tilastot.

Tilastojen ensimmäisessä osiossa on koostettuna poikkeamien määriä, sekä niiden suhdetta koko yrityksen tuottamiin kappalaisiin. PPM (Parts per million) on yleinen laadunhallinnassa käytetty mittari, jolla viallisten kappaleiden määrä saadaan suhtautettua tuotettuihin kappaleisiin. Luku saadaan laskettua jakamalla virheellisten kappaleiden määrä tuotannosta valmistuneiden kappaleiden määrällä ja kertomalla saadun luvun sadalla. Reklamaatioprosentti taas lasketaan jakamalla poikkeamien määrän toimitettujen myyntitilauksien määrällä. Lukujen laskentaa näytölle ei toteuteta manuaalisella toimenpiteellä, vaan tuotannosta valmistuneet kappaleiden sekä myyntitilausten määrä noudetaan automaattisesti Lemonsoft-toiminnanohjausjärjestelmästä päivittäin ensimmäisen kirjautumisen yhteydessä.

Korttien vieressä näytetään kuluvan vuoden poikkeamien määriä, jotka näkyvät ryhmiteltynä päiväyksen mukaisesti kuukausittain. Kaikille poikkeamatyypeille piirretään oma kuvaajansa, jolloin käyttäjä voi seurata kaikkien poikkeamatyyppien määriä yhdessä kaaviossa. Kaavioihin piirrettävä tieto noudetaan SQL Server-tietokannasta Entity Frameworkin avulla, joka muuntaa

koodin automaattisesti SQL-lauseiksi ja noutaa pyydetyt tiedot. Kaavion tiedot noudetaan tietokannannasta alapuolella esitetyllä tavalla.

```
[Route("api/GetCustomerComplaintCountByMonth")]
[HttpGet]
public IActionResult GetCustomerComplaintCountByMonth()
{
    var firstDayOfYear = new DateTime(DateTime.Now.Year, 1, 1);
    var lastDayOfYear = new DateTime(DateTime.Now.Year, 12, 31);

    var months = Enumerable.Range(1, 12)
    .Select(month => month)
    .ToList();

    var customerComplaints = _context.CustomerComplaints
       .Where(a => a.ComplaintDate >= firstDayOfYear && a.ComplaintDate <= lastDayOfYear).ToList();


    var q = from m in months
            join cc in customerComplaints on m equals cc.ComplaintDate.Month into joined
            from cc in joined.DefaultIfEmpty()
            select new
            {
                name = CultureInfo.CreateSpecificCulture("fi-FI").DateTimeFormat.GetAbbreviatedMonthName(m),
                value = joined.Count()
            };

    return Ok(q.Distinct());
}
```

Kuvio 10. Asiakaspoikkeamat kuukausittain.

Yläpuolella esitetty funktio noutaa listan kuukausista sekä kuukausille kohdistuvan yhteen lasketun poikkeamien määrän. Kyselystä palautettava kuukauden numero muutetaan vielä ennen tiedon palauttamista sen suomenkieliseksi lyhenteeksi, jolloin kaavion X-akseliin saadaan suoraa kuukausien lyhennetyt nimet.

Syykoodit-kaaviossa esitellään asiakaspoikkeamien yhteen lasketut määrät syykoodeittain ryhmiteltynä. Projektin toimeksiantaja koki tärkeäksi pelkkien asiakaspoikkeamien syiden esittämisen näytöllä, koska yksi projektin päätavoitteista oli päästä kiinni näiden toistuvuuteen ja tätä kautta juurisyyn korjaamiseen. Pylväskaaviossa käyttäjä näkee kaikki kuluvan vuoden asiakaspoikkeamien syyt yhteenlaskettuna sisältäen myös poikkeamat, joille ei syykoodia ole valittuna. Graafin tiedot noudetaan laskemalla poikkeamien syykoodit yhteen alla olevan esitetyllä tavalla

```
[Route("api/GetCustomerComplaintCountByReasonCode")]
[HttpGet]
public IActionResult GetCustomerComplaintCountByReasonCode()
{
    var firstDayOfYear = new DateTime(DateTime.Now.Year, 1, 1);
    var lastDayOfYear = new DateTime(DateTime.Now.Year, 12, 31);

    var customerComplaints = _context.CustomerComplaints
       .Where(a => a.ComplaintDate >= firstDayOfYear && a.ComplaintDate <= lastDayOfYear)
       .GroupBy(a => a.Reason.ReasonCode)
        .Select(a => new
        {
            name = a.Key ?? "Tyhjä",
            value = a.Count()

        });
    return Ok(customerComplaints);
}
```

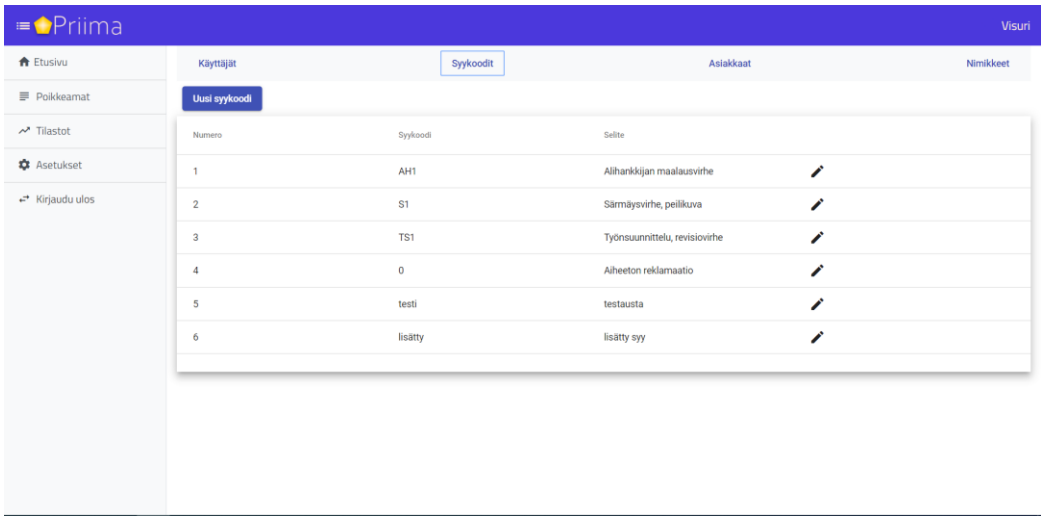
Kuvio 11. Asiakaspoikkeamat syykoodeittain.

Yläpuolella esitetty funktio laskee poikkeamilla olevat syykoodit yhteensä ja palauttaa listan niiden määristä. Kaikilla poikkeamilla ei välttämättä syykoodia kuitenkaan ole tallennettuna, joten syykoodittomat poikkeamat lasketaan myös yhteen ”Tyhjä”-otsikon alle.

Tilastot-näytön viimeisessä osiossa koostetaan yhteen poikkeamista aiheutuneita kuluja. Näytön korteissa käyttäjälle esitetään yhteenvetona sisäisten poikkeamien sekä asiakaspoikkeamien kuluja vuoden ja kuukauden alusta. Korttien vieressä kuluista koostetaan viivakaavio, joka summaa kuukausittaiset kulut ja jakaa ne kuukausille poikkeaman päivämäärien mukaisesti.

4.8 Asetukset

Asetukset-näyttö koostuu neljästä välilehdestä, jossa hallitaan käyttäjiä, syykoodeja, asiakkaita sekä tuotenimikkeitä. Käyttäjien ja syykoodien välilehdet muistuttavat pitkälti toisiaan. Uuden käyttäjän sekä syykoodin luontiin on luotu oma dialogi, jossa käyttäjälle annetaan käyttäjätunnus. Uuden käyttäjä luonnin jälkeen avautuu toinen dialogi, jossa tietoja voidaan muokata sekä käyttäjälle voidaan luoda salasana. Käyttäjille ja syykoodeille ei historiatietojen säilyttämisen vuoksi ole poisto-ominaisuutta tehty, jolloin käyttäjän poiston yhteydessä käyttäjä katoaisi kaikilta vanhoilta poikkeamilta. Käyttäjä voidaan kuitenkin muuttaa passiiviseksi, jolloin se katoaa käsittelijöiden listauksesta.



Kuvio 12. Asetukset.

Asiakkaiden sekä nimikkeiden päivitys onnistuu suoraa Lemonsoft-toiminnanohjausjärjestelmästä. Päivitys suoritetaan hakemalla Lemonsoftista kaikki asiakkaat, joiden asiakasnumeroa ohjelmistosta ei vielä löydy. Alla olevassa kuvassa luodaan SQL-proseduuri, jota ohjelmistosta voidaan kutsua nimikkeiden päivittämiseksi.

```
IF EXISTS (
    SELECT name
        FROM sys.objects
        WHERE object_id = OBJECT_ID(N'dbo.upd_priima_products')
)
DROP PROCEDURE upd_priima_products;
GO

CREATE PROCEDURE upd_priima_products

AS

INSERT INTO PriimaDB.dbo.products (productCode, ProductDescription)
SELECT product_code, product_description
FROM LemonDB1.dbo.products
WHERE product_code NOT IN (SELECT productCode FROM PriimaDB.dbo.products)

GO
```

Kuvio 13. Nimikkeiden päivitys.

5 YHTEENVETO

Työn tavoitteena oli rakentaa laatupoikkeamien käsittelyjärjestelmä käyttäen Angular- ja ASP.NET Core -ohjelmistokehyksiä sekä muita ohjelmistoteknologioita. Työssä käytiin läpi keskeisimpiä ohjelmiston rakennukseen käyttämiäni tekniikoita, ja havainnollistin valmista työtä kuvien ja kaavioiden muodossa. Työ antaa kuvan full stack -ohjelmistokokonaisuudesta, jolla sivuston käyttöliittymä ja palvelinratkaisu on luotu ja kuinka ne toimivat yhdessä.

Työn tutkimusongelmiin löydettiin ratkaisu ohjelmistossa käytettyjen teknologioiden kautta. Valmistuneen ohjelmiston Etusivulla esitetään käyttäjälle kohdistetut työt Angular Material -komponenttikirjaston avulla sekä poikkeamien määriä Bootstrap-ohjelmistokehyksen korttien avulla.

Poikkeamat-näytöllä ilmenneet tutkimusongelmat, kuten suurten listojen käsittely ratkaistiin listojen sivutuksella. Poikkeamalistoilla näytetään tieto sivutetusti (15 poikkeamaa yhtä sivua kohden), jolloin palvelimelta noudetaan kerralla vain yhden sivun poikkeamat. Käyttäjän vaihtaessa sivua sivun numerosta voidaan suoraa laskea seuraavat 15 noudettavaa riviä taulukkoon.

Laatupoikkeama-näytön tutkimusongelma, jossa asiakkaiden lataus pudotusvalikon taakse toteutettiin vaihtoehtoisella tavalla. Kun ohjelmiston otetaan tuotantokäyttöön, asiakas-pudotusvalikon taakse täytyisi noutaa tuhansia asiakkaita, jolloin näytön latausaika hidastuisi monin kertaisesti. Ongelma onnistuttiin korjaamaan muuttamalla asiakkaan vaihdos omaan dialogiinsa, jonka pudotusvalikkoon asiakkaat noudetaan arvaavan haun perusteella. Dialogissa käyttäjä alkaa kirjoittamaan hakuun asiakkaan nimeä, jolloin kirjoituksen pysähtyessä kentän alapuolelle ilmestyy 10 lähimpänä olevaa hakutulosta, joista käyttäjä voi valita asiakkaan poikkeamalle.

Tilastot-näytöllä tutkimusongelmaksi nousi tiedon syöttö kaavioihin. Poikkeamia laskettaessa yhteen saattoi vuoden aikana olla kuukausia, joilla ei poikkeamia ollut syntynyt. Tämä aiheutti käyttöliittymässä kuukauden katoamisen X-akselilta

kokonaan. Ratkaisuna ongelmaan tietokantakyselyyn tehtiin lista kuukausista, jolloin poikkeamat laskettiin niiden päiväyksen mukaisesti kuukausitaulukkoon. Tämä mahdollista myös tyhjien kuukausien esittämisen kaaviossa.

Vaikka työ saatiinkin valmiiksi ja käyttöönotto on alkamassa, voidaan ohjelmistosta silti jälkikäteen huomata pieniä vikoja. Ohjelmisto on luotu käyttäen näytön kokoon skaalautuvia elementtejä, mutta ohjelmiston toiminta mobiililaitteella ei silti ole virheetön. Tekstien rivitys mobiililaitteilla vaatisi vielä hiomista kuten myös kaavioiden kuvasuhde. Mobiilinäkymän viat eivät kuitenkaan vaikuta projektin käyttöön tuotantoympäristössä, koska se tullaan asentamaan yrityksen omalle palvelimelle sisäiseen verkkoon, josta siihen päästään käsiksi vain tietokoneilla.

Tulevaisuuden kehityskohde ohjelmistoon olisi sen automatisointi. Tämä vaatisi oman palvelimelle asennettavan palvelun, joka kirjaisi sähköpostiin tulleet poikkeamat automaattisesti ohjelmaan. Palvelu voisi sisältää myös automaattisen muistutusviestien lähetyksen, jolloin poikkeaman käsittelijä saisi aina viestin, kun uusi poikkeama ilmestyy hänen työlistalleen.

Opinnäytetyö oli kokonaisuutena haastava ja suuri, mutta kuitenkin yhden henkilön toteutettavissa. Ohjelmistokokonaisuuden luonti tyhjästä oli mielenkiintoinen prosessi, mutta se vaati paljon tutkimista eri tekniikoista ennen varsinaisen ohjelmoinnin aloittamista.

Projektin lopputulos oli mielestäni onnistunut. Toimeksiantaja ottaa uuden ohjelmiston virallisesti käyttöön 2020 kesäkuun aikana siirtäen alkuvuoden materiaalin Excelistä ohjelmaan. Ohjelmiston kehitys loi hyvän tietopohjan uusimmista teknologioista sekä kasvatti ohjelmistokehitystaitojani erittäin paljon. Ohjelmisto on myös osana toimeksiantajan kehitysprojektia, johon he saivat Business Finlandilta tukea 100 000 €.

LÄHTEET

Angular 2020. In-app navigation: routing to views. Viitattu 29.3.2020. https://angular.io/guide/router

Bootstrap 2020. About. Viitattu 12.2.2020. https://getbootstrap.com/docs/4.4/about/overview/

Chand, M. 2020. What Is C#. Viitattu 5.3.2020. https://www.c-sharpcorner.com/article/what-is-c-sharp/

Entity Framework Tutorial. 2020. What is Entity Framework? Viitattu 5.3.2020. https://www.entityframeworktutorial.net/what-is-entityframework.aspx

Fain, Y. & Moiseev, A. 2020. TypeScript Quickly. Viitattu 4.4.2020. https://learning.oreilly.com/library/view/typescript-quickly/9781617295942/kindle_split_009.html. o'Reilly.

Fain, Y. & Moiseev, A. 2018a. Angular Development with Typescript, Second Edition. Chapter 1. Viitattu 4.4.2020. https://learning.oreilly.com/library/view/angular-development-with/9781617295348/kindle_split_010.html. o'Reilly.

Fain, Y. & Moiseev, A. 2018b. Angular Development with Typescript, Second Edition. Chapter 2. Viitattu 4.4.2020. https://learning.oreilly.com/library/view/angular-development-with/9781617295348/kindle_split_011.html. o'Reilly.

JWT. 2020. Introduction to JSON Web Tokens. Viitattu 1.5.2020. https://jwt.io/introduction/

Material Design 2020. Viitattu 2.4.2020. https://material.io/

Node.js 2011. What is npm? Viitattu 4.5.2020. https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/

Peres, R. 2017. Mastering ASP.NET Core 2.0. Viitattu 15.3.2020. https://learning.oreilly.com/library/view/mastering-aspnet-core/9781787283688/3fdb7ebf-e1c9-4548-b80b-fd0e53d4e583.xhtml. o'Reilly.

VAMM Steel Oy. 2020. Viitattu 8.4.2020. http://vammsteel.fi/

Visuri, M. 2020. Laatupäällikkö. VAMM Steel Oy. Haastattelu 3.1.2020