Niko Sillvan

VERKKOSOVELLUS ÄLYKKÄÄSEEN LAITEOHJAUKSEEN

Tietojenkäsittelyn koulutusohjelma

2016

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

VERKKOSOVELLUS ÄLYKKÄÄSEEN LAITEOHJAUKSEEN

Sillvan, Niko
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Toukokuu 2016
Ohjaaja: Nieminen, Hans
Sivumäärä: 27

Asiasanat: ASP.NET, C#, esineiden internet, IoT

Opinnäytetyön toiminnallinen osuus toteutettiin Satakunnan ammattikorkeakoulun hankkeena vuoden 2014 aikana. Hankkeen tarkoituksena oli suunnitella ja toteuttaa IoT:n inspiroima verkkosovellus älykkääseen laiteohjaukseen. IoT on käsitteenä ajankohtainen, ja sitä pidetään internetin seuraavana evoluutiona.

Toteutettu verkkosovellus toimii portaalina IP-pistorasioihin kytkettyjen laitteiden etäohjaukseen. Laitteita voidaan ohjata langattomaan verkkoon yhdistettyjen tietokoneiden, tablettien ja älypuhelimien välityksellä. Sovellus toimii palveluperustaisesti, mikä tarkoittaa sitä, että käyttäjä voi valita haluamansa palvelut laitekohtaisesti.

Yksi palveluista on tarkoitettu moottoriajoneuvon esilämmityksen älykkääseen ajastukseen. Sovellus käyttää hyväkseen tuoreita säähavaintoja, joiden perusteella se pyrkii esilämmittämään ajastettua ajoneuvoa energiatehokkaasti vain tarpeen mukaan.

Tässä opinnäytetyössä esitellään lyhyesti, mistä IoT:ssa on kyse, käydään läpi sovelluksen kehityksessä käytettyjä työkaluja ja tekniikoita sekä esitellään toteutettu sovellus.

WEB APPLICATION FOR SMART CONTROL OF DEVICES

Sillvan, Niko
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Systems
May 2016
Supervisor: Nieminen, Hans
Number of pages: 27

Keywords: ASP.NET, C#, internet of things, IoT

The implementation of this thesis was commissioned by Satakunta University of Applied Sciences and it was implemented during the year 2014. The purpose of this thesis was to design and create an IoT-inspired web application for smart control of devices. The concept of IoT is currently a hot topic and it is said to be the next evolution of the internet.

The implemented web application acts as a portal for remote controlling the devices connected to IP sockets. The connected devices can be controlled by computers, tablets and smart phones connected to the wireless network. The application is based on services - meaning that the user can choose the services on demand for the selected devices.

The purpose of one of the services is to set a smart timer for pre-heating a motor vehicle. The application strives to pre-heat a motor vehicle energy efficiently based on the recent weather observations.

This thesis briefly describes the concept of IoT. After that we go through different tools and technologies used in the development of the application and finally the implemented application itself is showcased.

SISÄLLYS

TERMIT JA LYHENTEET

IPv6
Nykyisen IP-protokollan (IPv4) seuraajaksi kehitetty protokolla

MEMS
Microelectromechanical systems

OID
Object Identifier

Porttaus
Ohjelman lähdekoodin muunto alustalta toiselle

RFID
Radiotaajuinen etätunnistus

SNMP
Simple Network Management Protocol

UDP
User Datagram Protocol

XML
Extensible Markup Language

1 JOHDANTO

Opinnäytetyöni toiminnallisen osuuden projektin lähtökohtana oli suunnitella ja toteuttaa Internet of Things -käsitteen inspiroimana prototyyppi sovelluksesta, joka hyödyntää kolmannen osapuolen dataa sähkölaitteiden älykkääseen ohjaukseen. Lähtökohtien puitteissa toteutettiin palveluihin perustuva verkkosovellus, jonka kautta voidaan ohjata IP-pistorasioihin kytkettyjä laitteita SNMP-protokollan avulla. Sovellukseen kirjautunut henkilö voi lisätä laiteprofiileita ja liittää niihin haluamansa palvelut.

Palveluita toteutettiin kaksi (2). Ensimmäinen palveluista mahdollistaa laitteen päälle ja pois kytkemisen laiteohjaussivun kautta. Toinen palveluista mahdollistaa laiteohjaussivun kautta tapahtuvan ajastetun laitteen käynnistyksen Ilmatieteen laitoksen avoin data -rajapintaa hyödyntäen. Ohjaamisen älykkyys perustuu saatavilla oleviin säätietoihin. Palvelun käyttötarkoituksena voisi olla esimerkiksi ajoneuvon lämmityksen energiaa säästävä ajastus. Yleisellä tasolla palvelu pyrkii helpottamaan arkea ja säästämään energiaa.

Opinnäytetyöraportin alussa tutustutaan lyhyesti Internet of Things –käsitteeseen, sen ajankohtaisuuteen ja haasteisiin. Tämän jälkeen käydään läpi sovelluksen kehityksessä käytettyjä työkaluja ja tekniikoita sekä esimerkein niiden hyödyntämistä sovelluksessa. Seuraavaksi tarkastellaan kokonaisuudessaan projektin sovellusta. Viimeinen luku keskittyy projektin jälkipohdintaan.

Projekti liittyy Satakunnan ammattikorkeakoulun hankkeeseen, jonka yhteydessä sovellus toteutettiin. Projekti alkoi toukokuussa 2014 ja valmistui joulukuussa 2014.

2 INTERNET OF THINGS

Internet of Things (IoT) on käsite, joka hakee vielä muotoaan. Standardia, joka määrittelisi IoT:n, ei ole vielä olemassa, ja käsitys IoT:n määritelmästä riippuu hyvin pitkälti tulkitsijatahosta. Voidaan kuitenkin todeta, että IoT:n ajatellaan koostuvan internetin yli ulottuvien antureilla ja tunnistimilla varusteltujen laitteiden verkoista.

IoT:n ajankohtaisuus on korostunut muutamasta eri syystä. Vuonna 2012 esitelty IPv6-protokolla laajensi yksilöllisten IP-osoitteiden osoiteavaruutta mahdollistaen biljoonien fyysisten esineiden ja laitteiden verkostoa. RFID-tunnistimien sekä mikrosysteemeissä (MEMS) käytettyjen antureiden – kuten kiihtyvyysantureiden, gyroskooppien ja paineanturien – hinta on romahtanut viimeisen viiden (5) vuoden aikana.

Pilvipalveluiden käytön kasvu mahdollistaa esineiden antureista ja tunnistimista saadun datatulvan säilytyksen. Kun nämä yhdistetään tuoreimpiin analyyttisiin työkaluihin ja korkean suorituskyvyn tietokoneisiin, IoT:n saaman huomion ajankohtaisuus alkaa hahmottua.

Haasteita IoT:n laajempaan käyttöönottoon tuo teollisten standardien epäkypsyys. ”Tarvitsemme standardeja eri anturien ja tunnistimien keskustelun helpottamiseksi sekä avoimia arkkitehtuureja, jotka mahdollistavat ja sopeuttavat eri sovellukset.”, toteaa IEEE:n vanhempi jäsen Chonggang Wang. (IEEE 2016).

3 KÄYTETYT TYÖKALUT JA TEKNIIKAT

3.1 Microsoft Visual Studio

Microsoft Visual Studio on Microsoftin ohjelmointikehitysympäristö, joka tarjoaa kattavat työkalut ohjelmistokehitykseen. Visual Studiolla on laaja tuki eri ohjelmointikielille. C#, Visual Basic, F#, C++, HTML, JavaScript ja Python ovat tuettuina suoraan

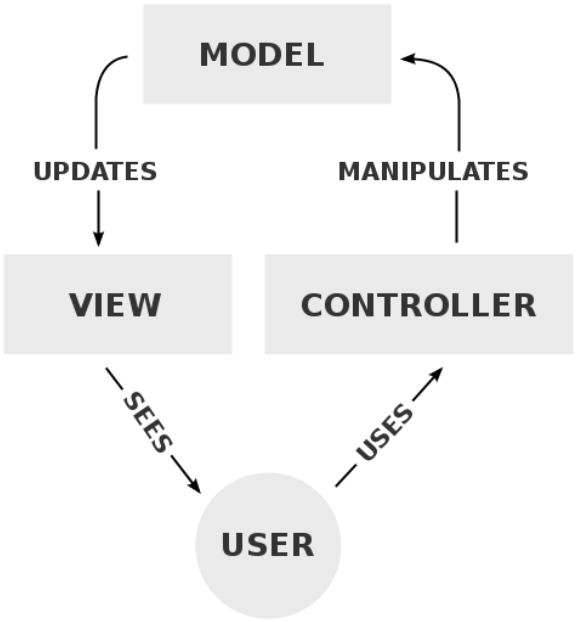
valinnaisina asennuksen yhteydessä kuten myös monet muut ohjelmointikielet laajennusten kautta. (Microsoft 2016A)

Visual Studiolla voi kehittää alustariippumattomia tietokoneohjelmia, verkkosivuja, verkkosovelluksia sekä verkkopalveluja. Visual Studion lähdekoodieditori tukee IntelliSenseä. Kyseessä on Microsoftin toteutus, joka sisältää mm. lähdekoodin automaattisen täydennyksen sekä syntaksin tarkastelun. Visual Studion virheenkorjaaja (debugger) pystyy tarkastelemaan virheitä sekä lähdekooditasolla että konekielitasolla. (Wikipedia 2016A)

Sovellukseni kehitykseen käytin Microsoft Visual Studio 2013 Professional –versiota. Koko ohjelmointiin liittyvä kehitysprosessi tapahtui Visual Studiossa, joka sisälsi kaikki tarvitsemani sovelluskehitysominaisuudet.

3.2 ASP.NET MVC

ASP.NET MVC on Microsoftin kehittämä Model-View-Controller (Kuva 1) –arkkitehtuuriin perustuva verkkosovellyskehys, joka käyttää Microsoftin .NET Frameworkin komponenttikirjastoa. MVC:n tarkoituksena on jakaa sovellus kolmeen eri loogiseen komponenttiin: Malliin (model), näkymään (view) ja käsittelijään (controller). Model edustaa liiketoimintalogiikkaa. Sitä voi pitää eräänlaisena järjestelmän tietokantakuvauksena. Controllerin tehtävänä on vastaanottaa käyttäjän syötteet ja käskyt ja näiden pohjalta muokata modelia. Controller vastaa myös päivittyneiden tietojen välityksen viewille. View huolehtii käyttöliittymän ulkoasun ja tietojen esittämisestä käyttäjälle. (Microsoft 2016B)



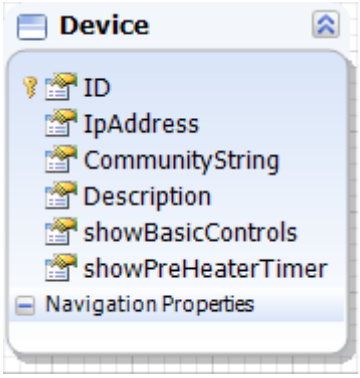
Kuva 1. Model-View-Controller (Wikipedia 2016)

Valitsin ASP.NET MVC verkkosovelluskehyksen projektin alustaksi pääosin siitä syystä, että se oli minulle jo ennalta tuttu ympäristö ja tiesin sen soveltuvan erinomaisesti projektille.

3.3 SQL Server Express

SQL Server Express on Microsoftin tarjoama ilmainen relaatiotietokantajärjestelmä. Käytin projektin sovelluksessa LocalDB:tä, joka on kevytversio SQL Server Expressin tietokantajärjestelmästä. LocalDB vaatii vain vähäistä asetusten muokkausta, ja sen voi sulauttaa sovellukseen, joka tarvitsee paikallisen tietokannan. Tästä syystä LocalDB oli ilmiselvä valinta sovelluksen tietokannaksi. (Microsoft 2016C)

Käytin sovelluksessa eroteltuina kahta LocalDB-tietokantaa. Toinen tietokannoista sisältää ASP.NET Identity -sovelluskehyksen tarjoamat käyttäjän identiteettiin liittyvät tietokantataulut. Toinen LocalDB-tietokanta sisältää ainoastaan yhden taulun (Kuva 2) laitteita varten.

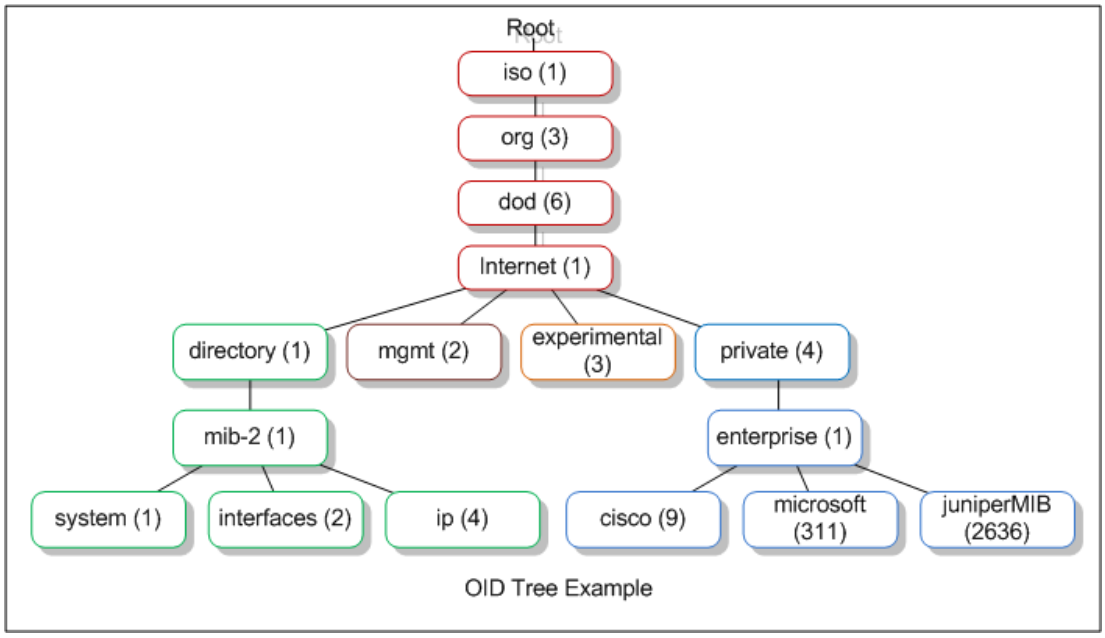


Kuva 2. Device-tietokantataulu

3.4 SNMP

SNMP on tietoliikenneprotokolla, joka kuuluu TCP/IP-protokollaperheen sovelluskerrokseen. SNMP-protokollaa käytetään verkon laitteiden seurantaan mm. ongelmien paikantamisessa. Protokollan avulla laitteista saadaan haettua tilatietoja ja laitteiden tilaa päästään myös muokkaamaan. Tyypillisiä SNMP-protokollaa tukevia laitteita ovat mm. reitittimet, kytkimet, palvelimet, työasemat, tulostimet ja IP-pistorasiat. SNMP on yhteydetön protokolla ja se perustuu UDP-protokollaan. (Wikipedia 2016C)

SNMP toimii oliokyselyillä. Olio edustaa jotain arvoa; esimerkiksi, onko laite päällä vai ei. Kun tehdään kysely laitteen tila –oliolle, palautetaan vastauksena muuttuja, joka kertoo kysyjälle, onko laite päällä vai ei. OID edustaa käytännössä olion osoitetta puumaisessa OID-hierarkiassa (Kuva 3). (Leskiw 2016)



Kuva 3. Esimerkki OID-hierarkiasta (Leskiw 2016)

Projektin sovellus käyttää SnmpSoftin SnmpSet–nimistä komentokehotesovellusta. Kun käyttäjä käyttää projektin sovelluksen tarjoamia perusohjaimia laiteohjaukseen (Kuva 4), kutsutaan controllerin metodia kuvassa 5. Tämä kutsuu edelleen metodia (Kuva 6), jonka tehtävänä on ajaa piilotettuna prosessina SnmpSet –sovellusta.

```
@model SNMPDemo.Models.Device

<h4>Basic controls</h4>
<p>
    @using (Html.BeginForm())
    {
        <input type="button" value="Turn On" class="btn btn-default" name="btn"
               onclick="Turn('@ViewBag.IP', '@ViewBag.CommunityString', 'On')" />
        <input type="button" value="Turn Off" class="btn btn-default" name="btn"
               onclick="Turn('@ViewBag.IP', '@ViewBag.CommunityString', 'Off')" />
    }
</p>
```

Kuva 4. Esimerkki Basic Controls –palvelun ohjainpainikkeista

```
[HttpPost]
public ActionResult _BasicControls(string Ip, string CommunityString, string Action)
{
    SNMPSet.SendSet(Ip, CommunityString, Action);
    return PartialView();
}
```

Kuva 5. Esimerkki controllerin metodista, joka ajetaan, kun ohjainpainikkeita käytetään.

```
public static void SendSet(string ip, string communitystring, string action)
{
    string val;

    if (action == "On")
        val = "1";
    else
        val = "0";

    System.Diagnostics.Process process = new System.Diagnostics.Process();
    System.Diagnostics.ProcessStartInfo startInfo = new System.Diagnostics.ProcessStartInfo();
    startInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
    startInfo.FileName = @"C:\Users\IoT\Downloads\1_SNMPDemo-master\SNMPDemo-master\"
                         +@"SNMPDemo\Resources\SnmpSet.exe";

    startInfo.Arguments = @"/C SnmpSet.exe -r:192.168.0.11 -c:"""
                         + communitystring
                         + @""" -o:0.1.3.6.1.4.1.21287.16.1.0 -val:"
                         + val;

    process.StartInfo = startInfo;
    process.Start();
}
```

Kuva 6. Esimerkki metodista, joka muokkaa laitteen tilaa.

SnmpSet ohjelmalle välitetään seuraavat parametrit:

- -r: IP-pistorasian IP-osoite, johon laite on kytketty
- -c: community string eli laitteen salasana
- -o: muuttujan OID, jota halutaan muokata
- -val: muuttujan arvo 1 asettaa laitteen päälle, ja arvo 0 asettaa laitteen pois päältä

3.5 Quartz.NET

Quartz.NET on C#:lla kirjoitettu avoimen lähdekoodin .NET Framework lähdekoodikirjasto. Quartz.NET on porttaus suositusta Quartz–nimisestä tehtävien ajastus –sovelluskehyksestä, joka on kirjoitettu Javalla. Quartz.NET on vikasietoinen, mikä tarkoittaa käytännössä sitä, että se muistaa ajastetut tehtävät, vaikka järjestelmä käynnistettäisiin uudelleen.

Ajastettujen tehtävien suorittamiseen liittyy kaksi termiä: laukaisin (Trigger) ja tehtävä (Job). Tehtävä suoritetaan aina, kun laukaisimeen liitetty ehto käy toteen. Tehtävät voidaan ajastaa suoritettavaksi esimerkiksi:

- Määriteltynä ajankohtana päivästä millisekunnin tarkkuudella.
- Määriteltyinä viikonpäivinä.
- Määriteltyinä päivinä kuukaudesta.
- Määriteltyinä päivinä vuodesta.
- Ikuisesti määritellyin väliajoin.
- Toistuvasti määriteltyyn ajankohtaan asti.
- Ikuisesti.

Tehtäville ja laukaisimille annetaan nimi, ja ne voidaan järjestää nimettyihin ryhmiin. Tehtävät voidaan ajastaa vain kerran, mutta niihin voidaan liittää useampi laukaisin. (Quartz.NET www-sivut, 2016)

Kuvassa 7 on esimerkki metodista, joka ajetaan ajastuksen asettamisen yhteydessä. Metodi vastaanottaa ajastetun ajoonlähtöajan sekä laitteen. Luodaan SetPreHeaterTimerJob –tyyppinen tehtävä, joka asetetaan laukeamaan 2,5 tuntia ennen ajoonlähtöaikaa. Tämän jälkeen tehtävä suoritetaan 15 minuutin välein maksimissaan kahdeksan (8) kertaa.

```
public static void SetTimer(string time, string DeviceId)
{
    SNMPDemoContext db = new SNMPDemoContext();
    Device dev = db.Devices.Find(Int32.Parse(DeviceId));

    string jobIdentity = "PreHeaterJob" + DeviceId;
    string triggerIdentity = "PreHeaterTrigger" + DeviceId;

    // Parse time
    DateTime dt = DateTime.ParseExact(time, "MM/dd/yyyy h:mm tt",
        CultureInfo.InvariantCulture);
    DateTimeOffset enddto = new DateTimeOffset(dt);
    DateTimeOffset startdto = enddto;
    startdto = startdto.AddHours(-2);
    startdto = startdto.AddMinutes(-30);

    // Build a job
    IJobDetail job = JobBuilder.Create<SetPreHeaterTimerJob>()
        .WithIdentity(jobIdentity, "group1")
        .UsingJobData("ip", dev.IpAddress)
        .UsingJobData("communityString", dev.CommunityString)
        .UsingJobData("deviceId", DeviceId)
        .UsingJobData("enddto", enddto.ToString())
        .Build();

    // Build a trigger
    ISimpleTrigger trigger = (ISimpleTrigger)TriggerBuilder.Create()
        .WithIdentity(triggerIdentity, "group1")
        .StartAt(startdto)
        .WithSimpleSchedule(x => x
            .WithIntervalInMinutes(15)
            .WithRepeatCount(8))
        .Build();

    // Schedule a job with the trigger
    MvcApplication.Scheduler.ScheduleJob(job, trigger);
}
```

Kuva 7. Esimerkki metodista, jota ajetaan ajastuksen asettamisen yhteydessä.

```
namespace Quartz
{
    public interface IJob
    {
        void Execute(JobExecutionContext context);
    }
}
```

Kuva 8. IJob-rajapinta

Suoritettavia tehtäviä voivat olla mitkä tahansa .NET Frameworkin luokat, jotka toteuttavat IJob-rajapinnan (Kuva 8). Kuvassa 9 näkyy esimerkki tehtävästä, joka tarkistaa ulkolämpötilan ja tekee sen pohjalta päätöksen, pistetäänkö laite päälle.

```
public void Execute(IJobExecutionContext context)
{
    JobKey key = context.JobDetail.Key;
    JobDataMap dataMap = context.JobDetail.JobDataMap;
    string ip = dataMap.GetString("ip");
    string communityString = dataMap.GetString("communityString");
    string enddtoString = dataMap.GetString("enddto");
    string deviceId = dataMap.GetString("deviceId");

    DateTimeOffset departureTime;
    departureTime = DateTimeOffset.Parse(enddtoString);

    double temperature = Double.Parse(GetTemperature(), CultureInfo.InvariantCulture);
    Debug.WriteLine(temperature);
    double hoursToWarm = 0.5 - (temperature * 0.1);

    TimeSpan hoursToDeparture = departureTime.Subtract(DateTimeOffset.Now);

    if (hoursToWarm >= hoursToDeparture.TotalHours)
    {
        Debug.WriteLine("Laite paalle!" + " hoursToWarm: " + hoursToWarm + " hoursToDeparture: "
            + hoursToDeparture.TotalHours + "  Kello on: " + DateTimeOffset.Now.ToString());
        SNMPSet.SendSet(ip, communityString, "On");
        MvcApplication.Scheduler.PauseJob(new JobKey("PreHeaterJob" + deviceId, "group1"));
        JobManager.TurnOffDevice(deviceId, departureTime);
    }
    else
        Debug.WriteLine("Odota..." + " hoursToWarm: " + hoursToWarm + " hoursToDeparture: "
            + hoursToDeparture.TotalHours + "  Kello on: " + DateTimeOffset.Now.ToString());

}
```

Kuva 9. Esimerkki metodista, jota ajetaan aina, kun laukaisimeen liitetyt ehdot käyvät toteen.

3.6 Bootstrap

Bootstrap on avoimen lähdekoodin käyttöliittymän toteutukseen tarkoitettu lähdekoodikirjasto, jota ylläpitää sen perustajaryhmä sekä pieni joukko sen ydinkehittäjiä. Bootstrap sisältää valmiita mallipohjia typografia, lomake, painike, navigaatio- ja muille käyttöliittymäkomponenteille. Bootstrap sisältää myös JavaScript-laajennuksia, ja sen päätarkoituksena on helpottaa dynaamisten verkkosivujen ja -sovellusten kehitystä. (Wikipedia 2016B)

Kuva 10. Esimerkki navigointikomponentista kapealla näytöllä.

Bootstrapin navigaatiokomponentti (Kuva 10) on hyvä esimerkki siitä, miten sen käyttö tuo responsiivisuutta käyttöliittymään. Kun näytön leveys käy liian kapeaksi,

piilotetaan navigaatiokomponentin linkit kuvakkeen alle, jota käyttämällä linkit avautuvat dynaamisesti eräänlaisena alasvetovalikkona.

Bootstrap on ASP.NET MCV -projektin mallipohjassa oletuksena käytössä. Tästä syystä oli luontevaa ja aikaa säästävää käyttää sitä käyttöliittymän ulkoasussa.

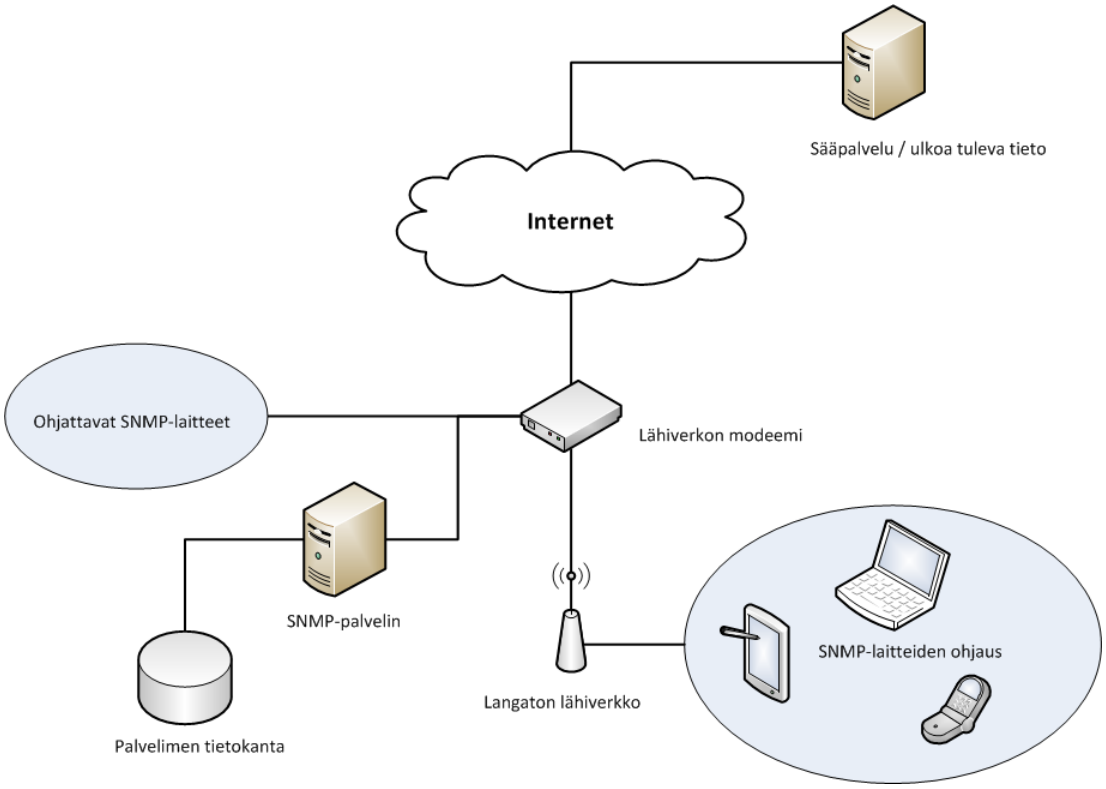
4 SOVELLUKSEN KUVAUS

Projektissa toteutettiin verkkosovellus, joka ohjaa IP-pistorasioihin kytkettyjä laitteita SNMP-protokollaa hyväksi käyttäen. Sovellus tarjoaa käyttäjälle WWW-käyttöliittymän, johon kirjautumalla käyttäjä pääsee luomaan laiteprofiileita ja liittämään niihin haluamansa laiteohjaukseen liittyvät palvelut.

4.1 Demoympäristö ja laitteet

Kaaviossa 1 näkyvät demoympäristön ja samalla kehitysympäristön laitteet ja rakenne. Demoympäristön keskiössä toimii palvelin, jossa sovellusta ajetaan. Sovellusta voidaan käyttää sekä suoraan palvelimelta että langattoman lähiverkon kautta. Sovellusta voi sujuvasti käyttää esimerkiksi pieniresoluutioisilla älypuhelimilla, mikäli selain ja tuki JavaScriptille löytyvät. Tätä tukee sovelluksen käyttöliittymän ulkoasussa käytetyn Bootstrap-lähdekoodikirjaston käyttö, joka muokkaa responsiivisesti sovelluksen käyttöliittymää päätelaitteen resoluution mukaan.

Ympäristön ohjattavina laitteina toimivat kaksi (2) pöytälamppua kytkettyinä IQtronicin IQSocket IQSW-IP10-F -sähköpistorasioihin.



Kaavio 1. Projektin demoympäristö (Hellsten & Sillvan 2014)

4.2 Sovelluksen palvelut

Laiteohjaus tapahtuu laiteprofiileihin liitettyjen palveluiden kautta. Sovellukseen toteutettujen palveluiden valinta perustui ajatukseen siitä, että käyttäjäryhmänä olisi kuluttaja.

Basic Controls –palvelu tarjoaa käyttäjälle perusohjauksen laitteen päälle ja pois kytkemiseksi. Smart Pre-Heater Timer –palvelu tarjoaa käyttäjälle älykkään, energiaa säästävän ajastetun laitteen käynnistyksen. Palvelu on tarkoitettu ajoneuvon esilämmittimen ajastukseen. Käyttäjä asettaa sovelluksen laiteohjaussivun kautta halutun ajoon lähtöajan. Sovellus hakee tuoreet säätiedot Ilmatieteenlaitoksen avoin data –rajapinnan kautta. Ensimmäinen datakysely suoritetaan 2,5 tuntia ennen asetettua ajoonlähtöaikaa, jonka jälkeen dataa haetaan 15 minuutin välein.

Jokaisen datakyselyn yhteydessä verrataan ulkolämpötilaa ja aikaintervallia ajastettuun ajoonlähtöaikaan. Mikäli laskettu lämmitysajan tarve on suurempi tai yhtä suuri kuin aikaintervalli ajastettuun lähtöaikaan, kytketään laite päälle ja luodaan ajastettu

tehtävä esilämmittimen pois päältä –kytkemiseksi viisi (5) minuuttia ennen ajastettua ajoon lähtöä. Jos lämmitysajan tarve on pienempi, tarkastetaan tilanne uudelleen 15 minuutin kuluttua.

Kuvaaja 1. Laskettu esilämmitysaika

Esilämmitysaika (Kuvaaja 1) perustuu karkeasti Motivan suosittelemiin esilämmitysaikoihin (Motiva Oy 2016), ja se lasketaan seuraavalla kaavalla:

*lämmitysaika = 0,5 - (ulkolämpötila * 0,1)*

4.3 Ilmatieteen laitoksen avoin data

Ilmatieteen laitos tarjoaa verkkopalvelun, jonka kautta voi maksutta hakea ja katsella Ilmatieteen laitoksen tuottamia tietoaineistoja. Palvelu vaatii rekisteröitymisen, jonka suoritettuaan käyttäjä saa palveluun tarvittavan tunnisteavaimen (fmi-apikey). Palvelu palauttaa kyselydataa INSPIRE-direktiivin mukaisen XML-tiedoston muodossa (Ilmatieteen laitos 2016).

```
public String GetTemperature()
{
    String query = "http://data.fmi.fi/fmi-apikey/9308aac9-b245-47fc-ac0a-aac240dd8a7e/wfs"
        + "?request=getFeature&storedquery_id=fmi::observations::weather::multipointcoverage"
        + "&place=pori&parameters=temperature";

    XDocument doc = XDocument.Load(query);
    XNamespace gml = "http://www.opengis.net/gml/3.2";

    String res = (String)
        (from e in doc.Descendants(gml + "doubleOrNilReasonTupleList")
         select e).First();

    String result = res.Trim().Split('\n').Last().Trim();

    return result;
}
```

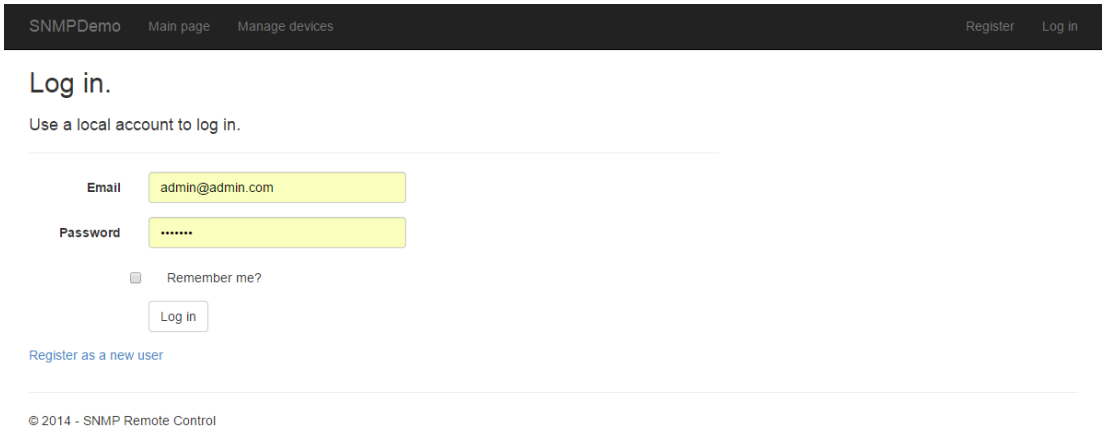
Kuva 11. Esimerkki tuoreimman Porin alueen lämpötilan hausta

Kuvassa 11 on esitelty sovellukseni metodi, joka hakee tuoreimman Porin alueen lämpötilasäähavainnon. Ilmatieteen laitoksen avoin data –palvelun tuoreimmat säähavainnot päivittyvät noin 10 minuutin välein. Tämän takia päädyin tekemään ajastetut kyselyt 15 minuutin välein. Alkuperäinen ajatus oli tarkistaa lämpötila 10 minuutin välein. Tämä olisi kuitenkin aiheuttanut sen, että pahimmassa tapauksessa sovellukseni olisi vastaanottanut tuoreimmat säätiedot vajaan 20 minuutin välein.

5 SOVELLUKSEN TOIMINNOT

5.1 Kirjautuminen

Sovelluksen etusivulle (Main Page) tultaessa ohjataan kirjautumaton käyttäjä suoraan kirjautumissivulle (Kuva 12). Etusivun lisäksi myös Manage Devices -linkkiä käyttämällä ohjataan kirjautumaton käyttäjä kirjautumissivulle. Kirjautumissivulle pääsee myös navigointipaneelin oikeasta yläkulmasta kohdasta Log in. Sovellus tukee usean käyttäjän rekisteröimistä, mutta kaikki käyttäjät jakavat samat laiteprofiilit. Kirjautumisen tarkoituksena on antaa käyttäjälle oikeudet käyttää sovelluksen tarjoamia palveluita: laiteprofiilien luontia, niiden muokkaamista ja laitteiden ohjausta.



Kuva 12. Sovellukseen kirjautuminen

5.2 Laitteiden ohjaus

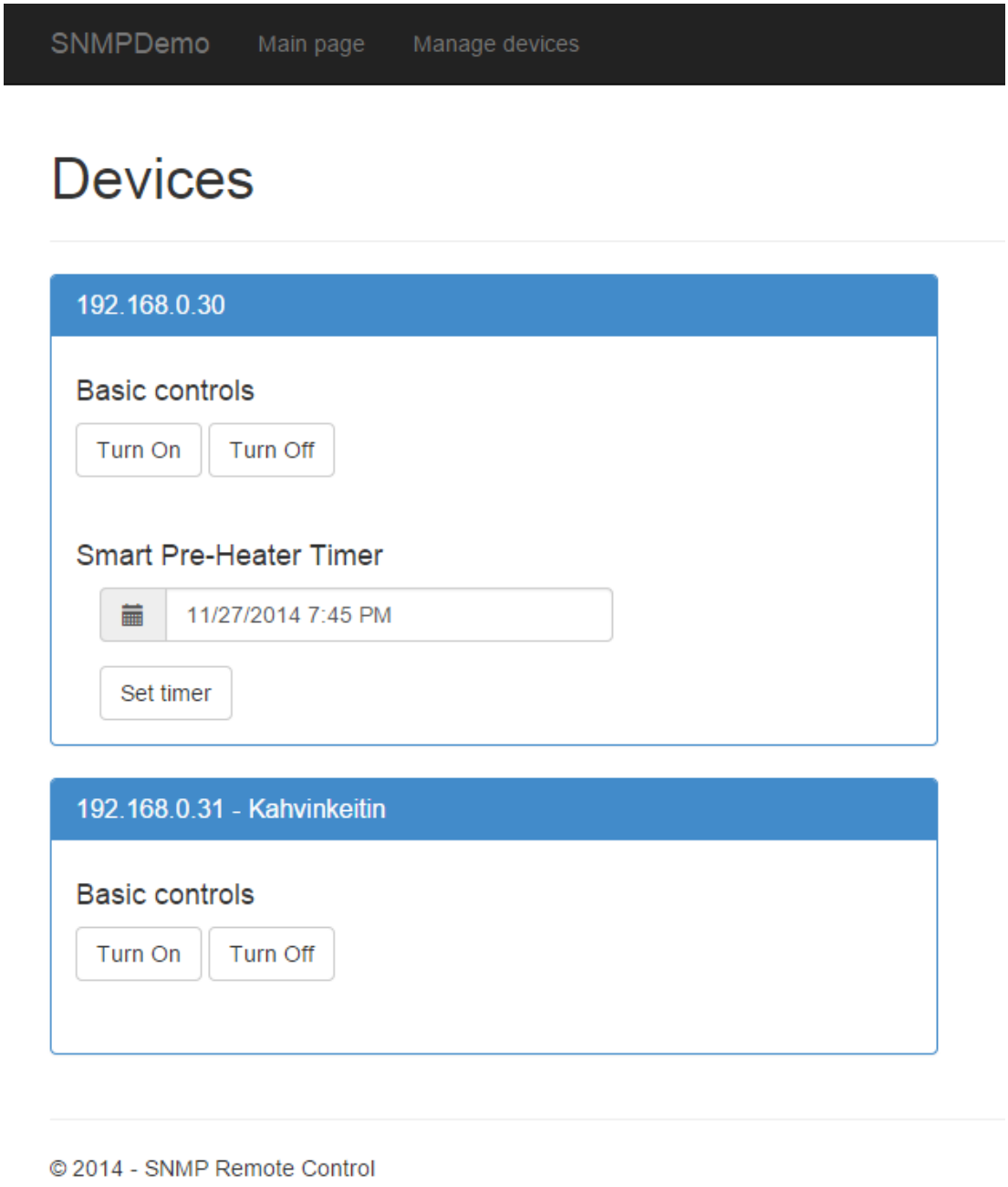
Laitteiden ohjaussivu (Kuva 13) tarjoaa käyttäjälle näkymän kaikkiin sovellukseen lisättyjen laitteiden palveluihin. Sivulle pääsee navigointipaneelin Main Page tai SNMPDemo -linkkien kautta. Laitteet on ryhmitelty allekkain, ja laitteiden palvelut on koottu sinireunaisiin laatikoihin. Sinireunaisen laatikon otsakeriviltä löytyy laitteen IP-osoite ja valinnaisesti annettu laitetta kuvaava nimi. Laatikon sisältä löytyy laiteprofiiliin liitetyt palvelut. Kuvan 13 yläosassa näkyvään laitteeseen on liitetty Basic Controls ja Smart Pre-Heater Timer -palvelu Alla olevaan laiteprofiiliin on liitetty ainoastaan Basic Controls –palvelu.

Basic Controls –palvelun Turn On kytkee laitteen päälle ja Turn Off kytkee laitteen pois päältä. Jos laite on jo päällä, kun Turn On -painiketta käytetään, ei tapahdu mitään. Sama koskee Turn Off -painiketta. Ohjainpainikkeet eivät ole tietoisia laitteen tilasta.

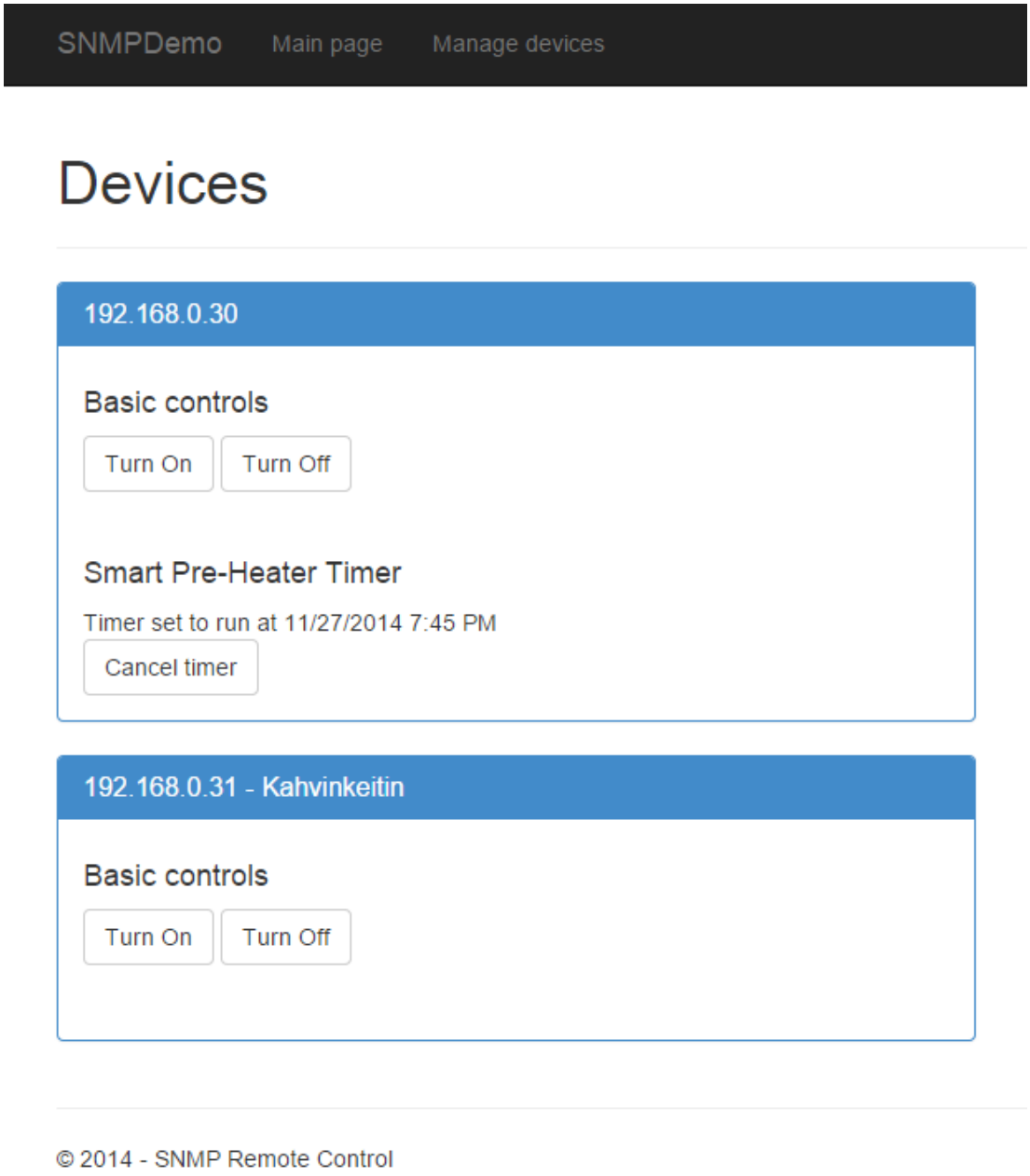
Smart Pre-Heater Timer –palvelu tarjoaa käyttäjälle mahdollisuuden valita haluttu ajastusaika kalenterikomponentista. Oletuspäivämääränä ja -aikana on nykyhetki. Set timer –painike asettaa laitteen ajastetun käynnistymisen tekstikentän ajan mukaan.

Kuvassa 14 on näkymä siitä, kun ajastus on asetettu. Asetettu aika toimii takarajana laitteen ajastetulle käynnistykselle. Tällöin näkyvissä on myös Cancel timer –painike, jota käyttämällä voidaan peruuttaa ajastus. Smart Pre-Heater Timer –palvelu ei Basic

Controls –palvelun tavoin ole tietoinen laitteen tilasta, mutta tunnistaa sen, jos laite on jo ajastettu. Laitteella voi olla vain yksi (1) aktiivinen ajastus kerrallaan.



Kuva 13. Laitteiden ohjaussivu (Main Page)

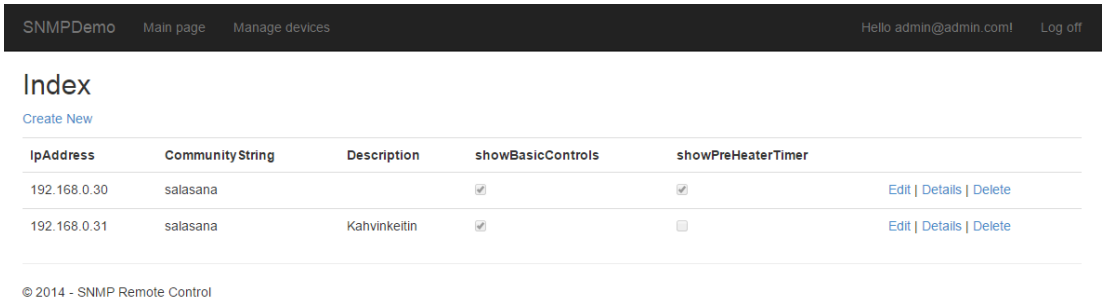


Kuva 14. Ajastettu laite

5.3 Laitteiden hallinta

Laitteiden hallintasivu (Kuva 15) tarjoaa käyttäjälle näkymän laiteprofiilien asetuksista. Sivulle pääsee navigointipaneelin Manage Devices –linkin kautta. Sivun päätarkoituksena on antaa käyttäjälle yleiskuva sovellukseen lisätyistä laiteprofiileista. Sovellukseen lisätyt laiteprofiilit on järjestelty allekkain. Create Now –linkin kautta käyttäjä pääsee lisäämään uuden laiteprofiilin. Jokaisen laiterivin oikeasta reunasta löytyy

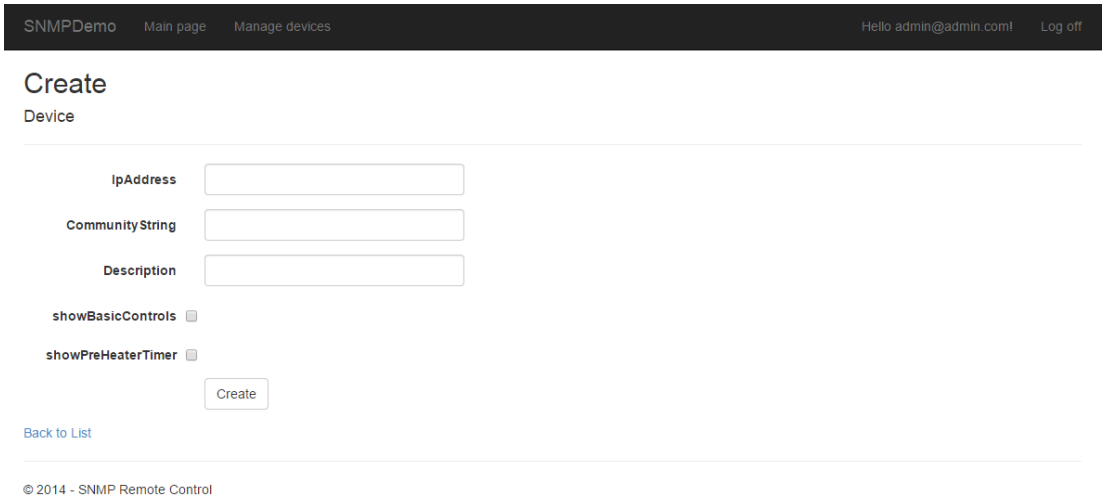
linkit laiteprofiilien muokkaukseen (Edit), yksittäiseen tarkasteluun (Details) sekä laiteprofiilin poistamiseen (Delete).



Kuva 15. Laitteiden hallintasivu (Manage Devices)

5.3.1 Laiteprofiilin lisääminen

Laiteprofiilien luomissivun (Kuva 16) kautta käyttäjä voi luoda uusia laiteprofiileita. Sivu tarjoaa käyttäjälle lomakkeen, johon käyttäjä syöttää laitetiedot sekä halutut laitteisiin liitettävät palvelut. Kun lomake on täytetty, käytetään Create-painiketta, joka lisää laiteprofiilin sovelluksen tietokantaan.

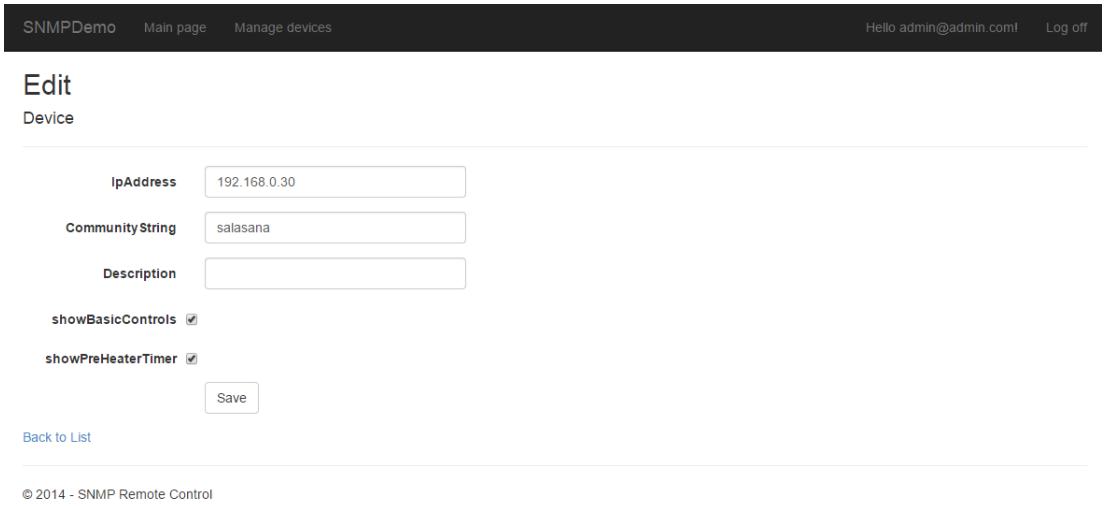


Kuva 16. Uuden laiteprofiilin luonti

5.3.2 Laiteprofiilin tietojen muokkaus

Laiteprofiilien tietojen muokkaussivun (Kuva 17) kautta käyttäjä voi muokata olemassa olevien laiteprofiilien tietoja. Sivu tarjoaa käyttäjälle olemassa olevien tietojen

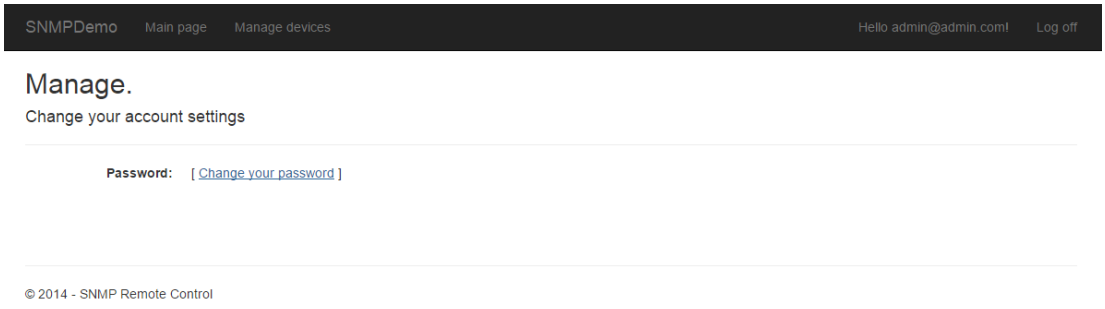
pohjalta esitäytetyn lomakkeen. Kun halutut muutokset laiteprofiiliin on tehty, voidaan muutokset tallentaa tietokantaan Save-painiketta käyttämällä.



Kuva 17. Olemassa olevan laiteprofiilin muokkaus

5.4 Käyttäjäprofiili

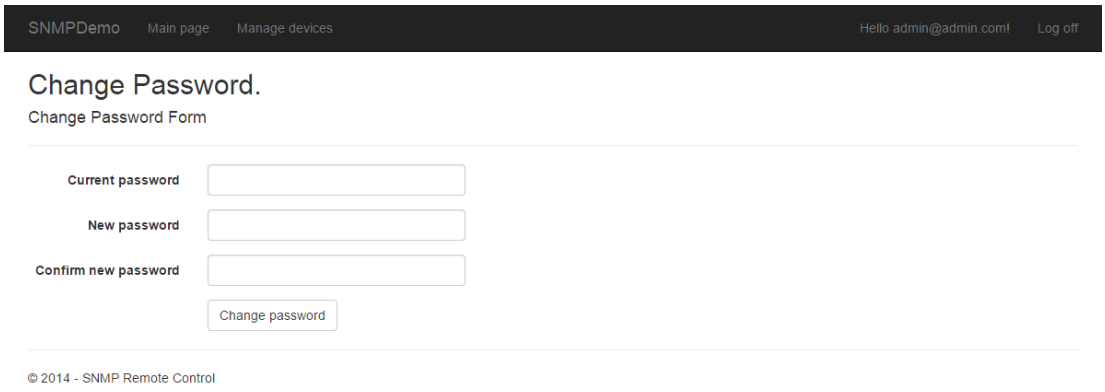
Käyttäjäprofiilin tarkastelusivun (Kuva 18) kautta käyttäjälle tarjotaan näkymä hänen omasta käyttäjäprofiilistaan. Käyttäjäprofiilin ainoa muokattavana oleva asetus liittyy käyttäjän salasanaan. Salasanan käyttäjä pääsee vaihtamaan Change your password –linkin kautta.



Kuva 18. Käyttäjäprofiilin tarkastelu

5.5 Salasanan vaihto

Salasanan vaihtosivun (Kuva 19) kautta käyttäjä pääsee muokkaamaan salasanaansa. Sivu tarjoaa tähän tarkoitukseen käyttäjälle lomakkeen, jonka hän täyttää Change password –painiketta käyttämällä. Salasanan muutokset tallentuvat tämän jälkeen tietokantaan.



Kuva 19. Salasanan vaihto

6 YHTEENVETO

Projektin päättymisestä alkaa olla aikaa kohta 1,5 vuotta. Minulla on ollut aikaa miettiä jälkikäteen, mikä meni hyvin ja mikä huonosti. Päällimmäisinä ajatuksina mielessäni ovat olleet sovellukseen liittyvät jatkokehityksen parannukset.

Sovellus tukee usean käyttäjän rekisteröintiä, mutta ei kuitenkaan yksilöllisiä laiteprofiileita. Tällä ei projektin tavoitteisiin nähden ole juurikaan vaikutusta, mutta se olisi joka tapauksessa askel kohti valmiimpaa tuotetta. Sovelluksen palveluista olisi voinut tehdä dynaamisempia esimerkiksi tekemällä palveluiden ohjainpainikkeista tilatietoisia. Tällöin olisi mahdollista saada selville laitteiden tila jo näkemällä ohjainpainikkeet.

Skaalautuvuutta ja palveluiden muokattavuutta ajatellen olisin voinut erotella ne omiksi tietokantatauluikseen. Se olisi mahdollistanut esimerkiksi laitekohtaiset tallennetut asetukset. Valmiille tuotteelle tämä olisi välttämätöntä. Käyttäjä pääsisi esimerkiksi muokkaamaan omaa sijaintiaan. Pidemmälle vietynä sijainnin tunnistuksenkin voisi automatisoida.

Palveluiden suunnittelussakin vain mielikuvitus on rajana. Eräänä ideana oli luokkahuoneen ilmanvaihdon säätö luennoille osallistuvien oppilaiden arvioidun lukumäärän perusteella. Tämä tieto ei suoranaisesti ollut saatavilla, mutta perusidea on sama – vastaanotetaan ulkopuolelta dataa jonka perusteella laiteohjaus tapahtuu.

Henkilökohtainen kohokohta projektissa oli, kun sain ensimmäisen kerran muokattua laitteiden tilaa sovelluksen kautta. Toinen vastaavanlainen onnistumisen tunne tuli siitä, kun yölliset testit ajastetusta laitteesta onnistuivat. Projekti päättyi joulukuussa 2014, ja totesimme saavuttaneemme sille asettamamme tavoitteet.

LÄHTEET

IEEE. 2016. Smarter Sensors. Viitattu 14.5.2016. http://theinstitute.ieee.org/technology-focus/technology-topic/smarter-sensors

Ilmatieteen laitos. 2016. Ilmatieteen laitoksen avoin data. Viitattu 13.5.2016. https://ilmatieteenlaitos.fi/avoin-data

Leskiw, A 2016 SNMP Tutorial Part2: Rounding Out the Basics. Viitattu 13.5.2016. http://www.networkmanagementsoftware.com/snmp-tutorial-part-2-rounding-out-the-basics/

Microsoft. 2016A. Visual Studio Community. Viitattu 13.5.2016. https://www.visualstudio.com/products/visual-studio-community-vs

Microsoft. 2016B. ASP.NET MVC Overview. Viitattu 13.5.2016. https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

Microsoft 2016C. Microsoft SQL Server 2014 Express. Viitattu 13.5.2016. https://www.microsoft.com/en-us/download/details.aspx?id=42299

Motiva Oy. 2016. Moottorin esilämmitys. Viitattu 12.5.2016. http://www.motiva.fi/moottorinesilammitys

Quartz.NET www-sivut. 2016. Viitattu 13.5.2016. http://www.quartz-scheduler.net/

Wikipedia. 2016A. Microsoft Visual Studio. Viitattu 13.5.2016. https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

Wikipedia 2016B. Bootstrap (front-end framework). Viitattu 13.5.2016. https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

Wikipedia 2016C. Simple Network Management Protocol. Viitattu 13.5.2016. https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

Kuva 1. Wikipedia 2016. Model-view-controller. Viitattu 13.5.2016. https://en.wikipedia.org/wiki/Model-view-controller

Kuva 3. Leskiw, A. 2016. SNMP Tutorial Part2: Rounding Out the Basics. Viitattu 13.5.2016. http://www.networkmanagementsoftware.com/snmp-tutorial-part-2-rounding-out-the-basics/