OULUN AMMATTIKORKEAKOULU

Ville Riekki

SÄHKÖNKULUTUSTA SEURAAVA MOBIILISOVELLUS

SÄHKÖNKULUTUSTA SEURAAVA MOBIILISOVELLUS

Ville Riekki
Opinnäytetyö
Kevät 2023
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä(t): Ville Riekki
Opinnäytetyön nimi: Sähkönkulutusta seuraava mobiilisovellus
Työn ohjaaja(t): Eino Niemi
Työn valmistumislukukausi ja -vuosi: Kevät 2023 Sivumäärä: 27

Opinnäytetyön tavoitteena oli toteuttaa Android-käyttöjärjestelmälle mobiilisovellus, joka helpottaa sähkönkulutuksen ja tulevaisuuden sähkön hintojen seuraamista. Mobiilisovelluksen lisäksi projektiin kuului ohjelma, joka haki sähkönkulutusta mittaavan mittarin dataa ja tulevaisuuden sähkön hintoja APIsta sekä tallensi tiedot Firebasen reaaliaikaiseen tietokantaan.

Työ aloitettiin tutustumalla sähkönkulutusta ja sähkön hintaa seuraaviin palveluihin, jonka jälkeen suunniteltiin ohjelmat sekä datan hakemiseen, tallentamiseen ja mobiilisovellus datan näyttämiseen. Mobiilisovellus, jota käytettiin datan näyttämiseen, tehtiin käyttäen Flutter-kehityspakettia Visual Studio Code -kehitysympäristössä. Flutter-sovelluksissa ohjelmointikielenä on Dart. Sovellus, joka haki datat ja tallensi ne tietokantaan, tehtiin Python-ohjelmointikielellä samassa kehitysympäristössä.

Lopputuloksena saatiin mobiilisovellus, jota voidaan käyttää nykyisen sähkönkulutuksen, aikaisemman sähkönkulutuksen ja tulevaisuuden sähkön hintojen seuraamiseen. Työ tehtiin omaan käyttöön ja idea siihen syntyi nousseista sähkön hinnoista sekä halusta seurata aikaisempaa tarkemmin omaa sähkönkulutusta.

Asiasanat: mobiilisovellukset, Android, energiankulutus, tietokannat

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

---

Author(s): Ville Riekki
Title of thesis: Mobile Application for Monitoring Energy Use
Supervisor(s): Eino Niemi
Term and year when the thesis was submitted: Spring 2023
Number of pages: 27

The aim of this thesis was to implement a mobile application to monitor electricity consumption and future electricity prices. The application was implemented for the Android operating system. In addition to the mobile application, the project included a program that retrieves energy consumption data measured by an electricity meter and future electricity prices from API, as well as storing the data in Firebase's real-time database.

The work began by getting to know the services that monitor electricity consumption and electricity prices. The second part was to design a program for retrieving, storing data and a mobile application for displaying the data. The mobile application that was used to display data was made using Flutter development kit in Visual Studio Code development environment. In Flutter applications, the programming language is Dart. The application that retrieved the data and stored it in a database was made using Python programming language in the same development environment.

The end result was a mobile application that can be used to monitor current electricity consumption, past electricity consumption and future electricity prices. The work was made for own use, and the idea for it was born from the increased electricity prices and the desire to monitor my own electricity consumption more closely than before.

Keywords: mobile applications, Android, energy consumption, databases

SISÄLLYS

TERMIT

Android	Mobiilikäyttöjärjestelmä
API	Ohjelmointirajapinta (Application Programming Interface)
Dart	Ohjelmointikieli
Energomonitor	Tuoteperhe energiankulutuksen mittaamiseen
ENTSO-E	Pörssisähkön hintoja tarjoava palvelu
Firebase	Taustapalveluita palveluna (Backend-as-a-service, BaaS) tarjoava kehitysalusta
Flutter	Käyttöliittymäohjelmistokehityspaketti
Python	Ohjelmointikieli
Realtime Database	Reaaliaikainen tietokanta
SDK	Ohjelmistokehityspaketti (Software Development Kit)
Visual Studio Code	Lähdekoodieditori

1 JOHDANTO

Opinnäytetyön aiheena on toteuttaa sähkönkulutusta ja sähkön hintaa seuraava mobiilisovellus Android-käyttöjärjestelmälle. Opinnäytetyön idea tuli sähkön hinnan noususta ja halusta seurata aikaisempaa tarkemmin omaa sähkönkulutusta ja miettiä mahdollisuuksia vähentää sitä.

Opinnäytetyön teknisessä toteutuksessa käytetään eri työkaluja eri osa-alueisiin. Mobiilisovelluksen tekemiseen käytetään käyttöliittymäohjelmistokehityspakettia Flutteria ja Microsoftin kehittämää tekstieditoria Visual Studio Codea. Mobiilisovellusta testataan manuaalisesti Android Emulaattorilla ja Android-puhelimella ja tämän lisäksi ohjelmaan tehdään automatisoituja testejä. Sähkönkulutus- ja sähkön hinnan datan hakemista APIsta varten on tehty Python-ohjelma, joka lisäksi tallentaa datat Firebasen Realtime Databaseen eli reaaliaikaiseen tietokantaan. Sähkönkulutusta seuraa Energomonitor-mittari, joka on kytketty omakotitaloon, tulevaisuuden sähkön hinta saadaan ENTSO-E:n APIsta.

Työssä keskitytään pääasiassa mobiilisovelluksen tekemiseen. Sovelluksen tärkeimmät ominaisuudet ovat nykyisen sähkönkulutuksen, aikaisemman sähkönkulutuksen ja tulevaisuuden sähkön hinnan näyttäminen.

2 TEKNOLOGIAT

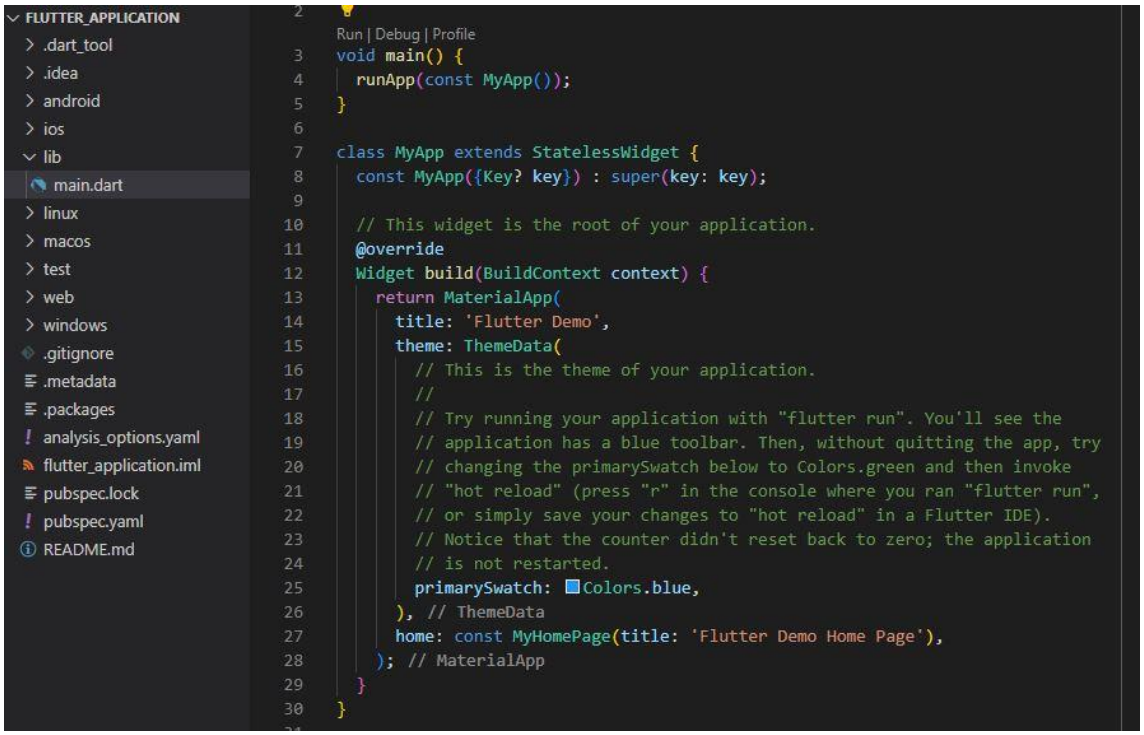
Tässä luvussa kerrotaan työssä käytetyt teknologiat. Mobiilisovellus tehtiin Flutter-käyttöliittymäohjelmistokehityspaketilla, jossa käytetään Dart-ohjelmointikieltä. Palvelinohjelma hakee dataa kahdesta eri ohjelmointirajapinnasta eli APIsta ja tallentaa sitä tietokantaan tehtiin Pythonilla. Tietokantana on Firebasen Realtime Database. Ohjelmointiin käytettiin Visual Studio Code -ympäristöä.

Sähkönkulutusta mitataan omakotitaloon kytketyllä Energomonitor-mittarilla, joka on yhteydessä Energomonitor Homebaseen. Energomonitor Homebase on liitetty Internetiin ja lähettää datan Energomonitor -palveluun. Sähkön hinta saadaan ENTSO-E-palvelusta.

2.1 Flutter

Flutter on vuonna 2017 julkaistu Googlen luoma käyttöliittymäohjelmistokehityspaketti. Sitä käytetään järjestelmäriippumattomien sovellusten kehittämiseen. Yhdestä koodikannasta voidaan kehittää sovelluksia esimerkiksi Androidille, iOS:lle, Linuxille, macOS:lle ja Windowsille. Flutter-sovellukset kirjoitetaan Dart-ohjelmointikielellä.
Flutter-sovelluksia voi tehdä joko Android Studio- tai Visual Studio Code kehitysympäristössä. Visual Studio Codessa Flutter-sovelluksen voi tehdä, kun on ensin ladannut Flutter SDK:n ja asentanut Visual Studio Codeen Flutter- ja Dart -lisäosat. (1.)

Uuden Flutter-projektin luominen on tehty yksinkertaiseksi. Ctrl + Shift + P painikeyhdistelmällä saa esiin komentolistan, josta valitaan uuden Flutter-projektin luominen. Projektityypin ja sen sijainnin valitsemisen sekä nimen syöttämisen jälkeen luodaan yksinkertainen esimerkkiohjelma, josta löytyy tarvittavat tiedostot järjestelmäriippumattoman sovelluskehittämisen aloittamista varten. Sovelluksen koodi löytyy lib-kansiossa olevasta main.dart-tiedostosta. Projektissa on valmiina myös test-kansio, jossa on valmiina yksinkertainen esimerkki automatisoituun testaamiseen (kuva 1).



KUVA 1 Flutter-esimerkkisovellus

2.2 Dart

Dart on Googlen kehittämä ohjelmointikieli, joka julkaistiin 10. lokakuuta 2011. Dartin uusin versio on 2.19.2, joka julkaistiin 8. helmikuuta 2023. Dart-ohjelmointikieli suunniteltiin valinnaiselle tyyppijärjestelmälle, jolloin tyypitystä ei tarvitse käyttää esimerkiksi pienissä projekteissa, mutta se on saatavilla esimerkiksi suurien projektien tarpeisiin. Kieltä voidaan suorittaa myös käännettynä JavaScriptiksi, jolloin sitä voidaan suorittaa verkkoselaimissa. (2.)

Dartissa on iso kokoelma ydinkirjastoja sekä tavallisiin että monimutkaisiin ohjelmointitehtäviin. Esimerkkinä tässäkin opinnäytetyössä käytetyt: kirjasto erilaisten dataesitysmuotojen kääntämiseen oikeanlaiseksi (dart:convert), kirjasto joka mahdollistaa asynkronisen ohjelmoinnin (dart:async) ja kirjasto joka tarjoaa HTML-elementtejä ja muita resursseja interaktioon webselaimen kanssa (dart:html). Ydinkirjastojen ja muiden Dartin julkaisemien pakettien lisäksi kolmannet osapuolet julkaisevat ja kehittävät jatkuvasti monia erilaisia kirjastoja ja paketteja helpottamaan Dart-ohjelmien tekemistä. Esimerkiksi Syncfusion on muun muassa julkaissut erilaisia paketteja kaavojen esittämiseen, joita käytetään tässäkin opinnäytetyössä. (3.)

2.3 Firebase

Firebase on taustapalveluja palveluna (Backend-as-a-service, BaaS) tarjoava kehitysalusta, joka tarjoaa tietokantapalveluiden isännöintiä ja työkaluja, esimerkiksi web- ja mobiiliapplikaatioiden parantamiseen ja kehittämiseen. Työkaluja ovat esimerkiksi reaaliaikainen tietokanta, autentikointi, pilvipalvelut ja mahdollisuus analytiikan hyödyntämiseen. Firebase myös tarjoaa eri ohjelmointikielille ja alustoille ohjelmistokehityspaketteja (SDK), jotka helpottavat Firebasen palveluiden käyttöönottoa. (4.)

Opinnäytetyössä Firebasen palveluista käytetään Realtime Databasea eli reaaliaikaista tietokantaa, joka on pilvipalvelussa isännöity tietokanta. Tietokannassa oleva data tallennetaan JSON-muodossa ja synkronisoidaan reaaliaikaisesti jokaiseen yhdistettyyn ohjelmaan. Reaaliaikaisessa tietokannassa jokainen yhdistetty ohjelma jakaa yhden tietokannan ja vastaanottaa automaattisesti päivitykset aina, kun uutta dataa on tarjolla. (5.)

2.4 Energomonitor

Energomonitor on reaaliaikainen sähkönkulutuksen, energiankulutuksen ja olosuhteiden seuranta- ja mittausjärjestelmä. Energiankulutuksen mittaamiseen kuuluu sähkö, vesi ja kaasu. Olosuhdemittauksiin kuuluu lämpötila -40:n ja +120 °C:n välillä, kosteus ja CO2. Mittariin voi myös laittaa jatkuvan valvonnan ja hälytykset päälle omilla asetuksilla, esimerkiksi jos lämpötila tippuu ja on jäätymisvaara tai jos sähkönkulutus on korkea tai matala. Kulutustiedot päivittyvät 90 sekunnin välein. Mitatun datan saa siirrettyä REST ja JSON APIn kautta tai ladattua joko Excel- tai CSV-tiedostoina. Energomonitor-mittausjärjestelmä on ollut markkinoilla vuodesta 2015 ja sitä kehitetään jatkuvasti. (6.)

2.5 ENTSO-E

ENTSO-E:n (European Network of Transmission System Operators for Electricity) tehtävänä on edustaa 39 kansallista kantaverkkoa hallinnoivaa jäsenyhtiötä 35 eri maasta Euroopassa. Ennen ENTSO-E:tä Euroopassa toimi kuusi eri yhteistyöjärjestöä, jotka ENTSO-E korvasi. ENTSO-E aloitti toimintansa vuonna 2009. (7.)

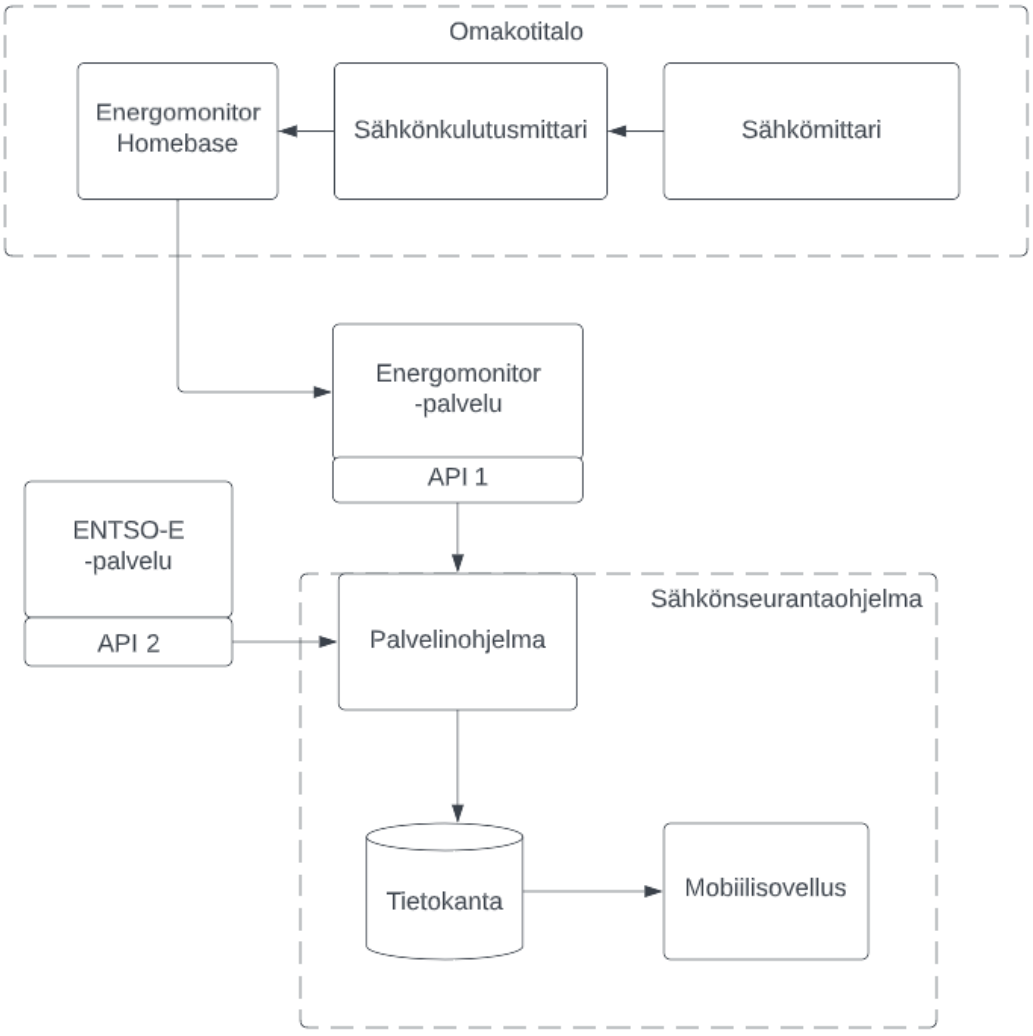
ENTSO-E:n Transparency Platform tarjoaa pääsyn Euroopan energiantuottamis-, energiansiirto- ja energiankulutusdataan ilmaiseksi (8.).

2.6 Python

Python on ohjelmointikieli, jota käytetään muun muassa webohjelmointiin, tieteelliseen tietojenkäsittelyyn, data-analysointiin, ja tekoälyyn. Se on tunnettu hyvästä luettavuudesta ja yksinkertaisuudesta ja näistä syistä se on suosittu ohjelmointikieli aloittelijoiden sekä kokeneempien ohjelmoijien keskuudessa. Python on yksi suosituimmista ohjelmointikielistä ja sillä on aktiivinen yhteisö, joka kehittää jatkuvasti monia erilaisia kirjastoja sekä ohjelmistokehyksiä, joten se kehittyy ja paranee koko ajan edellä mainituilla osa-alueilla. (9.)

3 TYÖN TOTEUTUS

Toteutettu sähkönkulutuksen seurantaohjelma lukee omakotitalon sähkönkulutusta sähkömittariin kytketyn mittarin avulla, joka lähettää tiedon Energomonitor Homebaseen, joka on liitetty omakotitalon langattoman verkon välityksellä Internetiin. Energomonitor Homebase lähettää tiedot Energomonitor-palveluun. Python-ohjelmointikielellä ohjelmoitu ohjelma lukee Energomonitor-palvelusta sähkönkulutusdataa ja ENTSO-E-palvelusta pörssisähkön hintadataa. Python-ohjelma eli palvelinohjelma myös tallentaa datat Firebase-tietokantaan ja mobiilisovellus lukee datat tietokannasta. Mobiilisovellus näyttää sähkönkulutuksen ja sähkön hinnan käyttäjälle (kuva 2).



KUVA 2 Projektin osat

Opinnäytetyön idea tuli sähkön hinnan nopeasta noususta ja halusta tarkkailla omaa sähkönkulutusta aikaisempaa tarkemmin. Sähkön hinta oli vuoden 2022 kolmannella neljänneksellä 40–60 prosenttia korkeampi kuin vuotta aikaisemmin (10). Tämä hinnan nopea muutos näkyy kuvassa 3, jossa esitetään, kuinka sähkön hinta (€/MWh) muuttui tammikuusta 2015 syyskuuhun 2022.

KUVA 3 Sähkön hinnan muutos 2015–2022 (11)

3.1 Sähkönkulutus

Sähkönkulutuksen seuraamiseen käytetään Energomonitor-mittaria (kuva 4). Mittari mittaa omakotitalon sähkönkulutusta ja on yhteydessä Energomonitor Homebaseen, joka on liitetty Internetiin ja on yhteydessä Energomonitor-sivuun. Sähkönkulutusdataa voi seurata Energomonitor-sivulta.

KUVA 4 Energomonitor Homebase vasemmalla ja Energomonitor-sähkönkulutusmittari oikealla (7)

Nettisivuilta voi joko ladata CSV- ja Excel-tiedostoja tai hakea dataa APIn kautta. Tässä työssä datan hakemiseen käytetään Python-ohjelmaa, joka hakee dataa tunnin välein Energomonitorin APIsta ja tallentaa sen tietokantaan. Sähkönkulutusdata näytetään mobiilisovelluksessa listana sekä kuvaajana.

Time (UTC)	Value [Wh]
1672640460 = 2023-01-02 06:21:00	12
1672640520 = 2023-01-02 06:22:00	11
1672640580 = 2023-01-02 06:23:00	11
1672640640 = 2023-01-02 06:24:00	8
1672640700 = 2023-01-02 06:25:00	12
1672640760 = 2023-01-02 06:26:00	14
1672640820 = 2023-01-02 06:27:00	14
1672640880 = 2023-01-02 06:28:00	12
1672640940 = 2023-01-02 06:29:00	12
1672641000 = 2023-01-02 06:30:00	10

KUVA 5 Sähkönkulutusdata Energomonitorin nettisivulla

3.2 Tulevaisuuden sähkön hinta

Tulevaisuuden sähkön hinnat on haettu ENTSO-E:n Transparency Platform APIsta. Datan hakemiseen käytetään samaa Python-ohjelmaa, joka hakee sähkönkulutusdataa Energomonitorin APIsta. Haetussa datassa on seuraavan vuorokauden jokaisen tunnin megawattituntihinta euroina. Hinta muutetaan €/MWh-muodosta snt/kWh-muotoon, jotta sitä voi hyödyntää sähkönkulutusdatan kanssa. Ohjelma hakee ENTSO-E:n APIsta dataa kerran vuorokaudessa ja tallentaa sen tietokantaan.

3.3 Tietokanta

Python-ohjelma ja mobiilisovellus on yhdistetty Firebase-tietokantaan, minne tallennetaan kaikki data. Firebaseen tallennetaan tietoa sähkönkulutuksesta ja tulevista hinnoista. Tässä projektissa tietokantana on Firebase Realtime Database. Se on NoSQL-tietokanta, joka tallentaa tiedot pilveen (eng. cloud-hosting) ja mahdollistaa datan reaaliaikaisen kirjoittamisen ja lukemisen.

Sähkönkulutusdataa tallennetaan tietokantaan tunnin välein. Datasta tehdään JSON-olio, johon kuuluu päivämäärä, kellonaika ja sähkönkulutus kilowattitunteina. Olion ID:ksi on laitettu tallennuksen aikainen aikaleima (kuva 6). Sähkön hinta on tallennettu muilta osin samalla tavalla, mutta olion ID:ksi on laitettu päivämäärä ja olioon kuuluu jokaiselle tunnille kilowattitunnin senttihinta avain-arvo-parina.

KUVA 6 Firebase-tietokantaan tallennettua sähkönkulutusdataa

3.4 Mobiilisovellus

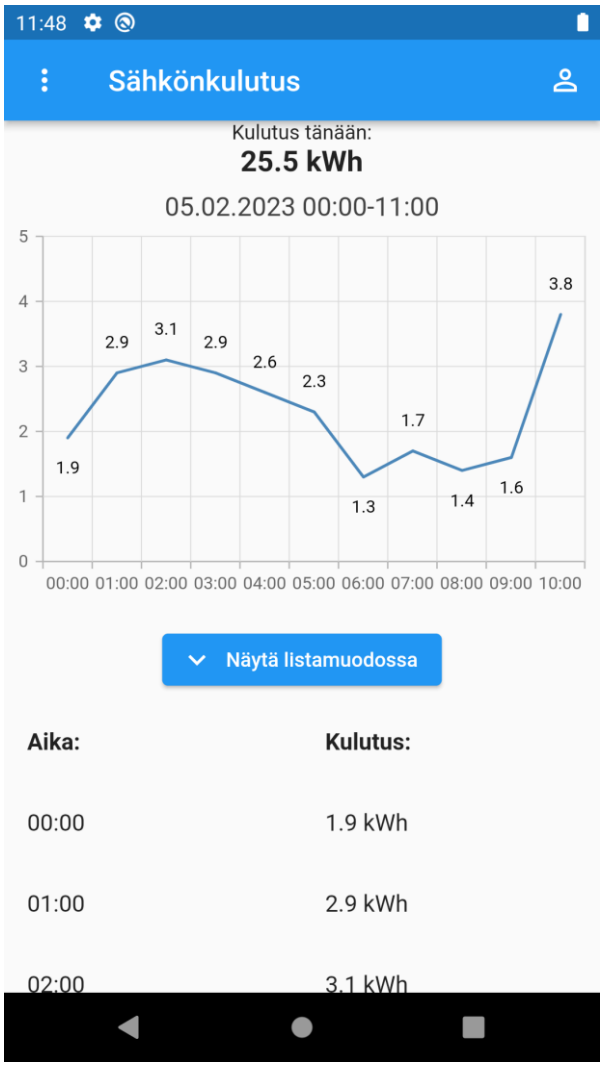
Mobiilisovellus tehtiin käyttäen Flutter-kehityspakettia ja ohjelmointikielenä oli Dart. Flutterilla olisi mahdollista tehdä Windows-sovellus, websovellus, iOS-sovellus ja Android-sovellus, mutta tässä projektissa tehtiin vain mobiilisovellus Android-käyttöjärjestelmälle. Mobiilisovellusta testattiin työn aikana manuaalisesti Android-emulaattorilla sekä Android-puhelimella, missä on käytössä Android-versio 11. Manuaalisen testauksen lisäksi sovellukseen tehtiin automatisoituja testejä.

Sovellukselle suunniteltiin kolme tehtävää ja jokaiselle tehtävälle oma näkymä. Tehtäviä ovat nykyisen sähkönkulutuksen seuraaminen, aikaisemman sähkönkulutuksen tarkastelu ja tulevaisuuden sähkön hinnan seuraaminen. Sovelluksessa käytetään listojen lisäksi kaavoja kuvaamaan sähkönkulutusta ja sähkön hintaa. Kaavat ovat Syncfusion Flutter Charts -kirjastosta, jonka luojana on Syncfusion.

3.4.1 Nykyinen sähkönkulutus

Näkymä nykyiselle sähkönkulutukselle on sovelluksen pääsivu ja se aukeaa, kun sovellus käynnistyy. Siinä näytetään yläosassa tämänpäiväinen kulutus kilowattitunteina. Esimerkissä (kuva 7) näkyy jokaisen tunnin kulutus keskiyöstä kuvanottohetkeen eli klo 11:00 asti. Kulutus näkyy ensin kuvaajana, jonka jälkeen on ”Näytä listamuodossa”-painike ja sitä painamalla sama tieto tulee esiin myös listamuodossa.

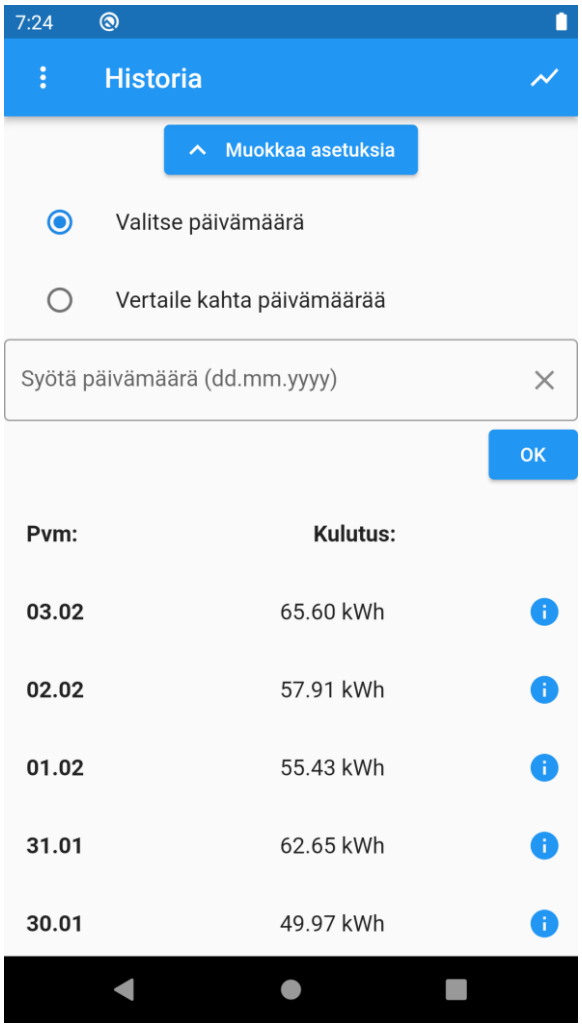
Näkymän yläpalkissa on kaksi painiketta. Vasemmalla on valikkopainike, jota painamalla voi siirtyä johonkin muuhun näkymään ja oikealla on käyttäjätietopainike, jota painamalla pääsee käyttäjätietoihin. Käyttäjätiedoissa ilmoitetaan, mihin Energomonitor-sivun tiliin sovellus on yhdistettynä.



KUVA 7 Nykyinen kulutus -näkymä

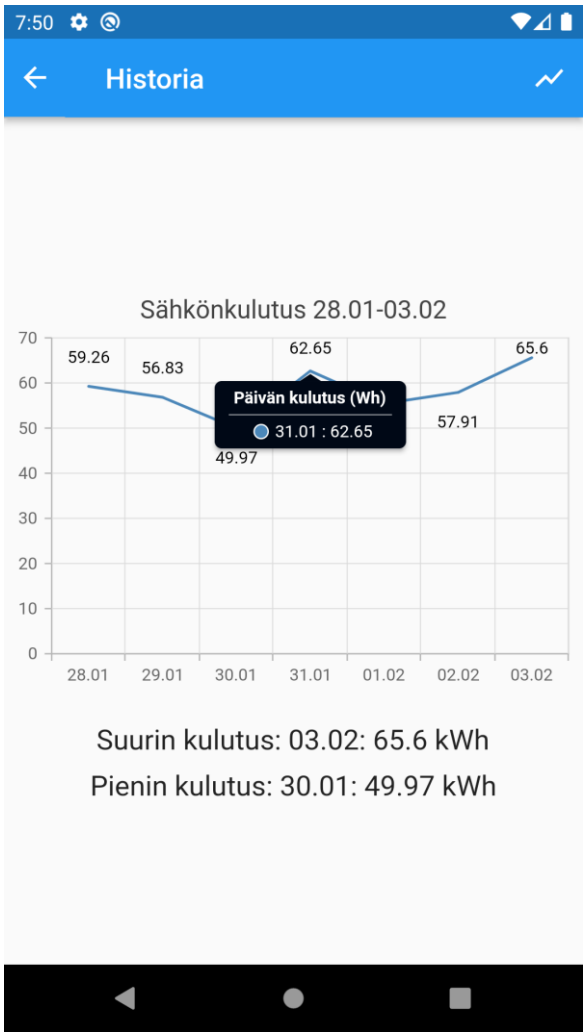
3.4.2 Historia

Historianäkymässä oletuksena näytetään seitsemän viime vuorokauden sähkönkulutus kilowattitunteina listamuodossa. Oletusnäkymän lisäksi voi valita myös eri näkymiä. Esimerkkikuvassa (kuva 8) näkyy yläosassa ”Muokkaa asetuksia”-painike, jota painamalla avautuu asetusvalikko. Asetusvalikossa voi valita päivämäärän, jota haluaa tutkia tarkemmin tai voi vertailla kahta eri päivämäärää. Asetusvalikko piilotetaan, kun painiketta painetaan uudestaan. Asetusvalikon alapuolella olevassa listassa vasemmalla on päivämäärä, keskellä sen päivämäärän sähkönkulutus kilowattitunteina Oikeassa reunassa on infopainike, jota painamalla pääsee uuteen näkymään, jossa näytetään kyseisen päivämäärän sähkönkulutustietoja tarkemmin eli infopainike on oikotie asetusvalikon ”Valitse päivämäärä”-toiminnolle.



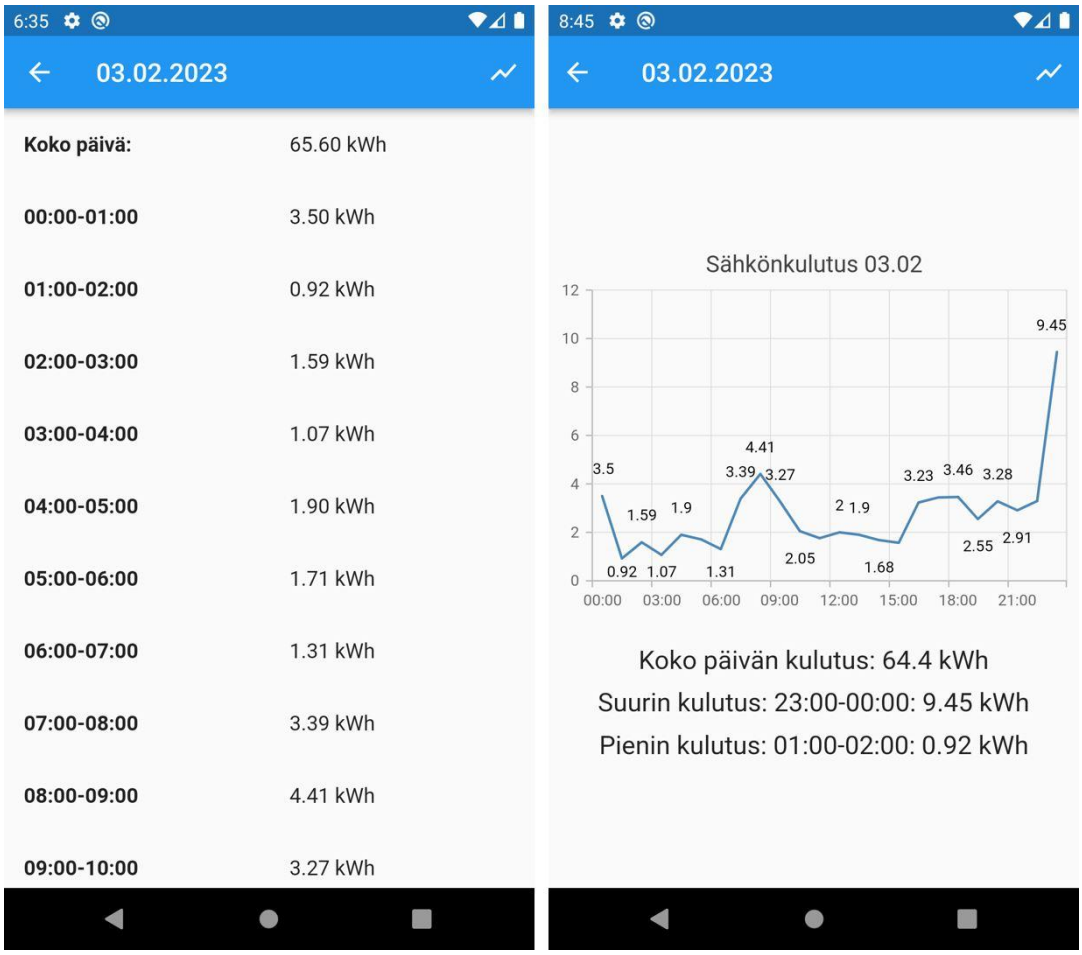
KUVA 8 Historianäkymä

Näkymän yläpalkin oikeassa reunassa on painike, josta avautuu kuvaaja (kuva 9). Kuvaajassa näytetään samat tiedot kuin listassakin eli viimeisen seitsemän vuorokauden sähkönkulutus kilowattitunteina. Jokaiselle vuorokaudelle on oma piste kuvaajassa. Lisäksi kuvaajan yläpuolella kerrotaan, miltä ajalta data on ja sen alapuolella kerrotaan ne vuorokaudet, jolloin oli suurin ja pienin kulutus. Kuvaajassa voi myös painaa jotain kohtaa, jolloin lähimmän pisteen tiedot eli päivämäärä ja sähkönkulutus näytetään hieman selvemmin.



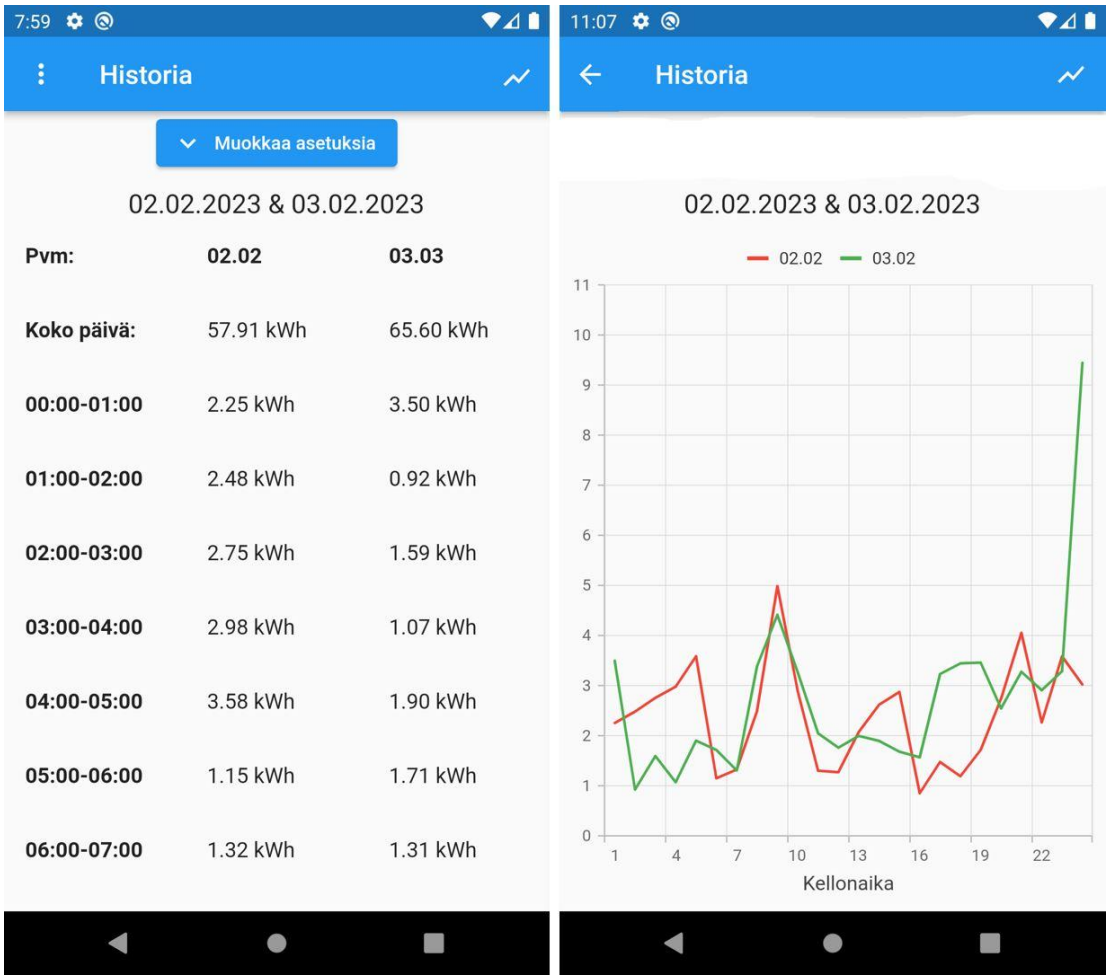
KUVA 9 Historianäkymän kuvaaja

Jos käyttäjä valitsee yhden päivämäärän tarkemman näkymän joko asetusvalikosta tai listassa olevasta infopainikkeesta, aukeaa näkymä, jossa aluksi näkyy koko päivän kulutus ja sen jälkeen jokaisen tunnin kulutus kilowattitunteina. Yläpalkin oikeassa reunassa olevasta painikkeesta voidaan avata kuvaajanäkymä, jossa näkyy jokaisen tunnin sähkönkulutusarvot kuvaajassa. Kuvaajan alapuolella kerrotaan myös koko kulutus yhteensä sekä suurimman ja pienimmän kulutuksen kilowattitunnit ja kellonajat, milloin ne tapahtuivat (kuva 10).



KUVA 10 Yhden päivämäärän tarkemmat tiedot historianäkymässä

Jos käyttäjä valitsee kahden päivämäärän vertailun, aukeaa näkymä, jossa aluksi kerrotaan päivämäärät, joita vertaillaan. Päivämäärien alapuolella näkyy kellonajat ja molemmat päivämäärät listassa. Listan alussa kerrottujen koko vuorokausien kulutuksen jälkeen näkyy kulutukset tunneittain. Vasemmalla on kellonaika, keskellä aikaisemman päivämäärän kulutus ja oikealla myöhemmän päivämäärän kulutus. Yläpalkin oikeassa reunassa olevasta painikkeesta voi avata kuvaajanäkymän, jossa näytetään molempien päivien joka tunnin sähkönkulutus kuvaajassa (kuva 11). Kuvaajassa punainen viiva on aikaisempi päivämäärä ja vihreä viiva on myöhempi päivämäärä.

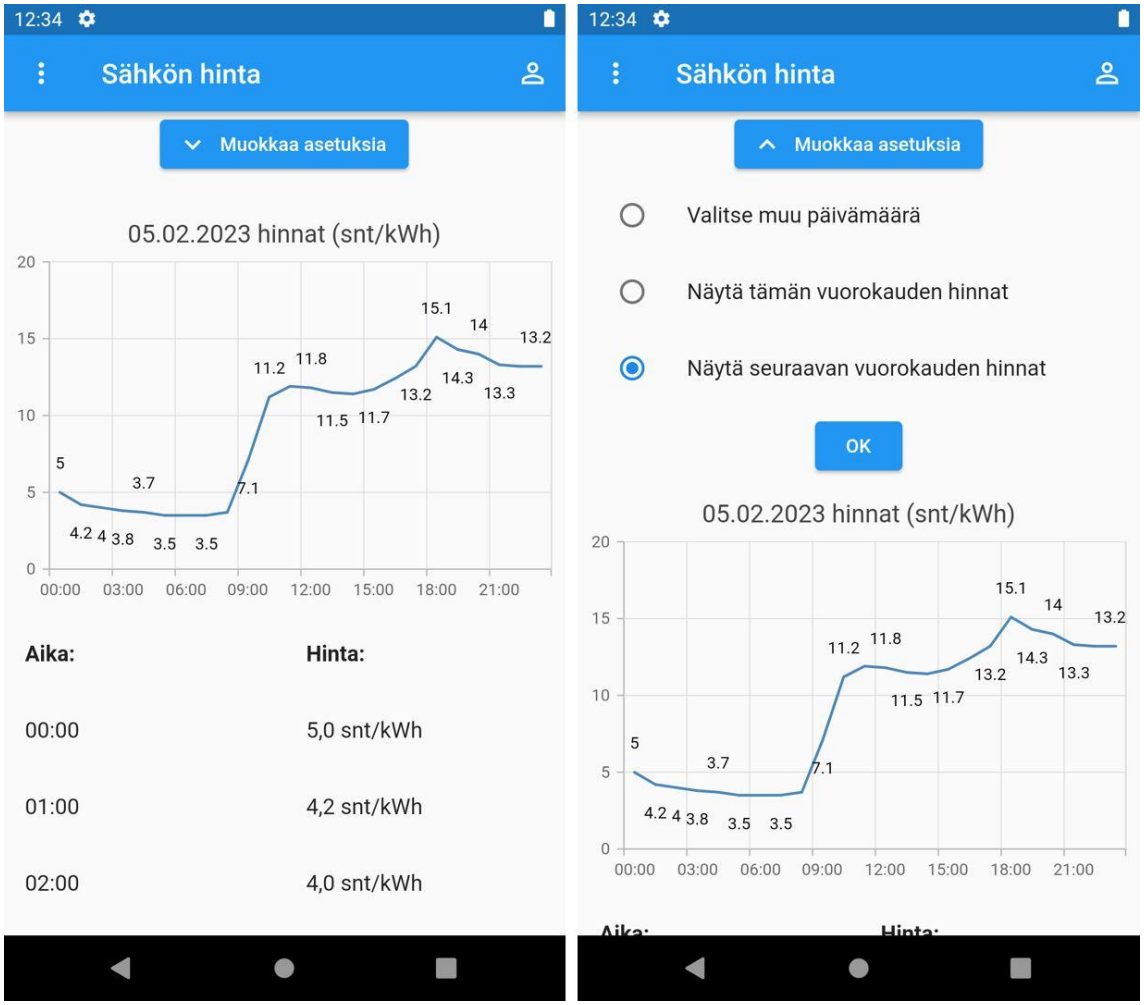


KUVA 11 Kahden päivämäärän sähkönkulutuksen vertaaminen

3.4.3 Tulevaisuuden hinnat

Näkymässä näytetään oletuksena, montako senttiä yksi kilowattitunti maksaa seuraavana vuorokautena. Hinnat näytetään ensin kuvaajana ja sen alapuolella listamuodossa (kuva 12). Molemmissa näytetään vuorokauden jokaisen tunnin hinta. Painamalla näkymän yläosassa olevaa ”Muokkaa asetuksia”-painiketta aukeaa asetusvalikko, josta voi valita jonkun muun päivämäärän. Jos valitsee ”Valitse muu päivämäärä” -kohdan, tulee esille tekstikenttä ja painamalla ”OK”-painiketta sen vuorokauden hinnat tulevat näkyviin. Nykyisen päivämäärän valitseminen on tehty nopeammaksi, koska nykyisen vuorokauden hintojen tarkkailu on tarpeellisempaa kuin sitä

vanhempien hintojen tarkkailu. Nykyisen päivämäärän hinnat saa näkymille valitsemalla ”Näytä tämän vuorokauden hinnat”-kohdan ja painamalla ”OK”-painiketta.



KUVA 12 Sähkön hinta -näkymä

3.4.4 Sovelluksen testaaminen

Mobiilisovellusta testattiin kehityksen aikana automatisoidusti ja manuaalisesti. Manuaalinen testaus tapahtui pääasiassa Android Emulatorilla, mutta sovellusta testattiin manuaalisesti myös Android-puhelimella, jossa oli käytössä Android-versio 11. Kehityksen aikana myös vertailtiin mobiilisovelluksessa näkyviä sähkönkulutusarvoja Energomonitor-palvelussa näkyviin arvoihin.

Mitä isommaksi sovellus kasvaa ja mitä enemmän toimintoja siinä on, sitä vaikeammaksi manuaalinen testaaminen menee. Automatisoidut testit auttavat tehokkaammin varmistamaan, että sovellus toimii halutulla tavalla ja että mahdolliset virheet löytyvät nopeasti.

Automatisoidut testit voidaan jakaa kolmeen eri kategoriaan: yksikkötestaaminen, pienohjelmatestaaminen ja integrointitestaaminen. Yksikkötestaamisessa testataan esimerkiksi yksittäistä funktiota tai metodia, pienohjelmatestaamisessa yhtä pienohjelmaa eli widgettiä ja integraatiotestaamisessa koko sovellusta tai isoa osaa siitä. Jos sovelluksen testit on tehty hyvin, siinä on yleensä useampia yksikkö- ja pienohjelmatestejä sekä sen verran integraatiotestejä, että kaikki tärkeät käyttötapaukset on saatu testattua ja varmistettua toimiviksi. Erilaisilla testeillä on erilaisia hyötyjä ja heikkouksia (kuva 13). (12.)

	Unit	Widget	Integration
Confidence	Low	Higher	Highest
Maintenance cost	Low	Higher	Highest
Dependencies	Few	More	Most
Execution speed	Quick	Quick	Slow

KUVA 13 Yksikkö-, pienohjelma- ja integrointitestaamisen vertailu (12)

Sovellukseen kuuluu useampia pienohjelmatestejä. Esimerkkikuvana on pääsivun testaus. Ensimmäisessä testissä tarkistetaan, että sivulta löytyy tekstiwidget, jossa on tekstinä ”Kulutus tänään:”. Toisessa testissä tarkistetaan, että aluksi sivulta ei löydy listaa, mutta ”Näytä listamuodossa”-painikkeen painamisen jälkeen sivulta löytyy yksi lista (kuva 14).

```
Run | Debug
void main() {
  Run | Debug
  group("Main page widget tests", () {
    Run | Debug
    testWidgets("Testing if Text shows up", (tester) async {
      await tester.pumpWidget(const MyApp());
      expect(find.text('Kulutus tänään:'), findsOneWidget);
    });

    Run | Debug
    testWidgets("Testing if ListView shows up when button is pressed",
        (tester) async {
      await tester.pumpWidget(const MyApp());
      expect(find.byType(ListView), findsNothing);
      await tester.tap(find.byIcon(Icons.expand_more));
      await tester.pump();
      expect(find.byType(ListView), findsOneWidget);
    });
  });
```

KUVA 14 Pienohjelmatestaus Flutter-sovelluksessa

Integrointitesteillä yleensä testataan, kuinka sovelluksen yksittäiset osat toimivat yhdessä. Testaamiseen käytetään integration_test-nimistä kirjastoa. Ennen testaamista sovellus pitää konfiguroida niin, että ajuri pääsee käsiksi käyttöliittymään ja funktioihin automatisoituja testejä varten.

Integrointitestin (kuva 15) alussa varmistetaan, että integrointitestin ajuri on alustettu, jonka jälkeen testataan, että listaa voidaan rullata alaspäin niin kauan, kunnes lista loppuu ja sen jälkeen takaisin ylös. Testistä tallennetaan yhteenveto, joka tallennetaan omaan tiedostoon projektin build-kansioon. (13.)

```
Run | Debug
void main() {
  Run | Debug
  group("Testing application performance tests", () {
    final binding = IntegrationTestWidgetsFlutterBinding.ensureInitialized();
    binding.framePolicy = LiveTestWidgetsFlutterBindingFramePolicy.fullyLive;

    Run | Debug
    testWidgets("List scrolling test", (tester) async {
      await tester.pumpWidget(MyApp());
      final listFinder = find.byType(ListView);
      await binding.watchPerformance(() async {
        await tester.fling(listFinder, Offset(0, -500), 10000);
        await tester.pumpAndSettle();
        await tester.fling(listFinder, Offset(0, 500), 10000);
        await tester.pumpAndSettle();
      }, reportKey: "list_scrolling_summary");
    });
  });
}
```

KUVA 15 Integrointitesti Flutter-sovelluksessa

4 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa omaan käyttöön mobiilisovellus, joka helpottaa oman sähkönkulutuksen seuraamista. Suunnitteluvaiheessa sovellukselle mietittiin kolme tärkeintä tehtävää: nykyisen kulutuksen seuraaminen, aiemman kulutuksen seuraaminen ja tulevien hintojen seuraaminen. Mobiilisovelluksen lisäksi työhön kuuluu Python-ohjelma, joka haki kulutetun sähkön dataa Energomonitorin APIsta ja tulevaisuuden sähkön hinnat ENTSO-E:n APIsta sekä tallensi molemmat tietokantaan.

Suunnitteluvaiheen jälkeen työ jatkui tutustumisella Energomonitoriin sekä ENTSO-E:hen ja erityisesti molempien ohjelmointirajapintoihin. Energomonitorin APIn käyttöönotto vaati vain käyttäjän luomista sivulle, ja ENTSO-E:hen pyydettiin käyttäjän luomisen jälkeen lupa tehdä API-pyyntöjä. Python-ohjelman luominen oli niin ikään helppoa, sillä ohjelmasta tuli melko yksinkertainen. Ohjelma on koko ajan päällä ja se hakee Energomonitorin dataa kerran tunnissa ja ENTSO-E:n dataa kerran vuorokaudessa. Tiedon tallentaminen tietokantaan tapahtuu heti sen jälkeen, kun data on haettu APIsta.

Mobiilisovellus tehtiin Flutter-ohjelmistokehityspaketin avulla Visual Studio Code-ympäristössä. Mobiilisovellus toteutettiin vain Android-käyttöjärjestelmille, eikä sitä testattu toimivaksi esimerkiksi iOS-puhelimella. Minulla oli jonkin verran aikaisempaa kokemusta mobiilisovelluksen ohjelmoinnista käyttäen Flutteria, joten pääsin alkuun hyvin. Sovelluksen haastavin asia oli tiedon hakeminen reaaliaikaisesta tietokannasta. Firebasen ja Flutterin sivuilta kuitenkin löytyi hyvä dokumentaatio ja lopulta sovelluksen yhdistäminen Firebasen tietokantaan olikin melko yksinkertaista.

Sovellukseen saatiin toteutettua kolme eri näkymää kolmelle eri tehtävälle eli nykyiselle sähkönkulutukselle, aikaisemmalle sähkönkulutukselle ja tuleville sähkön hinnoille. Pyrin pitämään mobiilisovelluksen mahdollisimman yksinkertaisena, mutta kuitenkin mahdollistamaan datan tarkkailun kuvaaja- ja listamuotoisena. Ohjelman oikea toiminnallisuus varmistettiin automatisoitujen testien lisäksi manuaalisella testaamisella, jonka aikana muun muassa vertailtiin mobiilisovelluksen näyttämiä sähkönkulutusarvoja Energomonitorin sivulla näkyviin arvoihin.

Projekti oli onnistunut ja sen tekeminen opetti paljon suunnittelusta ja mobiilisovelluksen kehittämisestä. Valittu aihe oli sopivan haastava. Jouduin jatkuvasti opettelemaan uutta mutta pysyin koko ajan aikataulussa ja pystyin etenemään teknisessä toteutuksessa sekä opinnäytetyön kirjoittamisessa tasaiseen tahtiin. Suurin osa tekniikoista oli entuudestaan tuttuja, mutta niiden osaaminen kehittyi työn aikana selkeästi.

LÄHTEET

1. Wikipedia 2023. Flutter (software). Hakupäivä 26.1.2023. https://en.wikipedia.org/wiki/Flutter_(software).
2. Wikipedia 2023. Dart (ohjelmointikieli). Hakupäivä 26.1.2023. https://fi.wikipedia.org/wiki/Dart_(ohjelmointikieli).
3. Dart. Dart overview. Hakupäivä 10.2.2023. https://dart.dev/overview.
4. Wikipedia 2023. Firebase. Hakupäivä 26.1.2023. https://en.wikipedia.org/wiki/Firebase.
5. Firebase 2023. Firebase Realtime Database. Hakupäivä 10.2.2023. https://firebase.google.com/docs/database.
6. OmaVahti 2023. Reaaliaikainen energiankulutus- ja olosuhdemittaus järjestelmä. Hakupäivä 26.1.2023. https://www.omavahti.fi/tuote/energomonitor/.
7. Wikipedia 2023. ENTSO-E. Hakupäivä 10.2.2023. https://fi.wikipedia.org/wiki/ENTSO-E.
8. Wikipedia 2023. European Network of Transmission System Operators for Electricity. Hakupäivä 26.1.2023. https://en.wikipedia.org/wiki/European_Network_of_Transmission_System_Operators_for_Electricity.
9. Wikipedia 2023. Python (ohjelmointikieli). Hakupäivä 10.2.2023. https://fi.wikipedia.org/wiki/Python_(ohjelmointikieli).
10. Pitkäranta, Pilvi 2022. Sähkön hinta nousi heinä-syyskuussa jopa 60 prosenttia viime vuodesta. Yle. Hakupäivä 9.2.2023. https://yle.fi/a/74-20007727.
11. Tilastokeskus 2022. Pandemiasta toipuminen ja Venäjän hyökkäys Ukrainaan johtivat energiahyödykkeiden hinnat jyrkkään nousuun. Hakupäivä 11.2.2023. https://stat.fi/julkaisu/cktyeqofs24270c529ia6x4i1.
12. Flutter 2023. Testing Flutter apps. Hakupäivä 8.2.2023. https://docs.flutter.dev/testing.
13. Codelabs 2023. How to test a Flutter app. Hakupäivä 8.2.2023. https://codelabs.developers.google.com/codelabs/flutter-app-testing#0.