OULUN AMMATTIKORKEAKOULU

Samuli Virkkula

NATLINK OY:N HUOLTOLOMAKKEEN PÄIVITYS

NATLNIK OY:N HUOLTOLOMAKKEN PÄIVITYS

Samuli Virkkula
Opinnäytetyö
Kevät 2024
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä: Samuli Virkkula
Opinnäytetyön nimi: Huoltolomakkeen teknillinen päivitys
Työn ohjaaja: Juha-Matti Huusko
Työn valmistumislukukausi ja -vuosi: Kevät 2024
Sivumäärä: 33

Tämän opinnäytetyön tilaajana toimi Natlink Oy. Yrityksellä oli tarve uusia tuotteiden huoltoon tarkoitettua selaimessa toimivaa huoltolomaketta. Tarve uudelle oli, koska vanhassa huoltolomakkeessa oli ongelmia toimivuuden kanssa. Tavoite opinnäytetyössä oli rakentaa uusi huoltolomake käyttäen Flutteria, jotta lomake toimisi selaimessa ja mobiilisovelluksessa.

Työ toteutettiin käyttäen Visual Studio Code -kehitysympäristöä, Flutteria-kehityspakettia ja Dart-ohjelmointikieltä. Flutter on Googlen kehittämä avoimen lähdekoodin käyttöliittymäkehitystyökalu, jonka avulla voidaan rakentaa natiiveja sovelluksia. Työn toteutus aloitettiin tutustumalla vanhaan lomakkeeseen ja tutkimalla lomakkeen ongelmia ja kehityskohtia. Työtä jatkettiin tutkimalla, miten lomakkeen taustalla toimivat kolmannen osapuolen järjestelmät toimivat. Huoltolomake kommunikoi esimerkiksi Microsoft Business Centralin kanssa. Kun tarvittavat tiedot saatiin, voitiin uuden lomakkeen kehittäminen aloittamaan.

Lopputuloksena saatiin toimiva huoltolomake, päivitetty käyttöliittymä ja dokumentaatio liittyen sovellukseen. Huoltolomake on käytössä yrityksen nettisivulta, mistä asiakas voi tilata huollon laitteelle. Työn dokumentaatio on tallennettu yrityksen pilvipalveluun, josta sitä voi tarvittaessa lukea. Tulevaisuudessa huoltolomake tulisi olemaan käytössä, myös mobiilisovelluksessa. Huoltolomake lähettää asiakkaan tiedot onnistuneesti rajanpinnalle, josta rajanpinta tallentaa ne Business Centraliin yrityksen luettavaksi. Käyttöliittymästä tehtiin moderni ja käyttäjäystävällisempi.

Asiasanat: Flutter, Dart, Microsoft Business Central, huoltopalvelut, Frontend-kehitys, alustojen välinen kehitys

ABSTRACT

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

Author: Samuli Virkkula
Title of thesis: Technical upgrade of service form
Supervisor: Juha-Matti Huusko
Term and year when the thesis was submitted: Spring 2024
Number of pages: 33

This thesis was commissioned by Natlink Oy. The company needed a new browser-based service form for product maintenance. There was a need for a new one, because the old service form had problems with the functionality. The aim of the thesis was to build a new service form using Flutter, so that the form would work in the browser and mobile application. The new form works in the browser as well as in the mobile app.

The work was implemented using the Visual Studio Code development environment, Flutter development package and Dart programming language. Flutter is an open-source UI development tool developed by Google to build native applications. The work started with a study of the old form and the problems and development issues of the form. The work continued by investigating how the underlying 3-part systems work. For example, the service form communicates with Microsoft Business Central. Once the necessary information was available, the development of the new form could begin.

The result was a working maintenance form, an updated user interface and documentation related to the application. The maintenance form is available on the company's website, where the customer can order maintenance for the device. The documentation of the work is stored in the company's cloud service, where it can be accessed when needed. In the future, the service form should be available, also on the mobile app. The maintenance form successfully sends the customer's data to the interface, from where the interface stores it in Business Central for the company to read. The user interface was made modern and more user-friendly. Translated with DeepL.com (free version)

Keywords: Flutter, Dart, Microsoft Business Central, maintenance services, front-end development, cross-platform development

SISÄLLYS

SANASTO

API	Ohjelmointirajapinta (Application Programming Interface)
Android	Android-käyttöjärjestelmä
DART	Ohjelmointikieli
Flutter	Käyttöliittymäohjelmistokehityspaketti
Hot-Reload	Flutter-kehityspaketin ominaisuus
IOS	Apple-mobiililaitteen käyttöjärjestelmä
JavaScript	Ohjelmointikieli
Konekoodi	Tietokoneen suorittimen ymmärtämä formaali kieli
Linux	Käyttöjärjestelmä
MacOS	Apple-tietokoneen käyttöjärjestelmä
Natiivi	Tietylle alustalle tarkoitettu ohjelma
WebAssembly	Binäärinen käskykanta virtuaalikoneelle
Widget	Flutter-sovelluksen rakennuspala
Windows	Windows-tietokoneen käyttöjärjestelmä

1 JOHDANTO

Opinnäytetyössä tutkitaan Natlink Oy:n käytössä olevaa web-pohjaista huoltolomaketta ja kehitetään tilalle uusi, web- sekä mobiilipohjainen huoltolomake, joka korjaa aikaisemman huoltolomakkeen ongelmat. Työn idea tuli yrityksen sisältä, kun huomattiin huoltolomakkeessa olevia ongelmia, jotka pitäisi korjata. Aikaisemman lomakkeen ongelmat aiheuttivat yritykselle ylimääräistä työtä ja sen myötä lisäkustannuksia. Työ edellyttää aikaisemman lomakkeen läpikäynti ja tutkimista, uuden suunnittelua, uuden ohjelmointia ja testausta, jotta uusi lomake voidaan ottaa käyttöön.

1.1 Natlink Oy

Natlink on Open Air Groupin lanseeraama yritys. Natlink on Euroopan suurin metsästysteknologiayritys (1). Aiemmin Open Air Groupin teknologiatoiminnan osana toimi kolme teknologian nimeä: Tracker, joka valmisti GPS-seurantalaitteita metsästyskoirille. WeHunt, joka valmisti metsästyssovelluksen ja riistakamera-asiantuntija Burrel. Natlink perustettiin, kun nämä kolme nimeä yhdistivät toimintansa vuonna 2024 yhdeksi yritykseksi. Yhdistymisen myötä Natlink on pohjoismaiden suurin metsästysteknologiayritys, jolla on yli 740 000 rekisteröitynyttä käyttäjää ja yli 20 miljoonaa euroa yhdistettyä liikevaihtoa. (2.)

1.2 Opinnäytetyön vaiheet ja työn tekniikka

Tehdessäni opinnäytetyötä on metsästyskauden niin sanottu low-season eli metsästystä ei juurikaan harrasteta. Yritys päätti, että huoltolomake olisi paras uusia low-seasonin aikana, koska laitteita ei tule juurikaan huoltoon. Huoltolomake on tärkeä saada toimimaan, ennen kuin high-season alkaa, eli kun metsästystä tapahtuu paljon. Opinnäytetyössä on monia vaiheita, joiden myötä työ saadaan valmiiksi. Vaiheita ovat suunnittelu, sovelluksen suunnittelu, sovelluksen toteutus ja sovelluksen testaaminen ja käyttöönotto. Näiden vaiheiden aikana on myös tarkoitus tehdä dokumentointia yritykselle ja samalla kirjoittaa tätä dokumenttia. Työ aloitettiin käymällä läpi, mikä ja miksi on tarve työlle. Tämän jälkeen kävimme yrityksen kanssa läpi, miten työ tulisi toteuttaa. Yritys päätyi käyttämään Googlen luomaa Flutter UI -ohjelmistokehityspakettia, koska tämä oli jo ennestään käytössä yrityksessä. Sovelluksen tulisi toimia sekä web-selaimessa että Android- ja iOS-puhelimilla ja tähän tarkoitukseen Flutter soveltuu erinomaisesti. Suunnittelun jälkeen pääsin aloittamaan

sovelluksen kehittämisen. Sovellusta kehittäessä pidimme palavereita kolmannen osapuolen yrityksiin, jotka toimivat eri rajapintojen kanssa. Näitä yrityksiä olivat Navakka Group Oy, Avanio Oy ja nShift.

Huoltolomake kehitettiin Visual Studio Codella käyttäen Flutter-käyttöliittymäkehitystyökalua ja Dart-ohjelmointikieltä. Flutter on erityisesti suunniteltu mobiilisovellusten ja web-sovellusten kehitämiseen. Flutterilla kehitetyt sovellukset toimivat sulavasti eri alustoilla, kuten Androidilla, iOS:llä ja web-selaimissa, mikä on tarpeellista huoltolomakkeelle. (3.) Toimivuus eri alustoilla on tärkeää, jotta käyttäjä voi tehdä huoltopyynnön selaimessa tai halutessaan mobiililaitteella.

2 TEKNOLOGIAT

Tässä luvussa kerrotaan teknologioista ja työkaluista, joita käytettiin opinnäytetyön aikana. Työkalut ja teknologiat valittiin Natlinkin puolesta jo ennen työn aloittamista. Valitut työkalut ja teknologiat olivat jo ennestään käytössä Natlinkillä. Huoltolomake suunniteltiin ja kehitettiin alustariippumattomaksi, jolloin lomake toimisi sekä web-selaimissa että mobiilisovelluksessa.

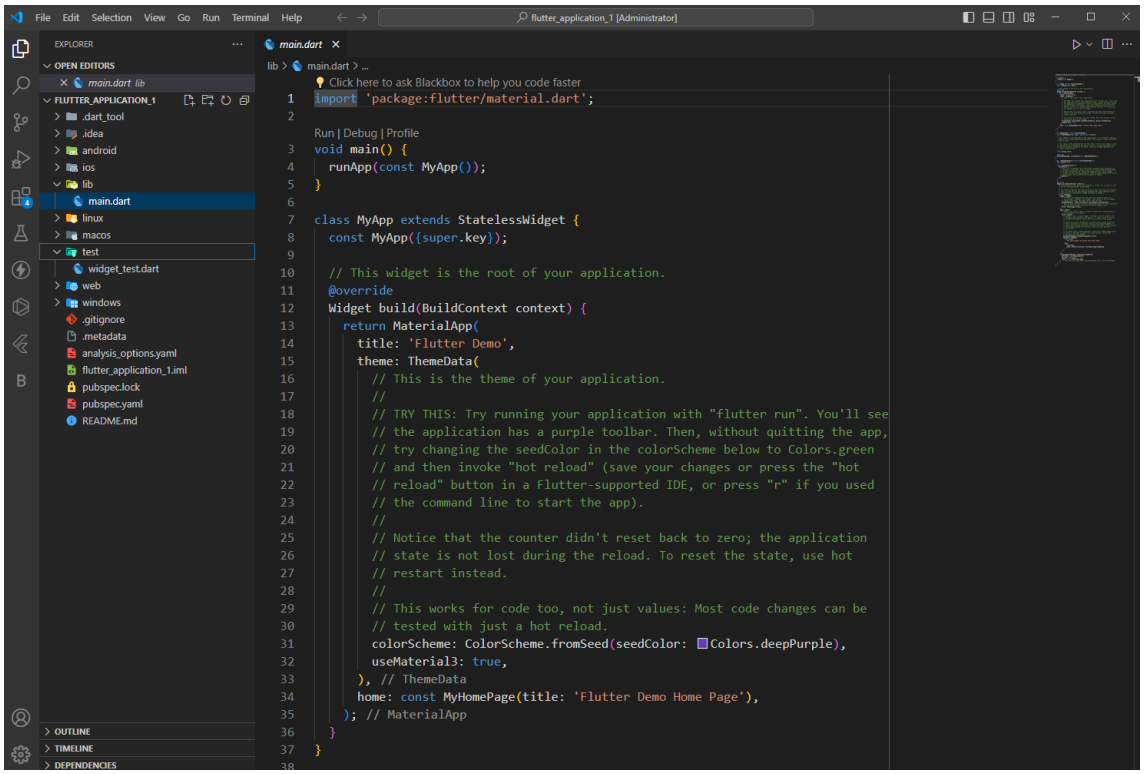
Lomake on kehitetty Flutter-käyttöliittymäohjelmistokehityspaketilla, joka perustuu Dart-ohjelmointikieleen. Lomake kommunikoi kolmannen osapuolen ohjelmistorajapinnan eli API:in kanssa, jota ylläpidetään Azure-palvelimella. API tallentaa käyttäjän täyttämän lomakkeen tiedot Azure palvelimelle. Kolmasosapuoli välittää käyttäjän tiedot Microsoft Business Centralin, josta yritys näkee tarvittavat huoltoon liittyvät tiedot.

2.1 Flutter

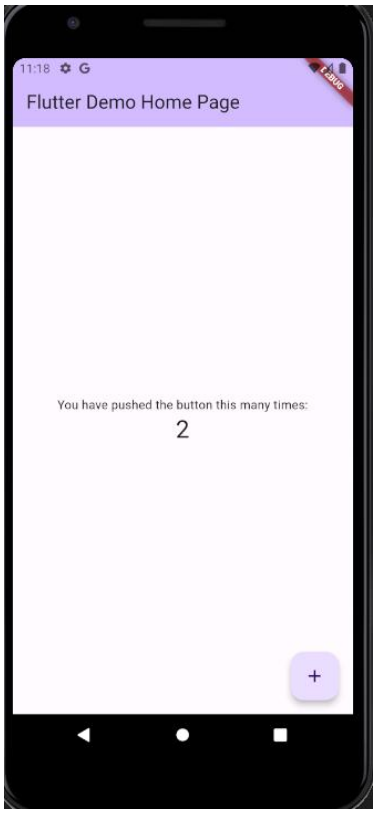
Flutter on avoimen lähdekoodin käyttöliittymäkehitystyökalu, jonka Google julkaisi 2017 keväällä. Flutter-sovellukset kirjoitetaan Dart-ohjelmointikielellä, jolla on oma rendering-moottorinsa, joka takaa natiivin tasoista suorituskykyä ja käyttäjäkokemusta eri alustoille. Flutterin avulla kehittäjä voi rankentaa natiiveja sovelluksia yhdestä koodipohjasta useille eri alustoille kuten, esimerkiksi iOS, Android, Windows ja macOS. (4.)

Flutter perustuu neljään pääkomponenttiin, joita ovat Dart-ohjelmointikieli, Flutter-moottori, Foundation-kirjasto ja Widget-kirjastot. Dart on Googlen kehittämä ohjelmointikieli, jota Flutter käyttää. Käyttämällä Dart-kieltä saavutetaan korkea suorituskyky ja Hot-Reload-ominaisuuden käyttäminen, joka tukee sovelluksen kehittämiskokemusta. Flutter-moottori, joka on pääasiassa kirjoitettu C++-ohjelmointikielellä, on vastuussa sovellusten perustoiminnasta. Moottori käyttää joko Googlen Skia-grafiikkakirjastoa tai räätälöityä Impeller-grafiikkatasoa, joka takaa alhaisen renderöintituen. Dart-kielellä kirjoitettu Foundation-kirjasto tarjoaa Flutter-sovelluksessa käytettävät perusluokat ja toiminnot, kuten Flutter-moottorin kanssa kommunikoivat API:t. Widgeteillä rakennetaan Flutter-sovelluksen käyttöliittymä, joten ovat käyttöliittymän peruselementtejä. Widgetit ovat itsenäisiä ja voivat pitää sisällään muita widgettejä, mikä luo hierarkisen rakenteen. (4.)

Flutter-projektin voi luoda käyttämällä eri ohjelmointiympäristöjä, kuten IntelliJ IDEA, Android Studio, Visual Studio Code ja Emacs. Näistä suosituimmat ovat Android Studio ja Visual Studio Code. Esimerkkinäkymä Visual Studio Codesta näkyy kuvassa 1. Flutter-projektin luominen esimerkksi käyttäen Visual Studio Codea on helppoa. Kirjoittamalla hakukenttään >Flutter: New Project, voidaan valita projektityyppi, projektin sijainti ja projektin nimi. Tämän jälkeen on luotu yksinkertainen Flutter-sovellus, jota voi lähteä kehittämään alustasta riippumatta (kuva 2). Projektista löytyy myös test-niminen kansio, joka on tarkoitettu sovelluksen automaattiseen testaamiseen.



KUVA 1. Flutter-demosovellus



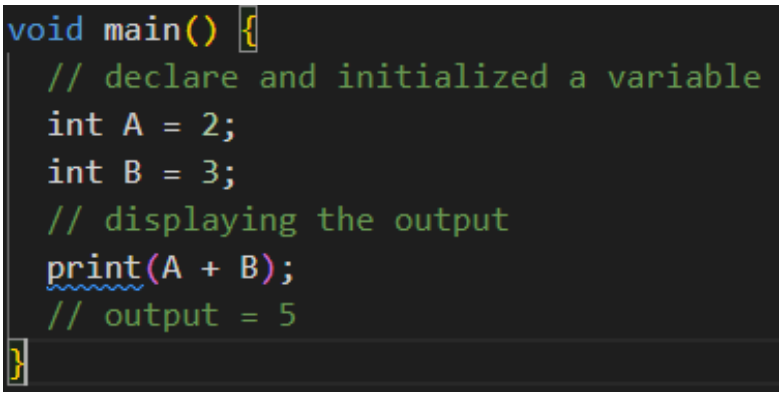
KUVA 2. Flutter-demosovelluksen käyttöliittymä

2.2 Dart

Dart on Googlen kehittämä ohjelmointikieli, jonka ovat suunnitelleet Lars Bak ja Kasper Lund. Se julkaistiin lokakuussa 2011, jonka jälkeen sitä on päivitetty tasaisesti. Dart on oliopohjainen, luokkaperusteinen ja automaattisella roskienkerääjällä varustettu kieli. Dart-kielen syntaksi on C-tyylinen (kuva 3 ja 4). Dartilla voi kehittää sovelluksia monille alustoille, kuten verkkoon, mobiililaitteisiin sekä palvelin- ja työpöytäsovelluksiin. Dart-koodin voi kääntää konekoodiksi, JavaScriptiksi tai WebAssemblyksi. Dart-kielestä on julkaistu myös ECMA-standardi (ECMA-408). (5.)

```dart
void main() {
  for (var i = 1; i <= 10; i++) {
    print(i);
  }
} // output prints number from 1 to 10
```

KUVA 3. Dart for loop



KUVA 4. Dart syntax

Dart-kieli on tyyppiturvattu. Dart käyttää staattista tyyppitarkastusta varmistaakseen, että muuttujan arvo vastaa aina muuttujan staattista tyyppiä. Tähän voidaan viitata nimellä sound typing. Tyyppien päättely on mahdollista, joten tyyppimerkinnät eivät ole pakollisia. Dartissa on sisään-rakennettu null safety-ominaisuus. Tämä tarkoittaa, että muuttujat eivät voi saada arvoa null, ellei sitä erikseen määritellä. Tämän avulla Dart suojaa kehittäjää null-poikkeus tilanteilta koodin ajon aikana staattisen koodianalyysin avulla. (6.)

Dart-kielelle on paljon erilaisia kirjastoja, jotka tarjoavat apua yksinkertaisiin tai monimutkaisiin ohjelmointitehtäviin. Esimerkiksi jokaisen Dart-ohjelman ydintoiminnot, sisäänrakennetut tyypit ja kokoelmat löytyvät dart:core-kirjastosta (7.) Datan käsittelyyn on kirjasto dart:convert, joka sisältää enkooderit ja dekooderit eri datamuotojen väliseen muuntamiseen (8.) Yleisiä datamuotoja ovat JSON ja UTF-8. Tässä opinnäytetyössä käytetään esimerkiksi dart:async- ja dart:convert-kirjastoa. Dart:async mahdollistaa asynkronisen ohjelmoinnin. (9.)

2.3 Azure

Azure on Mircrosoftin kehittämä julkinen pilvipalvelu, joka julkaistiin syyskuussa 2008. Azurea voidaan käyttää alustana virtuaalipalveluille tai kehitysalustana. Azure tarjoaa myös mobiililaitteiden hallintaa, dokumenttien suojaamista, suurten datamassojen analysointia, koneoppimista ja muita valmiita pilvipalvelukomponentteja. Azure tarjoaa myös tallennuspalveluja, joita tässä opinnäytetyössä hyödynnetään. Tallennuspalvelu tarjoaa REST- ja SDK-rajapintoja tiedon tallentamista ja

käyttämistä varten. Blob Service -palvelu mahdollistaa strukturoimattoman tekstin ja binääritiedon objektimuotoisen tallennuksen, jota voidaan käyttää HTTP(S)-polun kautta. (10.)

2.4 Visual Studio Code

Visual Studio Code on kehittäjien suosima avoimen lähdekoodin monialustainen tekstiedotori. Se äänestettiin suosituimmaksi editoriksi vuonna 2017 noin 70 %:n ääniosuudella. Visual Studio Coden ensijulkaisu oli huhtikuussa 2015. Editori on saatavilla Linuxille, MacOS:lle ja Windowsille. Visual Studio Code tukee kaikkia käytetyimpiä ohjelmointikieliä ja myös monia muita. Editorista löytyy monenlaisia ominaisuuksia, mutta merkittävimmät ovat virheenkorjaus, Git-versiohallinta, syntaksin korostus ja automaattinen koodin täydennys. Tärkeä osa editoria ovat myös laajennukset, joita käyttäjä voi asentaa tai tehdä itse. Laajennukset voivat esimerkiksi muuttaa editorin ulkonäköä, tuoda tekoälyn editoriin tai tukea uuttaa ohjelmointikieltä. Käyttäjällä on mahdollisuus muovata editoria omien tarpeidensa mukaan laajasti. (11.)

Visual Studio Code:ssa on myös erinomainen suorituskyky ja sitä on kevyt ajaa tietokoneella. Nämä erottavat sen monista muista editoreista. Editori on rakennettu Electron-ohjelmistokehykseen, joka takaa nopeuden ja responsiivisyyden. Visual Studio Code tukee etäkehitystä, mikä mahdollistaa työskentelyn etäkoneilla, virtuaalikoneilla ja Docker-konteissa. Tämän avulla pilvipohjainen kehitys ja DevOps-työskentely on helppoa, sillä kehittäjät voivat käyttää samoja työkaluja ja työtapoja riippumatta siitä, missä koodi on ajettu. Visual Studio Code:ssa on myös mahdollista monen kehittäjän työstää samaa koodia reaaliajassa Live Share-ominaisuuden avulla. Useampi kehittäjä voi muokata samaa tiedostoa samanaikaisesti, minkä myötä pariohjelmointi ja koodintarkastelu on helppoa ja tehokasta. (12.)

3 TOTEUTUS

Tässä osassa opinnäytetyötä käydään projektin toteutusta läpi suunnitteluvaiheesta käyttöönottoon asti. Ensimmäisessä luvussa käydään läpi projektin suunnittelua ja tarpeiden kartoitusta. Tarpeiden myötä laadittiin projektin kululle suunnitelma. Seuraavissa Luvuissa käydään läpi ohjelman eri osien toteutusta ja niissä käytettyjä menetelmiä. Luvuissa tarkastellaan muun muassa käytettyjä ohjelmistokehitysmenetelmiä ja koodaustekniikoita. Luvuissa käsitellään myös projektin edetessä tulleita ongelmia ja niiden ratkaisuja. Lopuksi tarkastellaan ohjelman testausta ja käyttöönottoa. Luvussa tarkastellaan, miten ohjelmaa testattiin ja sen toimivuus varmistettiin ennen käyttöönottoa.

3.1 Suunnittelu

Projektin suunnittelu oli aloitettu jo ennen kuin opinnäytetyön nimettiin minulle. Kun projekti nimettiin minulle, ensimmäinen vaihe oli käydä läpi työn tarve. Tarve työlle tuli yrityksen sisältä, kun käytössä olevassa lomakkeessa huomattiin ongelmia. Ongelmia oli lomakkeesta tulevan tiedon lähetyksessä, kun tieto ei tullut perille yritykselle asti, joten yritys joutui käyttämään ylimääräisiä resursseja tiedon käsin kirjaamiseen.

Kävimme suunnitteluvaiheessa useamman palaverin, jossa käytiin vanhan lomakkeen osia läpi kuten koodia ja sen toimivuuksia. Kävimme palavereita eri kolmansien osapuolien kanssa, jotka vastasivat lomakkeen eri rajapinnoista. Kolmannen osapuolen yrityksille ilmoitettiin suunnitelmasta uusia lomake ja uuden lomakkeen tarpeista. Kolmansien osapuolten yritykset auttoivat rajapintojen kehittämisessä ja varmistivat, että tiedonvaihto toimii uuden lomakkeen ja muiden järjestelmien välillä.

Käyttöliittymä vanhassa lomakkeessa oli myös tarpeellista uusia. Lomakkeen toiminnot ja ulkonäkö eivät vastanneet nykypäivän tasoa. Nämä heikensivät käyttäjän kokemusta lomakkeesta. Lomakkeessa olevat ongelmat olisi voitu myös korjata ilman uutta lomaketta, mutta se oli vaatinut enemmän resursseja kuin uuden kehittäminen. Yritys koki, että on parempi luoda uusi lomake kokonaan, sillä yrityksen verkkokauppaan oli myös suunnitteilla muita uudistuksia. Vanha lomake oli myös toteutettu PHP-kielellä, joka ei ole kovin moderni kieli. Päivittämällä lomaketta käyttäisi se modernimpaa ja ennestään jo yrityksessä käytössä olevaa ohjelmointikieltä.

Yritys näki uudessa lomakkeessa tärkeimpänä tarpeena korjata tiedon lähettämiseen liittyvät ongelmat. Ongelma oli, että kun asiakas täytti lomakkeen ja lähetti sen, kaikki asiakkaan tiedot eivät tulleet perille asti yrityksen Business Centraliin. Yritys joutui käsin hakemaan monen tuhannen rivin mittaisesta tietokannasta yhtä tiettyä riviä, josta asiakkaan tiedot löytyisivät. Tiedon löydyttyä yritys joutui kirjaamaan tiedon käsin Business Centraliin. Tämä ylimääräinen työ kulutti resursseja turhaan eikä ollut hyväksi yritykselle. Näiden ongelmien korjaamien vähentää ylimääräistä työtä yritykseltä ja parantaa käyttäjäkokemusta.

Yritykselle oli tärkeää myös projektin aikataulu, koska oli tärkeää ottaa käyttöön toimiva versio lomakkeesta ennen metsästyskauden aloitusta. Kun metsästystä ei tapahtunut paljoa, oli loistava tilaisuus suunnitella ja kehittää lomaketta, sillä huoltotilauksia ei tullut paljoa tänä aikana. Huoltolomake saattoi olla kokonaan poissa käytöstä asiakkailta päivällä, ja koska huoltotilauksia ei tullut paljoa tai ollenkaan, oli erinomainen mahdollisuus keskittyä lomakkeen kehittämiseen. Metsästyskauden ollessa päällä tilauksia saattoi tulla satoja viikossa. Tässä vaiheessa oli tärkeää saada lomake toimimaan ennen kuin huoltotilausten määrä kasvaisi merkittävästi.

Alkuperäisen huoltolomakkeen käyttöliittymä oli tehty yksinkertaiseksi ja asiakkaalle helppokäyttöiseksi. Käyttöliittymän päivittäminen ei ollut projektin tärkein osa yritykselle. Sitä tultaisiin päivittämään enemmän tulevaisuudessa. Käyttöliittymää tulisi kuitenkin päivittää ja parantaa projektin edetessä, siten että näkymä pysyisi ennallaan ja olisi edelleen yksinkertainen ja helppokäyttöinen. Vanhassa huoltolomakkeessa käyttöliittymä ei kuitenkaan ollut täydellinen, vaan se vaati muutamia korjauksia ja parannuksia. Lomakkeen käyttöliittymästä puuttui myös ominaisuuksia, jotka tekisivät asiakkaan käyttäjäkokemuksesta mieluisamman. Käyttöliittymästä tulisi myös tehdä enemmän ystävällinen käyttäjän silmille. Tällöin käyttöliittymä on miellyttävämpi ja sitä on helpompi katsoa. Flutterin ja sen pakettien avulla käyttöliittymää on helppo päivittää. Flutter myös takaa, että käyttöliittymän toiminnot ovat sulavia ja se on silmille miellyttävä. Suunnitteluvaiheessa tuli myös ilmi, että sama lomake tulisi toimimaan web-pohjaisena ja tulevaisuudessa myös mobiilisovelluksessa. Flutter antaa tähän hyvän mahdollisuuden.

Projektin suunnitteluvaiheessa kävimme myös läpi millaisia uusia ominaisuuksia lomakkeeseen olisi hyvä lisätä tarvittavien korjauksien lisäksi. Näiden lisäysten tarkoituksena on helpottaa yrityksen työtä ja laajentaa lomakkeen käyttäjäkuntaa. Lomakkeen käyttäjäkuntaa saadaan lisäämällä lomakkeelle uusia kieliä kuten norja ja ranska. Tämä kielten lisääminen tekee lomakkeesta entistä

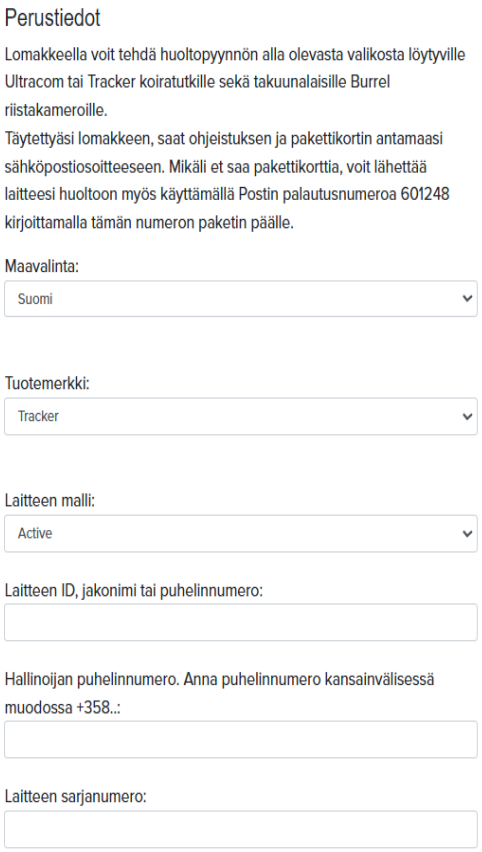
saavutettavamman kansainvälisille käyttäjille ja tarjoaa heille mahdollisuuden täyttää lomake omalla äidinkielellään. Kielten lisääminen myös laajentaa potentiaalista käyttäjien määrää. Tarkoituksena oli myös tehdä uusien kielien lisääminen tulevaisuudessa mahdollisimman helpoiksi.

Yrityksen työtä helpottavia kehitysideoita oli useita, ja ne tähtäävät prosessien tehostamiseen ja virheiden vähentämiseen. Parannusideoita olivat esimerkiksi automaattiset tietojen täyttämiset ja tarkistukset Business Centralissa. Nämä ovat merkittäviä parannuksia, joiden avulla voimme vähentää manuaalisen työn ja virheiden määrää. Tämä on erityisen tärkeää, kun käsitellään asiakastietoja, joiden tarkkuus ja luotettavuus ovat erittäin tärkeitä.

Kaikki nämä muutokset, parannukset ja korjaukset ovat tärkeitä, jotta lomake vastaisi paremmin yrityksen ja sen asiakkaiden tarpeisiin. Muutokset mahdollistavat paremman ja tehokkaamman tietojenkäsittelyn, mikä säästää resursseja ja parantaa asiakaskokemusta. Samalla ne tekevät lomakkeesta modernimman ja käyttäjäystävällisemmän, mikä on olennaista nykypäiväisessä lomakkeessa.

3.2 Käyttöliittymä

Sovelluksen käyttöliittymä säilyy pääosin samana uudessa versiossa, sillä vanhan lomakkeen käyttöliittymä todettiin toimivaksi eikä sen perusrakennetta tarvitse muuttaa merkittävästi. Uudessa versiossa käyttöliittymän rakenne pysyy pitkälti ennallaan, mutta muutoksia tehdään ulkonäköön ja käyttäjäkokemukseen. Lomakkeen käyttöliittymää päivitetään myös enemmän responsiiviseksi eri laitteelle. Vanha käyttöliittymä on toteutettu HTML-kielellä, jota käytetään verkkosivujen ja sovellusten käyttöliittymien rakenteen määrittämiseen. Käyttöliittymä on yksisivuinen, jossa asiakas voi täyttää ja lähettää lomakkeen. Käyttöliittymä koostuu kolmesta osasta: perustiedot, takuutiedot ja yhteystiedot, ja painikkeen mistä lomakkeen voi lähettää. Vanhasta käyttöliittymästä saatiin hyvä malli uudelle versiolle (kuva 5).



KUVA 5. Vanhan lomakkeen Perustiedot osio

Lomakkeen uusi käyttöliittymä on rakennettu käyttäen Flutterin widget-elementtejä, jotka tarjoavat joustavuutta ja tehokasta sovelluksen kehittämistä. Widgetit ovat kuin uudelleekäytettäviä rakennuspalikoita, joiden avulla sovellusten ulkoasun ja käyttäytymisen määrittäminen on helppoa.

3.2.1 Pää-widgetit

Uusi käyttöliittymä rakentuu kolmesta pää-widgetistä, jotka rakentavat toimivan ja sulavan käyttöliittymän lomakkeelle. Koodissa on määritelty MyFrom-luokka, joka perii StatelessWidget-luokan. Tämä kuvaa sovelluksen käyttöliittymän rakennetta (kuva 7). StatelessWidget tarkoittaa, että se ei säilytä tilaa eikä sen tila muutu luomisen jälkeen (13.) Ensimmäinen kolmesta pää-widgetstä on HeadingWidget, jonka tehtävänä on näyttää sovelluksen otsikko ja samalla kertoa käyttäjälle, mitä lomakkeella tehdään (kuva 7).

Toinen pää-widget on LanguageSelector, joka antaa käyttäjälle mahdollisuuden valita lomakkeeseen haluamansa kielen (kuva 7). Mahdollisuus vaihtaa lomakkeen kieltä halutessaan parantaa käyttäjäkokemusta merkittävästi, eritysesti kun lomake on käytössä monessa eri maassa. Widgetissä käytetään Flutter dropdown_button2 pakettia. Tästä paketista käytetään `DropdownButtonFormField2`-komponenttia, joka antaa alas vedettävän valikon, jossa on valittavat kielet (kuva 6). Flutter tarjoaa myös oman DropdownButtonFormField-komponentin, mutta tämä komponentti oli liian yksinkertainen eikä vastannut lomakkeen tarpeita. Tämän vuoksi päädyin käyttämään ulkopuolista pakettia. Jokaisella kielellä on maakohtainen lippu, joka helpottaa käyttäjää tunnistamaan valittavan kielen visuaalisesti. käyttäjä valitsee haluamansa kielen valikosta, jonka jälkeen lomake näytetään valitsemalla kielellä.

KUVA 6. DropdownButtonFormField2-komponentti

Tällä hetkellä lomakkeessa on kuusi eri kielivaihtoehtoa. Uusien kielten lisäämien on tehty helpoksi `flutter_localization`-paketin avulla. Paketti tarjoaa työkaluja käännöstiedostojen hallintaan ja kielten vaihtamiseen, mikä tekee uusien kielten lisäämisestä helppoa (13.) Kehittäjien on helposti mahdollista lisätä uusia kielivaihtoehtoja ja päivittää olemassa olevia käännöksiä tarvittaessa ilman suuria muutoksia koodiin. Kehittäjän tarvitsee vain luoda käännöstiedostotarvittavat muuttujat _changeLanguage nimiseen funktioon, jonka jälkeen

`flutter_localization`-paketti tekee loput tarvittavat muutokset. Nyt kehittäjän täytyy vain lisätä uusi vaihtoehto kielivalikkoon.

Viimeinen pää-widget FormWiget muodostaa lomakkeen rungon, jonka käyttäjä voi täyttää tarvittavilla tiedoilla. FormWidget sisältää lomakkeen erilaisia kenttiä ja niiden validointeja, jotka varmistavat, että oikeanlainen tieto syötetään kenttiin. Widget sisältää myös lähetys-painikkeen, josta käyttäjä voi lähettää täyttämänsä lomakkeen tiedot eteenpäin (kuva 7). Yhdistämällä nämä kolme pää-widgettiä saadaan toimiva ja helppokäyttöien käyttöliittymä lomakkeelle. FormWidget koostuu vielä kolmesta sisäisestä widgetistä. Näissä kolmessa widgettissä on lomakkeeseen liittyviä tietoja, asiakkaan täytettäviä kenttiä ja laitteen takuu-tietojen kentät.

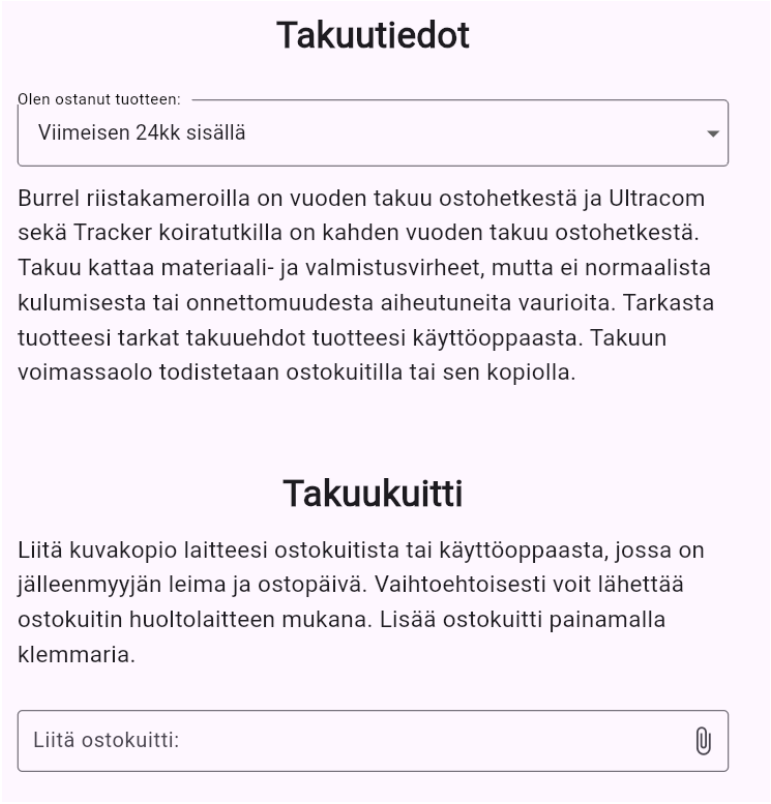
```
class MyForm extends StatelessWidget {
  const MyForm({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(AppLocalizations.of(context)!.title),
      ), // AppBar
      body: const SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: <Widget>[
            HeadingWidget(),
            SizedBox(height: 20),
            LanguageSelector(),
            SizedBox(height: 20),
            FormWidget(),
          ], // <Widget>[]
        ), // Column
      ), // SingleChildScrollView
    ); // Scaffold
  }
}
```

KUVA 7. MyFrom class

3.2.2 Pää-widgetin sisäiset widgetit

Ensimmäinen sisäinen widget on `form_info`. Tämä näyttää tekstiosion lomakkeen perustiedot osasta, josta käyttäjä voi lukea ohjeita lomakkeesta. Widgetissä on kolme tekstiosiota: basicInfo. basicText1 ja basicText2, jotka kääntyvät asiakkaan valitseman kielen mukaan. Teksteistä asiakasta saa tärkeää tietoa lomakkeen toimivuudesta. FormInfo käyttää Colum-widgettiä, mikä mahdollistaa lapsielementtien asettelun pystysuuntaan. Widgettiä käytettiin, koska se mahdollistaa tekstien asettelun pystysuuntaiseen järjestykseen helposti.

Toinen sisäinen widget on WarrantyInformationWidget. Tällä widgetillä asiakas pystyy täyttämään laitteen takuutiedot. Takuutiedoilla määritellään huollon maksupolitiikka, jos laite on yli 24 kk vanha tulee asiakkaan maksaa laitteen huolto itse. Laitteen ollessa alle 24 kk vanha voi asiakas lisätä lomakkeeseen ostokuitin laitteesta, jolloin yritys voi todistaa laitteen iän (kuva 8). Alle 24 kk vanhat laitteet menevät takuuseen, jolloin asiakas ei joudu maksamaan huollosta.



KUVA 8. Kenttä takuukuitin liittämistä varten

WarrantyInformationWidget-widgetissä käytetään myös DropdownButtonFormField2-komponenttia. Ensimmäisessä valikossa asiakas valitsee laitteen ostoajan: joko "Viimeisen 24 kk sisällä" tai "Yli 24 kk sitten". Riippuen asiakkaan valinnasta, lomake näyttää joko maksutavan valinnan tai kohdan, johon asiakas voi lisätä ostokuitin. Asiakkaan mahdolliset matkustavat riippuvat asiakkaan maasta. Postiennakko on käytössä vain Suomessa, muissa maissa on käytössä vain Klarna.

Kun asiakas valitsee "Yli 24 kk sitten" -vaihtoehdon, lomake näyttää erilaisia tekstiosioita, joissa kerrotaan tärkeää tietoa liittyen laitteiden takuuseen käyttäen Flutterin labelText-ominaisuutta. Tämä mahdollistaa selkeän ja informatiivisen tekstin näyttämisen, joka auttaa asiakasta ymmärtämään takuuehdot ja tarvittavat toimenpiteet.

Ostokuitin lisääminen on tehty mahdolliseksi käyttämällä `file_picker`-pakettia. Tämä paketti mahdollistaa tiedoston tai tiedostojen valitsemisen käyttäen laitteen natiivia resurssienhallintaohjelmaa (14.) Näin asiakas voi helposti liittää ostokuitin lomakkeeseen, mikä varmistaa laitteen iän todistamisen ja helpottaa takuun hyödyntämistä.

Kuva 9, ostokuitin lisääminen perustuu IconButton-widgettiin. Tämä widget käyttää kuvaketta, joka vaihtuu sen mukaan, onko ostokuittia valittua vai ei. Kuvake vaihtuu, jos `_fileNameController.text` sisältää tekstiä. Kun `f_fileNameController.text` sisältää tekstiä kuvake muuttuu poistamisikoniksi, muulloin ikoni on klemmari. Tiedoston valitseminen aloitetaan painamalla klemmaria, joka käynnistää `file_picker`-paketin toiminnon. Toiminto avaa resurssienhallintaohjelman, joka on määritelty hyväksymään vain näiden tyyppien tiedostot: jpg, jpeg ja pdf. Asiakkaan valittua tiedoston, sen nimi ja sisältö tallennetaan, ja tiedoston nimi näytetään lomakkeessa. Tämä antaa visuaalisen vahvistuksen onnistuneesta tiedoston lisäämisestä. Jos asiakas haluaa poistaa tiedoston, hän voi painaa poistoikonia, joka tyhjentää kentän.

```
suffixIcon: IconButton(
  icon: _fileNameController.text.isNotEmpty
      ? const Icon(Icons.clear)
      : const Icon(Icons.attach_file),
  onPressed: () async {
    if (_fileNameController.text.isNotEmpty) {
      setState(() {
        _fileNameController.clear();
      });
      widget.onFileSelected(
          '', null); // Notify file removal
    } else {
      final result = await FilePicker.platform.pickFiles(
        type: FileType.custom,
        allowedExtensions: ['jpg', 'jpeg', 'pdf'],
      );
      if (result != null) {
        final fileName = result.files.single.name;
        final fileBytes = result.files.single.bytes;
        setState(() {
          _fileNameController.text = fileName;
        });
        widget.onFileSelected(fileName, fileBytes!);
      }
    }
  },
), // IconButton
```

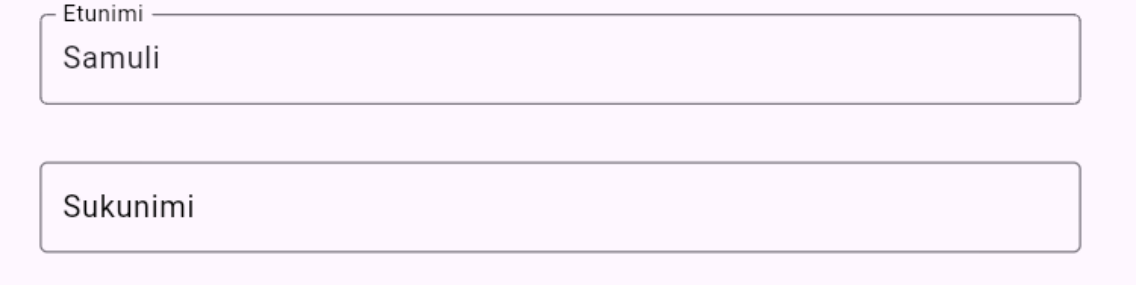
KUVA 9. Ostokuitin lisäämistä käsittelevä koodi

Kolmas sisäinen widget on ContactInfoWidget. Tämä widget pitää sisällään asiakkaan omat tiedot, kuten nimen, osoitteen ja puhelinnumeron. Widget on suunniteltu keräämään nämä tiedot käyttäjäystävällisellä ja intuitiivisella tavalla, samalla varmistaen, että tiedot ovat oikeassa muodossa.

ContactInfoWidget-widgetissä käytetään useita TextFormField-komponentteja keräämään eri tietoja, kuten etu- ja sukunimen, osoitteen, postinumeron, kaupungin, sähköpostin ja puhelinnumeron (kuva 10 ja 11). Kaikilla kentillä on oma valmiiksi asetetut tarkistustoiminnot, joka varmistavat, että käyttäjä täyttää kentät oikein. Esimerkiksi sähköpostikenttä tarkistaa, että sähköpostiosoite on syötetty ja että varmistussähköpostiosoite vastaa alkuperäistä. Jos sähköpostit eivät täsmää, lomake antaa virhettä lähetyksen yhteydessä ja pyytää asiakasta korjaamaan tämän.

```
child: TextFormField(
  cursorColor: AppStyles.cursorColor,
  controller: widget.firstNameController,
  decoration: AppStyles.inputDecoration.copyWith(
    labelText: AppLocalizations.of(context)!.fName,
  ),
  validator: (value) {
    if (value!.isEmpty) {
      return AppLocalizations.of(context)!.validatorFill;
    }
    return null;
  },
), // TextFormField
```

KUVA 10. Esimerkki TextFormField-komponentista



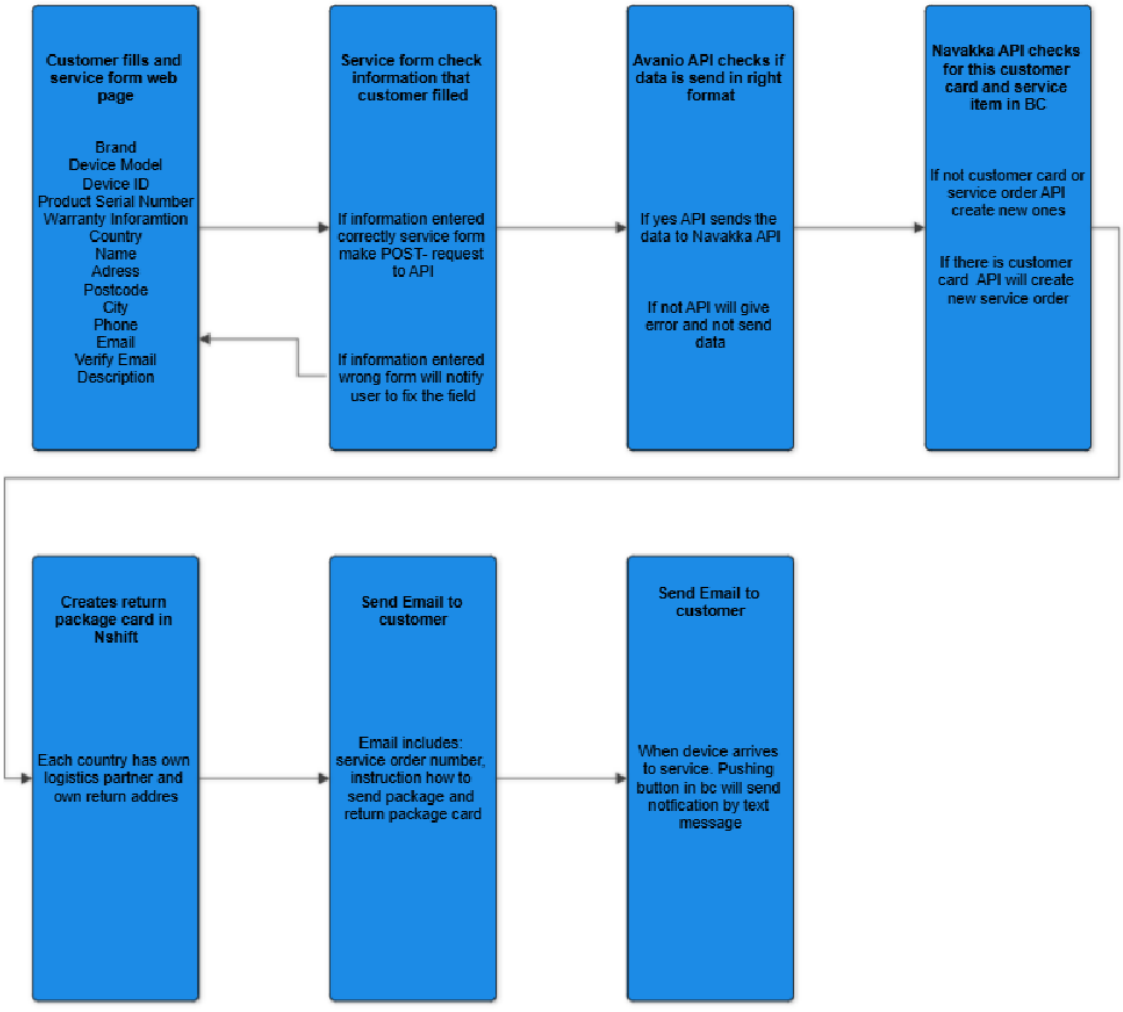
KUVA 11. TextFormField-komponentti sovelluksessa

Widget sisältää myös DropdownButtonFormField2-komponentin, joka mahdollistaa asiakkaan maan valinnan. Valittu maa vaikuttaa lomakkeen muihin kenttiin, kuten maksutavan valintaan. Jos valittu maa ei tue valittua maksutapaa, lomake nollaa maksutavan valinnan. Tämä varmistaa, että lomakkeen tiedot ovat aina oikeat. Lisäksi widget tarjoaa myös asiakkaalle mahdollisuuden lisätä viestin lomakkeeseen. Tämä saadaan aikaan käyttäen TextFormField-komponenttia, joka sallii usean tekstirivin syöttämisen. Tämä on erityisen hyödyllistä, jos asiakas haluaa antaa lisätietoja liittyen laitteen ongelmaan.

Uuden lomakkeen myötä voidaan todeta, että uuden käyttöliittymän rakentaminen Flutterilla tarjoaa merkittäviä etuja verrattuna aiempaan HTML-pohjaiseen ratkaisuun. Uusi käyttöliittymä on paitsi responsiivisempi ja visuaalisesti miellyttävämpi, myös käyttäjäkokemukseltaan paranneltu. Tämä saavutetaan hyödyntämällä Flutterin monipuolisia ja joustavia widgettejä, jotka mahdollistavat käyttöliittymän elementtien tehokkaan hallinnan ja muokkaamisen.

3.3 Business Centraliin liittäminen

Kun asiakas lähettää lomakkeen painamalla "lähetä" -painiketta, lomakkeen tiedot välitetään POST-pyynnöllä Avanio Oy ylläpitämälle Azure-palvelimelle. Tätä varten käytetään REST-rajapintaa, joka on integroitu osaksi Business Centralia. POST-pyynnön avulla lomakkeen tiedot siirtyvät Avanion palvelimelle, jossa ne käsitellään. Kun Azure-palvelin vastaanottaa POST-pyynnön, se analysoi sisältämät tiedot. Tämä prosessi sisältää tiedon tarkistamisen, sen tallentamisen tietokantaan, uuden asiakaskortin luomisen, huoltopyynnön luomisen, tilausten tekemisen tai muiden liiketoimintaoperaatioiden suorittamisen Business Centralissa. Avanion palvelin kommunikoi Navakka Group Oy rajapinnan kanssa, joka viime kädessä tallentaa tiedot Natlink Oy luettavaksi Business Centraliin (kuva 12). Kun tiedot on käsitelty ja tallennettu onnistuneesti, nShift lähettää asiakkaalle toimitus- ja maksuohjeet sähköpostitse ja puhelimitse.



KUVA 12. Sovelluksen vuokavio

Lomakkeen lähettää JSON multipart/forn-dataa. JSON multipart/form-data on tiedonsiirtotapa, jota käytetään lähettämään JSON-muotoista dataa palvelimelle POST-pyynnön avulla. Tiedonsiirtotapa mahdollistaa moniosaisen datan lähettämisen, mukaan lukien tekstitiedot ja tiedostot. Kun lomakkeen tiedot lähetetään JSON multipart/form-data-muodossa, ne jaetaan useisiin osiin tai kenttiin, jotka sisältävät erityyppisiä tietoja (Kuva 13). Lomake lähettää esimerkiksi: billing, customer, service ja warranty kenttiä. JSON-muotoiset tekstitiedot sijoitetaan yhteen tai useampaan osaan, jossa jokainen osa edustaa yhtä kenttää tai muuttujaa lomakkeella. Lisäksi tiedostot, kuten kuvat tai asiakirjat, lähetetään osina, joissa jokainen osa sisältää tiedoston tiedot ja sisällön.


```
{
  "device": {
    "model_number": "54634",
    "model_name": "Active",
    "serial_number": "123456798"
  },
  "billing": {
    "payment_method": "Postiennakko"
  },
  "customer": {
    "first_name": "Samuli",
    "last_name": "Virkkula",
    "address": {
      "address_1": "Esimerkki osote",
      "address_2": "",
      "postcode": "90570",
      "city": "Oulu",
      "country": "FI"
    },
    "phone": "+358931523713",
    "email": "etunumi.sukunimi@email.com",
    "languageCode": "FIN"
  },
  "service": {
    "number": "SRV_1310_31oksz7kz2jt0c7s7p0kpjlsiltexiwb",
    "description": "Esimerkki json multipart/form-data file "
  },
  "company": {
    "company_name": "",
    "business_id": "",
    "retailer_name": ""
  },
  "shippingagent": {
    "code": "POSTI",
    "service_code": "POSTIPAKET"
  },
  "additionaldescription": ""
}
```


KUVA 13. Esimerkki JSON multipart/form-datasta

Warranty-kenttään liitetään asiakkaan lataama ostotodistus käyttämällä uploadFile-funktiota (Kuva 14). Tämä funktio lähettää tiedostot palvelimelle. Se ottaa URL-osoitteen, tiedoston bittimuodossa ja tiedostonimen parametreina. Funktio luo POST-pyynnön, liittää tiedoston osaksi pyyntöä ja asettaa tarvittavat otsikot, jotta palvelin voi tunnistaa ja käsitellä tiedoston oikein. Lopuksi se lähettää pyynnön palvelimelle ja tulostaa tilan (onnistunut tai epäonnistunut) vastauksen statuskoodin perusteella. Tämä varmistaa, että asiakkaan lähettämä ostotodistus tallennetaan ja liitetään asianmukaisesti Business Centraliin käsiteltäväksi ja tietojen täydellisyyden varmistamiseksi.

```
Future<void> uploadFile(
    String url, List<int> fileBytes, String fileName) async {
  var request = http.MultipartRequest('POST', Uri.parse(url));

  request.files.add(http.MultipartFile.fromBytes(
    'warranty',
    fileBytes,
    filename: fileName,
    contentType: MediaType('application', 'pdf'),
  )); // http.MultipartFile.fromBytes

  request.headers['Content-Disposition'] = 'inline; filename="$fileName"';

  var response = await request.send();
  if (response.statusCode == 200) {
    print('File uploaded successfully.');
  } else {
    print('File upload failed with status: ${response.statusCode}');
  }
}
```

KUVA 14. uploadFile funktio

Yritykselle on tärkeää varmistaa, että asiakkaiden huoltotilaukset voidaan järjestelmällisesti tunnistaa ja seurata. Tätä varten käytetään uniikkeja huoltotilausnumeroita, jotka luodaan generateServiceCode-nimisellä funktiolla (kuva 15). Tämä funktio tuottaa yksilöllisiä koodisarjoja, jotka toimivat tunnistetietoina asiakkaan huoltotilauksille.

```
String generateServiceCode() {
  const String numbers = '0123456789';
  const String chars = 'abcdefghijklmnopqrstuvwxyz0123456789';
  final Random rnd = Random();

  String digits = String.fromCharCodes(Iterable.generate(
      4, (_) => numbers.codeUnitAt(rnd.nextInt(numbers.length)))); // Iterable.generate // String.fromCharCodes

  String alphanumeric = String.fromCharCodes(Iterable.generate(
      32, (_) => chars.codeUnitAt(rnd.nextInt(chars.length)))); // Iterable.generate // String.fromCharCodes

  return 'SRV_$digits' '_' '$alphanumeric';
}
```

KUVA 15. generateServiceCode funktio

Funktio käyttää satunnaisia numeroita ja merkkejä luodakseen uniikin koodin. Ensinnäkin se generoi nelinumeroisen numerosekvenssin ja sitten 32 merkin pituisen merkkijonon, joka sisältää sekä kirjaimia että numeroita. Nämä kaksi osaa yhdistetään luomaan lopullinen huoltotilausnumero, joka alkaa tunnisteella "SRV_" ja jatkuu ensin nelinumeroisella numerosarjalla ja sen jälkeen 32 merkin pituisella merkkijonolla. Näin saadaan uniikki sarjanumero, jonka avulla on helppo tunnistaa tilauksia.

4 TULOKSET JA JATKOKEHITYS

Projektin tuloksena saatiin toimiva huoltolomake, joka korjaa edellisen lomakkeen ongelmat ja parantaa lomakkeen visuaalista näköä. Asiakkaan lähettämät tiedot tulevat oikeassa muodossa Business Centraliin, ilman ylimääräistä työtä yritykseltä. Käyttöliittymästä saatiin moderni ja käyttäjäystävällinen, mikä helpottaa asiakkaiden lomakkeiden täyttämistä ja vähentää virheiden määrää.

Uusi huoltolomake on järjestelmäriippumaton, joten asiakkaat voivat täyttää sen helposti kaikilla laitteilla, olipa kyseessä tietokone, tabletti tai älypuhelin. Tämä varmistaa, että palvelua voidaan käyttää joustavasti missä ja milloin tahansa. Järjestelmäriippumattomuus mahdollistaa myös lomakkeen integroimisen tulevaisuudessa yrityksen mobiilisovelluksiin, mikä laajentaa sen käyttömahdollisuuksia ja parantaa asiakaskokemusta entisestään.

Uuden huoltolomakkeen merkittävin parannus on sen toimivuus lomakkeen ja Business Centralin välillä. Se korjaa edellisen version ongelmat ja varmistaa, että kaikki tiedot tulevat perille oikeassa muodossa ilman virheitä. Tämä vähentää manuaalisen työn tarvetta ja parantaa tietojen käsittelyn tarkkuutta, joka oli projekti päätavoite. Lomakkeen lähettämät tiedot saatiin oikeassa muodossa yritykselle rakentamalla uusi lomake, joka kommunikoi Azure-palvelimen kanssa. Tavoitteessa onnistuttiin suunnitelmien mukaisesti. Tavoitteeseen päästiin järjestämällä monia palavereita Navakka Group Oy:n ja Avanio Oy:n osapuolien kanssa, joissa keskustelimme uuden lomakkeen suunnitellusta ja tarpeista. Näiden myötä uuden lomakkeen kehittäminen oli helppoa.

Business Centraliin kehitettiin korjauksien lomassa myös parannuksia, jotka tekevät järjestelmästä entistä tehokkaamman ja joustavamman. Näihin parannuksiin kuuluvat muun muassa asiakkaan kielikoodin määrittäminen, toimitustavan automaattinen määritys ja kirjausryhmien hallinta asiakkaan maan perusteella. Näiden parannusten ansiosta järjestelmä pystyy käsittelemään kansainvälisiä asiakkaita entistä paremmin ja joustavammin.

Tuloksena saatiin lomakkeelle myös uudistettu ja paranneltu käyttöliittymä, joka on moderni ja käyttäjäystävällinen. Käyttöliittymän uusi version on visuaalisesti houkuttelevampi, joka edesauttaa asiakaskokemusta. Käyttöliittymä sai mallinsa vanhasta versiosta.

Uusi käyttöliittymä sisältää useita käyttäjäkokemusta parantavia muutoksia, jotka helpottavat lomakkeen täyttämistä ja tekevät siitä sujuvamman prosessin. Näihin muutoksiin kuuluu muun muassa kenttien selkeämpi asettelu ja ohjeistukset, jotka auttavat asiakkaita syöttämään oikeat tiedot. Lisäksi lomakkeeseen on lisätty uusia kenttiä, jotka vastaavat paremmin yrityksen tarpeita ja mahdollistavat tarkemman tiedonkeruun.

Lomakkeen jatkokehitystä on tarkoitus tehdä vielä tulevaisuudessa. Tähän kuuluu uusien ominaisuuksien kehittäminen ja vanhojen kehittäminen. Esimerkiksi automaattisten kenttien täyttämistä Business Centralissa voidaan vielä kehittää eteenpäin. Yksi suuri jatkokehityksen aihe on asiakkaan tietojen automaattinen täyttäminen. Tämä tapahtuisi niin, että kun asiakas antaa oman asiakasnumeron lomakkeelle, lomake hakee asiakkaan tiedot Firebase-tietokannasta ja täyttää lomakkeen näillä. Parannus olisi erittäin hyvä ja tekisi lomakkeesta entistä käyttäjäystävällisemmän. Saman tyyppinen jatkokehitys olisi myös yrityksille. Kun yritys syöttää esimerkiksi Y-tunnuksen hakee lomake yrityksen tiedot tietokannasta. Käyttöliittymälle jatkokehitystä on pitää se modernina ja käyttäjäystävällisenä.

5 POHDINTA

Opinnäytetyössä tavoitteena oli toteuttaa uusi huoltolomake Natlink Oy:n verkkosivuille. Vanhassa lomakkeen ongelmia ei todettu järkeväksi korjata, joten aloitettiin uuden suunnittelu ja kehittäminen. Työn päätavoitteena oli saada toimiva huoltolomake, joka korjaisi aiemman version ongelmat. Tavoitteessa onnistuttiin ja samalla pystyimme kehittämään käyttöliittymää ja tekemään muita parannuksia lomakkeeseen.

Huoltolomakkeen uudistusprojektiin sisältyi monenlaista työtä, kuten frontend- ja backend-kehitystä, suunnittelua, palavereita ja dokumentointia. Käyttöliittymän kehittäminen ei vienyt paljoa aikaa suunnitellusta, sillä uusi käyttöliittymä perustuu vanhan lomakkeen käyttöliittymään. Vaikka käyttöliittymän kehittäminen vei vähemmän aikaa, opin kuitenkin paljon uusia asioita Flutterista ja Dartista, mikä hyödyttää minua tulevaisuudessa.

Projektissa ehdottomasti suurin osa ajasta meni lomakkeen yhdistämiseen Business Centraliin. Ensimmäinen onnistunut POST-pyyntö saatiin nopeasti, mutta sen hiominen oikeanlaiseksi vei aikaa. Tässä vaiheessa oli muutamia ongelmia, jotka saatiin lopuksi ratkaistua. Ongelmia olivat esimerkiksi rajapintojen toimivuuksien välillä. Ongelmia ilmeni myös palvelimien toimivuudessa, osana aikaa. Palvelimet olivat kolmannen osapuolen hallussa, eivätkä aina kerenneet vastata pyyntöihini, joten tämä vei aikaa.

Dokumentoinnin merkitys nousi esiin työn edetessä, kun huomasin sen vaativan odotettua enemmän aikaa ja resursseja. Siirtyminen kehittämisen ja dokumentoinnin välillä oli haastavaa, koska molemmat vaativat omistautumista ja keskittymistä. Tässä vaiheessa tunnistin tarpeen tehokkaammalle ajanhallinnalle ja priorisoinnille, jotta saan varmistettua sekä työn laadun että dokumentaation kattavuuden.

Suunnittelun tärkeys korostui entisestään, kun törmäsin haasteisiin kehittämisen kanssa. Hyvin suunniteltu työ helpottaa työn etenemistä ja auttaa ennakoimaan mahdollisia ongelmia. Suunnitelman avulla tiedän, mitä pitää tehdä ja missä järjestyksessä. Tämä säästää aikaa ja vähentää tulevia ongelmia

Yhteenvetona voidaan todeta, että uuden huoltolomakkeen kehitysprojekti oli onnistunut. Kehitysprojekti paransi yrityksen prosesseja ja lomakkeen toimivuutta merkittävästi. Opinnäytetyön aika opin paljon uutta kehittämisestä, suunnitellusta, dokumentoinnista ja yrityksessä työskentelystä. Jatkossa voin hyödyntää työssä hankittuja osaamisia ja kokemuksia tulevissa projekteissa.

LÄHTEET

1. Natlink 2024. Etusivu. Hakupäivä 12.4.2024 https://natlink.se/language/fi/etusivu/

2. Natlink 2024. Open Air Group lanseeraa Natlinkin – Pohjoismaiden suurimman metsästysteknologiayrityksen. Hakupäivä 12.4.2024 https://natlink.se/language/fi/open-air-group-lanseeraa-natlinkin-pohjoismaiden-suurimman-metsastysteknologiayrityksen-2/

3. Flutter 2024. Etusivu. Hakupäivä 15.4.2024 https://flutter.dev/

4. Wikipedia 2024. Flutter (software). Hakupäivä 19.4.2024 https://en.wikipedia.org/wiki/Flutter_(software)

5. Wikipedia 2024. Dart (Programming language) Hakupäivä 3.5.2024 https://en.wikipedia.org/wiki/Dart_(programming_language)

6. Dart 2024. Dart overview. Hakupäivä 16.5.2024 https://dart.dev/overview

7. Dart API Docs 2024. dart:core library. Hakupäivä 18.5.2024. https://api.dart.dev/stable/3.4.2/dart-core/dart-core-library.html

8. Dart API Docs 2024. dart:convert library. Hakupäivä 18.5.2024 https://api.dart.dev/stable/3.4.2/dart-convert/dart-convert-library.html

9. Dart Docs 2024. Asynchronous programming: futures, async, await. cHakupäivä 18.5.2024 https://dart.dev/codelabs/async-await

10. Mircrosfot Azure 2024. Frontpage. Hakupäivä 20.5.2024 https://azure.microsoft.com/en-us

11. Wikipedia 2024. Visual Studio Code. Hakupäivä 20.5 https://en.wikipedia.org/wiki/Visual_Studio_Code

12. Visual Studio Code Docs 2024. Getting Started. Hakupäivä 20.5 https://code.visualstudio.com/docs

12. Flutter API Docs 2024 StatelessWidget class. Hakupäivä 25.5 https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html

13. Flutter API Docs 2024 Internationalizing Flutter apps. Hakupäivä 25.5 https://docs.flutter.dev/ui/accessibility-and-internationalization/internationalization

14. pub.dev 2024. file_picker 8.0.3. Hakupäivä 29.5 https://pub.dev/packages/file_picker