SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

CALENDARAPP

Kalenterimobiilisovellus ja palaute

TEKIJÄ/T: Arttu Väisänen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Arttu Väisänen	
Työn nimi Calendarapp kalenterimobiilisovellus ja palaute	
Päiväys 9.5.2019	Sivumäärä/Liitteet 23/1
Ohjaaja(t) Mikko Pääkkönen, lehtori, Keijo Kuosmanen, lehtori	
Toimeksiantaja/Yhteistyökumppani(t) Pelvic Fysio Oy	

Tiivistelmä

Opinnäytetyön tavoitteena oli toteuttaa monialustainen mobiilisovellus Pelvic Fysio Oy:lle, julkaista se heidän testiryhmälleen ja tehdä palautekysely. Sovelluksen oli tarkoitus olla kalenteri, johon käyttäjä voisi merkitä harjoituksia haluamiinsa aikoihin. Harjoituksista tulisi muistutus, jonka kautta käyttäjä voisi merkitä sen tehdyksi. Palautteesta oli tarkoitus laatia yhteenvetona raportti yritykselle.

Sovellus toteutettiin Xamarin-kehitysalustalla. Käyttöliittymä tehtiin XAML-merkintäkielellä ja ohjelma C#-kielellä. Palautekysely tehtiin Google Forms:lla.

Työn lopputuloksena oli toimiva mobiilisovellus Android-laitteille ja itsetehty palautekysely pienelle testiryhmälle. Erillistä raporttia palautteesta ei laadittu.

Avainsanat
Xamarin.Forms, XAML, betatestaus

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Arttu Väisänen			
Title of Thesis Calendarapp - a Calendar Mobile Application and Feedback			
Date	9 May 2019	Pages/Appendices	23/1
Supervisor(s) Mr. Mikko Pääkkönen, Senior Lecturer, Mr. Keijo Kuosmanen, Senior Lecturer			
Client Organisation /Partners Pelvic Fysio Oy			

Abstract

The objective of the thesis was to create a cross-platform mobile app for Pelvic Fysio Oy, to publish it to their test group and to collect feedback which then would be used to create a report. The application was meant to contain a calendar where a user could mark upcoming exercises at specific times. The app would then remind the user when an exercise is timed to happen. The user could mark the exercise done or undone when the reminder is tapped.

The application was created with Xamarin development tools. The entire user interface was created in eXtensible Application Markup Language (XAML) while the program was coded in C#. The feedback was collected with Google Forms.

The result of the thesis was a working application for Android devices and a self-made feedback survey for a small test group. No additional report was made from the results of the feedback but they are discussed in the thesis report.

Keywords
Xamarin.Forms, XAML, beta testing

SISÄLTÖ

1 JOHDANTO

Pelvic Fysio tarjoaa valmennuspaketteja äitiysliikuntaan, kehonhuoltoon ja kuntoutukseen. Valmennuspaketteja myydään yrityksen verkkokaupassa. Yrityksen ovat perustaneet fysioterapeutit Johanna Tossavainen ja Jannika Alanko. Aihe saatiin koulun kautta, johon yritys oli ollut yhteydessä. Aihe muuttui opinnäytetyön aikana. Uusi aihe katsottiin paremmin sopivaksi opinnäytetyöhön.

Työn tilaajat pyysivät mobiilisovellusta, jolla käyttäjä voisi asettaa muistutuksia tulevista harjoituksista sovelluksessa olevaan kalenteriin. Muistutukset tulisivat jossakin muodossa asetettuun aikaan. Lisäksi käyttäjä voisi merkitä harjoitukset tehdyiksi.

Työn tavoitteena oli kehittää mobiilisovellus, julkaista se testiryhmälle, kerätä palaute heiltä ja laatia raportti. Saadun palautteen perusteella sovellusta voitaisiin kehittää asiakaskäyttöön.

Tavoitetta ei kuitenkaan saavutettu ajoissa, vaan sovellus jäi julkaisematta tilaajien testiryhmälle. Sen sijaan sovellus julkaistiin vapaaehtoisille testaajille, jotka etsittiin Facebookista, ja heiltä kerättiin palaute. Palaute sisällytettiin osaksi opinnäytetyön raporttia.

2 PROJEKTI

Projektin aloituspalaveri oli toukokuussa 2018. Tilaajan tarve oli alun perin erilainen kuin toteutunut sovellus. Opinnäytetyön aihe vaihtui ja täsmentyi elokuussa 2018. Aiheeksi tuli mobiilisovelluksen kehittäminen Android- ja iOS-alustoille, sen julkaiseminen testikäyttäjille, palautteen kerääminen ja raportin laatiminen saadusta palautteesta.

Sovelluksessa oli tarkoitus voida lisätä, muokata ja poistaa muistutuksia tulevista harjoituksista sovellukseen sisällytettyyn kalenteriin. Muistutuksesta tulisi ilmoitus jossakin muodossa, kun asetettu aika lähestyy tai alkaa. Määritetyn ajan päästä ilmoituksesta tulisi kysymys, onko harjoitus suoritettu. Suoritetut ja suorittamattomat harjoitukset näkyisivät kalenterissa erilailla, jolloin käyttäjä voisi helposti tarkastella yleistä edistymistään. Kuukausinäkymässä näkyisi, onko päivän harjoitukset suoritettu vai ei, jos merkintöjä päivälle on. Muistutuksiin oli tarkoitus voida määrittää suoritettava harjoitus ja ajankohta. Käyttäjä voisi lisätä myös kommentteja tekstimuodossa ennen harjoitusta tai sen jälkeen.

Työn teko aloitettiin tutustumalla Xamarin-kehitykseen ja siinä käytettäviin tekniikoihin. Xamarin on monialustaiseen mobiilikehitykseen tarkoitettu kehitysalusta, joka on nykyisin osa Microsoft Visual Studio -ohjelmointikehitysympäristöä (TheWindowsClub, 2017). Kun sopiva kalenterikomponentti oli löydetty, alettiin sovelluksen toimintoja toteuttaa. Työtä tehtiin kotona omalla tietokoneella, joten työ ei ollut kokopäiväistä ja siihen kului useampi kuukausi. Lisäksi mobiilikehittämisestä oli vain vähän kokemusta, ja jotkin toiminnot vaativat runsaasti aikaa selvittämiseen ja toteutukseen. Sovelluksesta valmistui maaliskuussa 2019 Android-versio. iOS-versio jäi toteuttamatta, koska laitteita sen testaamiseen ei ollut.

Android-versio sovelluksesta annettiin kokeiltavaksi tilaajille. He ilmoittivat, ettei sovellus vastannut heidän toiveitaan visuaalisuudeltaan ja toiminnoiltaan, eivätkä he siksi voisi julkaista sovellusta testiryhmälleen. Toteutuneessa sovelluksessa pystyi lisäämään kalenteriin muistutuksia, joista tuli ilmoitus asetettuun aikaan notifikaation muodossa. Notifikaatiota painamalla avautui sovelluksen osa, jossa kysyttiin oliko harjoitus tehty. Harjoitukset näkyivät päiväkohtaisella sivulla listana, ja niitä oli mahdollisuus muokata, poistaa tai merkitä tehdyiksi/poistaa merkintä. Kuukausinäkymässä päivän kohdalla näkyi, jos siinä oli harjoituksia ja jos kaikki oli tehty tai ei.

Sovellus päätettiin julkaista itsekootulle testiryhmälle, jolta palautteen pystyi keräämään. Varsinaista raporttia ei palautteesta laadittu, vaan palautteen käsittely sisällytettiin opinnäytetyön raporttiin.

3 TEKNINEN TOTEUTUS

Mobiilisovellusten kehittämisessä on se vaikeus, että erilaisia alustoja on runsas määrä ja eri valmistajien ohjelmistoversiot eivät ole yhteensopivia. Tämä tarkoittaa, että perinteisesti Android-laitteilla toimiva sovellus ei toimikaan iOS-laitteilla, vaan molemmille täytyy tehdä sovellukset natiiveilla menetelmillä. Xamarin kuitenkin mahdollistaa mobiilisovellusten kehittämisen siten, että ne käyttävät samaa koodipohjaa, jolloin kerran kirjoitettu koodi toimii kaikilla kolmella alustalla (Windows, iOS ja Android) (Telerik, 2017).

Xamarin oli hyvä valinta kehitysalustaksi myös siksi, koska Visual Studio oli jo ennestään tuttu ohjelmointikehitysympäristö. Ohjelmointikielenä oli C#, joka oli niin ikään ennestään tuttu. XAML-merkintäkielestä ei ollut aiempaa kokemusta, mutta se oli helppo oppia, sillä se muistuttaa XML-syntaksia.

3.1 Xamarin.Forms

Alun perin Xamarin mahdollisti vain yhtenäisen lähdekoodin käytön sovellustasolla monialustaisille sovelluksille, minkä vuoksi käyttöliittymä täytyi tehdä erikseen jokaiseen projektiin. Xamarin.Forms kuitenkin mahdollistaa myös käyttöliittymän luomisen yhtenäisestä lähdekoodista, jolloin käyttöliittymäkoodi tarvitsee kirjoittaa vain kerran. Xamarin.Forms:lla luodut käyttöliittymäelementit kääntyvät natiiveiksi elementeiksi, kun sovellusta käytetään laitteella (Telerik, 2017). Xamarin.Forms:lla voi luoda elementtejä joko C#-koodilla tai XAML-merkintäkielellä. Toteutuneessa sovelluksessa käyttöliittymä on tehty kokonaan XAML-merkintäkielellä, kuvassa 1 on esimerkkinä sovelluksen eräs sivu.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="CalendarApp.EventFromNotificationPage">
    <ContentPage.Content>
        <StackLayout Orientation="Vertical" Spacing="10" HeightRequest="400" BackgroundColor="Linen">
            <StackLayout VerticalOptions="Start">
                <Button x:Name="didIt" Text="Tehty?" BackgroundColor="Orange" Clicked="ChangeIsDone"></Button>
            </StackLayout>
            <StackLayout>
                <Label x:Name="Date" HorizontalTextAlignment="Center"></Label>
                <Label x:Name="Time" HorizontalTextAlignment="Center"></Label>
                <Label x:Name="Subject" HorizontalTextAlignment="Center"></Label>
                <Image x:Name="oksign" IsVisible="False" Scale="0.5"></Image>
            </StackLayout>
            <StackLayout VerticalOptions="End">
                <Button Text="Kalenteriin" BackgroundColor="Orange" Clicked="ToTheCalendar"></Button>
            </StackLayout>
        </StackLayout>

    </ContentPage.Content>
</ContentPage>
```

Kuva 1. Sovelluksen sivu XAML-merkintäkielellä.

Xamarin.Forms sisältää myös tavan sitoa elementtien arvot muuttujiin, jolloin niitä voi käsitellä koodilla. Englanninkielinen termi tälle on data binding. Kun koodissa muutetaan vaikkapa Label-elementin Text-arvoa, näkyy muutos käyttäjälle välittömästi. Kuvissa 2 ja 3 näkyy, kuinka data binding toimii käytännössä. Niissä koodissa luotu CellViewModel-tyyppinen objekti sidotaan yksittäiseen

kalenterin soluun, jonka elementtien arvoina käytetään objektin muuttujia. Data binding liittyy olennaisesti MVVM-kuvioon (Model-View-Viewmodel), joka on yleisesti mobiilisovelluksissa käytetty arkkitehtuurimalli (Microsoft Xamarin From Data Bindings to MVVM, 2017).

```
<DataTemplate x:Key="eventtemplate">
    <Grid>
        <StackLayout Orientation="Vertical">
            <Label BackgroundColor="{Binding CellBgColor}" VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand"></Label>
        </StackLayout>
        <StackLayout Orientation="Vertical" BackgroundColor="{Binding CellBgColor}" IsVisible="{Binding IsCurrentMonth}">
            <Label Text="{Binding Date.Day}" HorizontalTextAlignment="Center" BackgroundColor="Linen"></Label>
            <StackLayout Orientation="Vertical" IsVisible="{Binding HasEvents}" >
                <Label FontSize="Micro" Text="Tapahtumia:"></Label>
                <Label FontSize="Micro" BackgroundColor="{Binding BgColor}" Text="{Binding Eventssum}" TextColor="black"></Label>
            </StackLayout>
        </StackLayout>
    </Grid>
</DataTemplate>
```

Kuva 2. XAML-merkintäkielellä luotu rakenne kalenterin soluille, missä hyödynnetään data binding:ia arvojen esittämiseen.

```
private void Calendar_OnMonthCellLoaded(object sender, MonthCellLoadedEventArgs args)
{
    CellViewModel cell = new CellViewModel(args.Date)
    {
        Date = args.Date,
        IsCurrentMonth = args.IsCurrentMonth
    };
    if (args.IsCurrentMonth)
    {
        cell.CellBgColor = Color.White;
    }
    else
    {
        cell.CellBgColor = Color.Linen;
    }
    args.CellBindingContext = cell;
}
```

Kuva 3. Objektin sitominen yksittäiseen kalenterin soluun data binding:n avulla.

3.2 Kalenteri

Sovelluksen toteutus aloitettiin tutkimalla mahdollisia kalenterikomponenttien tarjoajia. Käytettäväksi valittiin Syncfusionin SfCalendar. Syncfusion tarjoaa erilaisia käyttöliittymäkomponentteja Xamarin-mobiilisovelluksiin, web-applikaatioihin ja Windows-sovelluksiin. Komponentteja saa lisensseillä, jotka ovat maksullisia. Pienet yritykset ja yksityiset kehittäjät voivat kuitenkin saada komponentit myös ilmaiseksi Community-lisenssillä, jollaista tässä sovelluksessa on käytetty.

SfCalendar on monella tavalla kustomoitavissa, esimerkiksi viikon ensimmäisen päivän voi valita asetuksista. Lisäksi voidaan määrittää eri tapahtumia, kuten kun käyttäjä painaa kalenterin solua. Voidaan myös määrittää joko kuukausi- tai vuosinäkymä. Kuukausinäkymässä on kaikki kuukauden päivät, vuosinäkymässä ovat kaikki kuukaudet kerralla. Kuvassa 4 on luotu SfCalendar ja tehty siihen asetuksia. Erityistä huomioitavaa siinä on CellTemplate-arvo, johon on sidottu kuvassa 2 esitetty rakenne.

```
<Syncfusion:SfCalendar x:Name="sfCalendar" ToggleDaySelection="False" ShowNavigationButtons="True" EnableSwiping="False"
    ShowInlineEvents="False" HeightRequest="500" SelectionMode="SingleSelection" FirstDayofWeek="1"
    OnDateCellHolding="OnDateCellHolding" OnCalendarTapped="OnCalendarTappedHandler">
    <Syncfusion:SfCalendar.MonthViewSettings>
        <Syncfusion:MonthViewSettings CellTemplate="{StaticResource eventtemplate}" CellGridOptions="Both" BorderColor="black"/>
    </Syncfusion:SfCalendar.MonthViewSettings>
</Syncfusion:SfCalendar>
```

Kuva 4. SfCalendar:n luominen XAML-merkintäkielellä.

Toteutuneessa sovelluksessa kalenteri on suomenkielinen. Kalenterissa on vain kuukausinäkymä. Kuukaudesta toiseen navigoiminen tapahtuu nuolinäppäimillä, sillä pyyhkäisemällä siirtyminen vaikutti ongelmalliselta. Yksittäistä päivää pitkään painamalla avautuu sivu valitulle päivälle. SfCalendar sisältää kaksi käyttäjätapahtumaa solujen painamiselle. Loogisinta päivän valitsemiseen oli käyttää lyhyttä painallusta, joten päiväkohtaisen sivun avaamiseen käytettiin pitkää painallusta.

Sovelluksen edellytyksenä oli, että kalenterissa täytyisi jotenkin erotella päivät, joissa on merkintöjä. Niinpä täytyi tehdä kustomoitu rakenne päiville, mikä esitetään ylempänä kuvassa 2. Siinä päivät, joissa on kaikki tapahtumat merkitty tehdyiksi, saavat vihreän Label-elementin, jossa kerrotaan tapahtumien lukumäärä. Vastaavasti päivät, joissa kaikki tapahtumat eivät ole merkitty tehdyiksi, saavat punaisen Label-elementin. Päivät, joissa ei ole tapahtumia lainkaan, jätetään tyhjiksi. Kuvassa 5 on kalenterin kuukausinäkymä. Valittua päivää ei saatu korostettua eri värillä kustomoidun rakenteen monimutkaisuuden vuoksi, joten se sidottiin alla olevan Button-elementin Text-arvoon (kuva 6).

Kuva 5. Kuukausinäkymä.

```
<Button x:Name="AddButton" BackgroundColor="Orange" BindingContext="{x:Reference sfCalendar}"
Text="{Binding Path=SelectedDate,StringFormat='Lisää {0:dd. MMMM}'}" Clicked="OnAddEventHandler"
HeightRequest="100" WidthRequest="200" HorizontalOptions="Center" VerticalOptions="End"></Button>
```

Kuva 6. Nappi, jonka Text-arvoon on sidottu kalenterin valittu päivämäärä.

3.3 Päiväkohtaiset tiedot

Kuvassa 7 on sivu, joka avautuu kalenterin solua pitkään painamalla. Sivun yläosassa näkyy valittu päivä. Alempana listataan kyseisen päivän tapahtumat. Jos kaikki tapahtumat eivät mahdu kerralla näytölle, voi listaa rullata alas ja ylös. Alhaalla olevasta napista voi lisätä tapahtuman samalle päivämäärälle.

Kuva 7. Päiväkohtaiset tiedot, jossa näkyy päivälle merkityt tapahtumat listana.

Listattuja tapahtumia voi muokata, merkitä tehdyksi tai poistaa. Kuvassa 8 on muokkaussivu. Siinä ainoa muokattava tieto on tapahtuman kuvaus. Ajan muokkaaminen olisi ollut samalla tavalla toteutettavissa, mutta sitä ei katsottu tarpeelliseksi. Tehty-painikkeesta aukeaa ponnahdusikkuna, jossa voi vahvistaa harjoituksen tehdyksi. Merkinnän voi lisätä tai poistaa uudelleen, ja sen voi myös laittaa ennen kuin tapahtuma on mennyt. Poista-painikkeesta aukeaa puolestaan ponnahdusikkuna, jossa kysytään vahvistusta tapahtuman poistamiselle.

Kuva 8. Päivälle merkityn tapahtuman muokkaaminen.

3.4 Muistutukset

Muistutuksen muodoksi valittiin notifikaatio. Notifikaatiot jaetaan paikallisiin (local) ja etäisiin (remote tai push) (Stack Overflow, 2017). Paikallisen notifikaation sisältö luodaan sovelluksessa itsessään, etäiset notifikaatiot saavat sisältönsä palvelimelta. Toteutuneessa sovelluksessa käytetään paikallisia notifikaatioita.

Android- ja iOS-järjestelmillä on omat menetelmänsä notifikaatioiden toteuttamiseen. iOS:lla paikallisten notifikaatioiden luominen ja ajastaminen on melko yksinkertaista, sillä UILocalNotification-luokka sisältää arvon FireDate, jonka avulla voidaan asettaa haluttu aika notifikaatiolle UIApplication.SharedApplication-objektin metodilla ScheduleLocalNotification (Microsoft Xamarin Notifications in Xamarin.iOS, 2018). Androidilla asia on monimutkaisempi, ja seuraavissa luvuissa käsitelläänkin sen toteutusta.

3.4.1 Muistutuksen asettaminen

Käyttäjän täytyi pystyä lisäämään kalenteriin merkintöjä, joista tulisi muistutus tapahtuman alkaessa. Muistutukseen piti voida määrittää tapahtuman tekstimuotoinen kuvaus ja alkamisaika. SfCalendar sisältää luokan CalendarInlineEvent, jonka tyyppisiä objekteja kalenteriin voi lisätä ja joka sisältää edellä mainitut tiedot. Luomalla uuden luokan, joka periytyy luokasta CalendarInlineEvent, saatiin kalenteriin sopiva luokka, johon pystyi lisäämään enemmän arvoja (kuva 9).

```
public class EventModel : CalendarInlineEvent
{
    public bool IsDone { get; set; }
    public int ID { get; set; }
}
```

Kuva 9. Luokka, joka sisältää luokan CalendarInlineEvent arvot sekä uudet arvot IsDone ja ID.

Päivämäärä valitaan alustalle tyypillisellä päivämäärän valitsemiseen tarkoitetulla pienoisohjelmalla, kellonaika samoin. Täysin samalle ajankohdalle (päivämäärä ja aika) ei voi lisätä kahta tapahtumaa,

vaan niissä on oltava vähintään minuutin ero. Kuvassa 10 on käyttäjä syöttänyt tapahtuman päivämäärän ja ajan, mutta ei tekstiä.

Kuva 10. Sivu, jossa lisätään tapahtuma kalenteriin.

3.4.2 Muistuttaminen

Vaikka suurimman osan sovelluksesta pystyi tekemään jaetulla koodilla, jotkin toiminnot täytyi toteuttaa alustakohtaisesti, koska eri valmistajien laitteet ja käyttöjärjestelmät toimivat eri tavoin. Esimerkiksi notifikaatiot täytyi toteuttaa erikseen Android-projektissa ja iOS-projektissa. Jotta nämä toteutetut toiminnot saisi käyttöön jaetussa koodissa, jossa ohjelma varsinaisesti pyörii, täytyi käyttää palvelun paikantaja DependencyServiceä (Microsoft Xamarin DependencyService, 2018).

Ensiksi täytyi määrittää rajapinta, jossa on tarvittavat metodit. Nämä metodit ovat notifikaation asettaminen ja poistaminen (kuva 11).

```
public interface INotification
{
    void SetNotification(EventModel calendarevent);
    void CancelNotification(EventModel calendarevent);
}
```

Kuva 11. Rajapinta, joka määrittää tarvittavat metodit.

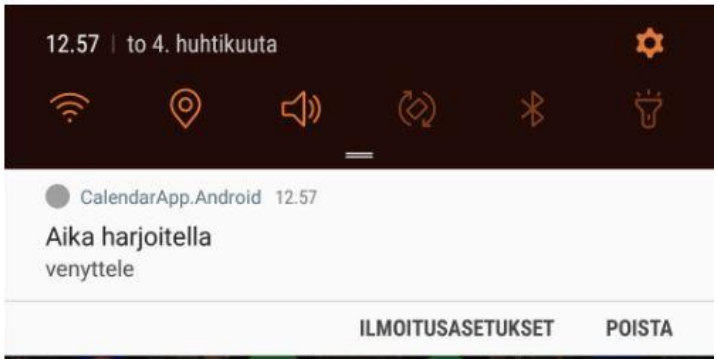
Seuraavaksi täytyi toteuttaa toiminnot alustakohtaisesti. Android-projektissa luotiin luokka, joka toteutti edellä mainitun rajapinnan eli se sisälsi metodit notifikaatioiden asettamiseen ja poistamiseen. Lisäksi luokka täytyi rekisteröidä DependecyServiceen metadata-attribuutilla, jotta DependencyService löytää oikean toteutuksen sovellusta ajettaessa. Jaetussa koodissa voitiin käyttää DependencyServicen metodeja, jotka kutsuvat alustakohtaisesti toteutettuja metodeja notifikaatioiden lisäämiseen ja poistamiseen (kuva 12).

```
DependencyService.Get<INotification>().SetNotification(calendarInlineEvent);
```

Kuva 12. DependencyService kutsuu rajapinnan määrittämää metodia ja etsii sille alustakohtaisen toteutuksen.

Toisin kuin iOS:lla, Androidilla ei ole valmista menetelmää notifikaatioiden ajastamiseen. Projektin suurin haaste olikin notifikaatioiden toteutus. Androidilla aika lasketaan millisekunneissa käyttäjän

asettamaan aikaan nykyhetkestä. Kun laskettu määrä millisekunteja on kulunut, Androidin AlarmManager laittaa lähetyksen (broadcast), joka otetaan vastaan BroadcastReceiver-luokasta periytyvässä luokassa. Siinä lähetyksen tiedot käsitellään ja luodaan notifikaatio, joka julkaistaan käyttäjälle. Notifikaatiossa näkyy asetettu kellonaika ja syötetty teksti (kuva 13).



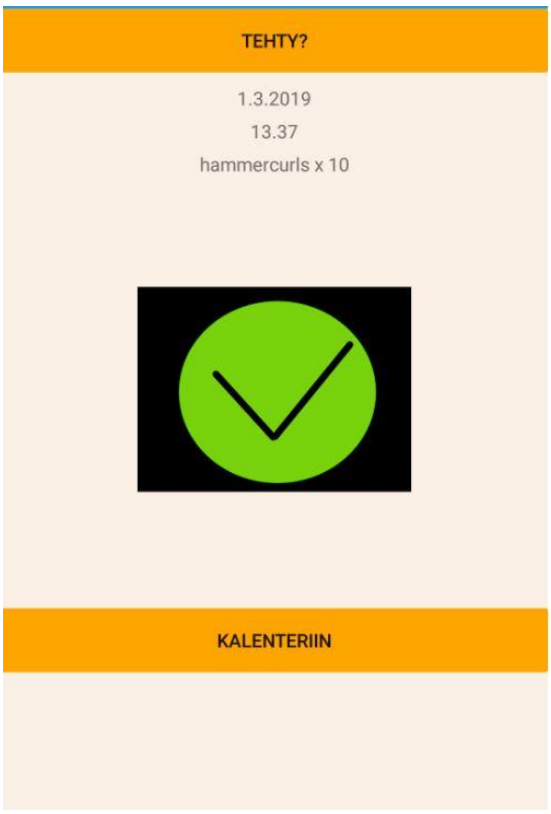
Kuva 13. Sovelluksen luoma notifikaatio käyttäjän asettamaan aikaan.

Android-systeemi pyrkii minimoimaan virrankulutuksen. Siksi se voi muuttaa aikaa, jolloin hälytys laukeaa, joten hälytykset eivät välttämättä tule asetettuun aikaan. Kun laite on jonkin aikaa paikoillaan ja käyttää akkuvirtaa sekä näyttö on pois päältä, laite menee unitilaan, jolloin se ei laukaise tavallisia hälytyksiä, koska silloin se heräisi ja kuluttaisi enemmän virtaa. Unitilassa laite kuitenkin herää hetkeksi pitenevin väliajoin, milloin se suorittaa tehtäviä, jotka ovat unitilan vuoksi tekemättä, esimerkiksi laukaisee hälytyksen (Android Developers Optimize for Doze and App Standby, 2019). Ratkaisuna tähän käytetään AlarmManagerin metodia SetExactAndAllowWhileIdle, joka varmistaa että hälytys laukeaa unitilasta huolimatta. Lisäksi hälytyksen tyypiksi valitaan RtcWakeup.

Koska iOS-laitteita ei ollut testaamiseen, iOS-projektia ei toteutettu. Muutoin olisi tehty toteutukset rajapinnan metodeista käyttämällä iOS-systeemin menetelmiä, kuten nyt tehtiin Androidilla.

3.4.3 Notifikaatio

Notifikaatiota painamalla avautuu sovelluksen sivu, jossa kysytään vahvistusta harjoituksen suorittamiselle (kuva 14). Ylempää nappia painamalla avautuu ponnahdusikkuna, jossa kysytään, onko harjoitus tehty. Jos on, ilmestyy kuvassa oleva merkki. Alemmasta napista käyttäjä ohjautuu kalenterinäkymään.



Kuva 14. Sivu, jossa vahvistetaan harjoituksen suorittaminen, kun notifikaatiota on painettu.

Kun notifikaatio luodaan, sille asetetaan SetContentIntent-metodilla toiminto, joka suoritetaan sitä painettaessa, sekä tekstimuotoinen extra-data. Tässä sovelluksessa toiminto on Androidin MainActivity, jossa tarkastetaan extra-data. MainActivity on luokka, joka on Android-sovellusten tulokohta eli kohta, josta sovellus alkaa, mutta tulokohta voi olla mikä tahansa toiminto. Extra-datan avulla määritetään, milloin sovellukseen tullaan notifikaatiota painamalla ja milloin sovellus on avattu normaalisti. Sieltä siirrytään takaisin jaettuun koodiin, joka näyttää kuvassa 12 olevan sivun.

Eräs vika sovellukseen jäi. Tilanteessa, jossa sovellus on jo julkaissut yhden notifikaation, jota ei ole avattu ja jos se silloin julkaisee toisen notifikaation, aiemmin julkaistu notifikaatio ohjaa käyttäjän uusimman notifikaation esittämiin tietoihin sitä painettaessa. Ongelmaa tutkittiin, mutta ratkaisua siihen ei keksitty. Testiryhmältä ei kuitenkaan tullut palautetta tästä ongelmasta, joten sen vaikutukset käyttäjäkokemukseen lienevät vähäiset.

3.4.4 Laitteen uudelleenkäynnistäminen

Androidin AlarmManager säilyttää asetetut hälytykset, kun laite on unitilassa, mutta ei säilytä niitä laitteen uudelleenkäynnistyksen yhteydessä (Android Developers AlarmManager, 2019). Hälytykset eli ajastetut notifikaatiot täytyi siis asettaa uudelleen silloin. Sitä varten sovellus tarvitsee luvan RECEIVE_BOOT_COMPLETED. Luvan pystyy asettamaan AndroidManifest-tiedostossa tai valitsemalla valikosta. Jotkin luvat tarvitsevat hyväksynnän käyttäjältä, näitä kutsutaan vaarallisiksi luviksi, mutta RECEIVE_BOOT_COMPLETED on normaali lupa eikä vaadi erillistä hyväksyntää.

Kun laite on käynnistynyt uudelleen, se laittaa yhden lähetyksen, jotta sovellukset tietävät tapahtumasta. Sovellus ottaa lähetyksen vastaan BroadcastReceiver-luokasta periytyvässä luokassa. Ohjelman osa saatiin suorittamaan ilman sovelluksen käynnistämistä käyttämällä JobIntentServiceä. Kun lähetys on vastaanotettu, JobIntentService aloittaa työn taustalla, mikä tässä tapauksessa on tulevaisuudessa olevien tapahtumien hälytysten uudelleen asettaminen. JobIntentService toimii eri tavoin Android Oreolla ja uudemmilla kuin sitä vanhemmilla versioilla (Android Developers JobIntentService, 2019). Siksi vanhempien versioiden varalta sovellus tarvitsi normaalin luvan WAKE_LOCK.

3.5 Tietojen tallennus

Käyttäjän syöttämät tiedot tallennetaan samaan paikkaan kuin sovellus, sovelluksen sisältämään kansioon. Tiedot muutetaan objekteista JSON-tekstiksi (serialisoidaan), minkä jälkeen se kirjoitetaan sellaisenaan tekstitiedostoon. Operaatiot toimivat niin, että JSON-teksti luetaan tiedostosta, muutetaan objektiksi (deserialisoidaan), muutetaan objektin arvoja, muutetaan objekti JSON-tekstiksi, kirjoitetaan JSON-teksti tiedostoon ja tallennetaan tiedosto.

Vaihtoehtoinen tapa tallentaa tietoja olisi ollut paikallinen tietokanta, joka olisi toteutettu SQLite.Net-paketilla. Tietokannan avulla olisi ehkä voinut toteuttaa useampia ominaisuuksia sovellukseen, kuten tilastointia tehdyistä harjoituksista.

4 SOVELLUKSEN JULKAISU

Normaalisti mobiilisovellusta kehitettäessä erilaisia testauksia tehdään pitkin prosessia ja tulokset voidaan dokumentoida. Tässä projektissa uusia ominaisuuksia lisättäessä sovellusta testattiin paikallisesti omalla laitteella, mutta tuloksia ei dokumentoitu. Kehityksen aikana tapahtuvaa testaamista kutsutaan alfatestaamiseksi, ja isommissa projekteissa sille voi olla oma ryhmänsä (Instabug, 2015).

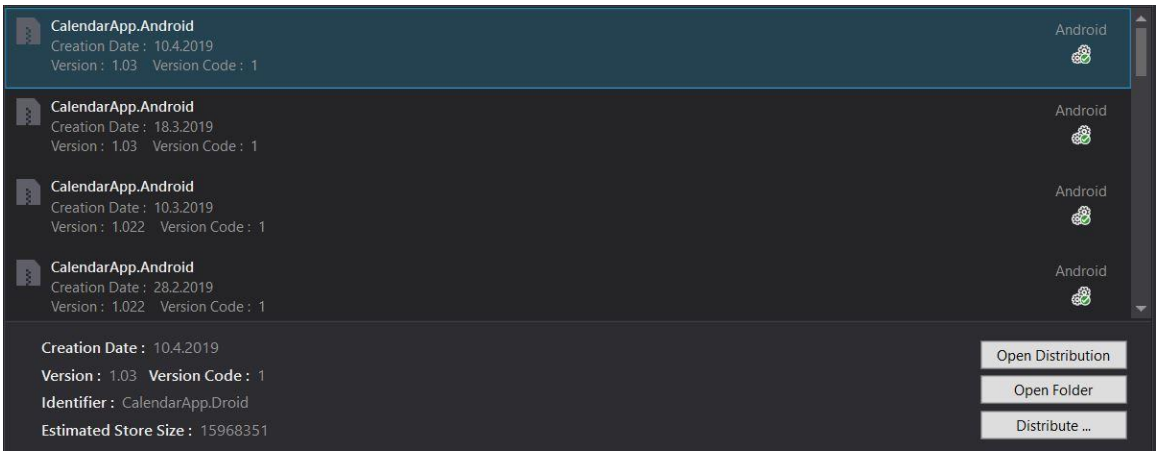
Alun perin sovellus oli tarkoitus betatestata tilaajien kokoamalla testiryhmällä. Niin ei kuitenkaan tehty, vaan sovellus julkaistiin testattavaksi itsekootulle pienelle ryhmälle.

4.1 Tekninen julkaisu

Ennen kuin Xamarin.Android-sovellus voidaan kääntää koodista jaettavaksi paketiksi, sille täytyy asettaa muutamia attribuutteja (Microsoft Xamarin Preparing an Application for Release, 2018). Sovellukselle on suositeltavaa asettaa ikoni, koska kaikki sovellusten jakelijat eivät hyväksy ikonittomia sovelluksia. Tässä sovelluksessa käytettiin Visual Studiossa valmiina olevaa Xamarin-ikonia. Sovellus tulee versioida, koska sovellukseen voidaan julkaista muutoksia. Versiotietoihin tulee asettaa numero sekä nimi.

Julkaistavan paketin voi kutistaa, jotta se veisi vähemmän tilaa asennettaessa laitteelle. Sovellus tulee suojata estämällä testaaminen (debug) ja käyttämällä Dotfuscator:ia, joka sisältyy Visual Studio 2017:aan. Muuten hyökkääjä voi uudelleenmallintaa sovelluksen ja lisätä siihen haitallista koodia. Lisäksi on muita suojausasetuksia, joita voi valita projektin ominaisuuksista.

Xamarin.Android-projekti käännetään paketiksi arkistoimalla. Archive Manager listaa kaikki aiemmin arkistoidut versiot (kuva 15).



Kuva 15. Visual Studion Archive Manager, jossa näkyy arkistoidut versiot sovelluksesta.

Distribute … -napista avautuu Distribution Channel -dialogi, jossa voidaan valita Ad-Hoc tai Google Play. Valitsemalla Google Play voidaan julkaista signeerattu APK-tiedosto Google Playhin. Valitsemalla Ad-Hoc luodaan signeerattu APK-tiedosto levylle, josta sen voi siirtää Android-laitteelle.

Signeeraukseen käytetään sertifikaattia, joka luotiin ensimmäisen julkaisun yhteydessä. Sertfikaatti sisältää tietoja kuten voimassaoloaika, maa ja nimi. Samaa sertifikaattia voi käyttää eri sovellusten signeeraamiseen. Kun sertifikaatin salasana on syötetty, valitaan levyltä tallennuspaikka APK-tiedostolle, josta se on siirrettävissä esimerkiksi yhdistettyyn Android-laitteeseen.

4.2 Testiryhmän keräys

Testaajien lukumäärä vaihtelee testattavan asian mukaan. Ei ole tiedossa, kuinka suuri testiryhmä työn tilaajilla olisi ollut, mutta betatestauksessa on suositeltavaa olla useita kymmeniä, ellei jopa satoja henkilöitä (Centercode, 2019). Tämä oli kuitenkin niin pienen mittakaavan projekti, että testiryhmään pyrittiin saamaan vain muutama henkilö.

Vapaaehtoisia testaajia kyseltiin Facebookista, josta heitä löytyi neljä. Vapaaehtoisia kyseltiin kertomalla testattavasta sovelluksesta ja miten tutkimus etenisi. Betatestaajia voi etsiä netistä, esimerkiksi Twitteristä tai siihen tarkoitukseen olevilta sivustoilta kuten Betalist (Instabug, 2015).

4.3 Sovelluksen jakaminen

Testaamisen voi suorittaa joko niin, että testaaja ja tutkija ovat samassa tilassa ja palaute kerätään samalla, tai etänä (Savvy Apps, 2018). Tässä betatestissä testaus suoritettiin etänä.

Vaihtoehdoksi sovelluksen jakamiseen testaajille harkittiin sähköpostilla lähettämistä, mutta parhaimmaksi vaihtoehdoksi katsottiin Google Driven kautta jakaminen. Google Driveen luotiin kansio, jonne sovellus tallennettiin. Sen jälkeen testaajien Gmail-sähköpostiosoitteille annettiin oikeudet ladata tiedosto kansiosta. Testaajille lähetettiin linkki, jolla he pääsivät kansioon. Jakaminen ja asennus onnistuivat vaikeuksitta, ja käyttäjät pääsivät testaamaan sovellusta.

5 PALAUTTEEN KERÄÄMINEN JA TULOKSET

Sukupuolijakauma testiryhmässä oli kaksi miestä ja kaksi naista. Ikäjakauma oli 26-29 vuotta. Aikaa testaamiseen annettiin seitsemän päivää, minkä jälkeen testaajille lähetettiin palautekysely. Suositeltava kesto testaamiselle on 6-10 viikkoa (Instabug, 2015), mutta sellaiseen ei ollut aikaa tässä tapauksessa.

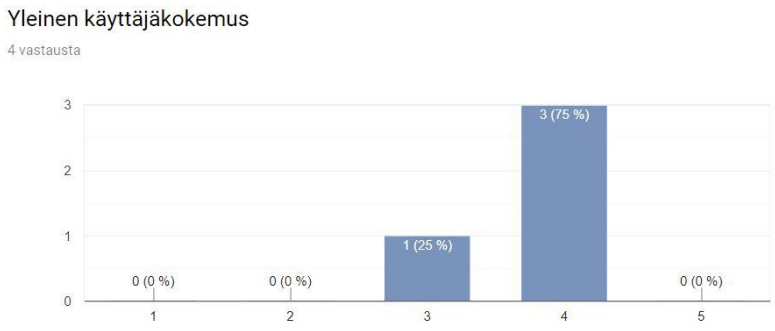
Ei ole tiedossa, millaisia asioita työn tilaajat olisivat testissä halunneet selvittää. Betatestissä tarkoitus on saada näkyvyyttä ulkomaailmaan ja tietoa, miltä sovellus vaikuttaa oikeiden käyttäjien käyttämänä (Instabug, 2015). Betatestissä voidaan selvittää sovelluksesta löytyviä ongelmia ja virheitä, jotka eivät tulleet ilmi alfatestauksessa, tai toivottuja lisäominaisuuksia.

5.1 Palautteen kerääminen

Palautekysely tehtiin Google Forms:lla (liite 1). Google Forms tarjoaa ilmaisen tavan luoda, lähettää ja tarkastella kyselyjä. Kysymykset voi lomakkeessa jaotella osioihin. Vastaukset voivat olla monivalintoja, asteikkoja, tekstiä, kellonaikoja tai päivämääriä ynnä muita.

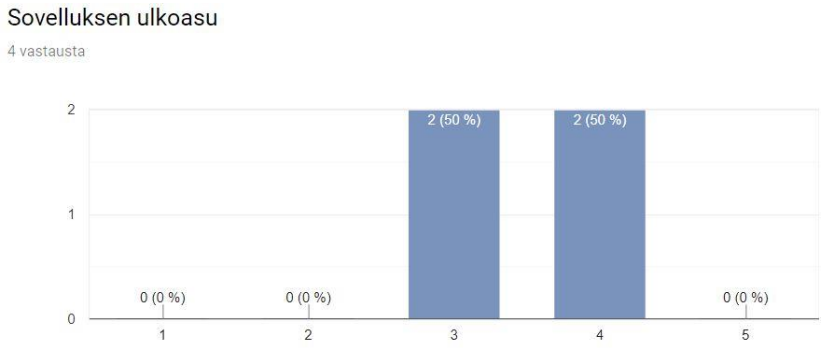
Kyselyllä haluttiin selvittää käyttäjien kokemuksia sovelluksesta, olivatko he löytäneet virheitä, miltä sovelluksen käyttö tuntui, oliko se visuaalisesti miellyttävä ja koettiinko sen käyttö hyödylliseksi. Kyselyssä käytettiin lineaarisia asteikkoja, monivalintaa ja vastaustekstejä. Lineaariset asteikot olivat välillä 1-5, jossa 1 on huono ja 5 hyvä.

5.2 Tulokset



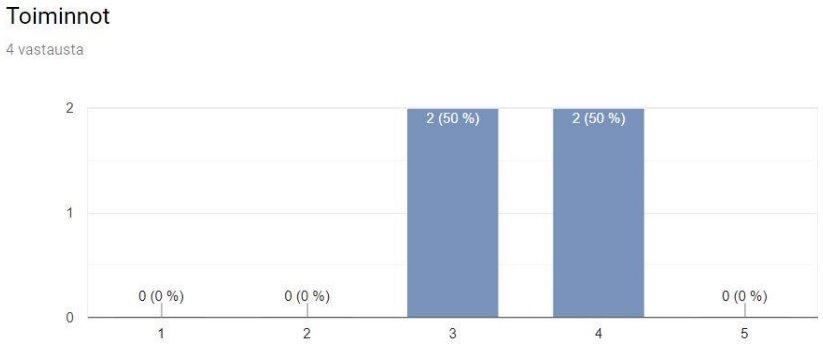
Kuva 16. Ensimmäinen kysymys lomakkeessa.

Yleinen käyttäjäkokemus oli kyselyn perusteella melko hyvä (kuva 16). Käyttäjäkokemuksessa olisi kuitenkin vielä parannettavaa. Työn tilaajien antaman palautteen perusteella käyttöliittymän hitaus lienee ollut vaikuttava tekijä tässä, mistä tuli myös sanallista palautetta kyselyssä.



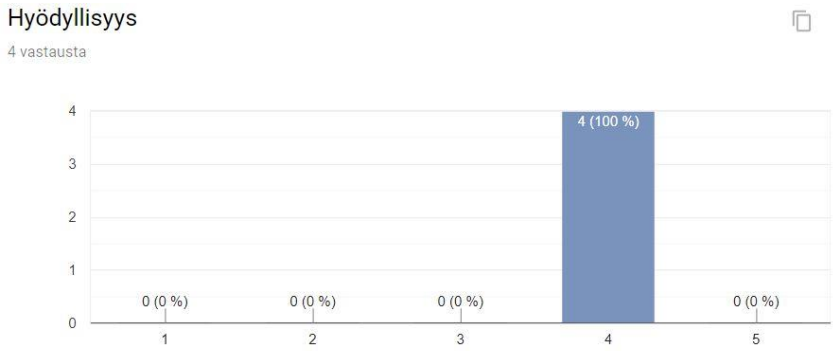
Kuva 17. Toinen kysymys lomakkeessa.

Sovelluksen ulkoasu oli vastausten perusteella melko hyvä, mutta siinä on parantamisen varaa (kuva 17). Työn tilaajat eivät pitäneet sovelluksen ulkoasua sopivana, mikä oli yksi syy sille, etteivät he voineet sitä julkaista testaajilleen.



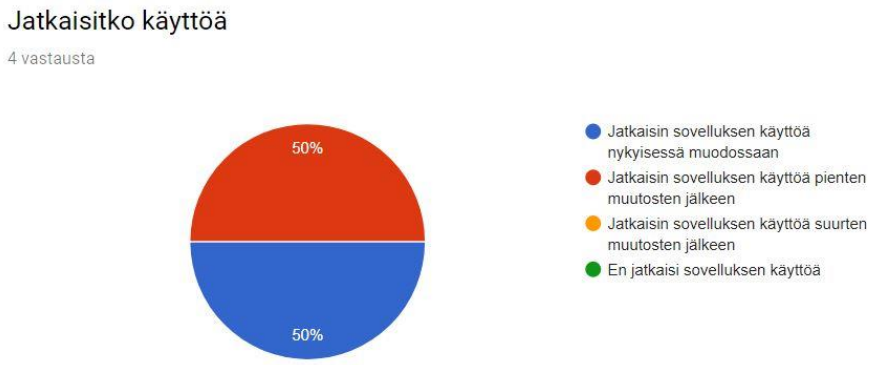
Kuva 18. Kolmas kysymys lomakkeessa.

Sovelluksen toiminnoissa on kyselyn perusteella parannettavaa (kuva 18), mikä liittynee edellä mainittuun hitauteen. Tarkemmin toimintojen ongelmista ei kerrottu, joten oletettavasti perustoiminnot olivat kuitenkin kunnossa.



Kuva 19. Neljäs kysymys lomakkeessa.

Sovellus koettiin melko hyödylliseksi (kuva 19). Kysymyksellä oli tarkoitus selvittää, pidettiinkö sovelluksen ideaa hyödyllisenä.



Kuva 20. Viides kysymys lomakkeessa.

Monivalintakysymyksessä (kuva 20) haluttiin selvittää, jatkaisivatko käyttäjät sovelluksen käyttöä. Puolet vastaajista jatkaisivat käyttöä sellaisenaan, puolet pienten muutosten jälkeen. Pienillä muutoksilla tarkoitettiin esimerkiksi muutoksia käyttöliittymään tai toimintoihin, ehkä uuden toiminnon lisäämistä. Vaihtoehto ”Jatkaisin sovelluksen käyttöä suurten muutosten jälkeen” tarkoitti, että ideaa sovelluksessa on pidetty hyvänä mutta vaatisi suuria muutoksia toimintoihin ja käyttöliittymään sekä paljon uusia toimintoja.

Vastaustekstillä kysyttiin parannusehdotuksia. Se kysymys jätettiin vapaaehtoiseksi vastata. Vastauksista kävi ilmi osin samoja asioita kuin työn tilaajien antamassa palautteessa. Päivän tapahtumien avaamiseen käytettävä kalenterin solun painaminen on koettu hieman liian pitkäksi. Sovelluksen värimaailmaan ja ulkoasuun on toivottu muutosta. Päivän tiedot toivottiin näkyviksi samalla sivulla kuin kalenteri erillisen sivun sijaan, kuten joissakin kalenterisovelluksissa on.

Toisella vastaustekstillä pystyi kirjoittamaan vapaata kommenttia käyttäjäkokemuksesta, sekään ei ollut pakollista. Vastauksista tuli ilmi, että sovellus oli tehnyt sen minkä oli luvannutkin.

5.3 Jatkokehitys

Vaikka työn tilaajat eivät ottaneet sovellusta käyttöönsä, he pitivät sitä kehityskelpoisena ja ehdottivat kehitystä jatkettavan esimerkiksi yhteistyössä Savonian Viretorin kanssa. Heidän mukaansa sovelluksen ulkoasu vaatisi yhtenäisempää visuaalista silmää ja paneutumista sovelluksen käyttöystävällisyyteen.

Testiryhmältä kerätyn palautteen perusteella sovelluksen värimaailma tulisi muuttaa. Eräässä palautteessa toivottiin päiväkohtaisten tietojen näyttämistä yhden painalluksen jälkeen samalla sivulla, varmaan kuten Android Kalenterissa on. Kalenterin solun pitkään painaminen on koettu hiukan liian pitkäkestoiseksi, joten sitä voisi muuttaa, mikäli mahdollista, riippunee kalenterikomponentin ominaisuuksista, joihin ei välttämättä voi vaikuttaa. Tosin jos päiväkohtaiset tiedot näkyisivät samalla sivulla, ei pitkään painamista tarvittaisi tähän tarkoitukseen lainkaan. Uusista toiminnoista testiryhmäläiset mainitsivat tapahtumien keston tilastoinnin ja kommentoinnin. Lisäksi toivottiin yhteenvetoa tapahtumista eri aikajaksoilla. Nämä toiminnot vaikuttavat järkeviltä lisäyksiltä jatkokehityksessä.

Jos sovelluksen kehittämistä jatkettaisiin itse, pyrittäisiin muuttamaan palautteessa esille tulleet asiat ja lisäämään ehdotetut toiminnot. Sovellusta voisi myös kehittää siten, että kalenterimerkinnät tulisivat esimerkiksi personal trainerilta asiakkaan sovellukseen, ja käyttäjä voisi sitten merkitä sen tehdyksi ajallaan, mistä tulisi ohjaajalle tieto. Opinnäytetyön loppuvaiheessa näyttää kuitenkin siltä, että sovelluksen kehittämistä ei jatketa.

6 YHTEENVETO

Opinnäytetyön lopputuloksena oli toimiva sovellus Android-alustoille, mitä työn tilaajat eivät kuitenkaan hyväksyneet, sekä itsenäisesti toteutettu palautekysely testiryhmälle. Sovellus sisälsi pääpiirteissään vaaditut toiminnot, mutta sen ulkoasu ei ollut sopiva. Sovellus julkaistiin itse kerätylle, pienelle testiryhmälle, jolle tehtiin palautekysely. Palaute oli pääosin myönteistä.

Haasteina opinnäytetyössä olivat sovelluksen toteutus monelle alustalle ja ajastetut muistutukset. Lisäksi haastavaa oli toteuttaa sovelluksen visuaalinen ilme ilman tarkkaa mallia. Toisin tekisin päiväkohtaisten tietojen näyttämisen. Tekisin sen näkymään samalla sivulla kuin kalenteri, kuten alun perin ajattelinkin. Toisin tekisin myös betatestin, olisi vaatinut enemmän aikaa sen tutkiminen ja toteuttaminen.

Opinnäytetyön edetessä opin Xamarinin käytöstä monialustaisessa mobiiliohjelmoinnissa. Lisäksi opin uusia asioita Android-ohjelmoinnista. Sovelluksen testaaminen käyttäjillä oli uusi asia, johon sain työstä kokemusta pienessä mittakaavassa.

LÄHTEET JA TUOTETUT AINEISTOT

Android Developers AlarmManager, 2019. [Viitattu 2019-04-04.] Saatavilla: https://developer.android.com/reference/android/app/AlarmManager.html

Android Developers JobIntentService, 2019. [Viitattu 2019-04-04.] Saatavilla: https://developer.android.com/reference/android/support/v4/app/JobIntentService

Android Developers Optimize for Doze and App Standby, 2019. [Viitattu 2019-04-14.] Saatavilla: https://developer.android.com/training/monitoring-device-state/doze-standby

Centercode. How Many Beta Testers Do You Need? 2019. [Viitattu 2019-04-15.] Saatavilla: https://www.centercode.com/blog/2018/12/how-many-beta-testers

Instabug. The Beginner's Guide for Beta Testing Your App. 2015. [Viitattu 2019-04-24.] Saatavilla: https://instabug.com/blog/the-beginners-guide-for-beta-testing-your-app/

Microsoft Xamarin. Introduction to DependencyService, 2018. [Viitattu 2019-04-14.] Saatavilla: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/dependency-service/introduction

Microsoft Xamarin. Notifications in Xamarin.iOS, 2018. [Viitattu 2019-04-23.] Saatavilla: https://docs.microsoft.com/en-us/xamarin/ios/platform/user-notifications/deprecated/local-notifications-in-ios

Microsoft Xamarin. Part 5. From Data Bindings to MVVM, 2017. [Viitattu 2019-04-14.] Saatavilla: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/xaml-basics/data-bindings-to-mvvm

Microsoft Xamarin. Preparing an Application for Release. 2018. [Viitattu 2019-04-05.] Saatavilla: https://docs.microsoft.com/en-us/xamarin/android/deploy-test/release-prep/?tabs=windows

Savvy Apps. How to Conduct User Testing For Your App. 2018. [Viitattu 2019-04-15.] Saatavilla: https://savvyapps.com/blog/how-to-user-testing-your-app

Stack Overflow. Which notifications should i use, Push notification or local notification? 2017. [Viitattu 2019-04-23]. Saatavilla: https://stackoverflow.com/questions/45343427/which-notifications-should-i-use-push-notification-or-local-notification

Telerik. What is Xamarin.Forms? 2017. [Viitattu 2019-04-01.] Saatavilla: https://www.telerik.com/blogs/what-is-xamarin-forms

TheWindowsClub. What is Xamarin? How does it help in cross-platform mobile app development? 2017. [Viitattu 2019-04-14.] Saatavilla: https://www.thewindowsclub.com/what-is-xamarin-and-cross-platform-mobile-development

LIITE 1: KÄYTTÄJÄKYSELY

4/25/2019 Käyttäjäkokemukset mobiilisovelluksesta

Käyttäjäkokemukset mobiilisovelluksesta

Tällä lomakkeella kerätään käyttäjäkokemuksia CalendarApp-mobiilisovelluksen käytöstä. Vastaukset käsitellään anonyymisti eikä sähköpostiosoitetta kerätä.

*Pakollinen

Yleinen käyttäjäkokemus*

	1	2	3	4	5	
Huono	○	○	○	○	○	Hyvä

Sovelluksen ulkoasu*

Miltä sovellus näytti, oliko se silmää miellyttävä vai vaikeaselkoinen.

	1	2	3	4	5	
Huono	○	○	○	○	○	Hyvä

Toiminnot*

Toimiko sovellus odotetusti. Oliko toiminnoissa virheitä.

	1	2	3	4	5	
Huono	○	○	○	○	○	Hyvä

Hyödyllisyys*

Koitko sovelluksen käytön hyödylliseksi vai oliko sen käyttö turhaa.

	1	2	3	4	5	
Turha	○	○	○	○	○	Hyödyllinen

Jatkaisitkokäyttöä*

Jatkaisitko käyttöä*

- ○ Jatkaisin sovelluksen käyttöä nykyisessä muodossaan
- ○ Jatkaisin sovelluksen käyttöä pienten muutosten jälkeen
- ○ Jatkaisin sovelluksen käyttöä suurten muutosten jälkeen
- ○ En jatkaisi sovelluksen käyttöä

Parannusehdotuksia

Kirjoita halutessasi, mitä muuttaisit sovelluksessa

Oma vastauksesi

Vapaa kommentointi

Kirjoita käyttäjäkokemuksestasi halutessasi

Oma vastauksesi

LÄHETÄ

Älä koskaan lähetä salasanaa Google Formsin kautta.