LAPIN AMK
Lapland University of Applied Sciences

MATKAPUHELINOHJELMAN SUUNNITTELU JA UUDEN PALVELINYMPÄRISTÖN PERUSTAMINEN

Lappalainen Ville

Opinnäytetyö

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

2021

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

Tekijä	Ville Lappalainen	Vuosi	2021
Ohjaaja(t)	Pekka Reijonen		
Toimeksiantaja	Kiinteistöyhtymä Lappalainen Väinö, Markku ja Virve		
Työn nimi	Matkapuhelinohjelman suunnittelu ja uuden palvelinympäristön perustaminen		
Sivu- ja liitesivumäärä	33		

Tämä opinnäytetyö perustui toimeksiantoon, jonka olen saanut metsäyhtymältä. Metsäyhtymän paperit säilytetään tällä hetkellä vain yhden metsänomistajan mapeissa ja haasteena koetaan se, ettei muut metsänomistajat pääse tarkistamaan metsänhoitoon liittyviä kuluja ja dokumentteja.

Opinnäyteyön aiheena oli luoda suunnitelma mobiilisovellukselle käyttämällä Android studio 4.1.1. -ohjelmaa. Luodun mobiilisovelluksen tarkoitus olisi jakaa tiedostoja palvelimelle, josta he voisivat löytää ne myöhemmin. Ohjelman kautta voisi myös ladata palvelimen tiedostoja suoraan Android-laitteille. Viitekehyksenä oli metsäyhtiön toiveidenmukainen mobiilisovellus.

Opinnäytetyössä oli tavoitteena luoda uusi Linux-palvelin, jossa ylläpidetään nettisivuja Androidille. Opinnäytetyössä käytiin läpi, mitä asioita pitää ottaa huomioon, jotta voidaan aloittaa mobiiliohjelman suunnittelu. Opinnäytetyössä käytiin myös läpi, miten saadaan palvelin toimimaan tietyssä osoitteessa ja miten palvelinta voidaan käyttää Androidin tiedostojen säilytykseen.

Opinnäytetyössä suunniteltiin alusta asti uusi ohjelma, jota voidaan käyttää mahdollisimman useassa laitteessa. Palvelimeen asennettiin Ubuntu, johon asennettiin palveluja, jotka mahdollistavat tiedostojen siirron ja tallennuksen palvelimen kovalevylle. Opinnäytetyön tekemisen lopputuloksena syntyi mobiiliapplikaatiosuunnitelma, jolla metsäyhtiö pystyisi jakamaan ja tallentamaan metsäyhtiön dokumentteja.

Avainsanat Mobiilisovellus, luonti, Android, tietokanta, Linux

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Bachelor's Degree
Bachelor of Business Administration

Author	Ville Lappalainen	Year	2021
Supervisor	Pekka Reijonen		
Commissioned by	Kiinteistöyhtymä Lappalainen Väinö, Markku ja Virve		
Subject of thesis	How to make a mobile application plan and a dedicated server.		
Number of pages	33		

This thesis is based on the assignment that I received from a forestry company. The forestry company's papers are currently stored in a physical file managed by one employee. The challenge is that other employees cannot check the documents related to forest management.

The topic of this thesis was to design a mobile application using Android 4.1.1. The purpose of the program is to save files onto a server where the files could be accessed on later. The program could also be used to access server files directly from any Android devices. In this thesis a new Linux server was created, which maintains websites for the Android devices. The frame of reference was to design a mobile application according to the forest company's wishes.

In this thesis I create a new Linux server to host websites for Android. The thesis covers everything that needs to be done to start designing a mobile application. The thesis also goes through how to configure a server to operate in a specific address and how the server could be used to store Android files.

In this thesis a new mobile application is designed in such a way, that it could be used in as many devices as possible. Ubuntu and three modules were installed to the server that allows file upload and download to the server's hard drive. The result of the thesis is a mobile application plan that would allow the forest company to share and store forest company documents.

Key words: Mobile application, create, Android, dedicated server, Linux

SISÄLLYS

KÄYTETYT MERKIT JA LYHENTEET

iOS	Applen kehittämä käyttöjärjestelmä mobiililaitteille ja televisioille.
SDK	Software development kit. Kokoelma koodaamiseen liittyviä työkaluja.
PHP	Yleensä palvelimen ja nettisivun kommunikaatiossa käytetty koodauskieli.
API	Application programming interface eli Ohjelmointirajapinta.
PIN	Personal identification number eli Tunnusluku.
HTML	Hypertext Markup Language eli hypertekstin merkintäkieli.
SQL	Structured Query Language eli rakenteinen kyselykieli.
XML	Extensible Markup Language eli jatkettu merkintäkieli.
LAN	Local area network eli lähiverkko.

1 JOHDANTO

Opinnäytetyössä tavoitteena on tuoda metsäyhtiö digitalisaation piiriin suunnittelemalla uusi mobiiliapplikaatio. Samalla tavoitteena on luoda oma Linux-palvelin, jonka kanssa tämä matkapuhelinohjelma kommunikoisi. Ohjelmalla on tarkoitus ottaa kuvia kuiteista, jotka siirtyvät suoraan palvelimelle. Mobiiliapplikaation kautta olisi myös helppo käydä tarkistamassa palvelimelle lähetetyt kuitit. Ohjelmalla olisi myös tarkoitus tallentaa tiedostoja palvelimelle ja hakea ne ohjelman avulla. Tavoitteena tässä opinnäytetyössä on tutkia, miten luodaan uusi matkapuhelinapplikaatio alusta loppuun saakka. Opinnäytetyössä lisäksi tutkitaan, miten luodaan uusi palvelin ja miten nämä sovellukset saadaan keskustelemaan keskenään.

Opinnäytetyön tarkoituksena on luoda suunnitelma ja esimerkkikoodi Android-ohjelmalle. Tälle applikaatiolle luodaan uusi Linux-palvelin, jota applikaatio käyttää datan säilyttämiseen. Tavoitteena on luoda matkapuhelinapplikaatiosuunnitelma, joka helpottaa kaikkien metsänomistajien juoksevien kulujen seurantaa, kun voidaan käyttää omaa applikaatiota, joka tallentaa ostokuitit ja muut metsänhoitoon liittyvä dokumentit suoraan palvelimelle. Näin fyysistä kopiota kuiteista ei tarvitse säilyttää tai lähettää metsäyhtymän kaikille omistajille erikseen. Applikaation tulisi olla helppo asentaa puhelimeen, ja applikaation sisällön tulisi olla selkeä ja loogisesti etenevä.

Tämä opinnäytetyö perustuu toimeksiantoon, jonka sain metsäyhtymältä. Metsäyhtymän paperit säilytetään tällä hetkellä vain yhden metsänomistajan mapeissa ja haasteena koetaan se, ettei muut metsänomistajat pääse tarkistamaan metsänhoitoon liittyviä kuluja ja dokumentteja. Tavoitteena olisi saada metsänhoitoon liittyvät paperit kaikkien luettavaksi ja tarkistettavaksi ajasta ja paikasta riippumatta.

Applikaation tarkoitus olisi olla yksityinen ja räätälöity metsäyhtiön tarpeita huomioiden. Applikaatioon suunnitellaan perustason lukitus, jonka tarkoituksena olisi estää tuntemattoman pääsy palvelimelle yksinkertaisella biometrisella tunnistautumisella.

2 TYÖN ALOITUS JA SUUNITTELU

2.1 Työn suunnittelu ja sen vaiheet

Mobiiliapplikaation kehitys alkaa ensin joko idean saamisella tai ongelman ratkaisulla (Varshneya 2019). Tässä opinnäytetyössä ratkaistaan ongelma, jossa kaikki metsäyrityksen omistajat haluavat päästä käsiksi dokumentteihin mobiililaitteen avulla. Yrityksessä halutaan myös saada dokumentit säästöön sähköiseksi fyysisen paperikopion lisäksi. Ongelma tullaan ratkaisemaan luomalla mobiiliapplikaatio, joka yhdistetään uuteen tietokantaan, joka luodaan ainoastaan mobiiliapplikaation käyttöön. Applikaation kautta on tarkoitus päästä näkemään kaikki dokumentit, jotka ovat sinne tallennettu.

Jotta voidaan tarkentaa mobiilisovelluksen tarkoitus, luodaan kysely yritykselle, jolla rajataan tarpeet mobiiliapplikaatiolle. Kyselyn tarkoituksena on selvittää, jos on tarvetta kehittää lisää toimintoja ja mitä ongelmia ohjelman täytyy ratkoa (Haselmayr 2013). Kun kaikki metsäyrityksen omistajat ovat vastannet kyselyyn, luodaan sen pohjalta Wireframe. Wireframessa visualisoidaan kaikki toiminnot, mitä mobiiliapplikaatiossa tulee olemaan.

Kun applikaatiosta on selvät suunnitelmat, on mahdollista aloittaa mobiiliapplikaation kehitys. Mobiiliapplikaation kehityksessä otetaan huomioon eteenkin sen toimivuus. Tällä tavalla varmistetaan, että applikaatiota on helppo ja mukava käyttää. (Varshneya 2019.) Mobiiliapplikaatio luodaan niin, että applikaatio pystyy keskustelemaan palvelimen kanssa, joka luodaan tätä applikaatiota varten. Lopuksi luodaan palvelin, johon tallennetaan tiedostoja ja dokumentteja. Luotu palvelin keskustelee mobiiliapplikaation kanssa ja on jatkuvasti kontaktissa luodun applikaation kanssa.

2.2 Kyselylomakkeen luonti, tulos ja analysointi

Sain toimeksiantona metsäyhtiöltä kehittää mobiiliapplikaation, jolla pääsee käsiksi dokumentteihin ja kuitteihin. Applikaation suunnittelu alkoi toimeksiantajan

kanssa palaverilla, jossa mietimme, mitä toimintoja mobiiliapplikaation pitäisi sisältää. Palaverin tulos oli, että saan vapaat kädet mobiiliapplikaation suunnitteluun.

Mobiiliapplikaatio on käytössä yrityksen sisäisesti. Kartoittaakseni metsäyhtiön tarpeet laadin sähköisen kyselylomakkeen, jonka lähetin jokaiselle metsäyhtymän omistajalle. Kyselylomake sisälsi neljä monivalintakysymystä, joiden perusteella lähdin toteuttamaan applikaatiota. Näillä kysymyksillä varmistetaan, että mobiiliapplikaatiolle tulee käyttöä ja se vastaa mahdollisimman paljon toimeksiantajan odotuksia.

Kyselylomakkeen tarkoituksena oli olla ymmärrettävä, jotta siitä saa helposti selvää ilman kokemusta mobiililaitteista. Kyselylomakkeessa oli vain tarkentavia kysymyksiä, jotta vältytään ylimääräiseltä datalta, jota ei tarvita (Jenn 2006).

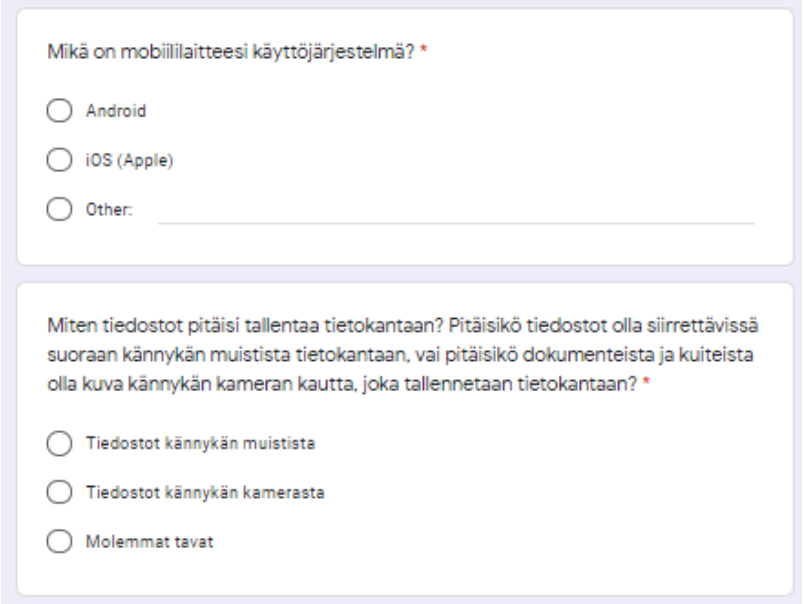
Käytin kyselyn luonnissa Google Formsia, joka on Googlen luoma kyselypohja. Forms on ilmainen ja antaa mahdollisuuden tallentaa kysymyksen tulokset Googlen omaan versioon Microsoft Excelistä. Tällä tavalla on helppo analysoida saatua dataa kyselystä. Kyselyyn otetaan mallia Cabotin (2020) luomasta kyselystä, jolla he kysyvät asiakkailta mobiiliapplikaation kehitykseen liittyviä kysymyksiä.

Ensimmäinen kysymys kyselyssä on: ”Mitkä ovat käyttäjien mobiililaitteiden käyttöjärjestelmät?” (Kuva 1.) Tällä kysymyksellä on tarkoitus saada tietoon, jos on tarvetta luoda mobiiliapplikaatio usealle rajapinnalle.

Tällä hetkellä markkinoilla on suurin osa mobiilikäyttöjärjestelmistä joko Android tai iOS. On olemassa muitakin käyttöjärjestelmiä kännykkään, mutta nämä käyttöjärjestelmät ovat vain 0,27 % kaikista käytössä olevista käyttöjärjestelmistä vuonna 2020 kesäkuussa. (Statista 2020.) Vastauksina kysymykseen on sen takia kyselyssä ”Android”, ”iOS” ja ”Muu, mikä?”.

Toisena kysymyksenä on: ”Miten tiedostot pitäisi tallentaa tietokantaan? Pitäisikö tiedostot olla siirrettävissä suoraan kännykän muistista tietokantaan, vai pitäisikö dokumenteista ja kuiteista olla kuva kännykän kameran kautta, jotka tallennetaan tietokantaan?” Kysymyksen tarkoituksena on saada selville, jos mobiiliapplikaa-

tioon pitää tuoda mahdollisuus ottaa kuvia kännykän kameralla, jonka luoma tiedosto sitten tallennetaan tietokantaan vai pitäisikö tiedosto olla vain mahdollista tallentaa suoraan puhelimen muistista. Puhelimen muistista tallennetut tiedostot voivat olla kaikki tiedostot, kuten esimerkiksi sähköpostista ladatut Excel-taulukot tai Word-tiedostot. Kyselyssä kysytään myös, jos metsäyrityksen omistajat haluavat kummatkin saada mahdollisuuden siirtää tiedostoja kummallakin tavalla.



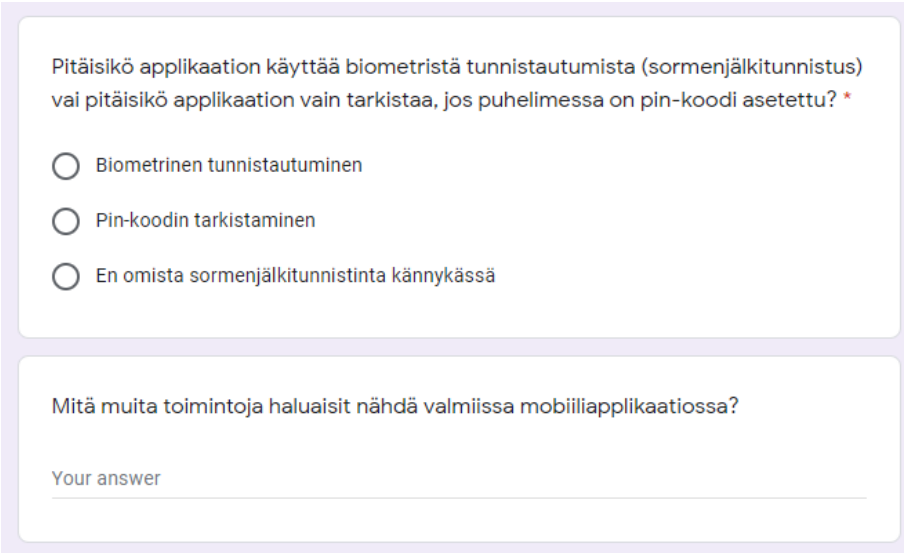
Kuva 1. Ensimmäiset kaksi kysymystä kyselyssä

Kolmantena kysymyksenä on: ”Pitäisikö applikaation käyttää biometristä tunnistautumista (sormenjälkitunnistus) vai pitäisikö applikaation vain tarkistaa, jos puhelimessa on pin-koodi asetettu?” (Kuva 2.) Tällä kysymyksellä kysytään, millä tavalla tunnistautuminen tehdään mobiiliapplikaatiossa.

Biometrisessä tunnistautumisessa Androidissa kysytään sormenjälkeä käyttäjältä. Tämä sormenjälki on tallennettu puhelimen muistiin (Android 2020). Toisena mahdollisuutena on vaatia käyttäjältä pin-koodi, jotta käyttäjä voi avata applikaation. Kysymykseen vastaukset ovat ”Biometrinen tunnistautuminen”, ”Pin-koodin tarkistaminen” tai ”En omista sormenjälkitunnistinta kännykässä”. Jos käyttäjä vastaa kysymykseen kolme, pitää olla muukin tapa tunnistautua kuin käyttämällä biometristä tunnistautumista.

Viimeisenä kysymyksenä on: ”Mitä muita toimintoja haluaisit nähdä valmiissa mobiiliapplikaatiossa?” Tällä kysymyksellä on tarkoitus saada tietoon, jos omistajilla

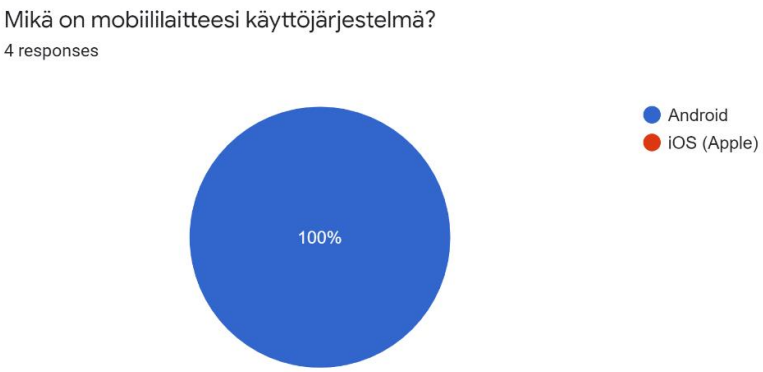
on jotain ideoita mobiiliapplikaatioon, joita voisi implementoida. Näin saadaan luotua mahdollisimman käytännöllinen tuote yrityksen omistajille.



Kuva 2. Kaksi viimeistä kysymystä

Luotuun kyselyyn vastasi neljä metsänomistajaa. Kysely luotiin täydentämään palaverissa käytyjä asioita, jotka liittyvät luotuun mobiiliapplikaatioon. Keskustelussa käytiin läpi, mitä asioita mobiiliapplikaatiossa täytyisi olla. Analyysin päätteeksi luodaan wireframe, jonka pohjalle rakennetaan mobiiliapplikaatio. Wireframe on tapa suunnitella ja visualisoida valmis ohjelma käyttämällä kuvia ja kuvioita (Athuraliya 2020).

Kaikki metsänomistajat omistivat Android-käyttöjärjestelmällä varustetun laitteen (Kuvio 1). Koodi tulee olemaan yhteensopiva vain Android-puhelimien kanssa.



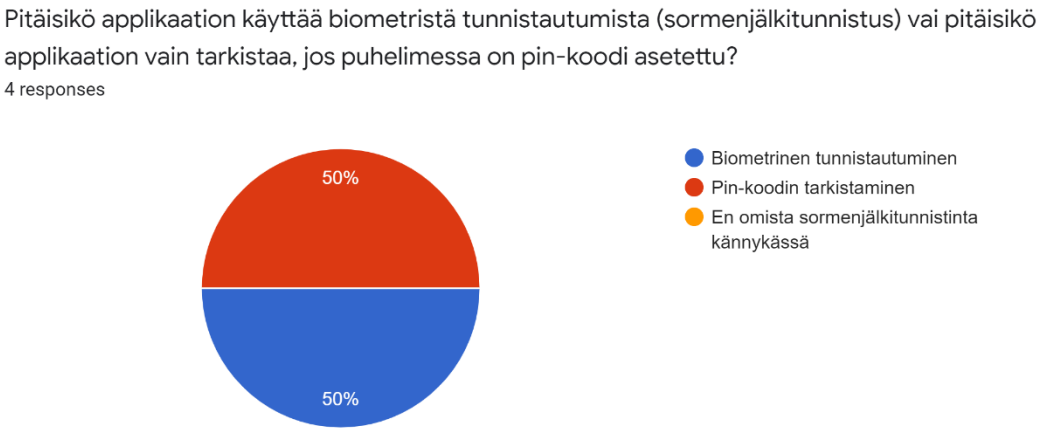
Kuvio 1. Metsänomistajien käyttämät käyttöjärjestelmät

Tulevaisuudessa, jos on tarvetta saada applikaatio toimimaan iOS-järjestelmissä, täytyy koodi luoda uudelleen. Androidin ja Applen koodit eivät toimi keskenään, koska Androidilla on eri komponentteja niiden käyttöjärjestelmässä verrattuna Appleen. (Lastovetska 2018.)

Kuten kuvio 2 osoittaa, yrityksessä 75 % vastaajista haluaisi, että he voivat ottaa kännykän kameralla kuvia dokumenteista ja haluavat myös mahdollisuuden siirtää puhelimen muistista tiedostoja. 25 % vastaajista haluaisi vain siirtää dokumentista otettuja kuvia tietokantaan. Mobiiliapplikaatiossa pitää siis olla mahdollisuus erikseen valita joko tiedostojen siirto tai kuvien ottaminen ennen tiedostojen lisäämistä. Mobiiliapplikaation pitää tukea eri dokumentteja ja kuvatiedostoja.

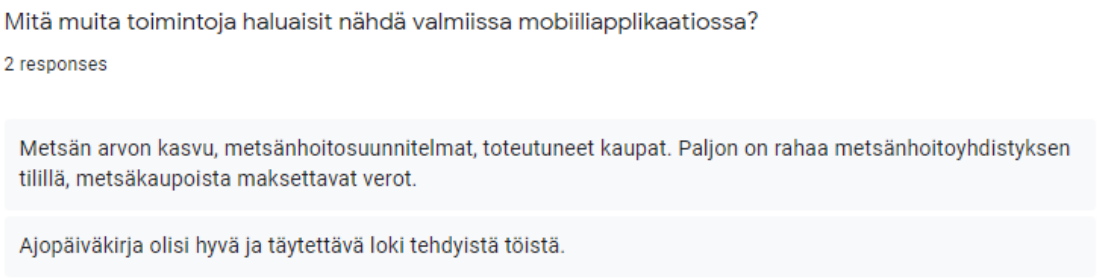
Kuvio 2. Metsänomistajien haluttu tapa tallentaa tiedostot palvelimelle

Kuvion 3 mukaisesti yrityksessä puolet (50 %) vastanneista haluaisi, että ohjelma tukee biometristä tunnistautumista. Biometrinen tunnistautuminen tulee olemaan ohjelmassa päätapa kirjautua sisään. Jos käyttäjät yrityksessä päättävät, että on pakollista olla PIN-koodilla kirjautuminen, täytyy applikaatioon lisätä mahdollisuus kirjautua PIN-koodilla. Tämä PIN-koodi säilytettäisiin palvelimella tietokannassa kryptatussa muodossa, jota ei voida ymmärtää. Kirjautuessa tämä koodi kuitenkin tarkistettaisiin PHP-skriptin kautta ja varmistetaan, jos annettu koodi vastaisi tätä kryptattua koodia. Tämä koodi olisi jokaiselle käyttäjälle oma tai yksi ja sama koodi kaikille.



Kuvio 3. Metsänomistajien haluama tapa tunnistautua mobiiliapplikaatioon

Viimeinen kysymys kyselyssä oli vapaamuotoinen. Applikaation käyttäjät halusi ohjelmaan lisätoiminnoiksi nähdä ”Metsän arvon kasvu, metsänhoitosuunnitelmat, toteutuneet kaupat. Paljon on rahaa metsänhoitoyhdistyksen tilillä, metsäkaupoista maksettavat verot.” Nämä tiedot voidaan nähdä dokumenteista, joita tallennetaan tietokantaan. Tämän takia ei ole tarvetta lisätä näitä toimintoja erikseen ohjelmaan. Omistajien ehdotus ajopäiväkirjasta ja loki tehdyistä töistä on mahdollista lisätä ohjelmaan.



Kuva 3. Vapaaehtoisen kysymyksen vastaukset

Toimeksiannosta ja kyselystä saatu data esitetään Wireframena. Wireframe toimii pohjana visuaaliselle kehitykselle, jota seurataan applikaation kehityksen ajan. Samalla on mahdollista esittää visuaalisesti kaikki toiminnot, joita tullaan koodaamaan applikaatioon.

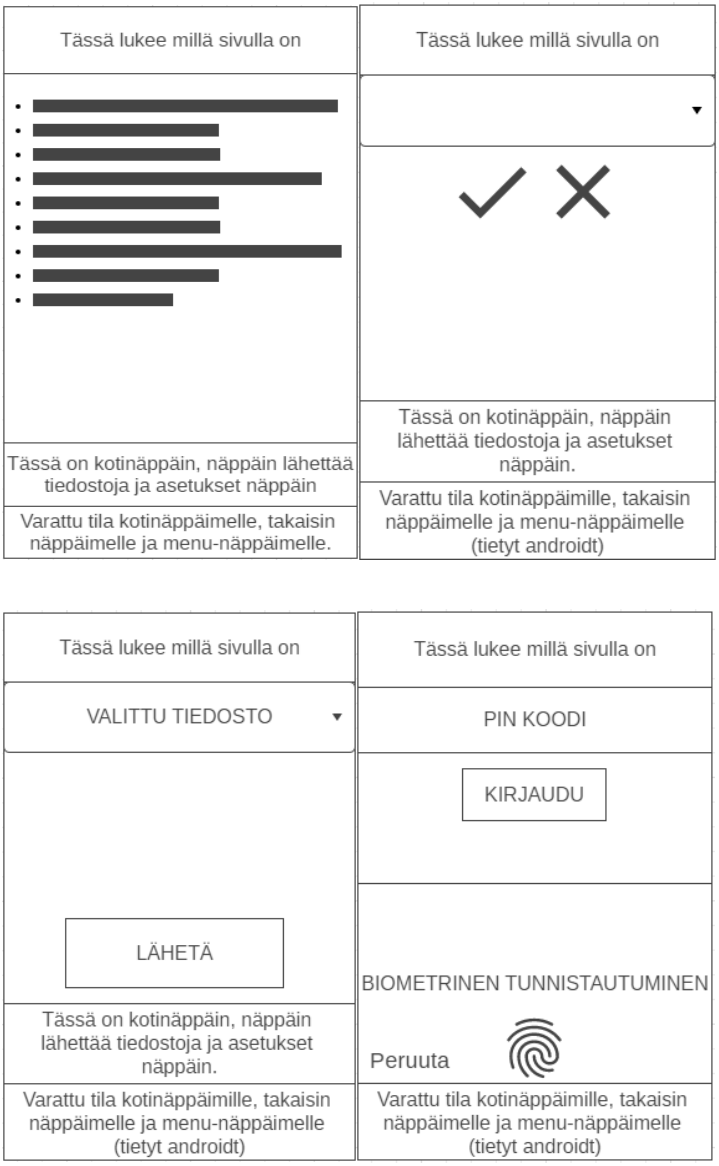
Applikaatiossa käyttäjälle kannattaa tuoda visuaalinen data mahdollisimman yksinkertaisessa muodossa. Tällä tavalla vältytään ylimääräistä rasitetta käyttäjälle ja tuodaan selkeyttä applikaatiosta etsittävästä datasta (Babich 2018).

Käyttäjien käyttöjärjestelmät ovat Android -pohjaisia, joten tämä tulee näkymään myös wireframessa. Etusivulla pitäisi olla näkymä suoraan tietokantaan. Näin käyttäjän ei tarvitsisi etsiä tiedostoja heti applikaation käynnistyttyä. Etusivun tiedostot tulee näkymään listana, josta voi linkkiä napsauttamalla ladata tiedoston suoraan kännykälle (Kuvio 4). Tämä toiminto tulee toimimaan HTML:n ja PHP:n kautta

Applikaation ”asennukset” sivulla on mahdollista muuttaa eri asetuksia. Tähän opinnäytetyöhön ei tule tarvetta lisätä asetuksia, mutta tulevaisuudessa tarvittaessa on mahdollista tälle sivulla lisätä esimerkiksi kieliasetuksia.

”Lähetettävät tiedostot” sivulla näkee napin, jolla voidaan valita tiedosto lähetettäväksi jonka voi nähdä kuvio 4:ssä tiedostojen lähetyksen wireframessa. Ohjelma menee suoraan ladatut kansioon, josta käyttäjän täytyy löytää oikeat tiedostot, jotka halutaan lähettää palvelimelle. Tiedosto lähetetään suoraan palvelimelle kansioon, jonka jälkeen sitä ei voida enää muokata.

Sormenjälkitunnistus tulee puhelimen omasta muistista, Applikaatio kysyy sormenjälkitunnistusta, mutta käyttäjä voi peruuttaa tämän toiminnon. Tulevaisuudessa applikaatioon kehitetään tapa kirjauta myös PIN-koodilla.



Kuvio 4. Etusivun wireframe, asetukset sivun wireframe, tiedostojen lähetys wireframe ja Kirjautumissivun wireframe

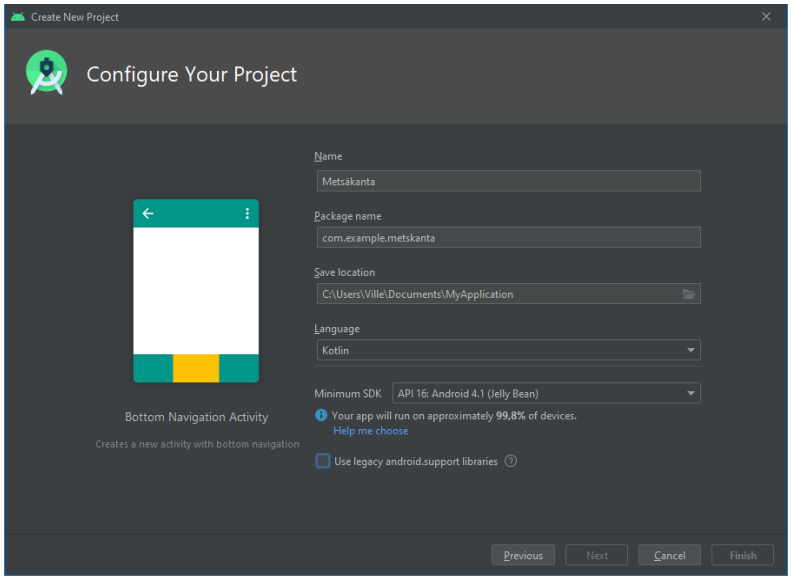
3 MOBIILIAPPLIKAATION KEHITYS JA SUUNNITTELU

3.1 Ohjelman valinta ja mobiiliapplikaation kehityssuunnitelma

Androidin koodaus tehdään Android studiolla. Android on luonut tämän ohjelman uusien applikaatioiden kehitykseen. Tästä ohjelmasta käytetään myös lyhennettä SDK eli Software developement kit. Android applikaatioissa käytetään kielenä Javaa, kotlinia ja C/C++. (Android 2020.)

Android ohjelman kehitys aloitetaan etusivusta, johon kaikki pääsevät kirjautumisen jälkeen. Ohjelman nimi on Metsäkanta ja projekti aloitetaan luomalla uusi pohja ”Bottom navigation activity”:llä, eli navigaatio näppäimet löytyvät ohjelman alaosasta. Paketin nimeksi tulee ohjelman nimi ilman ääkkösiä, jolla vältetään mahdolliset ongelmat tulevaisuudessa.

Kieleksi tulee Kotlin, jolla saadaan luotua ohjelma vähemmällä koodilla ja jota on helpompi kirjoittaa (Kotlin 2020). SDK:ksi valitaan API Level 16, joka tukee n. 99,8 % kaikista laitteista. Tiedoston sijainnilla on väliä. Tiedosto täytyy olla paikassa, johon Android Studio saa kirjoittaa.

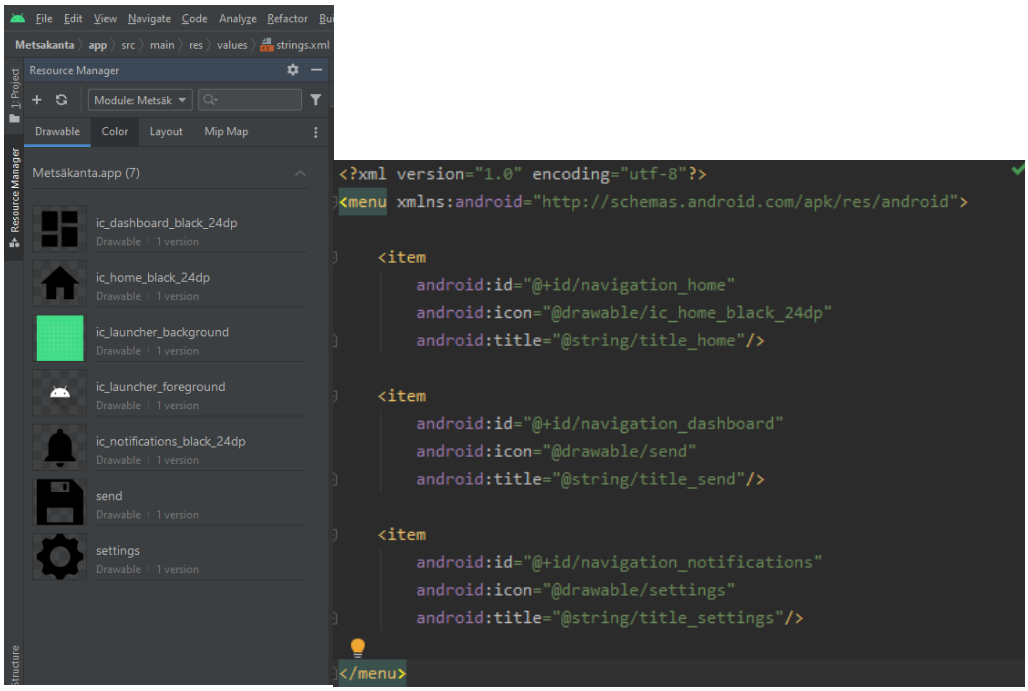


Kuva 4. Uuden projektin luonti

Projektissa muutetaan etusivun logot suomenkielisiksi. On tärkeää muuttaa ikonit ja koodi muistuttamaan haluttua lopputulosta, jotta näitä ei tarvitse muuttaa myö-

hemmin. Halutut ikonit tuodaan Android studioon klikkaamalla ”Resourse manager” ja viemällä halutut ikonit hiirellä vasempaan reunaan (Kuva 5). Kuvat löytyvät tämän prosessin jälkeen ”drawable”-kansiosta. Androidissa täytyy muokata ”strings.xml”-tiedostoa, josta Android tarkistaa kaikki muuttujat, kun niitä viitataan koodissa.

Projektissa tulee olemaan napit, jotka ovat nimeltään ”Koti, Lähetä tiedostoja ja asetukset”. Nämä näkyvät koodissa muokkauksen jälkeen nimillä ”title_home, title_send ja title_settings”. Seuraavaksi muutetaan alavalikkoa, jossa tuodaan muutetut nimet ja ikonit Androidin tietoon. Muokkaamalla android:icon ja android:title-tekstiä (Kuva 6), saadaan halutut nimet ja kuvat näkymään alavalikossa. Android vaatii, että muutokset lisätään myös ”mobile_navigation.xml”-kansioon muokkaamalla android:lable-tekstiä, jotta tulevaisuudessa on mahdollista upottaa valikot ja toiminnot muille sivuille applikaatiossa.



Kuva 5 ja 6. Resource manager (vasemmalla), Muokattu koodi (oikealla)

3.2 WebView’n luonti ja testaus

Etusivulle tullaan lisäämään WebView, jonka avulla voidaan nähdä tietokannassa olevat tiedostot. WebView yhdistää tietokantaan IP-osoitteen kautta ja näyttää sivun listana. Tämän kautta on mahdollista ladata tiedostoja suoraan mobiililaitteelle. WebView’n tarkoituksena on tuoda webbisivuja ohjelman käyttöön (Android 2020).

WebView rakennetaan koodiin tuomalla ensin Androidin tietoon, missä WebView on. WebView lisätään applikaatioon layout-kansioon halutulle sivulle kohtaan, mihin käyttäjä laskeutuu kirjautumisen jälkeen.

```
    <WebView
        android:id="@+id/webview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
/>
```

Kuva 7. Esimerkki WebView’stä koodissa

Tämän jälkeen tuodaan Androidille ylös, mihin osoitteeseen WebView yhdistää. WebView myös vaatii, että JavaScript on päällä, jotta applikaatio saa näytettyä sivun (Android 2020). Sivuksi asetetaan testiksi koodissa ”http://www.google.com”, joka näkyy kuvassa 8. Sivu muutetaan myöhemmin vastaamaan IP-osoitetta, joka ohjaa palvelimella pyörivään PHP-tiedostoon.

```
//Käynnistetään applikaation yhteydessä
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
//Etsitään etusivu ja siltä WebView
setContentView(R.layout.fragment_home)
val myWebView: WebView = findViewById(R.id.webview)
//Annetaan lupa Javascriptille
myWebView.settings.javaScriptEnabled = true
    //Annetaan sivu, mihin yhdistetään.
myWebView.loadUrl( url: "https://www.google.com")
```

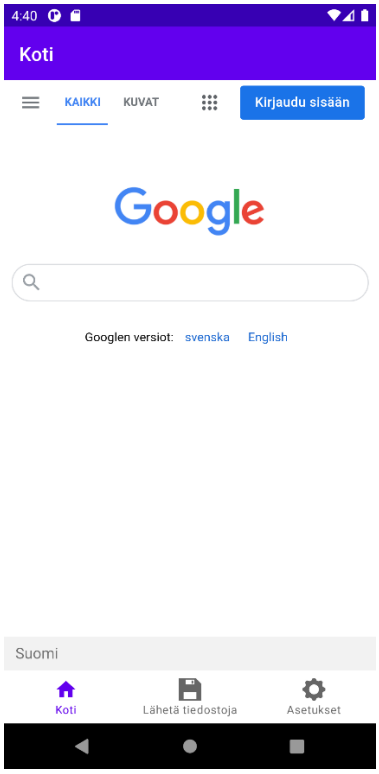
Kuva 8. Esimerkki WebView’stä koodissa

On tärkeää, että Androidille annetaan tietoon, että ohjelma vaatii Internet-yhteyden. Tämä täytyy lisätä ”AndroidManifest”-tiedostoon (Kuva 9). Jos tätä ei määritellä ohjelmassa, WebView ei toimi. Tämä johtuu siitä, että normaalisti ohjelmalla ei ole oikeuksia käyttää mitään komponenttia laitteessa

```
<!--Annetaan lupa käyttää internettiä -->
<uses-permission android:name="android.permission.INTERNET" />
```

Kuva 9. Esimerkki, jossa annetaan lupa internetin käytölle AndroidManifest-kansiossa

Tulosivulla on WebView, joten tämä käynnistetään heti applikaation käynnistyessä. WebView on ainoastaan näkyvillä Koti-sivulla, jonka näkee kuvassa 10. Tämä sivu tulee muuttumaan halutuksi sivuksi heti, kun koodissa muutetaan testisivu halutuksi IP-osoitteeksi



Kuva 10. Toiminnassa oleva WebView

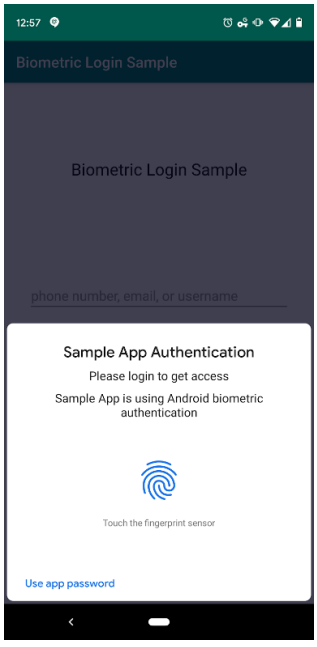
Lähetä tiedostoja -sivulla tarvitaan tapa, jolla voidaan siirtää tiedostoja palvelimelle suoraan Androidin muistista. Helpoin tapa on luoda uusi WebView. Tämä WebView tullaan yhdistämään IP-osoitteeseen, jossa on Apache serveri vastaanottamassa liikennettä. Sen sijaan että käyttäjä näkee tiedostot, WebView ohjaa käyttäjän lähetys sivulle. Palvelimella pyörii erittäin yksinkertainen HTML-tiedosto ja PHP-skripti, joka sisältää valinnan valita tiedoston ja napin lähetykselle palvelimelle (Kuva 11).

```php
<?php include 'fileupload.php';?>
<html lang="en">
  <head>
    <link rel="stylesheet" href="style.css">
    <title>Lataa tiedostoja palvelimelle</title>
  </head>
  <body>
    <div class="container">
      <div class="row">
        <!--Luodaan mahdollisuus valita tiedosto-->
        <form action="index.php" method="post" enctype="multipart/form-data" >
          <h3>Lataa tiedosto</h3>
          <input type="file" name="myfile"> <br>
          <div>
            <br>
            <br>
            <!--Nappi, jolla aktovoidaan tiedoston lähetys-->
          <button type="submit" name="save">Tallenna</button>
        </div>
        </form>
      </div>
    </div>
  </body>
</html>
```

Kuva 11. Esimerkki palvelimella pyörivästä HTML-sivusta

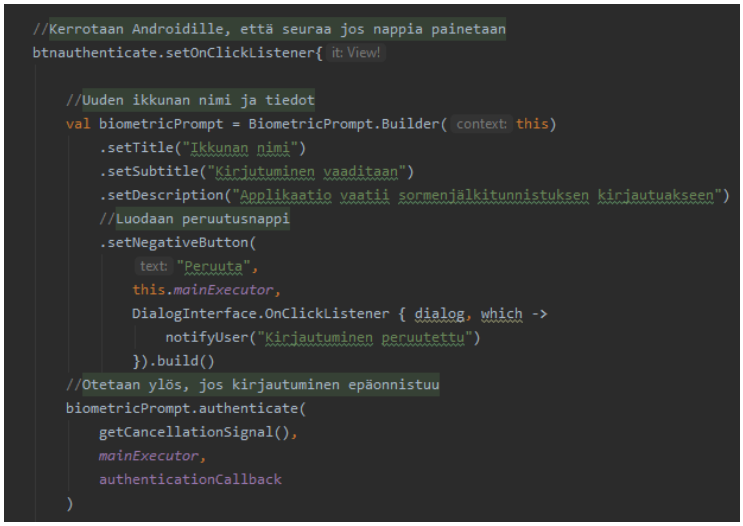
3.3 Biometrinen tunnistautuminen

Viimeisenä lisätään ”turvasivu”, jonka käyttäjä täytyy läpäistä, jotta pääsee käyttämään applikaatiota. Tämä sivu aukeaa heti, kun applikaation aukaisee. Tämä sivu sisältää vain napin, josta pääsee kirjautumaan sisään. Nappia painaessa tulee ilmoitus, jonka voi nähdä kuvassa 12.



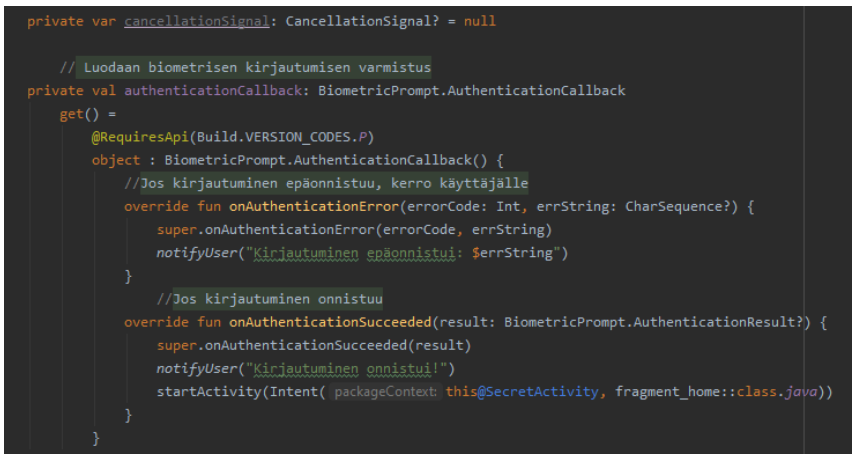
Kuva 12. Miltä biometrinen kirjautuminen näyttää Androidilla (Android 2020)

Ohjelma antaa mahdollisuuden skannata sormenjäljen tai peruuttaa skannauksen. Jos skannaus peruutetaan, ohjelma ei päästä eteenpäin tältä sivulta. Kirjautuminen onnistuu, jos sormenjälki skannataan ja käyttäjä ei peruuta kirjautumista. Onnistuneesta kirjautumisesta siirrytään nykyiseltä välilehdeltä ”activity_main”-välilehdelle, josta pääsee selaamaan applikaation muita ikkunoita.



Kuva 13. Esimerkki tunnistusikkunasta

Jos kirjautuminen epäonnistuu, applikaatio antaa siitä tiedon käyttäjälle ja kertoo, miksi kirjautuminen on epäonnistunut.



Kuva 14. Esimerkki kirjautumisen ilmoittamisesta

Kirjautuminen epäonnistuu, jos käyttäjä painaa peruuta näppäintä missään vaiheessa kirjautumista. Kirjautuminen estetään myös, jos käyttäjä ei ole aktivoinut sormenjälkitunnistinta, tai ei ole asettanut sormenjälkeä laitteeseen (Kuva 15).

```
//Ilmoitetaan käyttäjälle virheestä
private fun notifyUser(message: String) {
    Toast.makeText( context: this, message, Toast.LENGTH_SHORT).show()
}

//Tarkista jos käyttäjä painaa "peruuta" näppäintä
private fun getCancellationSignal(): CancellationSignal {
    cancellationSignal = CancellationSignal()
    cancellationSignal?.setOnCancelListener {
        notifyUser("Kirjautuminen keskeytyi käyttäjän toimesta")
    }
    return cancellationSignal as CancellationSignal
}

//Tarkista, jos laitteelle on asetettu sormenjälkitunnistus
private fun checkBiometricSupport(): Boolean {

    val keyguardManager = getSystemService(Context.KEYGUARD_SERVICE) as KeyguardManager

    if (!keyguardManager.isKeyguardSecure) {
        notifyUser("Sormenjälkitunnistusta ei ole aktivoitu asetuksista")
        return false
    }
```

Kuva 15. Esimerkki virheilmoituksista käyttäjälle

Käyttäjä myös ei voi käyttää sormenjälkitunnistinta, jos käyttäjä on estänyt applikaation oikeuden käyttää sormenjälkitunnistinta (Kuva 16).

```
        }
        //Tarkista jos käyttäjä on antanut luvan käyttää sormenjälkitunnistusta
        if (ActivityCompat.checkSelfPermission(
                context: this,
                android.Manifest.permission.USE_BIOMETRIC
            ) != PackageManager.PERMISSION_GRANTED
        ) {
            notifyUser("Et ole antanut lupaa käyttää sormenjälkitunnistusta")
            return false
        }
        //Tarkista jos käyttäjällä on sormenjälkitunnistin
        return if (packageManager.hasSystemFeature(PackageManager.FEATURE_FINGERPRINT)) {
            true
            //Jos mikään näistä virheistä ei tapahdu, päästä käyttäjä kirjautumaan
        } else true
    }
}
```

Kuva 16. Esimerkki virheilmoituksesta ja myös, jos virhettä ei tapahdu

4 PALVELIMEN PERUSTAMINEN JA KONFIGUROINTI

4.1 XAMPP ja palvelimen tiedostot

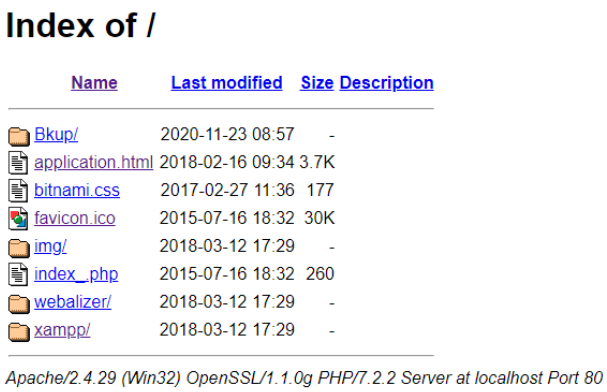
Palvelimen tarkoitus on säilyttää tiedostoja ja näyttää Androidin WebView'lle sivua, josta voi siirtää tietoja palvelimen kovalevylle säilöön. SQL- ja PHP-tiedostot luodaan ja testataan asentamalla XAMPP, joka on lyhenne sanoista Cross-Platform, Apache, MySQL, PHP ja Perl (Walia & Gill 2014). XAMPP on avoimen lähdekoodin ohjelmisto, jolla mahdollistetaan työn testauksen ja kehittämisen ilman palvelinta (Apache friends 2020).

XAMPP:lla tiedostot valmistellaan siirtoa varten Linux-ympäristölle, joka asennetaan palvelimelle. XAMPP sisältää Apache palvelimen ja MySQL-osan, jota tarvitaan WebView'tä varten (Apache friends 2020). Apachen tarkoitus on luoda yhteys palvelimen ja selaimen kanssa (Domantas 2020), ja MySQL antaa mahdollisuuden tallentaa tiedostoja palvelimelle (MySQL 2020).

Kehitysympäristönä toimii Windows 10, joka on yhdistetty LAN-verkkoon. Työkaluna on Chrome 87.0, jonka kautta pääsee avaamaan XAMPP:n tiedostoja. XAMPP asennetaan kehitysympäristöön lataamalla ”XAMPP for Windows” -tiedosto Apache Friends:n sivustolta. Tämä asennus asentaa kaikki tarvittavat palvelut, jolla voidaan testata tiedostojen toimivuus valmiiksi. Tällä tavalla ei ole tarvetta testata tiedostojen toimivuutta tiedostojen siirron jälkeen.

Verkkoon tuodaan myös toinen laite, jolla voidaan tarkistaa, että tiedostot toimii myös verkon yli. Tässä kehitysympäristössä tiedostojen toimivuus testattiin mobiililaitteella, johon on asennettu Android käyttöjärjestelmä. Kehitysympäristön tarkoitus on testata luotuja tiedostoja ja palveluita, jotka viedään Linux palvelimelle kehityksen tuloksena.

XAMPP tarkistaa asennuksen jälkeen ”xampp”-kansiosta, jos ”htdocs”-kansiossa on tiedostoja. Jos tässä kansiossa löytyy ”index.html”, XAMPP näyttää tämän tiedoston suoraan, kun palvelimen IP-osoitteeseen yhdistetään. Jos tätä tiedostoa ei löydy, XAMPP näyttää kaikki tiedostot kyseisessä kansiossa.



Kuva 17. XAMPP yhdistettynä localhostiin ilman index-tiedostoa

Jotta voidaan yhdistää tähän palvelimeen, vaaditaan kaksi porttia auki. Apcahe vaatii portin 80 ja 8000 aukinaiseksi, jotta selaimet voivat nähdä nettisivun (Apache 2020).

Androidin WebView'lle täytyy luoda sivu, josta voi ladata tiedostot palvelimelle ja sivu, mistä voidaan löytää tiedostot ja ladata ne laitteelle. Aloitetaan luomalla mahdollisuus ladata tiedostoja palvelimelle, jolla voidaan myöhemmin testata, jos tiedostot näkyy palvelimella latauksen jälkeen.

MySQL-palvelimelle luodaan uusi tietokanta, jotka pitää tiedon kaikista tiedostoista. Tämä palvelin sisältää kaksi taulukkoa, johon tieto tallennetaan. Tätä taulukkoa kysytään, kun tiedostoja halutaan ladata palvelimelta.

”htdocs”-kansioon luodaan uusi tiedosto nimeltään ”index.php”. Luotuun tiedostoon koodataan mahdollisuus ladata tietoa palvelimelle. Palvelimeen myös luodaan uusi kansio nimeltään ”tiedostot”, minne kaikki ladattu data menee. Index.php sisältää napin, jonka kautta voidaan ladata tiedosto palvelimelle (Kuva 11.), ja PHP-skriptin, joka avaa yhteyden SQL-palvelimelle ja tallentaa tämän tiedoston kovalevylle. SQL ja PHP näyttää palvelimen tiedostot eri sivulla, jotka voidaan linkkiä painamalla ladata laitteelle.

```php
<?php
// Yhdistä tietokantaan
//"localhost" muutetaan IP-osoitteeseen, missä palvelin toimii.
//"tietokanta" muutetaan tietokannan nimeksi
$conn = mysqli_connect('localhost', 'root', '', 'tietokanta');

// Tallennetaan tiedosto
if (isset($_POST['save'])) { // Jos käyttäjä painaa tallenna näppäintä, jatka
    // Hae nimi tiedostolle
    $filename = $_FILES['myfile']['name'];

    // Päätä sijainti tiedostolle
    $destination = 'tiedostot/' . $filename;

    // Ota selville, minkälainen tiedosto
    $extension = pathinfo($filename, PATHINFO_EXTENSION);

    // Luo tiedosto hetkellisesti palvelimelle
    $file = $_FILES['myfile']['tmp_name'];
    $size = $_FILES['myfile']['size'];

    if (!in_array($extension, ['zip', 'pdf', 'docx', 'jpg', 'xlsx', 'xlsm'])) {
        echo "Tiedostopääte on oltava .zip, .pdf, .docx, .jpg, tai .xlsx/xlsm";
    } elseif ($_FILES['myfile']['size'] > 20000000) { // Tiedosto ei saa olla yli 20 megatavua
        echo "Tiedosto on liian iso";
    } else {
        // Siirä väliaikainen tiedosto palvelimelle
        if (move_uploaded_file($file, $destination)) {
            $sql = "INSERT INTO files (name, size) VALUES ('$filename', $size)";
            if (mysqli_query($conn, $sql)) {
                echo "Tiedosto tallennettu!";
            }
        } else {
            echo "Tiedoston tallennus epäonnistui!";
        }
    }
}
```

Kuva 18. Esimerkki PHP-tiedostosta

Seuraavaksi luodaan Androidille sivu, josta voidaan nähdä kaikki tiedostot tietokannassa. Nämä tiedostot näytetään yksinkertaisessa luettelossa, josta voidaan linkkiä painamalla ladata tiedosto. ”index.php”-tiedostoon lisätään alkuun seuraava pätkä koodia, joka käyttää tiedoston lataamisen yhteydessä luotua koodia uudelleen.

```php
//kysytään, mitä tiedostoja tietokannassa on
$sql = "SELECT * FROM files";
$result = mysqli_query($conn, $sql);

$files = mysqli_fetch_all($result, MYSQLI_ASSOC);
```

Kuva 19. Lisätty koodi index.php-kansioon

Uusi sivu tuo esille kaikki tiedostot tietokannassa luomalla uusia rivejä käyttämällä PHP-skriptiä. Tämä skripti luo tiedostoille myös linkin, jonka kautta tiedostot voidaan ladata suoraan laitteelle. Tiedosto latautuu suoraan laitteen ”ladatut”-kansioon, josta voidaan ladattu tiedosto tämän jälkeen avata. Koodi tiedoston lataamiseen tullaan lisäämään ”fileupload.php”-kansioon edellisen koodin jälkeen

```php
// Jos scriptiin tullaan asetetulle file_id:llä,
if (isset($_GET['file_id'])) {
    $id = $_GET['file_id'];

    // Haetaan tiedsoto tietokannasta
    $sql = "SELECT * FROM files WHERE fileID=$id";
    $result = mysqli_query($conn, $sql);

    $file = mysqli_fetch_assoc($result);
    $filepath = 'tiedostot/' . $file['name'];

    // Aloitetaan tiedoston siirto
    if (file_exists($filepath)) {
        header('Content-Description: File Transfer');
        header('Content-Type: application/octet-stream');
        header('Content-Disposition: attachment; filename=' .basename($filepath));
        header('Content-Transfer-Encoding: binary');
        header('Expires: 0');
        header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
        header('Pragma: public');
        header('Content-Length: ' . filesize('tiedostot/' . $file['name']));
        ob_clean();
        flush();
        readfile('tiedostot/' . $file['name']);; //Absolute URL
        exit();

    }

}
```

Kuva 20. Fileupload.php-kansioon lisätty koodi edellisen koodin jälkeen

4.2 Linux-palvelimen asennus ja tiedostojen siirto

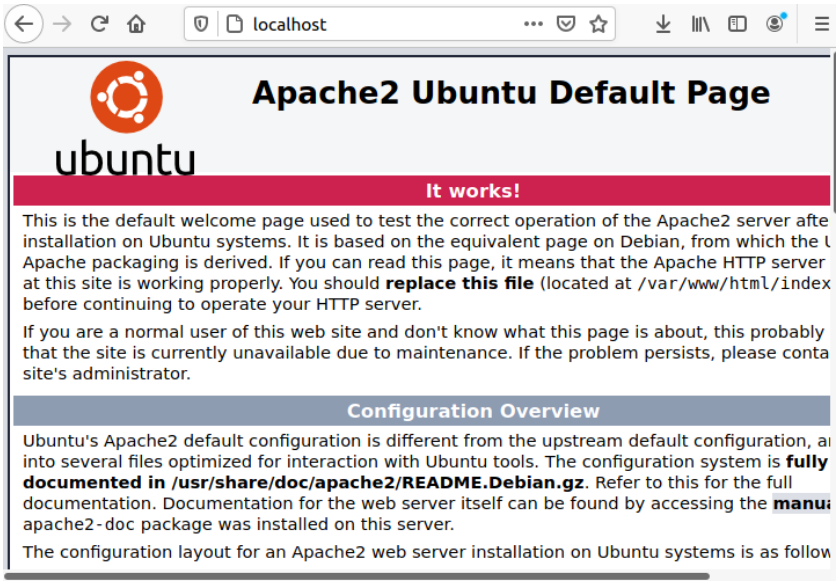
Linux valittiin palvelimeksi, koska Linux on ilmainen avoimella lähdekoodilla varustettu käyttöjärjestelmä. Tällöin on mahdollista välttyä lisenssi maksuilta, joka vähentää meneviä kuluja. (Leslie 2020.) Palvelimeen asennetaan Ubuntu, joka on käyttäjäystävällinen ja yksi suosituimmista Linux versioista (Hasan 2020). Tätä Ubuntua ei ole tarkoitus ohjata etäisesti, joten tämän takia palvelimeen ei asenneta Ubuntu Serveriä.

Linuxiin asennetaan Apache, MySQL ja PHP eli LAMP, joka tarkoittaa ”Linux operating system, with the Apache web server. The site data is stored in a MySQL database, and dynamic content is processed by PHP.”. (Bearnes 2016.) Nämä palvelut ovat samat kuin XAMPP:lla, joten luodut tiedostot toimivat Ubuntussa pelkällä tiedostonsiirrolla.

LAMP:n asennus alkaa lataamalla ja asentamalla Apache2 ohjelmiston, joka tapahtuu ensin päivittämällä ”apt-get”-kirjasto Ubuntussa, jonka jälkeen asennetaan itse ohjelmisto. Linuxilla terminaalissa nämä komennot ovat ”sudo apt-get update” ja ”sudo apt-get install apache2”. Tämän jälkeen testataan, jos asennus onnistui komennolla ”sudo apache2ctl configtest”.

Apache ilmoittaa onnistuneessa asennuksessa yhden virheen, joka on ilmoitus puuttuvasta globaalista palvelimen nimestä. Tämä virhe on harmiton ja se voidaan ohittaa. (Bearnes 2016.)

Asennuksen jälkeen avataan palvelimeen palomuurin portit Apachelle. Apachelle avataan portit 80 ja 443 komennolla ”sudo ufw allow in "Apache Full"”, jonka jälkeen testataan selaimessa, jos Apache on toiminnassa. Apache kiinnittyy palvelimen ulkoiseen IP-osoitteeseen, joka mahdollistaa palvelimeen yhdistämisen IP-osoitteella toiselta laitteelta. IP-osoitteen voi vahtaa Apachen config-tiedostosta haluamakseen. Jos kaikki on asennettu oikein, selain näyttää Apache2 testisivun (Kuva 21).

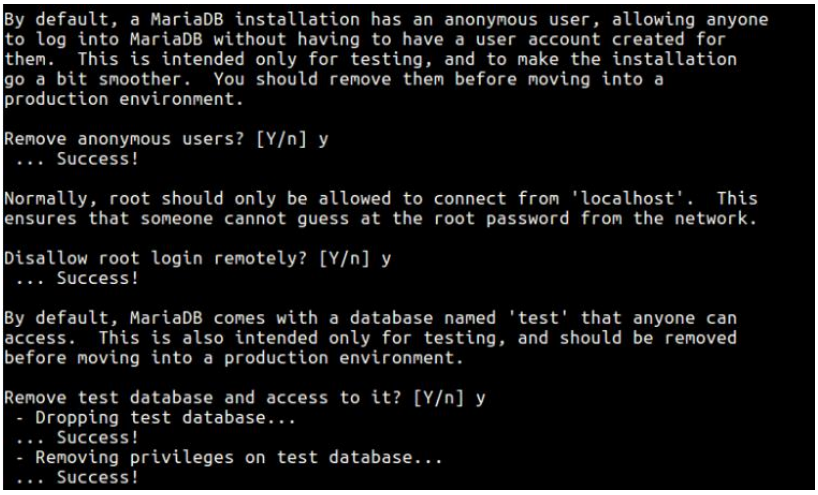


Kuva 21. Onnistunut Apachen asennus

MySQL palvelin asennetaan palvelimelle komennolla ”sudo apt-get install mysql-server”. Asennuksen jälkeen parannetaan tietoturvaa palvelimessa muuttamalla MySQL:n asetuksia. Asetuksen automaattien ohjaus käynnistetään komennolla ”mysql_secure_installation”. (Bearnes 2016.)

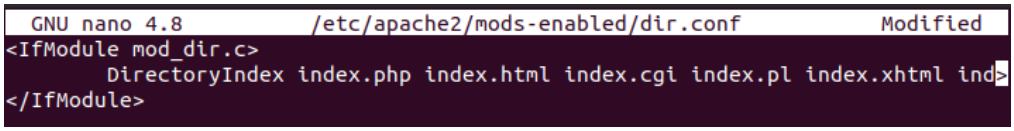
Asennus vaatii Linuxissa ”root”-oikeudet, jotka annetaan komennolla ”su -” ja antamalla salasanan. Asennuksen aikana kysytään, jos halutaan, että on mahdollista kirjautua MySQL-tietokantaan nimettömänä. Asennus kysyy, jos nämä nimettömät käyttäjät poistetaan. Tässä palvelimessa on vain yksi käyttäjä, eikä ole tarvetta olla muita käyttäjiä, joten tässä esimerkissä poistetaan kaikki nimettömät

käyttäjät. Samalla voidaan päättää, voiko käyttäjä ainoastaan kirjautua fyysisesti koneelta, eikä etänä. Tämä estetään koska palvelimella käydään ainoastaan fyysisesti eikä ole tarvetta etäohjaukselle. Viimeisenä asennus kysyy, jos poistetaan testitaulukko tietokannasta. Tähän taulukkoon pääsee kaikki käsiksi, joten ei ole järkevä pitää sitä palvelimella turhaan.



Kuva 22. Kysymykset secure installationissa (Kili 2017)

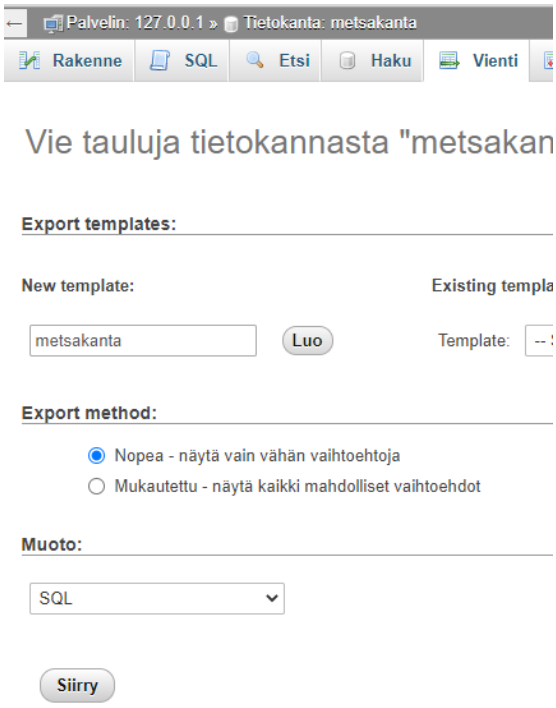
Viimeiseksi Linuxiin asennetaan PHP-palvelin. PHP:n asennuksessa lisätään mukaan komponentteja, jotka keskustelevat Apachen ja MySQL:n kanssa. Komento tähän asennukseen on ”sudo apt-get install php libapache2-mod-php php-mysql” (Bearnes 2016.) Asennuksen jälkeen kerrotaan Apachelle, että etsitään PHP-tiedostoja ennen HTML-tiedostoja. Tämä onnistuu komennolla ”sudo nano /etc/apache2/mods-enabled/dir.conf”. Tiedostossa muutetaan ”index.php” ensimmäiseksi ja jätetään ”index.html” toiseksi (Kuva 23). Lopuksi tallennetaan muutokset ja käynnistetään Apache uudestaan komennolla ”sudo systemctl restart apache2”.



Kuva 23. ”dir.conf”-tiedoston sisältö muutettuna

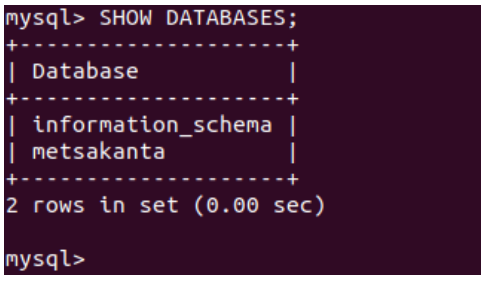
MySQL-tietokannan siirto onnistuu klikkaamalla tietokannan admin-ikkunassa haluttua tietokantaa, ja sen jälkeen viemällä tiedoston SQL-tiedostoksi (Kuva 23). Tekstitiedosto sisältää käskyjä, jotka luovat tietokannan uusiksi uudelle pohjalle.

Samalla otetaan kaikki tiedostot, jotka ovat ”htdocs”-kansiossa ja siirretään Ubuntulle USB-tikun kautta.



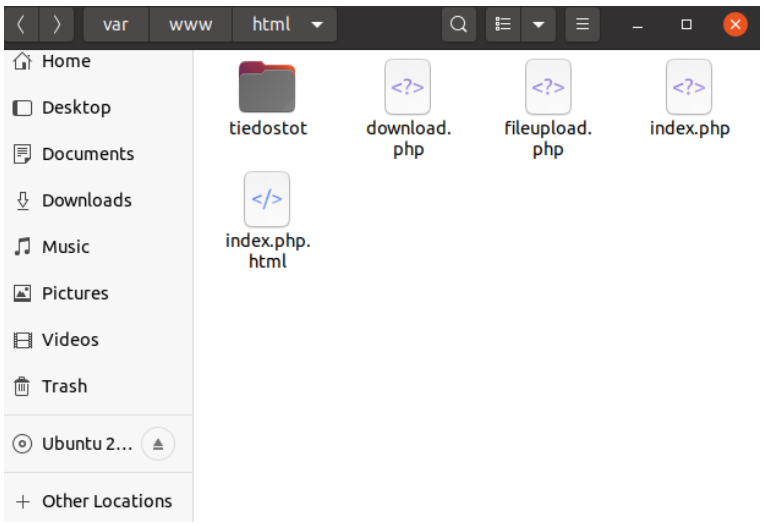
Kuva 24. Tietokannan vienti phpMyAdminissa

SQL-tiedosto tuodaan palvelimelle ensin avaamalla MySQL-terminaalissa jonka jälkeen luodaan uusi tietokanta, jonka nimi tässä esimerkissä on ”metsakanta” komennolla ”CREATE DATABASE metsakanta;”. Uuden tietokannan luonnin jälkeen suljetaan MySQL-ikkuna ja siirrytään terminaalissa ”Downloads” kansioon. Tämän jälkeen komennolla ”sudo mysql -u root -p metsakanta < metsakanta.sql” viedään kaikki asetukset SQL-tiedostosta uuteen MySQL-tietokantaan.



Kuva 25. Onnistuneesti viedyn tietokannan näkymä terminaalissa

PHP-tiedostot siirretään Apachen ”html”-kansioon, josta Apache osaa yhdistää oikeaan PHP-tiedostoon. Tämä kansio sijaitsee Ubuntussa ”/var/www/html/”. Htdocs-kansio puretaan Ubuntussa komennolla ”sudo unzip htdocs.zip -d /var/www/html”. Terminaalin pitää olla ladatussa kansiossa, ennen kuin komento annetaan, tai tulee virhe. Tämän jälkeen avataan terminaali ”html”-kansiossa, ja poistetaan ylimääräinen ”index.html”. Lopuksi PHP-tiedostossa vaihdetaan tiedostojen tiedot oikeaksi, eli tiedoston sijainnit vastaamaan Ubuntussa olevia tiedostoja ja IP-osotteita. Tämän jälkeen voidaan testata, jos IP-osoitteeseen yhdistämällä päästään lataussivustolle. Tämän jälkeen voidaan ladata tiedosto palvelimelle ja varmistaa, että tietokanta ja PHP toimii yhdessä.



Kuva 26. Valmis HTML-tiedosto

5 POHDINTA

Opinnäytetyössä otin selville, miten ohjelman suunnittelu aloitetaan. Suunnitelman tehtyä oli tärkeää, missä järjestyksessä kehityksen aloittaa. On myös tärkeää pitää suunnitelmasta kiinni, jotta applikaatioon ei tulisi ylimääräisiä toimintoja. Otin myös selville, miten onnistuisin luomaan kommunikointiyhteyden applikaation ja luodun palvelimen välille.

Lähdin kehittämään tätä opinnäytetyötä, koska halusin ottaa selvää, kuinka helppoa olisi luoda uusi mobiiliapplikaatio ja palvelin ilman erityisempää kokemusta koodaamisesta. Oli mielenkiintoista luoda uusi ympäristö metsäyhtymälle, joka ei ennestään ollut sähköistänyt dokumentaatiota. Tämä toimeksianto antoi hyvin vapaat kädet omalle suunnittelulle.

Opinnäytetyö aloitettiin kyselylomakkeen luomisella, jolla kartoitettiin toimeksiantajan tarpeet apukysymyksiä hyväksikäyttäen mobiiliapplikaation kehittämiseksi. Kyselylomakkeen luonti oli haasteellista, mutta palaverin jälkeen täydentävillä kysymyksillä pystyin hahmottamaan, millainen mobiiliapplikaatio olisi juuri oikea tämän yrityksen käyttöön.

Koodaaminen Androidille oli paljon vaikeampaa kuin luulin aluksi, vaikka nykyään saatavilla olevat koodaustyökalut ovat hyvinkin edistyneitä. Ubuntu-palvelimen luonti oli kiinnostavaa ja opetti, miten käytetään Debian-pohjaisia Linux-käyttöjärjestelmiä, asennetaan ja muokataan haluamaksi. Erityisesti terminaalin käyttöä oli paljon ja graafista-käyttöliittymään puolta tuli käytettyä yllättävän vähän. Linuxin valitsin palvelimeksi, koska se on ilmainen, eikä sen asentaminen tai päivitykset vaadi maksua.

Tämä oli minun ensimmäinen toimeksiantoni metsäyhtiölle, jossa oli minulle mahdollisuus oppia uusia asioita ja kehittää niitä. Tämä loi vankan pohjan tekemiselle ja oppimiselle.

Kehittämäni applikaatio on monipuolinen, joka vaatii tietokantojen käyttöä ja jota voi päivittää internetin välityksellä. Ensimmäiseksi työksi applikaation kehittäminen tuntuu hyvältä vaihtoehdolta opinnäytetyön aiheeksi. Projektiin kehittämiseen kului eniten aikaa koodatessa, koska en tunne käytettyjä koodikieliä hyvin.

Suurin haaste opinnäytetyössä oli mobiiliapplikaation luominen. Uuden kielen opettelu vei suurimman osan ajasta, jota käytin tässä projektissa. Muut kielet kuten HTML, XML, PHP ja SQL oli jo opittuna, mutta näitä kieliä ei ole tullut käytettyä ennen tätä opinnäytetyötä moneen vuoteen, mikä toi hankaluuksia alussa.

Työ oli mielenkiintoinen ja silmiä avaava kokemus koodaamiseen ja ohjelmien suunnitteluun. Tämä työ antoi hyvän esimerkin, mitä ohjelmistosuunnittelijan työ on ja mitä haasteita ja vastuita suunnittelijalla on. Tulevaisuudessa tulen keskittymään enemmän suunnitteluvaiheeseen. Olisi tärkeää saada tarkat suunnitelmat heti ennen kuin aloittaa mobiiliohjelman kehityksen. Tällä tavalla helpotettaisiin työtaakkaa, joka tulee koodaamisessa. Tämä opinnäytetyö oli käytännönläheinen ja uskon että tästä on hyötyä työelämässä.

Tietoturvallisuuteen olisi voinut painottaa enemmän palvelimen luonnissa. Olisin halunnut käydä enemmän läpi, miten voitaisiin estää ulkopuolisilta kirjautuminen PHP-skriptillä tai miten luodaan Admin-puoli, jonka kautta voitaisiin etänä poistaa tiedostoja tietokannasta. Parantamisen kohteita löytyy jatkuvasti ja näitä parannuksia tullaan luomaan yritykselle vuosien mittaan.

Opinnäytetyötä seuraamalla lukija voisi luoda uuden ympäristön ja pienen mobiiliapplikaation. Opinnäytetyön idea on olla dokumentointi ja ohjeistus vastaavanlaisen ympäristön perustamiseen. Tietenkin tekniikka muuttuu jatkuvasti ja parempia tapoja luoda vastaavanlainen ympäristö saattaa ilmestyä milloin tahansa.

LÄHTEET

Android 2020. Android Developers. Viitattu 15.11.2020 https://developer.android.com/studio.

Apache 2020. Binding to Addresses and Ports. Viitattu 23.11.2020 https://httpd.apache.org/ABOUT_APACHE.html.

Apache friends 2020. XAMPP Apache + MariaDB + PHP + Perl. Viitattu 16.2.2021 https://www.apachefriends.org/about.html.

Athuraliya, A. 2020. Step-by-Step Visual Guide to Mobile App Planning. Creately 9.9.2020. Viitattu 23.9.2020 https://creately.com/blog/diagrams/how-to-plan-an-app-visually/.

Babich, N. 2018. A Comprehensive Guide To Mobile App Design. Smashing magazine 12.2.2018. Viitattu 14.10.2020 https://www.smashingmagazine.com/2018/02/comprehensive-guide-to-mobile-app-design/.

Bearnes, B. 2016. How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 16.04. Digital Ocean 21.4.2020. Viitattu 17.12.2020 https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-16-04.

Cabot 2020. Mobile App Development Client Questionnaire. Viitattu 17.8.2020 https://www.cabotsolutions.com/public/Mobile-App-Development-Client-Questionnaire.pdf.

Domantas, G. 2020. What is Apache? An In-Depth Overview of Apache Web Server. Hostinger tutorials 17.6.2020. Viitattu 23.11.2020 https://www.hostinger.com/tutorials/what-is-apache.

Haselmayr, M. 2013. How To Build Your First Mobile App In 12 Steps: Part 1. Forbes 30.8.2013. Viitattu 11.8.2020 https://www.forbes.com/sites/allbusiness/2013/10/30/how-to-build-your-first-mobile-app-in-12-steps-part-1/.

Hasan, M. 2020. The 8 Most Popular Linux Distros Available Out There for 2020. Ubuntu Pit 11.12.2020. Viitattu 15.12.2020 https://www.ubuntupit.com/most-popular-linux-distros-available-out-there/.

Jenn, N. C. 2006. Designing A Questionnaire. Malays Fam Physician. 2006; 1(1): 32–35.

Kotlin 2020. Frequently asked questions (FAQ). Viitattu 16.11.2020 https://kotlinlang.org/docs/reference/faq.html.

Kili, A 2017. 12 MySQL/MariaDB Security Best Practices for Linux. Tecmint 1.12.2017. Viitattu 17.12.2020 https://www.tecmint.com/mysql-mariadb-security-best-practices-for-linux/.

Lastovetska, A. 2018. How to Convert an Android App to iOS or Vice Versa: 4-Step Process. MLSDev 6.12.2018. Viitattu 8.9.2020 https://mlsdev.com/blog/how-to-convert-android-app-to-ios.

Leslie, A. 2020. Linux Hosting vs. Windows Hosting (2020): 6 Differences & 6 Best Hosts. HostingAdvice.com 14.4.2020. Viitattu 15.12.2020 https://www.hostingadvice.com/how-to/linux-hosting-vs-windows-hosting/.

MySQL 2020. MySQL Documentation. Viitattu 23.11.2020 https://dev.mysql.com/doc/refman/8.0/en/introduction.html.

Statista 2020. Mobile operating systems' market share worldwide from January 2012 to July 2020. Viitattu 21.8.2020 https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.

Varshneya, R. 2019. A Step-by-Step Guide To Building Your First Mobile App. Entrepreneur 13.6.2019. Viitattu 11.8.2020 https://www.entrepreneur.com/article/231145.

Walia, S. & Gill, S. K. 2014. A Framework for Web Based Student Record Management System using PHP. International Journal of Computer Science and Mobile Computing 8.8.2014. Viitattu 20.11.2020 https://www.ijcsmc.com/docs/papers/August2014/V3I8201409.pdf.