Karelia-ammattikorkeakoulu
Tradenomi (AMK)

Fullstack-sovelluksen toteuttaminen Androidille

Roosa Kontinen

Opinnäytetyö, marraskuu 2022

www.karelia.fi

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Marraskuu 2022
Tietojenkäsittelyn koulutus

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Roosa Kontinen

Nimeke
Fullstack-sovelluksen toteuttaminen Androidille

Toimeksiantaja
Joensuun Eläinsuojeluyhdistys Ry

Tiivistelmä

Tämän toiminnallisen opinnäytetyön tavoitteena oli toteuttaa Joensuun Eläinsuojeluyhdistys Ry:lle varastokirjanpitosovellus Androidille. Sovelluksella käyttäjä voi seurata varastojen tarvikemääriä ja lisätä, poistaa tai muokata tarvikkeita.

Ennen sovelluksen kehitystä vertailtiin erilaisia kehitystyökaluja ja -menetelmiä, joiden pohjalta luotiin tietoperusta. Tietoperustaa käytettiin valittaessa kehitystyökaluja ja -menetelmiä. Sovellus kehitettiin Android Studiolla, joka yhdistettiin MySQL-tietokantaan. Front-end-kehitys tapahtui Java-ohjelmointikielellä ja back-end-kielenä toimi PHP.

Lopputuloksena saatiin toimiva mobiilisovellus, joka sisältää kaikki toimeksiantajan vaatimusmäärittelyssä ilmi tulleet ominaisuudet. Sovellus on tarkoitettu vain toimeksiantajan yksityiseen käyttöön, eikä sitä julkaistu millään alustalla. Jatkossa sovellusta voi kehittää luomalla hakumahdollisuuden sekä erilaiset kategorisoinnit tarvikkeille. On myös mahdollista, että natiivi mobiilisovellus vaihdetaan myöhemmin verkkosivustoksi, jotta kaikki käyttöjärjestelmät voivat käyttää sovellusta.

Kieli
suomi

Sivuja 46
Liitteet 0
Liitesivumäärä 0

Asiasanat
Android, fullstack, MySQL, mobiilisovelluskehitys

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
November 2022
Degree Programme in Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author (s)
Roosa Kontinen

Title
Fullstack Application Implementation for Android

Commissioned by
Animal Welfare Association of Joensuu

Abstract

The purpose of this functional thesis was to implement a storage inventory application based on Android for Animal Welfare Association of Joensuu. With the application, the user can track the number of supplies in the storage and add, delete or edit supplies.

Before developing the application, different development tools and methods were compared to create a knowledge base. The knowledge base was used to select the development tools and methods. The application was developed in Android Studio, which was connected to the MySQL database. The front-end development was done in Java programming language and back-end language was PHP.

The result was a functional mobile application with all the features requested by the client in the requirements specification. The application was intended for the private use of the client only and was not published on any platform. In the future, the application can be further developed by creating a search option and different categories for the supplies. It is also possible that the native mobile application will be changed to a website at a later stage, so that all operating systems can access the application.

Language	Pages 46 Appendices 0 Pages of Appendices 0
Finnish	

Keywords
Android, fullstack, MySQL, mobile application development

Sisältö

Sanasto

API	Ohjelmointirajapinta
APK	Android-sovelluksen pakettitiedosto tai asennustiedosto
Attribuutti	Määrittelee tiedoston tai objektin ominaisuuden
Big endian	Tavujen sarja, jossa isoin arvo tallennetaan ensin
C#	Vahvasti tyypitetty ohjelmointikieli
DDL	Data Definition Language, käytetään tietokannan objektien rakenteen luomiseen ja muokkaamiseen
Entiteetti	Olio tai itsenäinen kokonaisuus
Hostauspalvelu	Jonkin yrityksen tarjoama ja vuokrattavissa oleva palvelintila
INT	Tietotyyppi, joka on kokonaisluku
Java	Oliosuuntautunut ohjelmointikieli
JSON	Avoimen standardin tiedostomuoto tiedonvälitykseen ja tallennukseen
Kotlin	Avoimen lähdekoodin staattisesti tyypitetty ohjelmointikieli
Little endian	Tavujen sarja, jossa pienin merkitsevin arvo tallennetaan ensin
Natiivi sovellus	Nimenomaan mobiililaitteelle rakennettu sovellus, joka ei toimi muilla laitteilla
PHP	Avoimen lähdekoodin yleiskäyttöinen skriptikieli
PRIMARY KEY	Pääavain, joka yksilöi tietokantataulukon rivin
Relaatioskeema	Tietokannan riippuvuuksien kaava

SQL	Structured Query Language, eli kyselykieli, jolla voi tehdä tietokantaan erilaisia hakuja, muutoksia tai lisäyksiä
SQL DDL	SQL:n tiedonkäsittelykieli
UNIQUE	Rajoitustyyppi, jolla varmistetaan tietokannan sarakkeiden arvot erilaisiksi
VARCHAR	Tietotyyppi, joka sisältää määrittelemättömän pituisen merkkijonon

1 Johdanto

Mobiililaitteet, kuten esimerkiksi älypuhelimet ja tabletit, kehittyvät koko ajan ja ihmiset käyttävät enemmän aikaa mobiililaitteidensa parissa kuin ennen. Älypuhelinten ja mobiililaitteiden käyttäjien määrän on ennustettu olevan yli 5 miljardia vuonna 2019. (Statista Research Department 2016.) Tästä syystä on päädytty siihen, että yhä useampi yritys haluaa olla aallonharjalla mobiilisovellusten markkinoita mietittäessä. Mobiililaitteet ja ylipäätään älypuhelimet ovat läsnä jokaisen arjessa, ja syystäkin, koska älypuhelimet tarjoavat ennen tavoittamattomissa olleet ihmisryhmät paremmin. Mobiililaitteet ovat kytkettyinä verkkoon, jolloin niitä voidaan käyttää erilaisiin toimintoihin, kuten arkipäivän viestittelystä verkkokauppaostoksiin. Markkinoita hallitsevat tällä hetkellä Applen iPhone ja Googlen omat Androidiin perustuvat mobiililaitteet. Käytetyimmät käyttöjärjestelmät ovat iOS ja Android. Googlen Android sekä Applen iOS esiteltiin ensi kertaa vuonna 2007, jonka jälkeen molemmat näistä ovat tarjonneet säännöllisiä päivityksiä käyttöjärjestelmiinsä. Maailman johtavana mobiilikäyttöjärjestelmänä Android hallitsee 70 prosentin osuudella, kun taas iOS:n osuus käyttöjärjestelmistä on noin 28 prosenttia. (Laricchia 2022.)

Opinnäytetyön tarkoituksena oli luoda Joensuun Eläinsuojeluyhdistykselle heidän vaatimuksiensa mukainen mobiilisovellus. Android-laitteille suunnatulla sovelluksella oli tarkoitus helpottaa toimeksiantajan varastojen tarvikkeiden seurantaa. Mobiilisovellusta kehitettäessä panostettiin sovelluksen helppokäyttöisyyteen sekä visuaaliseen ilmeeseen. Tämä mobiilisovellus toteutettiin fullstackina käyttäen Android Studiota ja Java- sekä PHP-ohjelmointikieltä ja MySQL-tietokantaa. Näistä syistä tässä raportissa pureudutaan mobiilisovelluskehitykseen ja fullstackiin.

2 Opinnäytetyön lähtökohdat

2.1 Toimeksiantaja

Toimeksiantajana tälle opinnäytetyölle on Joensuun eläinsuojeluyhdistys Ry, eli JEY lyhennettynä. JEY on voittoa tavoittelematon yhdistys ja se on perustettu vuonna 1899, mutta on rekisteröity yhdistykseksi vasta 1920. Joensuun eläinsuojeluyhdistys on yksi Suomen vanhempia paikallisyhdistyksiä. Yhdistyksen toiminnan tarkoituksena on vastustaa eläintein kaltoinkohtelua ja eläinrääkkäystä, pelastaa hädässä olevia eläimiä, saada eläinten elinolosuhteet niiden tarpeita vastaaviksi ja herättää ihmisten rakkaus eläimiin. Toimialueena JEY:llä on koko Pohjois-Karjala ja se käsittää myös kaikki toimialueeseen kuuluvat eläimet. (Joensuun eläinsuojeluyhdistys 2022.)

2.2 Tarkoitus ja tavoite

Joensuun eläinsuojeluyhdistys Ry, eli JEY:n toiminta perustuu vapaaehtoisuuteen. JEY:n haasteisiin kuuluu muun muassa vapaaehtoisten kaukana sijaitsevat tarvikevarastot ja tietämättömyys siitä, mitä tarvikkeita on vielä jäljellä. Tällä hetkellä toiminta tapahtuu soittojen tai viestien välityksellä toisille vapaaehtoisille. Mikäli varmuutta ei ole, joutuu vapaaehtoinen matkustamaan välillä pitkänkin matkan varastoille selvittämään, onko tarviketta vielä jäljellä ja välillä myös turhan takia, koska tarvikkeet ovat loppu. JEY:n ilmaisemasta haasteesta olen keksinyt idean toteuttaa Android-tietokantasovellus, jolla voidaan helpottaa ja parantaa tarvikkeiden seurantaa. Myös turhat ajokilometrit voidaan näin välttää.

Kävin läpi eläinsuojeluun liittyviä opinnäytetöitä ja havaitsin, että eläinsuojeluyhdistyksille ei ole minun hakujeni mukaan tehty ainuttakaan natiivia sovellusta. Yhden jollain tapaa relevantin eläinsuojeluyhdistys-aiheisen opinnäytetyön löysin. Sen on tehnyt Iiris Kostiainen ja opinnäytetyön nimi on Seinäjoen seudun eläinsuojeluyhdistyksen Jäsenhuone-sivuston uudistus. Koska opinnäytetyö koskee verkkosivu-uudistusta, ei sekään ole verrattavissa omaan toteutukseeni. Opinnäytetyöstä kävi ilmi, että kovin monella eläinsuojeluyhdistyksellä ei ole omaa sivustoa. (Kostiainen 2022.) Tarkistin

asian myös Sey.fi -sivustolta ja vain 11 jäsenyhdistyksellä 39:stä oli omat verkkosivut. SEY, eli Suomen eläinsuojelu, on liitto suomalaisille eläinsuojeluyhdistyksille (Suomen eläinsuojelu 2022.)

3 Mobiilisovelluskehitys

3.1 Yleistä mobiilisovelluskehityksestä

Kun puhutaan mobiilisovelluksen kehittämisestä, tarkoitetaan sillä ohjelmiston ja sovelluksien kehitystä kannettaviin mobiilisiin päätelaitteisiin, kuten tabletteihin, matkapuhelimiin ja älypuhelimiin. Älylaitteisiin tehtävä sovelluskehitys on kasvava ala, kun katsotaan tilastoja esimerkiksi voittojen ja luotujen työpaikkojen perusteella. Mobiilisuunnittelussa varsin olennaista on mobiilikäyttöliittymän suunnittelu ja esimerkiksi sen responsiivisuus sekä saavutettavuus erilaisilla näytöillä. (Itewiki 2022a.)

Mobiilisovelluskehitys on joukko erilaisia prosesseja ja menettelyjä, jotka liittyvät ohjelmistojen kirjoittamiseen älylaitteille, kuten älypuhelimille ja tableteille. Mobiilisovelluskehitys juontaa alkujaan juurensa perinteisemmästä ohjelmistokehityksestä. Ratkaisevin ero tässä kuitenkin on, että mobiiliin tarkoitetut sovellukset luodaan tietyn mobiililaitteen ainutlaatuisten ominaisuuksien hyödyntämiseen. Esimerkiksi jokin peli voidaan luoda hyödyntämään iPhonen omaa kiihtyvyysanturia tai terveyssovellus älykellon omaa lämpötila-anturia. Nykyään kaksi merkittävintä mobiilialustaa ovatkin Applen iOS ja Googlen Android. (David, Novotny & Denman 2021.)

Vaikka mobiilisovelluskehityksestä on paljon hyötyjä, on siinä olemassa myös haittapuolensa. Erilaiset tietoturvauhkat sekä kyberhyökkäykset yleistyvät samaa tahtia, kun erilaisia uusia teknologioita kehitetään. Tietoturvallisuuden vaarantumisia tapahtuu jatkuvasti, ja niistä media tiedottaakin aina sellaisen synnyttyä. Hyvänä esimerkkinä voidaan pitää Goatse-ryhmän vuonna 2010 tekemää hyökkäystä, jossa se sai yli 100 000 iPadia käyttävän asiakkaan tietoja

haltuunsa. Toinen esimerkkitapaus on, kun Sony koki massiivisen hyökkäyksen omiin verkkopalveluihinsa vuonna 2011. Hyökkäys aiheutti 77 miljoonan ihmisen tietojen vaarantumisen ja vuotamisen rikollisille. (Hiltunen & Hiltunen 2014.)

3.2 Android

Android on Googlen omistuksessa oleva mobiililaitteille ja älylaitteille suunniteltu ohjelmistopino, joka sisältää käyttöjärjestelmän, väliohjelmistoja ja käyttäjän omia perusohjelmia. Androidin käyttöjärjestelmä perustuu Linux-ytimeen, jonka päälle on rakennettu järjestelmäkirjastoja, ohjelmistokehys ja sovellusohjelmia. Androidin koodikielenä toimii Java ja Google on kehittänyt erilaisia Java-kirjastoja tukemaan kehitystä. (Itewiki 2022b.)

Androidin käyttöjärjestelmä perustuu avoimeen lähdekoodiin, vaikkakin Google koettaa kontrolloida käyttäjiään. Sillä, että käyttöjärjestelmä on avoin, on haittapuolensa. Esimerkiksi tietoturvariskit ovat yksi suuri ongelma, koska Android ei toimita päivityksiä keskitetysti. Androidin kehittämisestä vastaava Open Handset Alliance on 34 eri teknologia-alan yrityksen perustama verkosto, jonka tarkoituksena on edistää mobiiliteknologioiden kehittymistä ja niiden yleistymistä. Tähän verkostoon kuuluu Googlen lisäksi T-Mobile, HTC, Qualcomm ja Motorola. (Itewiki 2022b.)

Androidille kehitettyjen sovellusten kauppapaikkana toimii Google Play. Tätä kauppapaikkaa kutsutaan avoimeksi, sillä sovelluksien kehittäjät saavat varsin vapaasti päättää esimerkiksi sovellustensa julkaisusta, kohdentamisesta ja hinnoittelusta. Android on tällä hetkellä maailmanlaajuisesti suosituin käyttöjärjestelmä mobiililaitteille. (Itewiki 2022b.)

Android-mobiilisovellusten kehityksessä hankalaa voi olla käyttöjärjestelmään perustuvien laitteiden valtava määrä sekä niihin asennettujen käyttöjärjestelmäversioiden kirjo. Android-sovelluksia kehitettäessä suosittuja ohjelmointikieliä ovat C++ sekä Java ja Kotlin. Kehittäjien mukaan Kotlin olisi

helpompi ohjelmointikieli oppia kuin Java, mutta vuosia käytössä oleva Java on silti suositumpi kehittäjien keskuudessa. Googlen oma Android IDE, eli ohjelmointiympäristö Android Studio tukee kaikkia edellä mainittuja kieliä. (Black 2020.)

3.3 Android Studio

Android Studio julkistettiin ensimmäistä kertaa Google I/O -tapahtumassa toukokuussa 2013 ja sen ensimmäinen vakaa versio julkaistiin joulukuussa 2014. Android Studio on saatavilla Windows-, Mac- ja Linux-alustoille. (TechTarget Contributor 2018.) Android Studio on kehitysympäristö erilaisten Android-pohjaisten sovellusten kehitykseen. Android Studio pohjautuu IntelliJ IDEA -ohjelmistoon ja tästä on otettu vaikutteita esimerkiksi tehokkaaseen koodieditoriin ja erilaisiin kehittäjätyökaluihin. (Android Developers 2022a.)

Android Studio käyttää Android-käyttöjärjestelmän sovelluskehityksen tukemiseen Gradle-pohjoista rakennusjärjestelmää, erillistä emulaattoria, koodimalleja ja Githubin integraatiota. Kaikilla Android Studion projekteilla on yksi tai useampi rakenne, jossa on lähdekoodi ja resurssitiedostot. Tähän rakenteeseen kuuluu erilaisia sovelluksen, kirjastojen ja Google App Enginen osia. Android Studion oma koodieditori tarjoaa kehittäjälle apua koodin kirjoittamisessa, täydentämisessä ja analysoinnissa, tehden sovelluskehityksestä helpompaa. Android Studiossa rakennetut sovellukset käännetään APK-muotoon (Android Package), jonka jälkeen sen voi ladata puhelimille esimerkiksi Google Play Storen kautta. (TechTarget Contributor 2018.)

3.4 Xamarin

Mobiilisovelluskehityksestä puhuttaessa on mahdotonta jättää huomioitta Xamarin. Xamarinia käytetään luomaan mobiilisovelluksia sekä Androidille että IOSille. Xamarin on kehitetty Miguel de Icazan johtamien kehittäjien toimesta ja

ensimmäisen kerran käyttöönotettu vuonna 2001. Xamarin-yhtiö taasen perustettiin vasta vuonna 2011. (Altexsoft 2020.)

Xamarin perustuu NET Frameworkiin ja tästä syystä myös kehitysympäristö käyttää kielenään C#:ää sovellusten luomiseen kaikille mobiilialustoille. Xamarin on hyvä vaihtoehto suorituskykyä vaativien sovellusten rakentamiseen. Hyvä puoli Xamarinin käytössä on koodin jakaminen eri alustoille, mikä nopeuttaa esimerkiksi suunnittelua. Xamarin ei myöskään vaadi vaihtamista erilaisten kehitysympäristöjen välillä, vaan kaiken voi rakentaa Xamarin-sovelluksella. Huonoimpia puolia Xamarinissa on tuen tarjoaminen uusimmille iOS- ja Android-versioille, koska muutosten toteuttaminen vie aikaa sekä se, että vaikka Xamarin on ilmainen avoimeen lähdekoodiin perustuva alusta, yrityksille ja yrityksien tarpeisiin Xamarin voi koitua maksamaan paljon. (Altexsoft 2020.)

4 Fullstack

4.1 Yleistä fullstackista

Fullstack sisältää sekä asiakas- että palvelinpuolen ohjelmistokehitykset, erilaiset kirjastot ja työkalut, joiden avulla luodaan tehokas sovellus. Fullstackiin liittyy yleensä myös selainohjelmointia, vaikka sovellus olisi toteutettu mobiiliin. Tyypillisimpiä selainohjelmoinnin kieliä ovat muun muassa JavaScript, jQuery, Angular ja Vue. Palvelinohjelmoinnin kieliä ovat taas Python, NodeJS ja PHP. (Lewis 2019.)

Fullstack-mobiilikehityksessä tärkeintä on sovelluksen kokonaiskuva. Kokonaiskuvaan sisältyy ulkoasu ja teemat, palvelimet ja tietokannat sekä erilaiset kehitysvaiheen prosessit, kuten testaus ja virheenkorjaukset. Fullstack-sovelluskehittäjän tärkeimpiä taitoja on projektinsa johtaminen. Hänen tulee hallita sovelluksen kaikki osa-alueet ja valita oikeat teknologiat tukemaan sovelluskehitystä. (Ready 2019.)

Yleensä ohjelmistokehittäjä keskittyy vain yhteen kehitystyön osa-alueeseen, eli front- tai back-endiin. Front-end pitää sisällään kaikki näkyvät komponetit, kuten käyttöliittymä. Back-end sisältää taasen taustalla tapahtuvat toimenpiteet, kuten tietokannan käsittelyn tai infrastruktuurin. Fullstack-kehittäjä hallitsee nämä molemmat. (Lewis 2019.)

4.2 Front-end

Front-end-kehitys sisältää kaiken näkyvän sovelluksessa tai verkkosivustossa. Esimerkiksi graafinen käyttöliittymä ja sen vuorovaikutus käyttäjän kanssa on front-end-kehitystä. Front-endissä käytetään useita erilaisia kieliä luomaan käyttöliittymä ja ulkoasu, joista tunnetuimpia ovat HTML, CSS ja JavaScript. (Al Mamun 2022.)

Mobiilikehityksen front-end voidaan jakaa kahteen versioon, natiiviin ja hybridiin. Jos sovellus tarvitsisi esimerkiksi suurempaa suorituskykyä, voidaan ja halutaan käyttää mahdollisimman natiivia sovellusta. Hybridikehityksen hyödyt tulevat yleensä siitä, että siinä käytetään web-työkaluja ja -rakenteita. Mobiilikehittäjät voivat yleensä valita näiden kahden vaihtoehdon väliltä, kumpi ratkaisu olisi parempi. Hybridikehityksellä on muitakin hyötyjä, esimerkiksi mahdollisuus käyttää yhtä ja samaa juurijoukkoa (eng. root set) komponenttien ja muotoilun osalta usealla eri laitteella. Natiivisovelluksia luodessa tämä ei ole mahdollista. (Ready 2019.)

Front-endin tarkoituksena on kehittää ennen kaikkea helppokäyttöinen ja visuaalisesti kaunis alusta. Kehityksestä front-endillä on tullut haastavampaa, kun käyttäjät ovat vuorovaikutuksessa sovellusten ja sivustojen kanssa erilaisten laitteiden, kuten tablettien, älypuhelimien ja pöytäkoneiden kautta. Front-end-kehittäjien on varmistettava, että sovellus tai sivusto näyttäytyy oikein eri laitetyypeillä ja käyttöjärjestelmillä. (University Of Toronto 2022.)

4.3 Back-end

Back-end nimensä mukaisesti sisältää kaiken taustalla tapahtuvan tai näkymättömän osan sovelluksesta tai sivustosta. Esimerkiksi palvelinpuolen toiminnanohjausprosessit kuuluvat back-endiin. Back-end-kehittäjä keskittyy yleensä muun muassa palvelimien luomiseen ja konfiguroimiseen sekä tietokantojen ja sovellusten rajapintojen kanssa työskentelyyn.

Mobiilisovellusten kehittäminen palvelintasolla on suurelta osin samanlaista kuin työpöytä- tai web-palvelinsovellusten kehittäminen. Muutamia poikkeuksiakin on, muun muassa rajattu tapa, jolla uudet sovellukset voidaan asentaa mobiililaitteeseen, eli Google Play tai Apple Store, palvelinpuolen toiminnot, jotka on sidottu yhteen ja tiettyyn käyttöjärjestelmään ja erilaiset tavat, joilla voidaan päivittää laitesovelluksia. Samankaltaisuuksia ovat taasen tapa, jolla tietojen kanssa keskustellaan API-kutsujen avulla, sisällön päivittäminen versionhallintaa apuna käyttäen sekä skaalautuvuuden varmistaminen. (Ready 2019.)

Back-endin kanssa työskennellessä kehittäjät suorittavat erilaisia taustatapahtumia, joihin käytetään esimerkiksi seuraavia kieliä, kuten Java, PHP, Ruby, Python ja C. Eri kielivaihtoehtojen on intregroiduttava front-endin infrastruktuuriin ja kehittäjän onkin hyvä pitää tämä mielessä sovelluksen tai sivuston back-endiä suunnitellessa. Tietokantojen ja kielten tuntemus on pakollista, jotta voidaan toimia vuorovaikutuksessa erilaisten tietokantojen, esimerkiksi MySQL:n, Oraclen tai Microsoft SQL -palvelinten kanssa. (Al Mamun 2022.)

5 Tietokannat

5.1 Yleistä tietokannoista

Tietokanta (Database, tai DB) on jollain tapaa toisiinsa liitännäisten tietojen joukko, jota voidaan käsitellä tietokantakielellä, esimerkiksi SQL-kielellä.

Tietokannan tietoja hallinnoi ohjelmisto, tietokannanhallintajärjestelmä, lyhennettynä TKHJ (Database Management System, DBMS). Tunnetuimpia hallintajärjestelmiä ovat esimerkiksi Oracle, Microsoft SQL Server, MySQL ja Access. Tietokannat itsessään tarjoavat ohjelmoijille sekä käyttäjilleen monenlaisia mahdollisuuksia. Tietoja tallennetaan ja haetaan tietokannasta, tieto pysyy yhtenäisenä ja muuttumattomana, kun se säilytetään tietokannassa. Mikäli TKHJ:ta ei olisi, käyttäisimme luultavasti tiedostoja ja puhuttaessa monimutkaisemmista tietokokonaisuuksista ohjelmointi olisi paljon työläämpää, tietokannan eheys ja yhteneväisyys olisi heikkoa ja erilaisten tietojen hakeminen vaikeaa. (Hovi, Huotari & Lahdenmäki 2005, 4.)

Erilaisten tietokantojen suunnittelu on tärkeää monestakin eri syystä. Esimerkiksi tarve tallentaa tietoa tietokantoihin kasvaa koko ajan. Lisäksi halutaan, että tarvittavat tiedot löytyvät tietokannoista, kun niitä tarvitaan. Suunnittelussa täytyy ottaa huomioon tietokannan rakenne ja luoda se sellaiseksi, että sitä on helppo muuttaa tarvittaessa. Nykyään uudet tietojärjestelmät tehdään relaatiotietokannoiksi. (Hovi, Huotari & Lahdenmäki 2005, 2.) Yrityksien kannalta tiedot ovat varsin tärkeä resurssi, ja jotta ne voitaisiin säilyttää levyllä, se vaatii investointeja. Kaikki vähänkin tärkeät tietojärjestelmät käyttävätkin tietokantatekniikkaa tietojen tallentamiseen. Monesti yritykset ovat riippuvaisia omistamistaan tietokannoista. Tärkeänä seikkana on, että tiedot on tallennettu sellaiseen muotoon ja siten, että tallennetuista tiedoista saadaan nopeasti ja vaivattomasti yhdistelmiä erilaisiin tarpeisiin. (Hovi, Huotari & Lahdenmäki 2005, 4.)

5.2 ERDPlus

ERDPlus on yksi tehokkaimmista relaatiokaaviomallityökaluista, jota voidaan käyttää erilaisten entiteettisuhteiden, relaatioskeemojen, tähtikaavioiden ja SQL DDL -lausekkeiden luomiseen. ERDPlussan merkintätapa tukee säännöllisten ja heikkojen entiteettien, erityyppisten attribuuttien ja kaikkien mahdollisten suhteiden kardinaalisuusrajoitusten piirtämistä. (Tiwari 2022.)

ERDPlus mahdollistaa myös ER-kaavioiden muuntamisen relaatioskeemoiksi. Tämä nopeuttaa huomattavasti ER-kaavioon perustuvan relaatioskeeman luomista. Tällä sovelluksella voi myös luoda SQL-kyselyitä relaatioskeemoista ja tähtikaavioista. ERDPlus toimii useimpien nykyaikaisten tietokannanhallintatyökalujen kanssa, mukaan lukien Oracle, MySQL, PostgresSQL ja Microsoft Access. (ERDPlus 2021.)

Oracle, Microsoft SQL Server, MySQL ja Access ovat ehkä tunnetuimpia tietokantoja, mutta mobiilikehitykseen ne eivät välttämättä sovellu parhaiten. Suosituimmiksi mobiilikehitykseen liitetyt tietokannat ovatkin esimerkiksi MySQL, Firebase, SQLite, MariaDB ja MongoDB. Valitessa sovellukselle parasta vaihtoehtoa on mietittävä kokonaiskuvaa. Millainen rakenne tietokantaan tulisi, millaisia datamääriä on tarkoitus säilyttää tai mitä mobiilisovellusalustaa aiot käyttää. Nämä esimerkiksi on hyvä miettiä etukäteen, jotta saataisiin mahdollisimman mutkattomasti toimiva sovellus. (Samsukha, 2022.)

5.3 MySQL

MySQL AB:n suunnittelema MySQL siirtyi Sun Microsystemsin omistukseen vuonna 2008 ja siitä sitten Oraclen omistukseen, kun Oracle osti Sunin vuonna 2010. Ohjelmoijat ja kehittäjät voivat käyttää MySQL:ää GNU-lisenssillä (General Public License), mutta esimerkiksi yritysten on hankittava kaupallinen lisenssi Oraclelta. Monien johtavien sivustojen ja yritysten, kuten Facebookin, Twitterin ja Youtuben, taustalla pyörii MySQL ja sitä pidetään edelleenkin suosittuna relaatiotietokantojenhallintajärjestelmänä. MySQL on avoimeen lähdekoodiin perustuva relaatiotietokantojen hallintajärjestelmä. (RDBMS), jonka perusta on tehty SQL-kielelle (Structured Query Laguange). MySQL toimii lähes jokaisella alustalla, esimerkiksi Windowsissa, UNIXissa ja Linuxissa. Yleensä MySQL yhdistetään verkkosovelluksiin ja verkkojulkaisemiseen, mutta sitä voidaan käyttää myös monessa muussa. (Moore 2018.)

Asiakas-palvelin-malliin perustuva MySQL:n ydin on palvelin, joka käsittelee ja toteuttaa kaikki tietokannalle syötettävät ohjeet tai komennot. Palvelin on saatavana erillisenä ohjelmistonaan, jota käytetään palvelinverkkoympäristössä tai vaihtoehtoisesti kirjastona, joka voidaan linkittää erillisiin sovelluksiin. MySQL sisältää useita eri apuohjelmia, jotka auttavat MySQL-tietokantojen hallinnassa. Komentoja lähetetään MySQL-palvelimelle asiakasohjelman kautta, joka voidaan asentaa tietokoneelle. Alun perin MySQL on kehitetty käsittelemään suuria määriä tietokantoja ja niiden sisältöä nopeasti. Vaikkakin MySQL-palvelin asennetaan yleensä vain yhteen tietokoneeseen, se kykenee lähettämään tietokantojaan useisiin eri paikkoihin asiakasliittymien kautta. Nämä mainitut asiakasliittymät käsittelevät SQL-lauseet ja näyttävät sitten komentojen tulokset. (Moore 2018.)

5.4 Firebase

Firebase on reaaliaikainen pilvipalveluna toimiva tietokanta. Tiedot, joita halutaan lisätä tietokantaan, tallennetaan JSON-muotoisena (JavaScript Object Notation) ja synkronoidaan reaaliajassa jokaiseen yhdistettyyn asiakasliittymään. Jos halutaan rakentaa alustarajoja ylittäviä sovelluksia esimerkiksi Android- tai Apple-alustojen sekä JavaScriptin SDK:iden avulla, kaikki asiakasliittymät voivat jakaa saman synkronoidun tietokannan keskenään. Firebase luokitellaan NoSQL-tietokannaksi, ja sen toiminnot eroavat relaatiotietokannoista. Firebasen API (Application Programming Interface) on suunniteltu niin, että se sallii vain operaatiot, jotka voidaan suorittaa nopeasti. Näin voidaan taata reaaliaikainen kokemus ja synkronointi laitteiden välillä. (Firebase 2022.)

Firebasen avulla voidaan rakentaa lukemattomia erilaisia sovelluksia. Ainoat rajoitukset koskevat alustoja. Firebasen ensisijaisia käyttötarkoituksia ovat iOS- ja Android-alustat, mutta sitä voidaan käyttää myös verkkosivuilla sekä Flutterin ja Unityn kanssa. Firebasella on erilaisia ohjelmistokehityksen työkaluja, kuten FirebaseUI-kirjasto, joka tarjoaa hyödyllisiä apuohjelmia, jotka tekevät Firebasen kanssa työskentelyn vieläkin helpommaksi. (Stevenson 2018.)

5.5 SQLite

SQLite on sisäinen prosessikirjasto, joka toteuttaa itsenäisen palvelittoman, konfiguroimattoman ja transaktionaalisen SQL-tietokantajärjestelmän. SQLiten koodi on itsessään julkista omaisuutta, joten sitä voidaan käyttää vapaasti mihin tahansa tarkoitukseen, yksityiseen tai kaupalliseen. SQLite on maailman laajimmin käytetty tietokanta, jolla on muun muassa lukemattomia sovelluksia, jotka käyttävät sitä. SQLite on integroitu tietokantajärjestelmä, jossa ei ole erillistä palvelinprosessia taustalla. SQLite nimittäin kirjoittaa ja lukee suoraan tavallisista levytiedostoista. Tietokanta, jossa on useita tauluja, indeksejä ja näkymiä, sisältyy kaikki yhteen tiedostoon. Tiedostomuoto SQLitessä on alustarajat ylittävä, eli tietokantaa voidaan vapaasti kopioida 32-bittisten ja 64-bittisten järjestelmien välillä tai vaihtoehtoisesti big-endian- ja little-endian-arkkitehtuurien välillä. Näitten ominaisuuksiensa takia SQLite on suosittu valinta sovelluskehityksen tietokantajärjestelmiä mietittäessä. (SQLite 2022.)

SQLite on itsessään kirjasto, ja kun sen kaikki ominaisuudet ovat käytössä, kirjaston koko voi olla jopa alle 750 kilotavua kohdealustasta riippuen. Muistin käytön ja nopeuden välillä on kompromissi, joka tarkoittaa sitä, että SQLite toimii yleensä sitä nopeammin, mitä enemmän muistia sille on varattu. Suorituskyvyn sanotaan olevan myös kohtalaisen hyvä vähämuistisissa ympäristöissä. Riippuen käytöstä, SQLite voi olla nopeampi kuin suora tiedostojärjestelmän I/O (Input/Output). (SQLite 2022.)

SQLiteä pidetään luotettavana, koska ennen jokaista kehitysjulkaisuaan, se testataan huolellisesti. Lähdekoodista suurin osa on omistettu pelkästään testaukseen ja todentamisesta varten. SQLiten oma automaattinen testipaketti ajaa miljoonia testitapauksia, joihin voi sisältyä satoja miljoonia yksittäisiä SQL-lauseita ja tästä syystä voidaan saavuttaa 100 prosentin kattavuus. Koodipohjaa tukee kansainvälinen kehittäjätiimi, joka työskentelee SQLiten parissa. Kehittäjät laajentavat jatkuvasti uusia ja paranneltuja ominaisuuksia,

tinkimättä suorituskyvystä. SQLiten lähdekoodi on kaikille ilmainen, ja tästä syystä varsin suosittu. (SQLite 2022.)

6 Käyttöliittymä

6.1 Yleistä käyttöliittymistä

Käyttöliittymä (UI, User Interface) on vuorovaikutuksen kohteena ihmisen ja tietokoneen viestinnän välissä. Tähän voi liittyä näytöt, näppäimistöt, hiiri ja työpöydän ulkoasu. Se on myös tapa, jolla käyttäjä voi olla vuorovaikutuksessa sovelluksen tai verkkosivustojen kanssa. Käyttäjäkokemusta parantaessa monet yritykset ovatkin ottaneet käyttöliittymän kehityksen tärkeäksi prioriteetiksi. (Churchville 2022.)

Käyttöliittymästä puhutaan yleensä yhdessä käyttäjäkokemuksen (UX, User Experience) kanssa, johon kuuluu muun muassa laitteiden ulkonäkö, vasteaika ja käyttäjälle käyttöliittymän yhteydessä esitetty sisältö. Optimaalisemman käyttäjäkokemuksen luomiseen on kiinnitetty yhä suurempi huomio, mikä onkin saanut jotkut luomaan uraa UI- ja UX-asiantuntijoina tai -suunnittelijoina. (Churchville 2022.)

Mobiilisovellusten yleistyminen on vaikuttanut käyttöliittymiin ja sen myötä on syntynyt niin sanottu mobiilikäyttöliittymä. Mobiilikäyttöliittymässä on kyse käyttökelpoisten ja vuorovaikutteisten käyttöliittymien luomisesta älypuhelinten ja tablettien pienemmille näytöille ja eritysominaisuuksien, kuten esimerkiksi kosketusohjauksen parantamisesta. (Churchville 2022.)

6.2 Figma

Figma on digitaaliseen suunnitteluun ja prototyypitykseen tehty työkalu. Se on UI- ja UX-suunnittelusovellus, jolla voidaan luoda verkkosivustoja, sovelluksia

tai käyttöliittymäkomponentteja, jotka voidaan helposti integroida muihin projekteihin. Figma on vektoripohjainen työkalu, joka toimii pilvipalveluna, joten käyttäjien on helppo työskennellä paikkariippumattomasti selaimen kanssa. Vertailukelpoisina työkaluina voi pitää esimerkiksi Sketch-, Adobe XD-, Invision- tai Framer-tuotteita. Monien muiden UI-työkalujen tavoin Figmaa tukee vankka suunnittelijoiden ja kehittäjien yhteisö, joka jakaa liitännäisiä toiminnallisuuden lisäämiseksi ja työnkulun nopeuttamiseksi. Kuka tahansa voi osallistua ja jakaa omia liitännäisiään. Figmaa käyttävät suuret yritykset, kuten Slack, Twitter, Zoom, Dropbox ja Walgreens. Pelkästään näistä nimistä voi jo päätellä, että Figma on hyvä suunnittelutyökalu lähes minkälaiseen projektiin vain. Figma on myös saatavilla ilmaiseksi, ja sitä kannattaakin kokeilla, jos työskentelee digi-alalla. (Cousins 2019.)

7 Suunnitteluvaihe

7.1 Vaatimusmäärittely

Vaatimusmäärittely on yleensä jonkinlainen dokumentaatio siitä, mitä ohjelmistolta vaaditaan. Siihen kuuluvat määritelmät siitä, mitä ohjelmistolla olisi tarkoitus tehdä, millaisiin ongelmiin se kartoittaa ratkaisun tai millaisia ominaisuuksia kyseiseltä ohjelmistolta halutaan. (Matilainen 2022.)

Vaatimusmäärittely tehdään ennen ohjelmiston hankintapäätöstä ja koko ohjelmistoprojekti pohjautuu siihen. Vaatimusmäärittelyyn luodun dokumentaation pohjalta palveluntarjoajalla pitäisi olla selkeä kuva siitä, mitä ohjelmistolta halutaan ja millainen on ohjelmistoprojektin lopputulos. (Matilainen 2022.)

Suunnittelu lähti liikkeelle toimeksiantajan kanssa käydyistä keskusteluista. Keskusteluista kävi ilmi, että Joensuun eläinsuojeluyhdistys ei pysty mitenkään reaaliajassa seuraamaan, mitä eläimille tarkoitettuja tarvikkeita varastoilta löytyy. Tästä lähti idea toteuttaa yhdistykselle mobiilisovellusta, joka helpottaisi

tarvikkeiden seurantaa. Tämä sovellus tulisi myös vähentämään turhia bensakustannuksia, kun varastolla ei tarvitsisi käydä toteamassa, että siellä ei juuri tarvitsemaa tarviketta löydykään.

Aihe oli selvillä, joten aloin suunnittelemaan sovelluksen vaatimuksia tarkemmin ensin itsenäisesti. Tämän jälkeen kysyin toimeksiantajan mielipiteitä, millaisia vaatimuksia he tarvitsisivat sovellukselleen. Tein seuraavanlaiset vaatimusmäärittelyt toimeksiantajan kanssa käydyn keskustelun perusteella:

- Käyttäjän on pystyttävä rekisteröitymään.
- Käyttäjän on pystyttävä kirjautumaan sovellukseen.
- Mikäli käyttäjällä on tarvittava attribuutti, hänen pitää pystyä muokkaamaan, poistamaan tai lisäämään tarvikkeita.
- Peruskäyttäjä voi vain selata tarviketietokantaa.
- Mobiilisovelluksen on tuettava Android-käyttöjärjestelmää.
- Ulkoasun pitää olla helppokäyttöinen.

Tavoitteena oli luoda Androidille mobiilisovellus, joka auttaa Joensuun eläinsuojeluyhdistyksen jäseniä saamaan ajankohtaisen tiedon tarvikkeista varastoilla ja vähentää näin ollen matkoihin kuluvia bensakustannuksia.

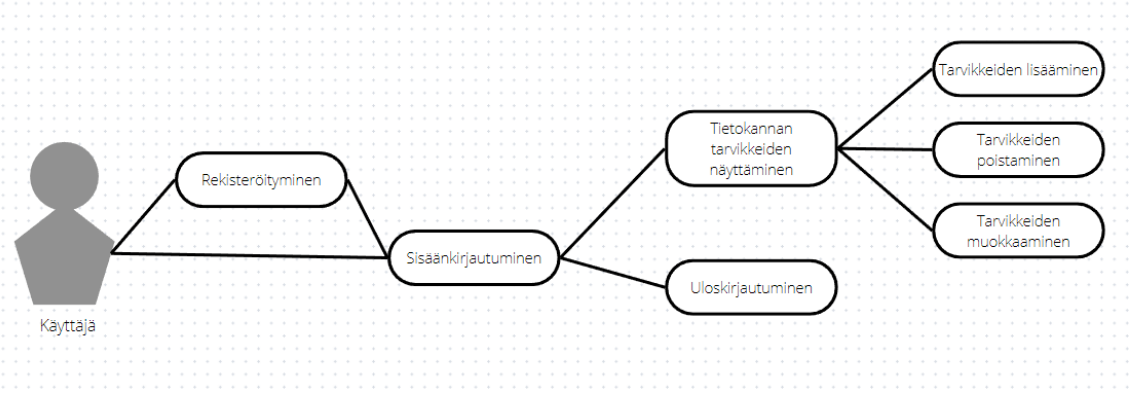
7.2 Käyttötapaukset

Käyttötapaus on menetelmä, jota käytetään järjestelmävaatimusten tunnistamiseen, selventämiseen ja järjestämiseen. Käyttötapaus koostuu joukosta mahdollisia järjestelmien ja käyttäjien välisiä vuorovaikutussarjoja jossakin tietyssä tilanteessa ja tiettyyn tavoitteeseen liittyen. Menetelmällä on tarkoitus luoda dokumentaatio kaikista vaiheista, joita käyttäjä tarvitsee suorittaakseen jonkin toiminnon. (Brush 2020.)

Käyttötapauksia käytetään useissa ohjelmistokehityksen vaiheissa, kuten järjestelmän vaatimusten suunnittelussa, suunnittelun varmentamisessa, testaamisessa ja erilaisten verkkohakemistojen ja käyttöohjeiden hahmotelman luomisessa. Käyttötapauksista syntyvä dokumentaatio auttaa kehitystiimiä

tunnistamaan ja ymmärtämään, missä tapahtuman osissa voi esiintyä virheitä. (Brush 2020.)

Aloin miettiä sovellukselle käyttötapauskaaviota. Mobiilikäyttöliittymän toimintojen tulisi mahdollistaa rekisteröityminen ja sisäänkirjautuminen. Sisään kirjautunut käyttäjä voi hallita tarviketietokannan tietoja muokaten, poistaen tai lisäten, mikäli käyttäjälle on annettu tarvittava attribuutti sitä varten. (kuva 1).



Kuva 1. Käyttötapaukset.

7.3 Käytettävät teknologiat

Alun alkaen suunnittelin käyttäväni Xamarinia mobiilisovelluksen kehitykseen, mutta kävi ilmi, että toimeksiantaja haluaa nimenomaan Androidille tuotetun sovelluksen. Päädyin valitsemaan kehitysympäristöksi Android Studion, joka toimisi tässä tapauksessa front-endinä. Android Studiosta minulla itselläni oli jo aikaisempaa kokemusta parin oman projektin osalta sekä Android Studiosta löytyi myös paljon tietoa ja oppaita.

Pohdin käytettävää tietokantaa jonkin aikaa ja mietin, voisinko hyödyntää minulla jo olemassa olevaa hostauspalvelua ja MySQL-tietokantaa. Olin yhteydessä toimeksiantajaan ja mietimme yhdessä, haluaako se oman hostauspalvelun vai voisimmeko käyttää minulta löytyvää. Tulimme siihen tulokseen, että käytämme jo olemassa olevaa MySQL-tietokantaa. Tämän koimme järkevimmäksi, koska se ei tule kustantamaan ensimmäiseen vuoteen

toimeksiantajalle mitään ja tulen kuitenkin itse toimimaan mobiilisovelluksen ylläpitäjänä myös tulevaisuudessa.

MySQL-tietokannan kanssa olin aikaisemmin käyttänyt PHP:tä, joten se vaikutti suuresti back-end-kielen valintaan. Toteutusvaiheelle oli kuitenkin suunniteltu käytettäväksi kuukausi, joten en tiennyt riittäisikö aika opetella uutta kieltä. Lopulta PHP tuntui siis luotettavalta valinnalta aikataulun pitämisen suhteen.

Projektille tarvittiin myös versionhallinta, joten valitsin käytettäväksi Gitin ja Githubin, koska olin käyttänyt niitä jo aikaisemmissa projekteissani. Asiaan vaikutti myös se, että Android Studio tukee suoraa integraatiota Gitin sekä Githubin kanssa.

8 Toteutusvaihe

8.1 Käyttöliittymän näkymät

Toteutus alkoi käyttöliittymän suunnittelulla Figma-työkalua käyttäen. Vaatimusmäärittelyn kautta sai jo hyvän käsityksen siitä, millaisia näkymiä sovelluksella pitäisi olla. Myös käyttötapauskaaviota katsomalla sai pääteltyä, mitä kaikkea sovellukseen tarvittiin.

Sovellukselle tehtiin kirjautumisnäkymä (kuva 2a), joka olisi päänäkymä koko sovelluksessa, kun se avataan. Kirjautumisnäkymässä käyttäjän pitäisi kirjautua sisään tai vaihtoehtoisesti päästä rekisteröitymissivuille, jos tunnuksia ei vielä olisi. Rekisteröitymisnäkymä (kuva 2b) muistuttaa paljon kirjautumisnäkymää, mutta se eroaa syötekenttiensä sekä yläotsikkonsa puolesta tunnistettavasti. Rekisteröintinäkymässä käyttäjän on tarkoitus rekisteröityä ja näin ollen luoda tunnukset, joita voidaan käyttää kirjautumisnäkymässä.

Etusivunäkymän on tarkoitus avautua, kun käyttäjä on kirjautunut onnistuneesti sisään. Etusivunäkymiä (kuva 2c ja 2d) on kaksi riippuen siitä, onko käyttäjällä

tarvittava attribuutti tarvikkeiden lisäämiseen, muokkaamiseen tai poistamiseen. Tarvikkeiden lisäämiselle tein oman näkymän (kuva 2e). Lisäämisnäkymässä on tarkoitus syöttää tarvikkeen nimi ja määrä ja nappia painamalla siirtää ne tietokantaan.

Kuva 2. Näkymät vasemmalta oikealle a) kirjautumisnäkymä, b) rekisteröitymisnäkymä, c) etusivunäkymä muokkausoikeudella, d) etusivunäkymä ilman muokkausoikeutta, e) lisää tarvike -näkymä.

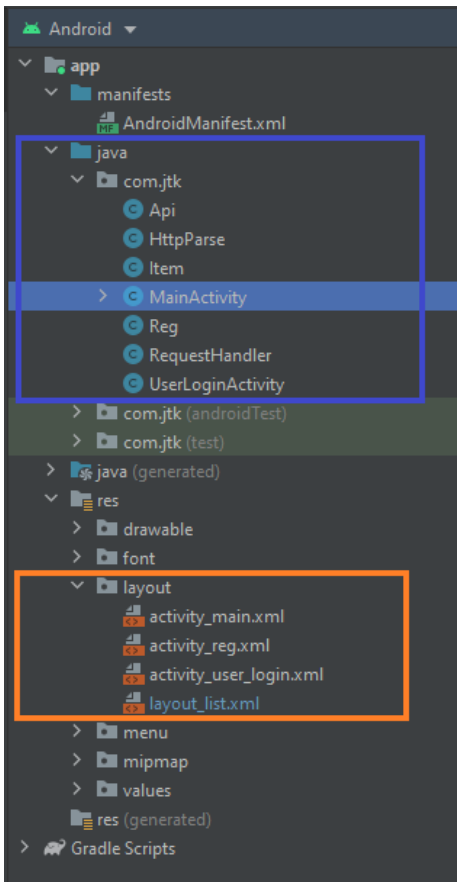
8.2 Front-end Android Studiolla

XML on lyhenne sanoista Extensible Markup Language. XML on HTML:n kaltainen merkintäkieli, jota käytetään tietojen kuvaamiseen. XML on johdettu alkujaan Standard Generalized Markup Language -kielestä. XML-tunnisteet määrittelevät tiedot ja niitä käytetään erilaisten tietojen tallentamiseen ja järjestämiseen. (Geeksforgeeks 2022.)

Android Studiossa XML:ää käytetään käyttöliittymään liittyvien osien tietojen toteuttamiseen. Android Studion käyttöliittymä on rakennettu tärkeimpien asettelujen, eli widgettien hierarkiana. Asettelut ovat ViewGroup-objekteja tai niin kutsuttuja säiliöitä, jotka ohjaavat sitä, miten alanäkymä asetetaan näytölle. Widgetit ovat näkymäobjekteja eli esimerkiksi painikkeita ja tekstilaatikoita. (Geeksforgeeks 2022.)

Toteutuksen niin kutsuttu front-end aloitettiin luomalla XML-tiedostoja (kuva 7) Android Studioon. Jokaiselle omalle näkymälleen piti luoda XML-tiedosto, joka

sisältää tiedot näkymien asetteluista. XML-tiedostot tarvitsevat myös luokkansa, missä toiminnallisuus määritellään koodin kautta. Luokat löytyvät hierarkiassa java-kansion (kuva 7) alta.



Kuva 7. XML-tiedostot ja luokat hierarkiapuussa.

Ensimmäisenä tehtiin rekisteröitymis- ja kirjautumisnäkymät. Näissä haluttiin panostaa yksinkertaiseen ulkoasuun, mutta kuitenkin hienompaan kuin Figmalla suunnitellut raakaversiot. Kirjautumis- ja rekisteröitymisnäkymä ovat tyyliltään ja kentiltään miltei identtiset, joten niiden XML-koodikin (kuva 8) näytti todella samalta.

```
<TextView
    android:id="@+id/textview1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:fontFamily="@font/staatliches"
    android:gravity="center"
    android:letterSpacing="0.05"
    android:padding="50dp"
    android:text="Kirjaudu"
    android:textColor="@color/white"
    android:textSize="30dp"
    android:textStyle="bold">
</TextView>
<EditText
    android:id="@+id/username"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="20dp"
    android:background="#a73737"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:fontFamily="@font/roboto_medium"
    android:hint="Käyttäjätunnus"
    android:minHeight="55dp"
    android:padding="10dp"
    android:textColor="#FFFFFF"
    android:textColorHint="#FFFFFF" />
<EditText
    android:id="@+id/password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginBottom="20dp"
    android:focusable="true"
    android:focusableInTouchMode="true"
    android:fontFamily="@font/roboto_medium"
    android:hint="Salasana"
    android:inputType="textPassword"
    android:minHeight="55dp"
    android:padding="10dp"
    android:background="#a73737"
    android:textColor="#FFFFFF"
    android:textColorHint="#FFFFFF" />
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/Login"
    android:text="KIRJAUDU SISÄÄN"
    android:textStyle="bold"
    android:textSize="20dp"
    android:minHeight="55dp"
    android:backgroundTint="@color/button"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginBottom="20dp"/>
<TextView
    android:id="@+id/noAccount"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:text="Ei tunnuksia? Rekisteröidy."
    android:textColor="@color/white"
    android:gravity="center"
    android:textSize="18dp"
    android:onClick="goToReg"
    android:clickable="true"/>
```

Kuva 8. Kirjautumisnäkymän XML-koodi.

Ainoa selkeä ero oli rekisteröitymisnäkymän tietosuojaseloste-kohdassa, johon tehtiin yksinkertainen AlertDialog (kuva 9) kasaamaan tietosuojaseloste. AlertDialog täytyi sisällyttää Android Studion luokkakoodiin onCreate()-funktion sisälle. Android Studion generoimaan luokkakoodiin sisältyy aina onCreate()-funktio ja tätä funktiota kutsutaan, kun näkymä luodaan (Geeksforgeeks 2021.)

```
public void Privacy(View view) {
        AlertDialog.Builder builder = new AlertDialog.Builder(Reg.this);

        builder.setTitle("Tietosuojaseloste")
                .setMessage("Esimerkki teksti")

        .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {

                    }
        })
        .setIcon(android.R.drawable.ic_menu_help)
        .show();
}
```

Kuva 9. AlertDialogin luonti tietosuojaselosteelle.

Seuraavana luotiin päänäkymä, eli sisään kirjautuneen käyttäjän näkymä. Näkymässä on listattuna tietokannan tarvikkeet, joten siihen käyttettiin ListView-objektia (kuva 10), jonka tarkoitus on näyttää luettelo erilaisista asioista, kuten tässä tapauksessa yksittäisistä tarvikkeista (Android Developers 2022b.).

```
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/linearLayout2"
    android:divider="@color/black"
    android:layout_above="@id/bottomnav"
    android:dividerHeight="2dp"/>
```

Kuva 10. Listview-objekti.

Muutoksia tuli myös alkuperäiseen suunnitelmaan tarvikkeiden lisäyksestä ja se sisällytettiin samalle sivulle listauksen kanssa. Tämä siksi, että käyttäjän olisi helppo käyttää sovellusta, eikä hänen tarvitsisi hyppiä sivulta toiselle, vaan kaikki tarvittava tekeminen olisi yhdellä sivulla.

Toivomuksena oli myös yksinkertainen navigaatio sovellukseen ja tähän käytettiin BottomNavigationViewiä (kuva 11), jotta alanavigaatio saatiin luotua. BottomNavigationView on Googlen materiaalikirjastossa sijaitseva valmis vaihtoehto navigaation luomiseen.

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottomnav"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:background="?android:attr/windowBackground"
    app:itemIconTint="@drawable/selector"
    app:itemTextColor="@drawable/selector"
    app:menu="@menu/nav"
/>
```

Kuva 11. BottomNavigationView-objekti.

Alanavigaation tarkoituksena oli antaa käyttäjälle mahdollisuus kirjautua ulos sovelluksesta. Tästä syystä täytyi myös luoda toiminnallisuus (kuva 12) päänäkymän omaan luokkakoodiin onCreate()-funktion sisään.

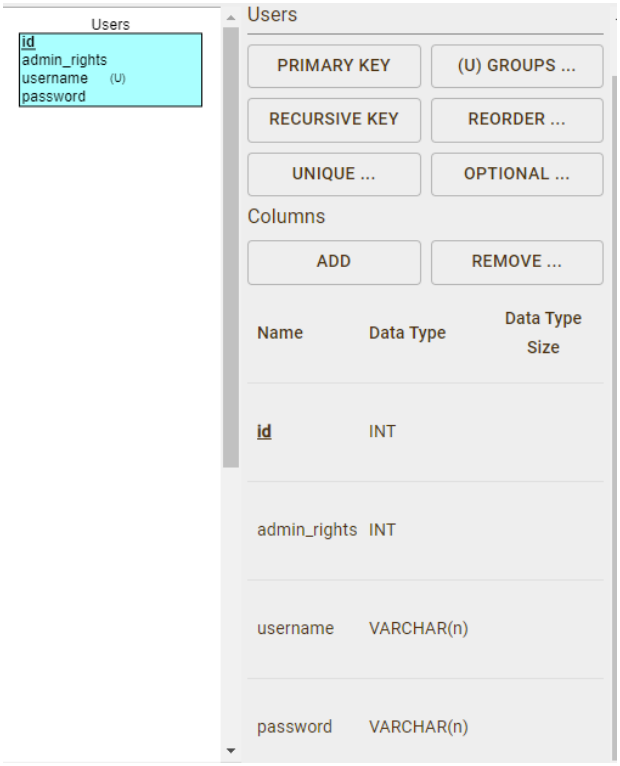
```
@Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      //Initialize And Assign Bottom navigation
      BottomNavigationView bottomNavigationView = findViewById(R.id.bottomnav);

      //Perform ItemSelectedListener
      bottomNavigationView.setOnItemSelectedListener(new NavigationBarView.OnItemSelectedListener() {

          @Override
          public boolean onNavigationItemSelected(@NonNull MenuItem item) {
              switch (item.getItemId()) {

                  case R.id.exit:

                      AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

                      builder.setTitle("Kirjaudu ulos")
                              .setMessage("Haluatko kirjautua ulos?")
                              .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
                                  public void onClick(DialogInterface dialog, int which) {
                                      finish();
                                      Intent intent = new Intent(MainActivity.this, UserLoginActivity.class);
                                      startActivity(intent);
                                      Toast.makeText(MainActivity.this, "Kirjauduttu ulos", Toast.LENGTH_LONG).show();
                                  }
                              })
                              .setNegativeButton(android.R.string.no, new DialogInterface.OnClickListener() {
                                  public void onClick(DialogInterface dialog, int which) {
                                  }
                              })
                              .setIcon(android.R.drawable.ic_dialog_alert)
                              .show();
              }
              return false;
          }
      });
  }
```

Kuva 12. BottomNavigationViewin toiminnallisuus.

8.3 Tietokannan rakenne

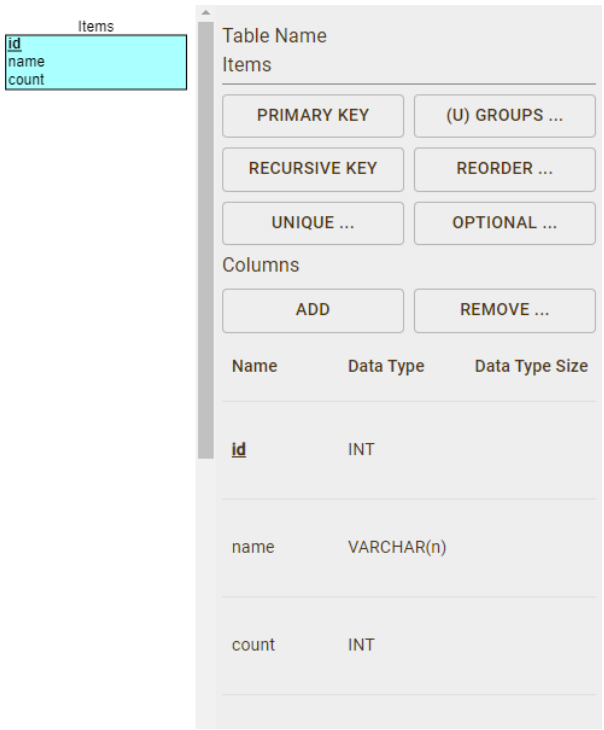
Tietokannan toteutus alkoi luomalla Erdplus-sovelluksella relaatiokaavio tietokannassa tarvittavista tauluista. Taulut luotiin ensin Erdplus-sovelluksella, että olisi helpompi hahmottaa, mitä kaikkea tarvitsee ottaa huomioon ennen oikean tietokannan luontia.

Käyttäjien rekisteröintiin ja kirjautumiseen tarkoitettuun Users-tauluun (kuva 13) tarvittiin INT-tyyppinen admin_rights sekä id, joka on myös PRIMARY KEY. Koska id attribuutti on PRIMARY KEY, täytyy sen olla myös INT-tyyppinen, eli kokonaisluku. PRIMARY KEY mahdollistaisi sen, että id:n avulla voisi hakea käyttäjää esimerkiksi kirjautuessa. Admin_rights valittiin INT-tyyppiseksi, koska sen arvo määritellään joko 0:ksi tai 1:ksi. Tämä tarkoittaa käytännössä sitä, että mikäli käyttäjällä on admin_rights -attribuutti, jonka arvo on 0, ei hän voisi kirjautua sisään ja mikäli arvo olisi 1, käyttäjä voisi kirjautua sisään. VARCHAR-tyyppiset username ja password, joista username attribuutti on UNIQUE, koska en halunnut samoja käyttäjänimiä rekisteröityvän useampaa.



Kuva 13. Users-taulu.

Tarvikkeiden listausta varten tarvittiin Items-taulu (kuva 14). Tähän tauluun luotiin INT-tyyppiset count ja id, joka toimi PRIMARY KEY:nä. Count-attribuutti on INT-tyyppinen, koska se kuvaa määrää, joka voidaan esittää vain kokonaislukuna. Jokaiselle tarvikkeelle tarvitaan myös nimi, joten tästä syystä tehtiin VARCHAR-tyyppinen name.



Kuva 14. Items-taulu.

Taulujen luomisen jälkeen ne oli helppo ladata SQL-muotoon ja siirtää suoraan MySQL-tietokantaan. Siirrätteässä SQL-muotoista tiedostoa tietokantaan, ei tarvinnut huolehtia tulevatko attribuutit samalla tavalla, vaan ne olivat juuri siinä muodossa kuin ne oli ERDPlussalla määritelty. Näin tietokannan luominen oli helppoa ja vaivatonta.

8.4 Back-end toteutus

API on lyhenne sanoista Application Programming Interface, joka tarkoittaa sovellusohjelmointirajapintaa. Rajapinta on ohjelmiston välittäjä, jonka avulla kaksi eri sovellusta voi kommunikoida keskenään. Esimerkiksi joka kerta, kun käytät sääsovellusta tarkistaaksesi säätiedot, käytät API:a. (Salesforce 2022.)

Sovelluksen back-end aloitettiin selaamalla oppaita, kuinka yhdistetään MySQL-tietokanta ja jo luotu front-end yhteen. Kävi ilmi, että tämä onnistuisi tehdä API:n kautta. Rekisteröitymiselle ja kirjautumiselle oli luotu oma taulunsa MySQL-tietokantaan, joten rajapinnan avulla sovellus ja tietokanta saatiin kommunikoimaan keskenään. Rekisteröityessä rajapinta välittää

rekisteröityneen käyttäjän syöttämät tiedot POST-pyyntönä palvelimelle. Tätä varten tarvittiin luoda RequestHandler-luokka Javalla tehtynä (kuva 15). RequestHandler-luokan tarkoitus on lähettää pyyntöjä rajapinnalle ja ottaa pyyntöjä vastaan rajapinnalta.

```
public class RequestHandler {

    //Method to send httpPostRequest
    //This method is taking two arguments
    //First argument is the URL of the script to which we will send the request
    //Other is an HashMap with name value pairs containing the data to be send with the request

    public String sendPostRequest(String requestURL,
                                  HashMap<String, String> postDataParams) {
        //Creating a URL
        URL url;

        //StringBuilder object to store the message retrieved from the server
        StringBuilder sb = new StringBuilder();
        try {
            //Initializing Url
            url = new URL(requestURL);

            //Creating an httplurl connection
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();

            //Configuring connection properties
            conn.setReadTimeout(15000);
            conn.setConnectTimeout(15000);
            conn.setRequestMethod("POST");
            conn.setDoInput(true);
            conn.setDoOutput(true);

            //Creating an output stream
            OutputStream os = conn.getOutputStream();

            //Writing parameters to the request
            //We are using a method getPostDataString which is defined below
            BufferedWriter writer = new BufferedWriter(
                    new OutputStreamWriter(os, "UTF-8"));
            writer.write(getPostDataString(postDataParams));
            writer.flush();
            writer.close();
            os.close();
            int responseCode = conn.getResponseCode();

            if (responseCode == HttpsURLConnection.HTTP_OK) {

                BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                sb = new StringBuilder();
                String response;
                //Reading server response
                while ((response = br.readLine()) != null) {
                    sb.append(response);
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
        return sb.toString();
    }
```

Kuva 15. RequestHandler-luokan POST-metodi.

Salasanoille luotiin erillinen enkryptaus-metodi, joka varmistaisi sen, etteivät salasanat olisi selkokielisenä missään vaiheessa. Tähän käytettiin MD5-salausprotokollaa. Koska sovellus ei ole julkisessa jaossa, vaan saatavissa vain JEY:n omistuksessa koettiin, että MD5-salausprotokolla tarpeeksi turvalliseksi

suojelemaan tietokantaan vietäviä salasanoja. Sovellus ei tule sisältämään mitään niin kutsuttua arkaluontoista sisältöä, mistä kukaan ulkopuolinen voisi jotenkin hyötyä. Myöskään ei ole pelkoa siitä, että tietokannasta tai käyttäjätunnuksesta löytyisi henkilötietoja, mistä käyttäjän voisi tunnistaa tai siitä voisi koitua käyttäjälle itselleen jotain vahinkoa.

Koska tiettyyn URL-osoitteeseen lähetetty POST- ja GET-pyynnöt pitäisi saada myös otettua rajapinnassa kiinni, tehtiin niille PHP:llä omat rajapintakäskyt. Esimerkiksi jos tahdottaisiin paikallisesti luoda käyttäjä, kutsuttaisiin osoitetta https://localhost/index.php/api=createuser, jolloin tähän osoitteeseen lähetetyt käyttäjätunnus ja salattu salasana voitaisiin välittää tietokantaan PHP-kielellä tehdyllä API:lla (kuva 16). API palauttaa JSON-muotoisen vastauksen Android Studiolla tehdylle RequestHandlerille, joka käsittelee tiedon. Tähän API-tiedostoon luotiin myös muut tarvittavat pyynnöt esimerkiksi käyttäjän hakemiselle, tarvikkeiden listaukselle sekä päivittämiselle ja poistamiselle.

Tuotantopalvelimella, eli verkkosivustojen ja web-sovellusten isännöintipalvelulla, URL-osoite muuttuu, kun ei olla enää paikallisessa verkossa, eli localhostissa. Tuotantopalvelimelle täytyy viedä tarvittavat tiedostot, jotta ne muodostavat oikean URL-osoitteen. Tässä tapauksessa tuotantopalvelimelle täytyisi viedä index.php-tiedosto. URL-osoite, eli verkko-osoite voisi olla tässä tapauksessa esimerkiksi https://www.esimerkki.fi/index.php/api=createuser (One 2022a).

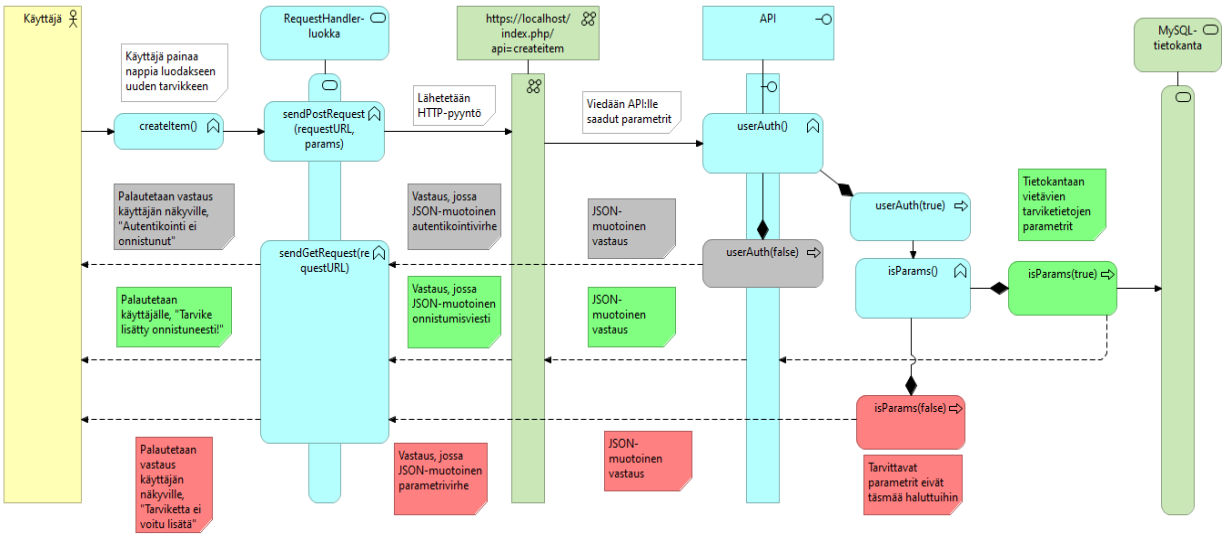
```php
<?php

    //getting the dboperation class
    require_once '../includes/example.php';

    //function validating all the paramters are available
    //we will pass the required parameters to this function
function isTheseParametersAvailable($params){
    //assuming all parameters are available
    $available = true;
    $missingparams = "";

    foreach($params as $param){
        if(!isset($_POST[$param]) || strlen($_POST[$param])<=0){
        $available = false;
        $missingparams = $missingparams . ", " . $param;
        }
    }

    //if parameters are missing
    if(!$available){
        $response = array();
        $response['error'] = true;
        $response['message'] = 'Parameters ' . substr($missingparams, 1, strlen($missingparams)) . ' missing';

        //displaying error
        echo json_encode($response);

        //stopping further execution
        die();
    }
}
    //an array to display response
    $response = array();

    //if it is an api call
    //that means a get parameter named api call is set in the URL
    //and with this parameter we are concluding that it is an api call
    if(isset($_GET['api'])){

        switch($_GET['api']){

            //the CREATE operation
            case 'createuser':
            //first check the parameters required for this request are available or not
            isTheseParametersAvailable(array('name','password'));

            $db = new example();

            //creating a new record in the database
            $result = $db->createUSer(
            $_POST['name'],
            $_POST['password']
            );
        }
    }

if($result){
    $response['error'] = false;
    $response['message'] = 'Käyttäjätunnus luotu!';
}
else{
    $response['error'] = true;
    $response['message'] = 'Jokin meni vikaan!';
}
echo json_encode($response);
```

Kuva 16. PHP API-koodi.

Sovellukselle määritettiin luokkakohtaiset koodit Android Studiossa rekisteröitymiseen, kirjautumiseen ja tarvikkeiden käsittelyyn. Rekisteröitymisen ja kirjautumisen luokkakoodi tallentaa sovelluksen kenttiin kirjoitetun käyttäjätunnuksen ja salasanan ja välittää sitten tiedot RequestHandler-oliolle. Myös tarvikkeiden poisto ja muokkaus tapahtuu samoin. Tarvikkeiden listauksessa huomioitavaa oli myös se, että täytyi ottaa vastauksena tullut JSON-muotoinen tietojoukko ja purkaa se yksittäisiksi tarvikkeiksi.

Sovelluksen keskeinen toiminta on havainnollistettu UML-sekvenssikaavion (kuva 17) avulla back-end-toteutuksesta. Kuvassa on kerrottu kuinka tieto ja

parametrit kulkevat käyttäjän yrittäessä luoda tarviketta. Kuten kaaviosta käy ilmi, käyttäjä on syöttänyt tarvikkeen tiedot, eli nimen ja määrän. Tämän jälkeen käyttäjä painaa nappia, jolla lisätään tarvike. Seuraavaksi kutsutaan jo mainittua RequestHandler-luokan oliota ja välitetään tieto siitä, että tämä on POST-tyyppinen pyyntö. RequestHandler-luokka ottaa parametrit sekä tarvittavan URL-osoitteen ja lähettää sen http-pyyntönä API:lle. API vastaanottaa parametrit ja luo tarkistuksen, onko käyttäjä kirjautunut userAuth()-funktion avulla. API-kerrokseen toteutettu kirjautumisen tarkastus mahdollistaa sen, että voidaan todentaa käyttäjän olevan kirjautunut silloin, kun pyyntö lisätä tarvike on jo lähetetty. Tämä poissulkee mahdollisia väärinkäytöksiä, esimerkiksi SQL-injektion avulla. SQL-injektio on tapa, jolla hakkeri voi päästä näkemään arkaluontoisia tietoja (One 2022b.) Mikäli käyttäjä on kirjautunut, siirrytään tarkastamaan onko tarvittavat parametrit kelvollisia isParams()-funktiolla. Jos parametrit ovat kunnossa, voidaan tarvike lisätä tietokantaan. Kun tarvike on onnistuneesti lisätty tietokantaan, API vastaanottaa tästä vastauksen, jonka se lähettää takaisin JSON-muotoisena RequestHandler-luokan sendGetRequest()-funktiolle. sendGetRequest()-funktion tarkoitus on kääntää JSON-muotoinen vastaus ja viedä se käyttäjän käyttöliittymään selkokielisenä vastauksena. Tällöin vastaus olisi ”Tarvike on lisätty onnistuneesti”, kuten kaavio kuvaa. Tässä kaaviossa selitetty rakenne toimii samoin myös kirjautumisen sekä tarvikkeiden muokkauksen ja poiston suhteen. Kaikki toiminnot siis suorittavat samankaltaisen kaavion polun, ainoastaan data polun sisällä vaihtuu.



Kuva 17. UML-sekvenssikaavio tarvikkeen lisäyksestä.

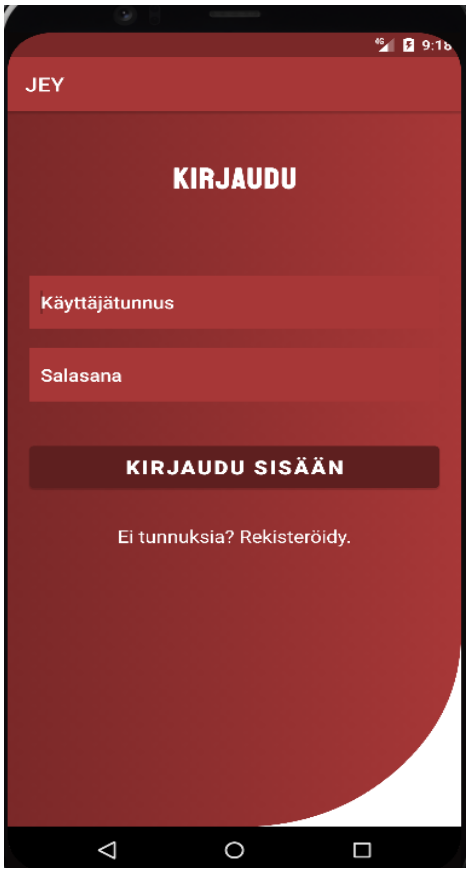
9 Pohdinta

9.1 Tulokset

Opinnäytetyön toiminnallisen osuuden tuloksena luotiin toimeksiantajan vaatimusmäärittelyn mukainen mobiilisovellus, joka toimii Android-käyttöjärjestelmässä. Aikataulu toiminnalliselle osuudelle piti miltei suunnitelmien mukaan ja sain palautettua sovelluksen toimeksiantajalle, kuten oli sovittu.

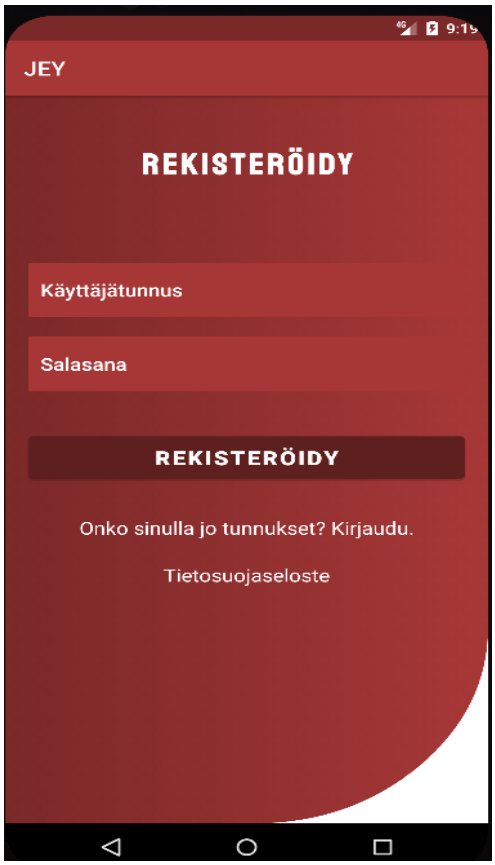
Ohjelma sisältää tällä hetkellä kaikki vaatimusmäärittelyissä määritellyt kohdat. Toistaiseksi sovellukseen ei saatu toteutettua hakutoimintoa ajan puutteen vuoksi. Tämä toiminto on kuitenkin suunnitelmissa tulevaisuudessa helpottamaan tarvikkeiden listausta.

Sovelluksen käynnistyessä avautuu kirjautumisnäkymä (kuva 18). Kirjautumisnäkymässä käyttäjän on tarkoitus kirjautua sovellukseen sisään tai vaihtoehtoisesti siirtyä rekisteröitymisnäkymään, jos tunnuksia ei ole.

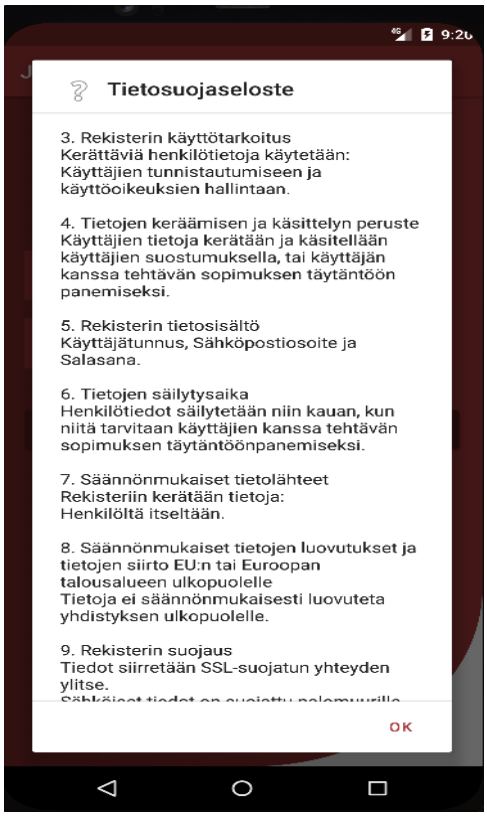


Kuva 18. Kirjautuminen.

Rekisteröitymisnäkymä (kuva 19) on pääpiirteittäin samanlainen kuin kirjautumisnäkymä. Rekisteröitymisnäkymän tarkoitus on luoda käyttäjätunnus, jolla käyttäjä voi kirjautua sisään. Samasta näkymästä käyttäjä voi myös katsoa tietosuojaselosteen (kuva 20). Tietosuojaseloste pitää sisällään käyttäjän oikeuksia ja rekisterin sekä tietokannan ylläpitäjän tiedot.



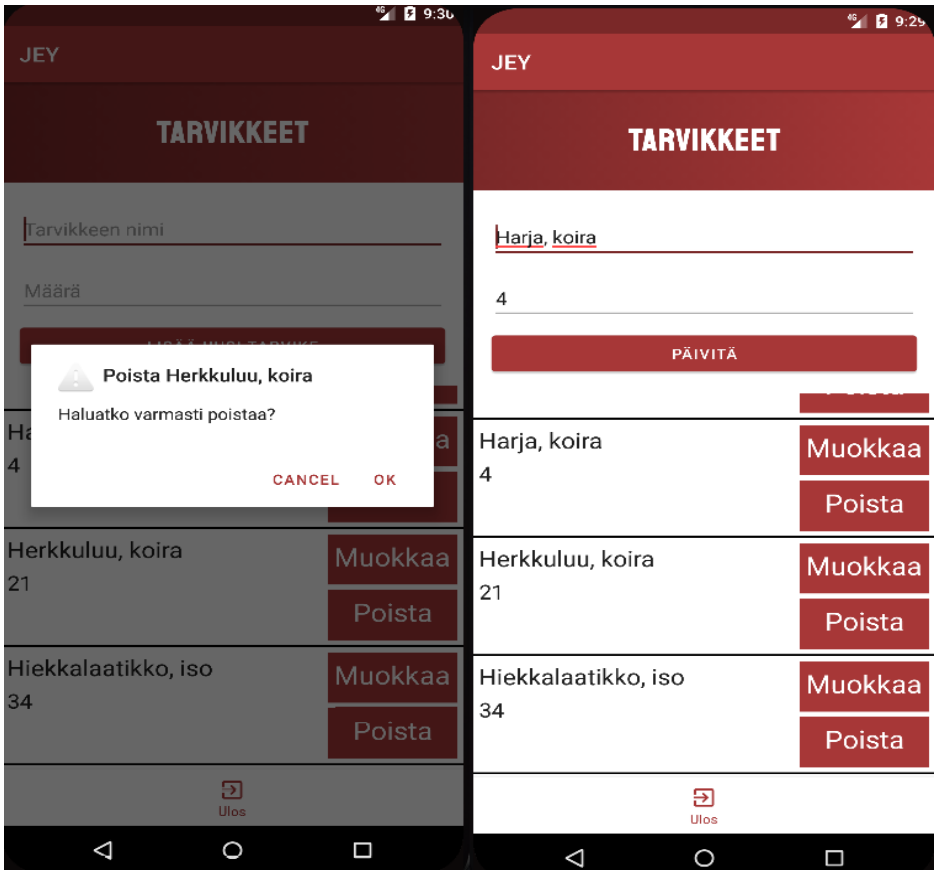
Kuva 19. Rekisteröityminen.



Kuva 20. Tietosuojaseloste.

Mikäli käyttäjällä on oikeus kirjautua sovellukseen, avautuu päänäkymä (kuva 21). Päänäkymässä on listaus tietokannan tarvikkeista sekä mahdollisuus lisätä, poistaa (kuva 22a) tai muokata (kuva 22b) tarvikkeita. Listattavat tarvikkeet päivittyvät automaattisesti aakkosjärjestykseen luotaessa tai päivittäessä jo olemassa olevia tarvikkeita sekä silloin kun tarvikkeet haetaan tietokannasta.

Kuva 21. Päänäkymä.



Kuva 22a, 22b. Tarvikkeiden poisto ja muokkaus.

Päänäkymän alanavigaatio sisältää tällä hetkellä vain uloskirjautumisen, mutta ajatuksena olisi lisätä sinne myös tarvikkeiden hakua vastaava painike. Uloskirjautumispainiketta painamalla käyttäjä kirjautuu ulos sovelluksesta ja sovellus palauttaa käyttäjän takaisin kirjautumisnäkymään.

9.2 Johtopäätökset

Alun perin minulla oli ajatus tehdä Joensuun eläinsuojeluyhdistykselle sovellus, joka helpottaisi vapaaehtoisten värväämistä. Kävimme keskustelua toimeksiantajan kanssa ja siinä tuli ilmi tarve tietokantasovelluksesta. Tämän keskustelun johdosta syntyi opinnäytetyön tarkoitus. Tämän sovelluksen tarkoitus oli helpottaa JEY:n varastoilla sijaitsevien tarvikkeiden seurantaa ja vähentää bensakustannuksia.

Opinnäytetyön tavoitteena oli luoda Android-mobiilisovellus Joensuun Eläinsuojeluyhdistykselle. Tavoitteeseen pääsin tekemällä toimeksiantajan kanssa vaatimusmäärittelyn ja toteuttamalla mobiilisovelluksen sekä saamalla sen ajoissa valmiiksi. Aikataulutus oli tärkeässä roolissa tavoitteen saavuttamiseksi, koska opinnäytetyön aikataulu on ollut vain kolme kuukautta. Mobiilisovellus on ollut JEY:llä käytössä siitä asti, kun se on yhdistykselle luovutettu.

Miettiessäni miten lähteä toteuttamaan tätä tietokantasovellusta ajattelin, onko natiivi mobiilisovellus sittenkään se oikea valinta. Mahdollisena vaihtoehtona pidin toteuttamista suoraan web-ympäristöön, jolloin se olisi toiminut kaikissa käyttöjärjestelmissä. Tällä hetkellä, mikäli JEY joskus vaihtaa käyttöjärjestelmäänsä esimerkiksi iOS:iin, ei sovellus enää toimikaan. Päädyin kuitenkin mobiilisovellukseen, koska halusin saada kokemusta myös mobiilisovelluskehityksestä ja toimeksiantajan tarve oli vain Android-käyttöjärjestelmälliselle sovellukselle.

Koska Android on toinen maailman suosituimmista käyttöjärjestelmistä, löytyi Android Studiolla kehittämiseen paljon ohjeita. Ohjeista oli paljon hyötyä kehityksen ollessa jumissa ja varsinkin back-end-puolen kanssa. Ongelmia tuotti esimerkiksi Android-sovelluksen ja MySQL-tietokannan liittäminen keskenään, koska minulla ei ollut tästä aiempaa kokemusta. MySQL tarjoaa paremman indeksointijärjestelmän kuin Firebase, johon tutustuin opinnäytetyöprosessin aikana, joten MySQL oli mielestäni valintana onnistunut. Firebase olisi tarjonnut 10GB:seen asti ilmaista tietokantatilaa, mutta tämän ylittyessä olisi siitä täytynyt maksaa. Koska JEY:n ylimääräiset kustannukset halutaan pitää pienenä, ei tämä sopinut tietokantavaihtoehdoksi. MySQL:n tilalla SQLite olisi tarjonnut vain paikallisesti kirjoitetun tietokannan, joka tallentuu puhelimeen itseensä, eikä se olisi palvellut käyttäjiä siinä, että kaikkien sovellukseen pääsyn omaavien olisi pitänyt päästä muokkaamaan samaa tietokantaa.

Front-endin kehitysvaiheessa havahduin ajatukseen, onko sovellus sittenkään niin helppokäyttöinen kuin se voisi olla. Miettiessäni käyttäjäkokemusta ja UX-

suunnittelua en tullut ajatelleeksi sitä, onko Figmalla suunnittelemani lisää tarvike -näkymä tarpeellinen. Lisää tarvike -näkymä selkeytti sovellusta siinä määrin, että käyttäjän on helppo nähdä, missä tarvikkeita lisätään, mutta samalla se aiheutti ylimääräisiä painalluksia, mikäli halusi lisätä tarvikkeen ja nähdä sen jälkeen listauksen. Tästä syystä päädyin ratkaisuun, jossa kaikki toiminta tapahtuu yhdellä sivulla. Näin ollen käyttäjän lisätessä tarvike, hän näkee suoraan, kun se päivittyy listaukseen, ja voi muokata sitä vaikka saman tien.

Koko opinnäytetyön haastavin osa oli varmasti kirjallinen osuus. Kirjallisen prosessin aikana kehitystä on tapahtunut niin lähteiden käytössä kuin kirjoitusasun ja asiatyylisen tekstin luomisessa. Itse lähteiden hakeminen tietoperustaan ja tietoperustan kirjoittaminen ei ollut vaikeaa, koska tietoa löytyi kattavasti. Ainoana ongelmana oli englanninkielisten lähteiden kääntäminen ja varsinkin tiettyjen sanojen kääntäminen suomeksi. Joillekin sanoille ei tuntunut löytyvän järkevää suomennosta.

Opinnäytetyön lähtökohdissa mainitsemani opinnäytetyö Seinäjoen seudun eläinsuojeluyhdistyksen Jäsenhuone-sivuston uudistus sai pohtimaan, miksi opinnäytetöitä ei ole tehty enemmän eläinsuojeluyhdistyksille. Vaikka niille ei toteuttaisi natiiveja sovelluksia, voisivat ne hyötyä esimerkiksi verkkosivuista. Mikäli eläinsuojeluyhdistyksellä olisi omat verkkosivut kokisin, että yhdistykset olisivat helpommin saatavilla. Luulen, että eläinsuojeluyhdistykset pitävät tärkeimpänä asiana tulla nähdyksi verkkosivujen muodossa kuin saada oma natiivi sovellus. Uskon kuitenkin, että oma opinnäytetyöni on hyvä lisä tuomaan tietoutta tällaisista sovellusmahdollisuuksista myös eläinsuojeluyhdistyksille. On myös mielenkiintoista nähdä, miten tulevaisuudessa natiivisovellukset yleistyvät eläinsuojeluyhdistysten keskuudessa.

Jos nyt aloittaisin opinnäytetyöni uudelleen, muutama asia muuttuisi. Ensimmäisenä tekisin vaatimusmäärittelystä vielä laajemman. Lisäisin sinne mahdollisesti kohdan, jossa olisi ei-pakolliset, mutta toivotut ominaisuudet. Ja mikäli aikaa olisi ollut ylimääräisenä, olisi näitä toivottuja ominaisuuksia voinut toteuttaa. Toisena muutoksena ottaisin käyttöön jonkin projektinhallinnan

työkalun, esimerkiksi Azure DevOpsin. Tämä olisi helpottanut huomattavasti muun muassa ajan seurantaa ja sitä, mitä pitäisi vielä tehdä. Välillä tuskastuin kehittäessäni sovellusta, sillä joitain asioita tai pieniä muutoksia olisi pitänyt tehdä aiemmin. Luulen, että tällä tavoin olisi ollut vieläkin helpompaa pysyä aikataulussa.

9.3 Ammatillinen kasvu ja kehitys

Tämä opinnäytetyö on opettanut minua monella tapaa. Olen kehittynyt tiedon hankinnassa, erilaisten suunnitelmien laatimisessa ja toteuttamisessa, vaatimuksien määrittelyssä sekä mobiilisovelluskehityksessä. Olen myös saanut paljon uutta tietoa siitä, miten mobiilisovelluksia kannattaa tai pitäisi tehdä. Esimerkiksi salausprotokollat ja niiden merkitys tuli minulle uutena asiana esille kehitysvaiheessa. Myös Android Studio osoittautui varsin aloittelijaystävälliseksi käyttää. Sen kanssa oli helppoa luoda front-end ja saada asiat toimimaan niin kuin oli suunnitellut. MySQL-tietokanta ja PHP toimivat varsin hyvin tähän käyttötarkoitukseen ja tämän kokoluokan sovellukseen. Koska MySQL on relaatiotietokanta, antaa se paljon mahdollisuuksia esimerkiksi taulujen välille luotavien yhteyksien kanssa tulevaisuuden ominaisuuden, kategorisoinnin lisäykseen.

Uutena ja mielenkiintoisena asiana tuli myös tehdä projektia yksin alusta loppuun saakka. Opintojeni aikana olemme tehneet paljon tiimiprojekteja, joten itsenäinen kehitys on jäänyt hyvin vähäiseksi. Itsenäisen työskentelyn hyötyjä ovat olleet omasta työstä vastaaminen, aikataulutus ja se, että on saanut itse päättää asioista. Toisaalta välillä tiimistä tai työparista olisi varmasti ollut hyötyä. Hankalien tilanteiden pohtiminen ja ajatusten jakaminen olisi saattanut helpottaa ja nopeuttaa etenemistä. Koska fullstack-kehittäjät tekevät työtään yksin, uskon, että olen nähnyt yhdenlaisen kokonaiskuvan siitä, millaista fullstack-kehittäjänä projektityöskentely voi olla.

9.4 Luotettavuus ja eettisyys

Koska sovellus ei ole tulossa julkiseen jakoon, vaan ainoastaan JEY:n käyttöön, on sillä hyvin vähän eettisiä kysymyksiä. Tietenkin on mahdollista, että joku JEY:n vapaaehtoinen päättääkin vaikka tuohtuneena muille sotkea koko tietokannan sisällön sovelluksen kautta. Tästä syystä toteutin sovellukseen rekisteröitymis- ja kirjautumisvaiheen. Kaikki sovelluksen omistajat voivat rekisteröityä, mutta heiltä on pääsy evätty kirjautua sovellukseen ennen kuin tarvittava attribuutti löytyy. Mikäli attribuuttia ei ole, ei ole myöskään oikeuksia tehdä mitään tuhoa tarviketietokannan suhteen. Myös käyttäjien tiedot salataan, kun ne viedään tietokantaan, joten salasanojen ei pitäisi voida vuotaa ainakaan selkokielisenä kenellekään.

Sovellukseen luotiin myös tietosuojaseloste, jonka tarkoituksena on kertoa käyttäjälle, mihin hänen tietojaan käytetään ja kuka tietoja säilyttää. Tietosuojaseloste on luotu, koska käyttäjillä pitää olla oikeus tarkistaa ja oikaista heistä rekisteriin kerättyjä tietoja. Tässä sovelluksessa esimerkiksi käyttäjien on hyvä tietää kehen ottaa yhteyttä, jos haluavat, että heidän käyttäjätietonsa poistetaan.

Tietosuojaselosteessa on myös kerrottu tietojen säilytysaika. Tämä on tärkeää, koska yhdistyksessä jäsenissä voi olla vaihtuvuutta. Miettien vuosienkin päähän, voi yhdistyksen jäsenistö olla vaihtunut, jolloin tietokannassa olevat tunnukset eivät enää ole ajankohtaisia säilyttää. Jos rekisterinpitäjällä ei olisi oikeutta poistaa sovelluksen käyttäjien tunnuksia, riskinä olisi mahdollinen sovelluksen väärinkäyttö. Tietosuojaseloste lisää näin ollen myös sovelluksen luotettavuutta.

Vaikka en voi taata kokonaisvaltaista luotettavuutta sovellukselle, ottaen huomioon esimerkiksi tietoturvariskit palvelinpuolella, voin ainakin varautua asioihin suunnittelemalla ja pohtimalla näitä eettisiä kysymyksiä.

9.5 Hyödynnettävyys ja jatkokehitys

Sovellusta voisi hyödyntää jatkossa myös Suomen laajuisesti. Esimerkiksi pienillä muokkauksilla sitä voisi tarjota myös muille suomen eläinsuojeluyhdistyksille, mikäli tällaiselle sovellukselle olisi kiinnostusta tai tarvetta.

Jatkokehitystä mietittäessä on tullut jo ilmi erilaisia ominaisuuksia, joita olisi hyvä olla sovelluksessa tulevaisuudessa. Näitä ovat esimerkiksi tietojen varmuuskopiointi, jos puhelin ei ole yhdistettynä internettiin, tai admin-hallintapaneeli sisään kirjautuvien käyttäjien attribuuttien hallintaan. JEY itse on ilmaissut, että jatkokehitysideana voisi olla hakutoiminto, jolla voisi hakea tiettyä tarviketta nimeltä. Myös erilaisten tarvikkeiden kategorisoimisesta on puhuttu.

En myös pidä mahdottomuutena luoda tämän projektin pohjalta verkkosivu, jolla olisi samat ominaisuudet kuin sovelluksella, mutta toimisi selaimessa. Samalla voisi myös hyödyntää verkkosivuja siihen, että JEY saisi omat sivunsa näkyvämmin esille, kun ne ovat nyt Sey.fi -sivujen alla.

Lähteet

Al Mamun, A. 2022. Full Stack. https://urly.fi/2UbP. 11.09.2022.

Altexsoft. 2020. The Good and The Bad of Xamarin Mobile Development. https://urly.fi/2UbN. 11.09.2022.

Android Developers. 2022a. Meet Android Studio. https://developer.android.com/studio/intro. 11.09.2022.

Android Developers 2022b. ListView. https://urly.fi/2UbQ. 22.11.2022.

Black, R. 2020. Essential mobile app development tools and programming languages. https://urly.fi/2UbR. 11.09.2022.

Brush, K. 2020. Use case. https://urly.fi/2UbS. 22.11.2022.

Churchville, F. 2022. User Interface (UI). https://urly.fi/2UbT. 22.11.2022.

Cousins, C. 2019. What is Figma? A 101 Intro. https://urly.fi/2UbU. 22.11.2022.

David, M., Novotny, A. & Denman, J. 2021. Mobile application development. https://urly.fi/2UbV. 08.09.2022.

ERDPlus. 2021. A database modeling tool for creating Entity Relationship Diagrams, Relational Schemas, Star Schemas, and SQL DDL statements. https://www.erdplus.com. 22.11.2022.

Firebase. 2022. Firebase Realtime Database. https://firebase.google.com/docs/database. 12.09.2022.

Geeksforgeeks. 2021. Activity Lifecycle in Android with Demo App. https://urly.fi/2UbW. 22.11.2022.

Geeksforgeeks. 2022. A Complete Guide to Learn XML For Android App Development. https://urly.fi/2UbX. 22.11.2022.

Hiltunen, E. & Hiltunen, K. 2014. Teknoelämää 2035. Miten teknologia muuttaa tulevaisuuttamme?. Helsinki: Talentum. E-kirja.

Hovi, A., Huotari, J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo. E-kirja.

Itewiki. 2022a. Mobiilikehitys. https://www.itewiki.fi/opas/mobiilikehitys/. 07.09.2022.

Itewiki. 2022b. Android. https://www.itewiki.fi/opas/android/. 10.09.2022.

Joensuun eläinsuojeluyhdistys. 2022. Yhdistys. https://urly.fi/2UbY. 22.11.2022.

Kostiainen, I. 2022. Seinäjoen seudun eläinsuojeluyhdistyksen Jäsenhuone-sivuston uudistus. https://www.theseus.fi/bitstream/handle/10024/752983/Kostiainen_Iiris.pdf?sequence=2&isAllowed=y. 28.11.2022.

Laricchia, F. 2022. Market share of mobile operating systems worldwide 2012-2022. https://urly.fi/2U6k. 12.09.2022.

Lewis, S. 2019. Full-stack developer. https://www.theserverside.com/definition/full-stack-developer. 12.09.2022.

Matilainen, M. 2022. Millainen on hyvä vaatimusmäärittely Hubspotin hankintaa varten?. https://urly.fi/2UbZ. 22.11.2022.

Moore, L. 2018. MySQL. https://www.techtarget.com/searchoracle/definition/MySQL. 12.09.2022.

One. 2022a. Mikä on domain?. https://urly.fi/2Uc0. 22.11.2022.

One. 2022b. Mitä on SQL injektio (SQL injection)?. https://www.one.com/fi/verkkosivuston-turvallisuus/mita-on-sql-injektio. 27.11.2022.
Ready, K. 2019. Getting started with full-stack mobile development. https://urly.fi/2UbP. 11.09.2022.
Salesforce. 2022. What is an API? (Application Programming Interface). https://urly.fi/2Uc1. 22.11.2022.
Samsukha, A. 2022. Best Databases For Mobile Apps 2022 – Choosing the Best One.
SQLite. 2022. About SQLite. https://www.sqlite.org/about.html. 12.09.2022.
Statista Research Department. 2016. Mobile phone users worldwide 2015-2020. https://urly.fi/2Uc2. 12.09.2022.
Stevenson, D. 2018. What is Firebase? The complete story, abridged. https://urly.fi/2Uc3. 11.09.2022. https://urly.fi/2UBe. 29.11.2022.
Suomen eläinsuojelu. 2022. SEY. https://sey.fi/medialle/. 27.11.2022.
TechTarget Contributor. 2018. Android Studio. https://urly.fi/2Uc4. 11.09.2022.
Tiwari, R. 2022. 9 ER Model Tools to use in 2022: A Comprehensive List. https://hevodata.com/learn/er-model-tools/#l8. 22.11.2022.
University Of Toronto. 2022. What Is a Full Stack Developer & What Do They Do?. https://urly.fi/2Uc5. 11.09.2022