Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2023

Nico Kranni

Mobiilisovelluksen kehittäminen eläinsuojeluyhdistykselle

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2023 | 29 sivua

Nico Kranni

Mobiilisovelluksen kehittäminen eläinsuojeluyhdistykselle

Tässä opinnäytetyössä keskityttiin mobiilisovelluksen kehittämiseen yhteistyössä eläinsuojeluyhdistys Dewi ry:n kanssa. Opinnäytetyön tavoitteena oli luoda toimiva mobiilisovellus, joka tarjoaa käyttäjille nopean ja helppokäyttöisen tavan saada viimeisimmät ohjeistukset sijaiskotina toimimisesta. Sovelluksen pääasiallisena tarkoituksena on tarjota käyttäjille pääsy eläinsuojeluyhdistyksen hoito-oppaaseen.

Projektin alussa Figma-työkalua käytettiin ulkoasun suunnitteluun. Ulkoasuun sisällytettiin navigointipainikkeita ja ankkureita ohjeiden selailua varten. Hoito-oppaaseen lisättiin zoomaustoiminnallisuus, joka oli yksi tärkeimmistä ominaisuuksista. Asiakkaalle esiteltiin suunnitelma, ja saadun palautteen perusteella tehtiin muutoksia ulkoasuun.

Mobiilisovelluksen kehittämisessä hyödynnettiin React Native -ohjelmointikieltä, jonka ainutlaatuinen ominaisuus mahdollistaa sovelluksen rakentamisen samanaikaisesti sekä Androidille että iOS:lle. Tämä lisäsi projektin tehokkuutta ja nopeutti kehitysprosessia huomattavasti. Sovelluksen interaktiivisuutta parannettiin ja sen ulkoasua hiottiin tarkasti. Lopullisessa vaiheessa sovellus testattiin kattavasti ja julkaistiin molemmissa sovelluskaupoissa. Koko projekti toteutettiin yhteistyössä Dewi ry:n kanssa, ja lopputuloksena saatiin aikaan monipuolinen mobiilisovellus. Sovellus tarjoaa eläinsuojeluyhdistyksen ohjeita sijaiskodeille ja tukee niiden tehokkaampaa toimintaa.

Asiasanat:

Ohjelmistokehitys, mobiilisovellus, oppiminen, suunnittelu, hyväntekeväisyys

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information & Communications Technology

2023 | 29 pages

Nico Kranni

Building a mobile application for an animal welfare organization

This thesis focused on developing a mobile application in collaboration with the animal welfare association Dewi ry. The goal of the thesis was to create a functional mobile application that provides users with a fast and user-friendly way to access the latest guidelines for acting as a foster home. The primary purpose of the application is to provide users with access to the animal welfare association's care manual.

In the thesis, I initially used the Figma tool for designing the user interface. The design included navigation buttons and anchors for browsing the instructions. An important feature was also adding zoom functionality to the care manual. The design was presented to the client, and based on feedback, changes were made to the user interface.

The development of the mobile application utilized the React Native programming language, a key feature of which is its ability to simultaneously build applications for both Android and iOS. This significantly enhanced the efficiency and expedited the development process. The interactivity of the application was increased, and the user interface was meticulously polished. In the final stage, the application underwent comprehensive testing and was subsequently released in app stores. The entire project was a collaboration with Dewi ry, resulting in a versatile mobile application that provides guidelines for foster homes from the animal welfare association and aids in their more efficient operation.

Keywords:

Software development, mobile application, learning, designing, charity

Sisältö

Kuvat

Sanasto

Back-End	Backend sisältää kaiken palvelimella suoritettavan sovelluksen tai verkkosivuston osaset, johon kuuluvat esimerkiksi yhteydet tietokantoihin, lomakkeiden ja tiedostojen käsittelyt, sovellukseen tai sivustolle kirjautuminen sekä salasanojen tarkistaminen [1].
CLI	Komentorivi eli Command Line Interface on eräs mahdollisuus käyttöjärjestelmien ja muiden tietokoneohjelmien käyttöliittymäksi. Se on tapa olla vuorovaikutuksessa tietokoneohjelman kanssa syöttämällä tekstimuotoisia komentoja peräkkäin [2].
Expo	Expo on avoimen lähdekoodin kehyssovellus, joka toimii natiivisti Androidilla, iOS:llä ja verkossa. Expo yhdistää parhaat puolet mobiilista ja verkosta ja mahdollistaa monia tärkeitä ominaisuuksia sovelluksen rakentamiseen ja skaalaamiseen [3].
Figma	Figma on pilvipohjainen suunnittelusovellus ja vektoripiirto-ohjelma. Figman avulla voidaan tehdä kaikenlaisia graafisia suunnittelutöitä, kuten luoda verkko- tai mobiilisovelluksista malleja/pohjia, suunnitella sosiaalisen median viestejä, tehdä esitteitä ja kaikkea siltä väliltä [4].
iOS	iOS (aiemmin iPhone OS) on Applen kehittämä käyttöjärjestelmä, joka on käytössä Applen iPhone-, iPod Touch-, iPad- ja Apple TV -laitteissa [5].

node.js	Node.js on avoimen lähdekoodin, monialustainen JavaScript-suoritusympäristö [6].
npm	npm tarkoittaa Node Package Manageria. Se on kirjasto ja rekisteri JavaScript-ohjelmistopaketeille [7].
React	React on JavaScript-kirjasto, joka erikoistuu auttamaan kehittäjiä rakentamaan käyttöliittymiä eli UI:ta [8].
React Native	React Native on avoimen lähdekoodin JavaScript-kehys, suunniteltu sovellusten rakentamiseen useille alustoille, kuten iOS, Android ja myös web-sovelluksille, hyödyntäen täysin samaa koodipohjaa [9].

1 Johdanto

1.1 Dewi ry

Eläinsuojeluyhdistys Dewi ry on Turun seudulla toimiva voittoa tavoittelematon eläinsuojeluyhdistys, joka on erikoistunut populaatiokissojen auttamiseen. Tämä yhdistys toimii täysin vapaaehtoisvoimin ja sen tavoitteena on paitsi konkreettisesti auttaa kissoja, myös nostaa kissan arvoa Suomessa. Yhdistys jakaa tietoa populaatiokissoista ja kissatietoutta yleisesti laajalle yleisölle [10].

Dewi ry:n erityinen fokus on arkojen kissojen asemassa löytöeläiminä ja uusien kotien etsimisessä niille. Yhdistys toimii sijaiskotien pohjalta, ilman kissataloa tai yleistä osoitetta. Tämä lähestymistapa mahdollistaa löytöeläimille kodinomaisen ympäristön ja paremman mahdollisuuden sopeutua tulevaan pysyvään kotiinsa.

1.2 Mobiilisovelluksen tarve

Tässä opinnäytetyössä keskitytään mobiilisovelluksen kehittämiseen eläinsuojeluyhdistykselle sekä projektin eri vaiheisiin. Nykyaikana mobiilisovellukset ovat keskeisiä viestintävälineitä, tarjoten käyttäjille laajan kirjon toiminnallisuuksia ja intuitiivisia ratkaisuja erilaisiin tarpeisiin. Projekti sai alkunsa henkilökohtaisesta motivaatiosta kehittää hyväntekeväisyysorganisaatiolle mobiilisovellus, ja eläinsuojeluyhdistys Dewi ry osoitti kiinnostusta yhteistyöhön.

Opinnäytetyön päätavoitteena on suunnitella ja toteuttaa toimiva mobiilisovellus eläinsuojeluyhdistykselle. Sovelluksen tarkoituksena on mahdollistaa käyttäjille nopea ja vaivaton pääsy ajantasaisiin ohjeistuksiin sijaiskotitoiminnasta. Sovelluksen kautta käyttäjät voivat tutustua eläinsuojeluyhdistyksen hoito-oppaaseen. Keskeinen vaatimus on sisällön päivittämisen mahdollisuus lähes reaaliajassa ilman monimutkaisia prosesseja. Tämän vaatimuksen myötä on tarpeen tutkia ja arvioida erilaisia teknisiä ratkaisuja ja niiden soveltuvuutta projektin tavoitteisiin.

2 Työsuunnitelma

2.1 Vaatimukset

Ensimmäisessä vaiheessa määritellään sovelluksen rakenne, tarvittavat näkymät sekä yleiset vaatimukset. Tämä sisältää kolme pääosaa: ohjekirjan näkymän, jossa esitetään ohjeiden sisältö; otsikkojen näkymän, joka listaa kaikki ohjekirjan otsikot ja tarjoaa mahdollisuuden siirtyä suoraan valittuun kohtaan; sekä infonäkymän, joka sisältää hyödyllisiä tietoja ja yhteystietoja. Jokaisen näkymän suunnittelu keskittyy käytettävyyteen ja käyttäjäkokemuksen optimointiin.

2.2 Käyttöliittymän suunnittelu

Toisessa vaiheessa keskitytään elementtien valintaan ja käyttöliittymän suunnitteluun. Tässä vaiheessa valitaan sovelluksen käyttöliittymäelementit, kuten painikkeet, tekstikentät ja ikonit. Käyttöliittymän luonnokset laaditaan käyttäen Figma-työkalua, painottaen käytettävyyttä ja visuaalista selkeyttä.

2.3 React Nativen perehtyminen

Kolmannessa vaiheessa syvennän React Native -osaamistani React Nativen ja Expon dokumentaation avulla. Dokumentaatio tarjoaa kattavan ja käytännönläheisen opetusmenetelmän, joka keskittyy sovelluskehityksen olennaisiin osa-alueisiin. Dokumentaation aikana kerrotaan tärkeitä konsepteja ja parhaita käytäntöjä, jotka auttavat kehittämään entistä tehokkaampia ja käyttäjäystävällisempiä käyttöliittymiä.

2.4 Mobiilisovelluksen kehitys

Tässä vaiheessa valmistellaan ohjelmointiympäristö ja aloitetaan sovelluksen koodaus. Tämä sisältää React Native -ympäristön, Node.js:n ja Visual Studio Coden asennuksen ja konfiguroinnin. Koodausvaiheessa kehitetään erikseen kunkin näkymän ja varmistetaan niiden yhteensopivuus ja sujuva toiminta. Sovelluksen testaus eri laitteilla on tärkeää, jotta voidaan varmistaa sen toimivuus Android- ja iOS-alustoilla.

2.5 Testaus

Viidennessä vaiheessa sovellus julkaistaan beta-versiona, jotta voidaan kerätä käyttäjäpalautetta. Saadun palautteen perusteella tehdään tarvittavat muutokset sovellukseen, jotta voidaan parantaa sen käytettävyyttä ja toiminnallisuutta.

2.6 Julkaisu

Viimeisessä vaiheessa suoritetaan sovelluksen viimeistely ja valmistaudutaan sen julkaisuun sovelluskaupoissa. Lisäksi laaditaan suunnitelma sovelluksen jatkokehitystä ja mahdollisia päivityksiä varten, varmistaen sovelluksen pitkän aikavälin kestävyyden ja käyttäjien tyytyväisyyden.

3 Mobiilisovelluksen suunnittelu

3.1 Asiakastapaaminen, kartoituspalaveri

Opinnäytetyöni aiheen ideointivaiheen ja potentiaalisten yhteistyöorganisaatioiden kartoittamisen jälkeen aloitettiin tunnettujen paikallisten eläinsuojeluyhdistysten kontaktointiprosessin. Tavoitteena oli löytää yhteistyökumppani, joka hyötyisi mobiilisovelluksesta toiminnassaan. Lähetin useille yhdistyksille ehdotuksen kehittää mobiilisovellus, joka toimisi heidän toimintansa käsikirjana. Tämä sovellus olisi osa opinnäytetyötäni. Positiivisen vastauksen antoi Dewi ry, joka osoitti välitöntä kiinnostusta projektiin. Yhdistyksen tarve päivittää ohjeistuksiaan kotihoitajille sopi hyvin ehdotetun sovelluksen tavoitteisiin.

Projektin suunnitteluvaiheessa oli ensiarvoisen tärkeää määritellä sovelluksen tekniset vaatimukset ja sisältö. Sovelluksen keskeinen elementti olisi kattava, mutta helppokäyttöinen käsikirja. Tämän käsikirjan tulisi olla jatkuvasti päivitettävissä, jotta uusimmat ohjeet olisivat aina saatavilla. Käsikirjan suunnittelussa korostettiin, että sen tulisi olla kokonaisuudessaan saavutettavissa ilman tarvetta siirtyä useiden erillisten näkymien välillä. Lisäksi oli tärkeää, että tietyt keskeiset aiheet, kuten kissojen madotus tai ensiapuohjeet, löytyisivät nopeasti ja vaivattomasti.

Sovelluksen käyttäjäkokemusta ajatellen keskityimme navigointiratkaisujen kehittämiseen. Koska ohjeisto on laaja, oli olennaista suunnitella intuitiivinen navigointijärjestelmä, joka helpottaisi käyttäjiä löytämään tarvitsemansa tiedot nopeasti. Alkuperäisenä ideana oli ankkureiden käyttö, jossa otsikoihin yhdistetyt painikkeet ohjaavat suoraan kyseiseen kohtaan käsikirjassa. Tämä ratkaisu edellyttäisi huolellista suunnittelua ja toteutusta, jotta käyttökokemus olisi mahdollisimman sujuva ja tehokas.

Toinen keskeinen tekninen haaste oli sovelluksen päivitysmekanismin suunnittelu. Vaikka PDF-tiedoston upottaminen oli yksi vaihtoehto, se ei tarjonnut toivottua joustavuutta ja helppoutta päivitysten suhteen. Siksi

aloitimme kehittämään ratkaisua, joka mahdollistaisi sisällön päivittämisen suoraan sovelluksen kautta. Tämän ratkaisun tulisi olla sekä yksinkertainen ylläpitää että käyttäjäystävällinen.

3.2 Ulkoasun suunnittelu

Ulkoasun suunnitteluun valitsin Figma-työkalun, joka oli tullut tutuksi aiempien koulukurssien kautta. Figma on tunnettu sen aktiivisista päivityksistä, jotka tuovat jatkuvasti uusia ominaisuuksia käyttöliittymäsuunnittelun alueelle. Lisäksi Figma on laajasti tunnustettu ja käytetty työkalu käyttöliittymäsuunnittelun ammattilaisten keskuudessa.

Ennen ulkoasun suunnittelun aloittamista Figmalla toteutin taustatutkimusta yleisimmin käytössä olevista puhelinten näyttöresoluutioista, jotta voisin määrittää sopivan kuvasuhteen suunnittelun lähtökohdaksi. Tutkimusten perusteella en löytänyt yksiselitteistä suositusta tietylle resoluutiolle, mikä johtuu markkinoilla olevien puhelinmallien laajasta kirjosta ja niiden julkaisutiheydestä [11]. Lisäksi eri mallien resoluutiot voivat vaihdella merkittävästi [12].

Figma tarjoaa työkaluja, jotka mahdollistavat eri puhelinmallien resoluutioiden perusteella luotavien valmisten runkojen käyttämisen. Tämä ominaisuus helpotti suunnitteluprosessia merkittävästi, sillä käytettävissä oli valmiiksi määritetty runko ja kuvasuhde. Valitsin suunnittelun perustaksi iPhone 14 Pro -mallin resoluution, koska se on yksi Figma-työkalun uusimmista puhelinmalleista. Lisäksi sen resoluutio vastaa useiden Android-puhelimien resoluutiota, mikä antaa kattavan kuvan suunnittelun soveltuvuudesta eri laitteille.

Sovelluksen ulkoasun suunnittelussa valittiin Figma-työkalu sen aiemman tuttuuden ja alan tunnustuksen perusteella. Figma on tunnettu säännöllisistä päivityksistään, jotka tuovat uusia ominaisuuksia käyttöliittymäsuunnittelun alueelle. Ennen suunnitteluprosessin aloittamista toteutettiin tutkimus nykyisistä puhelinmallien näyttöresoluutioista, mikä auttoi määrittämään sopivan kuvasuhteen. Tutkimus paljasti, että laajan kirjon puhelinmallien resoluutiot vaihtelevat, mikä tekee yhden resoluution valinnan haastavaksi.

Figma tarjoaa työkaluja suunnitteluun eri puhelinmallien resoluutioiden perusteella, mikä helpotti suunnitteluprosessia. Valinnaksi päätyi iPhone 14 Pro -mallin resoluutio, koska se vastaa useiden Android-puhelimien resoluutiota.

Yhteistyökumppanilta, Dewi ry:ltä, saatiin projektin kannalta olennaiset materiaalit, jotka olivat ratkaisevassa roolissa sisällön suunnittelussa. Tämä mahdollisti suunnittelun aloittamisen yhdistyksen olemassa olevan materiaalin pohjalta. Väliaikaisen sisällön tuottamiseen, ennen yhdistyksen lopullisen käsikirjan valmistumista, käytettiin ChatGPT:tä. ChatGPT generoi tekstiä annettujen otsikkojen perusteella, auttaen hahmottamaan käsikirjan mahdollista pituutta ja rakennetta.

Käyttöliittymäelementtien suunnittelussa käytettiin Font Awesome -kirjastoa, joka tarjoaa laajan valikoiman valmiita kuvakkeita ja ikoneita [13]. Väriksi valittiin Dewi ry:n pääväri, vihreä. Figman prototyyppiominaisuutta hyödynnettiin interaktiivisten painikkeiden toteutuksessa, joka simuloitiin näkymien vaihtoa, antaen realistisen esikatselun suunnitellusta käyttöliittymästä. Tämä esikatselu oli keskeinen osa suunnitteluprosessia, mahdollistaen sujuvan siirtymisen sovelluksen kehitysvaiheeseen. Ensimmäinen luonnos (Kuva 1) toteutui nopeasti, vaikka tuntuen vajavaiselta.

Kuva 1. Sovelluksen ensimmäinen luonnos.

Alkuperäinen suunnitelma Figman käytöstä kohtasi teknisiä haasteita. Interaktiivisuus, erityisesti otsikoiden pikalinkkien ankkurointi, ei toiminut odotetulla tavalla. Interaktiiviset polut toiseen näkymään oli mahdollista luoda, mutta suora yhteys otsikon ja tekstissä olevan kohdan välillä ei onnistunut. Tämän seurauksena oli tarpeen aloittaa komponenttien luominen alusta, mikä toi mukanaan uuden oppimisen ja kehityksen mahdollisuuden.

Ratkaisuna otettiin käyttöön menetelmä, jossa jokaiselle tekstiosiolle luotiin oma komponenttinsa, jotka sitten yhdistettiin samaan kehykseen tai ryhmään. Vaikka tämä lähestymistapa ei suoraan ratkaissut alkuperäistä ankkurointiongelmaa, se mahdollisti käyttöliittymän toiminnallisuuden, joka replikoi lähimmin oikean mobiilisovelluksen toimintaa.

Lopullisessa ulkoasun suunnittelussa tehtiin vielä muutoksia painikkeiden kuvakkeisiin, jotta ne paremmin kuvaisivat niiden toimintoja. Esimerkiksi etusivun talokuvakkeen vaihtaminen kirjakuvakkeeksi oli osa tätä muutosta. Kirjakuvake oli valittu edustamaan ohjekirjaa, joka on keskeinen elementti

kyseisessä näkymässä. Nämä muutokset viimeisteltyään suunnitelma (Kuva 2) esiteltiin asiakkaalle hyväksyttäväksi.

Kuva 2. Sovelluksen toinen luonnos.

Tehtyjen muutosten seurauksena sovelluksen ulkoasu alkoi vastata suunniteltuja vaatimuksia sekä toiminnallisuudeltaan että visuaalisesti. Se simuloitiin niin, että se jäljitteli valmiin mobiilisovelluksen toimintaa mahdollisimman tarkasti. Tässä vaiheessa projektia oli oleellista saada asiakkaan näkemyksiä ja palautetta suunnitellusta ulkoasusta. Tämän vuoksi lähetettiin asiakkaalle Figman interaktiivinen prototyyppi, joka esitteli sovelluksen käyttöliittymän ja sen toiminnot. Asiakkaalta pyydettiin palautetta erityisesti käyttöliittymän suunnitteluun liittyen, jotta varmistettaisiin, että lopputulos vastaisi heidän tarpeitaan ja odotuksiaan.

3.3 Ulkoasun muutokset ja hyväksyntä

Käyttäjältä saadun palautteen perusteella aloitin välittömästi suunnitellut muutokset sovelluksen ulkoasuun. Yksi keskeinen kehityskohde oli parantaa

painikkeiden interaktiivisuutta lisäämällä niihin pieni visuaalinen efekti painalluksen yhteydessä, mikä jäljittelisi fyysisen painikkeen toimintaa. Lisäksi olin integroinut yläosaan kiinteän bannerin, joka alun perin seurasi käyttäjän vieritystä alas ohjeiden läpi. Saadun palautteen mukaan banneri tulisi jättää yläosaan eikä annettava sen seurata vieritystä, ja lisäksi sen kokoa tulisi pienentää.

Uusien ominaisuusehdotusten joukossa oli hakuominaisuuden lisääminen ja fontin koon säätämisen mahdollistaminen. Näiden ehdotusten pohjalta aloin viimeistellä ulkoasusuunnitelmaa. Yläosan bannerin muuttaminen kiinteäksi ja sen koon säätäminen oli suoraviivaista, sillä olin alun perin suunnitellut bannerin niin, että se voisi seurata käyttäjää. Lukitsemalla bannerin yläosaan ja säätämällä sen kokoa, varmistin sen visuaalisen sopivuuden ja toiminnallisuuden. Kehityksen tueksi latasin Figman mobiilisovelluksen puhelimeeni, jotta sain paremman käsityksen bannerin optimaalisesta koosta ja käyttökokemuksesta.

Fonttien kokomuutoksen implementointi Figmassa suunnitellulla tavalla kohtasi teknisiä rajoitteita. Alun perin tavoitteena oli mahdollistaa ohjeiden fonttikoon dynaaminen muuttaminen käyttäjän toimesta. Vaikka tämä toiminnallisuus toteutettiin onnistuneesti, se aiheutti toimintahäiriöitä navigointipainikkeissa.

Teknisen arvioinnin ja asiantuntijakonsultaation jälkeen todettiin, että Figma ei tue kyseistä toiminnallisuutta optimaalisesti. Yksinkertaisemmaksi ja tehokkaammaksi ratkaisuksi valittiin tekstien zoomaustoiminto, joka on standardi toiminnallisuus mobiililaitteilla. Tämä ratkaisu mahdollistaa käyttäjille helpon tekstin suurentamisen ja parantaa lukukokemusta. Tämän jälkeen lähetin ulkoasun (Kuva 3) uudelleen testattavaksi.

Kuva 3. Sovelluksen kolmas luonnos.

Lopulta toteutetut muutokset saivat asiakkaan hyväksynnän. Sovelluksen lopullinen käyttöliittymäsuunnitelma oli käytännöllinen ja täytti asiakkaan vaatimukset.

4 Mobiilisovelluksen kehittäminen

4.1 Teoriaan perehtyminen

Ennen opinnäytetyöni kehittämisprosessia tarvitsin lisää tietämystä mobiilisovellusten kehittämisestä. Valitsin React Native -ohjelmointikielen, koska sen avulla on mahdollista tuottaa sekä Android- että iOS-versiot samanaikaisesti. React Nativella on yhteys React-ohjelmointikieleen, josta minulla oli jo aiempaa kokemusta.

React Nativen sekä Expo-dokumentaation avulla pystyin perehtymään React Nativelle ominaisiin piirteisiin ja sen toimintalogiikkaan, minkä jälkeen ohjeistettiin lataamaan kehitystyökalut: Node.js, Android Studio, GIT ja Visual Studio Code. Vaikka Android Studiota suositellaan emulaattorina, päätin käyttää omaa laitteistoani testaamiseen ja kehittämiseen.

Sovelluksen kehittämisen alussa tehdään valinta React Native CLI:n ja Expo CLI:n välillä. Expo CLI valittiin sen helppokäyttöisyyden ja aloittelijaystävällisyyden vuoksi, vaikka se karsii joitain ominaisuuksia. Dokumentaatiossa käytiin läpi Expo CLI:n asennusta, Expo-sovelluksen käynnistämistä ja sovelluksen käyttöönottoa omalla laitteella [14].

Dokumentaatiossa näytettiin React Nativelle tyypillinen rakenne ja sen eri menetelmät, funktiot ja komponentit. Lisäksi kerrottiin komponenttien tyylin muokkaamista käyttäen CSS-tyylisovelluksia ja StyleSheet.create-funktiota. Dokumenttien avulla saavutettiin kattava ymmärrys siitä, miten React Native -sovelluskehitystä lähestytään ja toteutetaan.

4.2 Mobiilisovelluksen kehittäminen

React Native ympäristön tutustumisen jälkeen Dewi ry:n mobiilisovelluksen kehitysprojektin valmistelu aloitettiin. Projektin teknologiset tarpeet määriteltiin

ensin. Valituiksi teknologioiksi identifioitiin React Native -ohjelmointikieli, Node.js-ympäristö, npm-pakettien hallintajärjestelmä sekä Visual Studio Code -tekstieditori.

Tämän valmisteluvaiheen jälkeen aloitin Expo-projektin luomisen käyttäen create-expo-app-komentoa. Tämä komento luo perustan ja rakenteen sovelluksen kehittämistä varten. Komennon suorittaminen käynnistää prosessin, jossa kysytään projektin perustietoja, kuten nimi, perusta ja joitakin asetuksia, jotka ovat välttämättömiä projektin konfiguroinnissa.

Sovelluksen toteutuksessa valitsin Model-View-Controller (MVC) -arkkitehtuurin, joka perustuu sen modulaariseen lähestymistapaan ohjelmistosuunnittelussa. MVC-arkkitehtuuri erottaa tehokkaasti sovelluksen datan (Model), käyttöliittymän (View) ja sovelluslogiikan (Controller) toisistaan. Tämä eriyttäminen mahdollistaa paremman koodin hallinnan ja helpottaa yksikkötestausta.

4.2.1 Näkymien luonti

Näkymien luonti alkoi tekemällä View-kansion MVC:n mukaan, johon tein tiedostot jokaiselle näkymälle. Kaikilla näkymillä on sama header sekä footer, mikä oli seuraava kohde, minkä puoleen käännyin. Jotta rakenne pysyisi mahdollisimman selvänä, tein headerille ja footerille oman kansion.

Headeriä kehittäessä käytin React Nativen Image-komponenttia saadakseni Dewi ry:n logon esille sovelluksen sisällä. Jotta kuva skaalautuisi hyvin eri kokoisissa näytöissä, käytin UseWindowDimension-koukkua, mikä mahdollisti laittamaan kuvan leveyden olemaan sama kuin puhelimen näytön leveys.

Footerin kehityksessä käytin suorakulmaista palkkia, jonka päälle laitoin ulkoasussa käyttämiäni FontAwesome-ikoneita. Vähentääkseni tiedostojen määrää sovelluksessa, käytin FontAwesome-npm-pakettia, joka hakee ikonit ja asentaa ne osana rakennusprosessia. Tämä vähentää virheiden mahdollisuuksia. Käytin ikoneiden kanssa TouchableOpacity-komponenttia,

joka teki ikoneista interaktiiviset, joka pystyisi tulevaisuudessa vaihtamaan näkymää samalla kuin antoi visuaalisen vihjeen painalluksen rekisteröinnistä.

Footerin ja headerin ollessa valmiit, importtasin kyseiset komponentit kaikkiin näkymiin. Käytin näkymissä Flexbox-layout-mallia. Flexbox on tapa järjestää elementtejä säiliössä niin, että ne voivat skaalautua ja sopeutua eri näyttökokoja ja -suuntia varten.

Footer ja header sisällytettiin View-komponentin sisälle, mikä tarjoaa joustavan tavan ryhmitellä ja asetella muita komponentteja. View-komponentille voidaan antaa erilaisia tyylejä, joista yksi on Flexbox.

4.2.2 Siirtyminen

Sovelluksen kolmen eri näkymän perusrakenteen valmistuttua siirtyi projektin painopiste navigoinnin toteutukseen näiden näkymien välillä. Tämän haasteen ratkaisemiseksi valittiin React Navigation -kirjasto, jonka avulla navigointi ja siirtymät sovelluksen sisällä voidaan toteuttaa sujuvasti ja tehokkaasti.

React Navigation -kirjaston [15] tarjoamista navigointiratkaisuista valikoitui käyttöön StackNavigator, joka perustuu näkymäpinoon (stack). StackNavigatorin toimintaperiaatteena on, että kun uusi näkymä avataan, se lisätään pinon päälle, kun taas navigoinnin yhteydessä takaisin liikuttaessa aiemmat näkymät poistetaan pinosta. Tämä navigointimalli oli optimaalinen valinta käsiteltävälle sovellukselle, joka sisältää kolme erillistä näkymää.

Sovelluksen päänavigointilogiikka toteutettiin listamalla kaikki näkymät StackNavigatorin konfiguraatioon. StackNavigator automaattisesti hallitsee ja näyttää aktiivisen näkymän, joka on pinon päällimmäisenä. Interaktiivisuuden ja navigoinnin toteutuksessa hyödynnettiin sovelluksen alatunnisteessa (footer) sijaitsevia ikoneita. TouchableOpacity-komponentin avulla ikoneista tehtiin käyttäjän kanssa vuorovaikutteisia, mahdollistaen niiden käytön navigoinnin hallinnassa. Jokaiselle ikonille määriteltiin yksilöllinen näkymä, johon sovellus siirtyy käyttäjän painaessa kyseistä ikonia.

Tämän lähestymistavan avulla saavutettiin sujuva ja intuitiivinen käyttökokemus, joka tukee sovelluksen keskeistä navigointilogiikkaa. React Navigation -kirjaston ja StackNavigatorin käyttö osoittautui tehokkaaksi ratkaisuksi navigointitarpeiden hallintaan, tarjoten modulaarisen ja joustavan tavan toteuttaa siirtymiä eri näkymien välillä.

4.2.3 Ohjekirjan lisääminen

Toteutusvaiheessa seuraavaksi keskityttiin ohjekirjan integroimiseen sovellukseen. Tätä varten hyödynnettiin aiemmin laadittua ohjekirjan HTML-versiota. Ohjekirjan visuaalinen esittäminen sovelluksessa toteutettiin käyttämällä WebView-komponenttia, joka on standardi ratkaisu web-sisällön näyttämiseen natiivissa sovelluksessa. WebView-komponentin avulla HTML-tiedosto, joka sisältää ohjekirjan kopion, renderöitiin sovelluksen käyttöliittymään.

HTML-tiedoston esitystavan hallinta sovelluksessa saavutettiin WebView-komponentin erilaisten ominaisuuksien avulla. Erityisesti scalesPageToFit-ominaisuus otettiin käyttöön, mikä mahdollisti web-sisällön skaalauksen hallinnan. scalesPageToFit-ominaisuuden asettaminen arvoon false poisti HTML-tiedoston automaattisen skaalauksen, mikä johti parannettuun visuaaliseen esitykseen sovelluksessa.

4.2.4 Otsikoiden linkittäminen

HTML-tiedoston otsikkojen ankkurointiin liittyvän toiminnallisuuden toteutuksessa käytin hyväksi React Nativessa tarjottavaa WebView-komponenttia, joka mahdollistaa web-sisällön integroinnin sovellukseen. Tavoitteenani oli kerätä tiedot HTML-tiedoston otsikoista (ankkureista) ja muuttaa ne interaktiivisiksi elementeiksi sovelluksen käyttöliittymässä. Tähän päämäärään päästäkseni hyödynsin WebView-komponentin injectedJavaScript-ominaisuutta.

injectedJavaScript-ominaisuuden avulla suoritin määrätyn JavaScript-koodin ennen web-sisällön latautumista WebView-komponentissa. Tämä koodi käytti document.querySelectorAll-metodia etsiäkseen kaikki <h1> ja <h2> HTML-elementit, ja muuntamaan ne JavaScript-objektien taulukoksi. Tämä taulukko, nimeltään headers, sisälsi kunkin otsikon tekstin sekä niiden yksilöllisen id-tunnisteen. Käyttämällä JavaScriptin .map-metodia headers-taulukolle, sain luotua listan, joka sisälsi tarvittavat tiedot kustakin otsikosta, mukaan lukien tekstisisällön ja id-tunnisteen.

Koodin suorituksen päätteeksi käytin window.ReactNativeWebView.postMessage-metodia, joka mahdollisti kerättyjen tietojen välittämisen takaisin React Native -sovelluksen puolelle. Tämän mekanismin avulla sain siirrettyä otsikoiden tiedot JavaScript-ympäristöstä React Native -sovelluksen kontekstiin. Tietojen vastaanoton jälkeen sovellus pystyi dynaamisesti luomaan käyttöliittymäelementtejä, kuten painikkeita, jotka vastasivat HTML-tiedoston otsikoihin liittyviä toimintoja.

HTML-tiedoston otsikoiden painikkeet lähettävät aikaisemmin saadut tiedot ohjekirjan näkymään, jotta saataisiin implementoitua ominaisuus, joka mahdollistaa automaattisen skrollauksen tiettyyn dokumentin osioon WebView-komponentissa. Tämä toteutettiin käyttämällä Reactin tilanhallintaa (useState), viitteitä (useRef), sekä sivuvaikutusten hallintaa (useEffect). Tilan ja viitteiden alustus tapahtui useState-hookin avulla, jolla alustettiin muuttuja currentHeaderId, varastoimaan skrollattavan otsikon tunnisteen (ID), ja useRef-hookilla luotiin webViewRef, joka on viite WebView-komponenttiin, mahdollistaen suoran interaktion komponentin kanssa. Reititysparametrien käsittelyssä ensimmäinen useEffect-hookki valvoo route.params-objektia, ja kun selectedHeaderId sisältyy reititysparametreihin, currentHeaderId-tila päivitetään kyseisellä ID:llä. Skrollauksen suoritus perustuu toiseen useEffect-hookkiin, joka reagoi currentHeaderId-tilan muutoksiin, ja kun tila päivittyy ja webViewRef on määritelty, suoritetaan WebView-komponentissa JavaScript-skripti, joka käyttää document.getElementById-metodia halutun elementin löytämiseksi DOM-puusta ja suorittaa scrollIntoView-metodin animoidun skrollauksen

aikaansaamiseksi. WebView komponentin käyttö sisältää ref-viitteen lisäämisen, joka yhdistetään webViewRef-viitteeseen, mahdollistaen JavaScript-koodin suorittamisen komponentissa injectJavaScript-metodin avulla, ja lisäksi WebView-komponenttiin injektoidaan CSS-tyylejä injectedJavaScript-ominaisuuden kautta, tarkoituksena parantaa näytettävän sisällön ulkoasua. Näiden teknisten toteutusten ansiosta sovellus kykenee reagoimaan käyttäjän navigaatioon ja suorittamaan halutun osion skrollauksen WebView-komponentissa dynaamisesti.

5 Sovelluksen julkaisu

5.1 Sovelluksen testaus

Vaatimusten täyttämisen jälkeen toteutettiin sovelluksesta Android-käyttöjärjestelmälle tarkoitettu .apk-tiedosto, joka kapseloi kaikki välttämättömät resurssit Android-sovelluksen asentamista ja suorittamista varten. Tämä pakettitiedosto toimitettiin asiakkaalle arvioitavaksi ja samanaikaisesti jaettiin useammille potentiaalisille käyttäjille testitarkoituksessa.

Saadun palautteen pohjalta tehtiin parannuksia sovelluksen HTML-tiedoston esitykseen, keskittyen erityisesti tiedostonäkymän parantamiseen. Lisäksi tehtiin visuaalisia muutoksia, kuten otsikkoankkureiden tekstitasauksen muuttaminen vasemmalle. Hakutoiminnon implementointi oli yksi saaduista ehdotuksista, ja sen suunnittelua on aloitettu, mutta se päätettiin rajata tämän opinnäytetyön ulkopuolelle ja jättää mahdollisesti myöhemmän kehityksen piiriin.

Tämä prosessi kuvastaa ohjelmistokehityksen iteratiivista luonnetta, jossa sovelluksen kehittäminen ja parantaminen on jatkuvaa, perustuen käyttäjäpalautteeseen ja markkinoiden tarpeisiin. Tällainen lähestymistapa mahdollistaa joustavuuden ja varmistaa, että tuotettu sovellus vastaa loppukäyttäjien vaatimuksiin ja toiveisiin.

5.2 Mobiilisovelluksen julkaisu

Testausvaiheen ja käyttäjäpalautteen keräämisen jälkeen seuraavana askeleena oli mobiilisovelluksen julkaisu Google Playssa ja App Storessa. Julkaisuprosessi vaati Dewi ry:n käyttöoikeuksien saamista molempiin sovelluskauppoihin. Google Play -kaupan osalta tarvittiin yhdistyksen käyttöoikeudet, kun taas App Storessa julkaiseminen edellytti yhdistyksen Apple ID -tunnusten käyttämistä, koska App Storessa ei ole käyttöoikeuksien jakamisen mahdollisuutta kuten Google Playssä.

Julkaisuprosessissa perehdyin Googlen ja Applen asettamiin vaatimuksiin sovelluksen julkaisulle heidän alustoillaan. Nämä vaatimukset sisälsivät sovelluksen kuvauksen, kuvakaappauksia sovelluksesta eri resoluutioilla ja selvityksen sovelluksen tarvitsemista käyttöoikeuksista. Esimerkiksi, jos sovellus käyttää kameraa, on perusteltava, miksi se on tarpeen. Dewi ry:n käsikirjasovellus vaati ainoastaan internet-yhteyden käyttöoikeuden, jotta se voi hakea käsikirjan yhdistyksen Google-tililtä.

6 Yhteenveto

Opinnäytetyön tavoitteeksi asetettiin Dewi Ry:lle suunnattavan mobiilisovelluksen suunnittelu ja toteutus. Sovelluksen primäärinen funktio oli tarjota käyttäjilleen nopea ja vaivaton pääsy sijaiskotitoimintaa koskeviin ajantasaisiin ohjeistuksiin. Projektissa keskityttiin mobiilisovelluksen kehittämiseen siten, että se toimii sijaiskodeille ohjekirjana, jonka sisältöä voidaan jakaa ja päivittää reaaliajassa.

Projektin alkuvaiheessa suoritettiin tarvekartoitus, jossa identifioitiin mobiilisovelluksen kriittiset vaatimukset ja suunniteltiin käyttäjäystävällinen käyttöliittymä. Vaatimusmäärittelyssä korostuivat Android- ja iOS-tuet, ohjekirjan näkyvyys ja päivitysmahdollisuus sekä ohjekirjan otsikoiden linkitys painikkeisiin. Julkaisuun liittyvää kartoitusta tehtiin, mutta sovelluksen julkaisu lykkääntyi ohjekirjan keskeneräisyyden vuoksi.

Lopputuloksena mobiilisovellus saavutti käyttövalmiuden määriteltyyn tarkoitukseensa, huolimatta siitä, että HTML-tekstin esitys ei ole vielä optimaalinen. Opinnäytetyöprosessin aikana havaittiin useita potentiaalisia kehityskohteita, joiden parissa työskentelyä jatketaan projektin päättymisen jälkeen. Kehityskohteina mainittakoon hakutoiminnon lisääminen, alaotsikoiden ja pääotsikoiden parempi integraatio sekä ohjekirjan synkronoinnin tehostaminen sovelluksen sisällä. Näiden kehityskohteiden toteuttaminen on suunniteltu jatkuvan, kunnes sovelluksen ohjekirja on valmis ja sen toiminnallisuus sovelluksen sisällä on varmistettu.

Lähdeluettelo

[1] ISE, ”www.ise.fi,” N.d. [Online]. Available: https://ise.fi/sanastoa/backend/. [Haettu 10. lokakuu 2023].

[2] T. Lahtonen, N.d. [Online]. Available: https://appro.mit.jyu.fi/itkp1011/luennot/cli/#TOC1. [Haettu 28. lokakuu 2023].

[3] 650 Industries Inc., ”Expo,” N.d. [Online]. Available: https://docs.expo.dev/overview/. [Haettu 28. lokakuu 2023].

[4] T. Piispanen, ”Webguru,” 14. lokakuu 2021. [Online]. Available: https://webguru.fi/figma/. [Haettu 28. lokakuu 2023].

[5] Wikipedia, ”Wikipedia,” N.d. [Online]. Available: https://fi.wikipedia.org/wiki/IOS. [Haettu 28. lokakuu 2023].

[6] OpenJS Foundation, ”Node.js,” N.d. [Online]. Available: https://nodejs.org/en/learn. [Haettu 28. lokakuu 2023].

[7] N. Abramowski, ”CareerFoundry,” 28. marraskuu 2022. [Online]. Available: https://careerfoundry.com/en/blog/web-development/what-is-npm/. [Haettu 16. marraskuu 2023].

[8] S. Morris, ”Skillcrush,” N.d. [Online]. Available: https://skillcrush.com/blog/what-is-react-js/. [Haettu 28. lokakuu 2023].

[9] P. Paterska, ”El Passion,” 25. tammikuu 2023. [Online]. Available: https://www.elpassion.com/blog/what-is-react-native-and-when-to-use-it. [Haettu 28. lokakuu 2023].

[10] Dewi ry, ”Dewi Eläinsuojeluyhdistys,” N.d. [Online]. Available: https://www.dewi.info/. [Haettu 28. lokakuu 2023].

[11] A. Lanxon, ”CNET,” 15. elokuu 2023. [Online]. Available: https://www.cnet.com/tech/mobile/why-we-dont-need-new-phone-releases-every-year/. [Haettu 28 lokakuu 2023].

[12] C. Johnson, ”Worship,” 21. heinäkuu 2023. [Online]. Available: https://worship.agency/mobile-screen-sizes-for-2023-based-on-data-from-2022. [Haettu 28. lokakuu 2023].

[13] Figma Inc., ”Figma Learn,” N.d. [Online]. Available: https://help.figma.com/hc/en-us/articles/360040449513-Add-icons-to-text-layers-with-icon-fonts. [Haettu 28. lokakuu 2023].

[14] Meta Platforms Inc., ”React Native,” 29. elokuu 2023. [Online]. Available: https://reactnative.dev/docs/environment-setup. [Haettu 28. lokakuu 2023].

[15] R. Navigation, ”React Navigation,” N.d. [Online]. Available: https://reactnavigation.org/. [Haettu 16. marraskuu 2023].