TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Taavi Paal 184957IADB

Hübriid-mobiilirakendus beebide ja väikelaste une jälgimiseks ja parandamiseks

Bakalaureusetöö

Juhendaja: Meelis Antoi
MSc

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Taavi Paal

17.05.2021

Annotatsioon

Antud bakalaureusetöö eesmärgiks oli luua hübriid-mobiilirakenduse prototüüp, mis aitaks jälgida ning parandada beebide ja väikelaste und. Töö eesmärk on ajendatud asjaolust, et järjest rohkem pöördutakse laste une probleemidega ekspertide poole. Loodud lahendus oleks üheks abinõuks lastevanematele, võimaldades üles märkida lapse unega seotud tegevusi ning seeläbi jälgida kujunenud mustreid.

Töö analüütilises osas viidi läbi küsitlus potentsiaalsete rakenduse kasutajate hulgas, mis oli abiks funktsionaalsete ning mitte-funktsionaalsete nõuete väljaselgitamisel. Lisaks analüüsiti populaarsemaid olemasolevaid analoogilisi lahendusi, selgitati tehnoloogiate valikut ning teostati andmebaasi disain.

Töö rakenduslikus osas loodi rakenduse prototüüp, mis koosneb ASP.NET Core raamistikul arendatud veebiteenusest ning React Native raamistikule arendatud hübriid-mobiilirakendusest. Veebiteenuse eesmärk on kasutaja konto haldamine, andmete varundamine ning klientseadmete vahel andmete sünkroniseerimine.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 6 peatükki, 10 joonist, 2 tabelit.

Abstract

Hybrid Mobile Application for Tracking and Improving the Sleep of Infants and Toddlers

The aim of the current bachelor's thesis was to develop a hybrid mobile application prototype for monitoring and improving sleep of infants and toddlers. The aim of the thesis was inspired by the problem that more and more parents turn to experts for help with child sleep problems. The developed application could help parents to monitor their child's sleep patterns and thus improve their child's sleep schedule.

The analytical part of the thesis consisted of creating a survey amongst potential users of this application. The survey was an input to defining the functional and non-functional requirements. Additionally, the analytical part consisted of analysing the most popular applications currently available, explaining the selection of chosen technological stack and creating the database design.

The practical part consisted of developing the application prototype that consisted of an ASP.NET Core backend application and React Native based hybrid mobile application. The aim of the backend application was to manage user accounts and data retention and synchronisation between devices.

The thesis is in Estonian and contains 31 pages of text, 6 chapters, 10 figures, 2 tables.

Lühendite ja mõistete sõnastik

ACID	*Atomicity, consistency, isolation, durability* – atomaarsus, konsistentsus, isoleeritus, püsivus. Andmebaasihaldurite puhul tehingutöötluse põhikarakteristikud, et tagada töökindlus.
API	*Application Programming Interface* – rakendusliides
CSS	*Cascading Style Sheets* – kaskaadlaadistik
HTML	*Hypertext Markup Language* – hüpertekst-märgistuskeel
HTTP	*Hypertext Transfer Protocol* – hüperteksti edastusprotokoll
HTTPS	*Hypertext Transfer Protocol Secure* – hüperteksti edastusprotokoll üle turvasoklite kihi
IDE	*Integrated development environment* – integreeritud programmeerimiskeskkond
JSON	*JavaScript Object Notation* – JavaScripti objektide notatsioon
JWT	*JSON Web Tokens* – andmevahetuse krüpteerimiseks kasutatav standard
MBaaS	*Mobile backend as a service* – mobiilirakendustele mõeldud pilveteenus, kus tagarakendus on hallatud teenusepakkuja poolt
MVP	*Minimum viable product* – minimaalne jätkusuutlik toode
NoSQL	*Not only SQL* – mitte-relatsiooniline alternatiiv klassikalistele relatsioonilistele andmebaasidele
ORM	*Object-relational mapper* – objekt-relatsioonvastendus. Programmimehhanism, mis võimaldab andmebaasis andmeid otsida ja käidelda.
PWA	*Progressive Web Application* – progressiivne veebirakendus
REST	*Representational state transfer* – esitusoleku edastus, tarkvaraarhitektuuri laad.
SQL	*Structured Query Language* – struktuuripäringukeel, mida kasutatakse relatsiooniliste andmebaaside päringute tegemiseks
UI	*User Interface* – kasutajaliides
URL	*Uniform Resource Locator* – internetiaadress ehk üldine infoallika asukohamääraja
UWP	*Universal Windows Platform* – universaalne Windows platvorm

XAML	*Extensible Application Markup Language* – laiendatav rakenduse märgistuskeel
XML	*Extensible Markup Language* – laiendatav märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Beebide ja väikelaste uneprobleemid on aina sagenev nähtus, millega lapsevanemad spetsialistidelt abi otsivad. 2020. aasta algusest Eestis tööd alustanud imikute ja väikelaste unenõustamisteenust pakkuv Unehaldjas [1] on aastaga kasvanud kolme nõustajaga meeskonnaks, kuid sellegipoolest on konsultatsioonidele järjekorrad ning abisoovijaid oluliselt rohkem kui suudetakse ära teenindada. Samuti on ääretult populaarseks osutunud Unehaldja veebiseminarid erinevas vanuses laste vanematele. Teaduspõhise ja tõendatud uneabi järele on Eestis nii vajadus kui nõudlus, kuid kuna teenus ei ole kaetud Haigekassa rahastusega, on see paljude vanemate jaoks liiga kulukas. Seetõttu oleks tarvis luua rakendus, millel oleks nii hariv kui praktilist abi pakkuv funktsioon ning mis oleks taskukohase hinna tõttu kättesaadav kõikidele soovijatele.

Antud töö eesmärgiks on kirjeldada lahendatava probleemi olemust Eesti kontekstis ning selgitada välja loodava lahenduse funktsionaalsed ning mittefunktsionaalsed nõuded. Lisaks võrreldakse antud töös väljapakutud lahendust juba olemasolevate analoogsete mobiilirakendustega, analüüsitakse probleemi lahenduseks kasutatavate tehnoloogiate valikut ning kirjeldatakse loodud lahendust.

Käesolevas töös välja pakutud hübriid-mobiilirakendus oleks suureks abiks lapsevanematele, kelle lastel esineb raskusi unega, võimaldades üksikasjalikult üles märkida lapse une ja päevakavaga seotud sündmusi ning saada kohest tagasisidet võimalikest probleemidest ning soovitusi nende probleemide lahendamiseks.

2 Ülevaade probleemist

Laste unealaste teadusuuringute kohaselt esineb ligi kolmandikul kuni kolmeaastastest lastest unehäireid [2], kusjuures käitumuslikud unehäired on oluliselt levinumad kui ebanormaalse polüsomnograafiaga unehäired (nt parasomniad, uneapnoe, narkolepsia) [3]. See omakorda tähendab, et suur osa lapse esimeste eluaastate jooksul esinevatest unehäiretest on ennetatavad või korrigeeritavad käitumuslike sekkumistega. Lisaks sellele on uuringud näidanud, et imiku- ja väikelapseea uneprobleemid on seotud hilisema ülekaalulisuse, käitumis- ja tähelepanuhäirete ning emotsioonide regulatsiooni probleemidega [4]. Imiku uneprobleemid on ka üks olulisemaid sünnitusjärgse depressiooni riskitegureid [5]. Seega on uneprobleemide ennetamine ja lahendamine imiku- ja väikelapseeas oluline nii lapse kui vanema seisukohalt.

Uuringud ja unenõustajate praktika on näidanud, et imikuea uneprobleeme on võimalik hariduslike meetmetega efektiivselt ennetada [6]. Selleks vajavad lapsevanemad usaldusväärset teaduslikku infot lapse unevajaduse, une arengu ning une parandamise seisukohalt efektiivsete sekkumiste kohta. Kuigi unealane info on internetis aina kergemini kättesaadav, leidub palju vastukäivat teavet ning infomüras orienteerumine on vanemate jaoks keeruline ja aeganõudev. Individuaalne nõustamine on aga uneprobleemide ennetuse seisukohalt liiga ressursimahukas ning akuutsete uneprobleemide lahendamisel küll väga efektiivne, kuid kallis ning Eesti kontekstis unenõustajate puuduse tõttu ka raskesti kättesaadav.

Siinkohal võiks loodav lahendus olla uneprobleeme ennetav meede. Lahendus oleks kättesaadav piiramatule hulgale inimestele ning võimaldaks efektiivselt uneprobleeme ennetada vähese aja- ja rahakuluga.

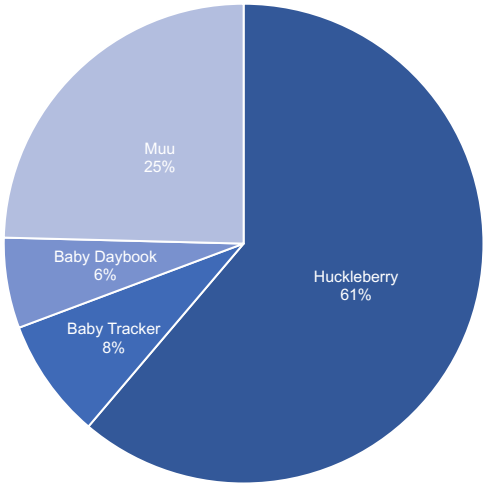
Unehaldja poolt välja töötatud „Rahuliku Une Raamistik“ võtab kokku teadusel põhinevad und mõjutavad tegurid. Nendest kolm on muutumatud tegurid – bioloogia (kuidas uni kehas toimib), areng (millises arenguetapis laps hetkel on ning kuidas see tema und mõjutab) ja kiindumus/kasvatus (millised on vanema hoiakud une osas ning kasvatusmeetodid) ning kolm muudetavat tegurit – unekeskkond (millises keskkonnas

laps magab), päevarütm (milline on lapse eakohane päeva- ja ööune vajadus, optimaalne toitumine) ning uneassotsiatsioonid (kuidas on laps harjunud uinuma, millised on käitumuslikud mustrid seoses unega).

Une jälgimise rakendus aitaks vanematel optimeerida kõiki kolme muudetavat tegurit ning annaks vanematele infot muutumatute tegurite kohta.

2.1 Eksisteerivad lahendused

Unehaldja klientide seas läbiviidud uuringust (Lisa 2) selgus, et varasemalt mõnda une jälgimise rakendust kasutanud klientide hulgas osutusid kõige populaarsemateks Huckleberry (61,2% vastanutest) [7], Baby Tracker (8,1%) [8] ning Baby Daybook (6,1%) [9] (Joonis 1). Kui Huckleberry on peamiselt keskendunud laste une jälgimisele ja parandamisele, andes kasutajatele perioodilist tagasisidet, siis viimased kaks lahendust on mõeldud üleüldiseks lapsega seotud toimingute (näiteks magamine, toitmine, kasvamine, mängimine, arsti juures käimine jne) üles märkimiseks kuid ei anna kasutajale sisestatud andmete põhjal tagasisidet.



Joonis 1. Populaarsemad alternatiivsed lahendused Unehaldja klientide seas.

2.1.1 Huckleberry

Huckleberry [7] on ülekaalukalt kõige populaarsem laste une jälgimise mobiilirakendus Unehaldja klientide hulgas. Rakenduse põhiliseks eesmärgiks on une aegade ülesmärkimine, kuid lisaks võimaldab Huckleberry üles märkida ka muid tegevusi, nt

söötmine, ravimite tarvitamine, lapse kasvamine, kraadimine jne. Huckleberry unikaalne funktsionaalsus on *Sweetspot*, mis võimaldab ennustada parimaid lapse uinaku aegasid võttes arvesse eelnevalt sisestatud une andmeid. Rakenduses on ka mitmeid tasulisi funktsionaalsusi, nt detailsem *Sweetspoti* seadistamine, lapse päevakava koostamine ning individuaalne nõustamine.

Unehaldaja kliendid tõid välja, et Huckleberry puuduseks on see, et sellel on liiga palju funktsionaalsust, mis hajutab kasutaja fookust ning lisaks suurendab õpikõverat. Lisaks pakub Huckleberry individuaalset tagasisidet, mis on aga kallis ning nõuab iga kord nõustaja sekkumist. Selleks, et saada korduvat tagasisidet tuleb maksta üsna suurt kuumaksu.

Huckleberry on loodavale lahendusele kõige sarnasem ning rakenduse analüüs andis palju mõtteid kuidas loodava lahenduse kasutajakogemust optimeerida.

2.1.2 Baby Tracker

Baby Tracker [8] on lihtne ja minimalistlik laste toitmise, une ja muude toimingute ülesmärkimise rakendus. Rakendus võimaldab ajaliselt toiminguid üles märkida ning vaadata ülesmärgitud sündmuste kohta statistikat ning kalendrivaatest kõikide toimingute ülevaadet. Baby Tracker ei anna kasutajale toimingute kohta tagasisidet vaid eeldab, et kasutaja teeb kalendrivaate ja statistika põhjal ise järeldused oma lapse une seisundist. Baby Tracker on küll tasuta, kuid sisaldab reklaame, mida kasutajale iga vaate juures kuvatakse. Rakendus ei võimalda kasutaja konto loomist ning seetõttu puudub ka andmete sünkroniseerimise võimalus mitme seadme vahel. Siiski on võimalus oma sisestatud andmed eksportida ning hiljem teises seadmes importida. Rakendusel puudub ka tume värvirežiim, mis teeb rakenduse kasutamise öösel ebamugavaks.

2.1.3 Baby Daybook

Baby Daybook [9] on oma sisu poolest sarnane Baby Trackerile, kuid võimaldab üles märkida rohkem erinevaid tegevusi, näiteks vannitamine, arstil käimine, mängimine, õues jalutamine jne. Baby Daybook võimaldab vaadata iga tegevuse kohta eraldi statistikat ja tegevuse ajalist graafikut. Rakenduse puuduseks on kalendrivaade, mis on väikese kirjasuurusega ja sümbolitega ning ei võimalda suurendamist, mis võib olla takistuseks halvema nägemisega inimestele. Baby Daybookil on olemas ka eestikeelne kasutajaliides, kuid kohati on terminid halvasti tõlgitud või puudub tõlge üldse, mis tähendab, et kasutaja

peab kogu funktsionaalsuse kasutamiseks siiski inglise keelt oskama. Rakenduse põhifunktsionaalsus on tasuta ning vaated sisaldavad reklaame, kuid reklaamide eemaldamiseks ning sisse kantud tegevustest detailsema ülevaate saamiseks tuleb kasutada tasulist rakendust. Baby Daybookile unikaalseks funktsionaalsuseks on võimalus kontole lisada pereliikmed, mis võimaldab mitmelt seadmelt korraga andmeid sisestada. Lisaks on rakendusel olemas ühilduvus Google Assistantiga, mis võimaldab kasutada häälkäsklusi tegevuste sisestamiseks.

2.2 Esitatava lahenduse skoop

Antud hetkel puudub eestikeelne väikelaste ja imikute une jälgimise nutirakendus. Lisaks sellele puudub ka rakendus, mis annaks lapsevanematele kohest tagasisidet lapse une kohta ning soovitusi une parandamiseks. Seetõttu on loodav lahendus unikaalne.

Olemasolevatel lahendustel on küll olemas uneaegade ülesmärkimise funktsionaalsus, kuid selle kohta ei saa kasutaja mingit tagasisidet või siis on tagasiside kallis ning nõuab individuaalset lähenemist rakenduse-väliselt isikult.

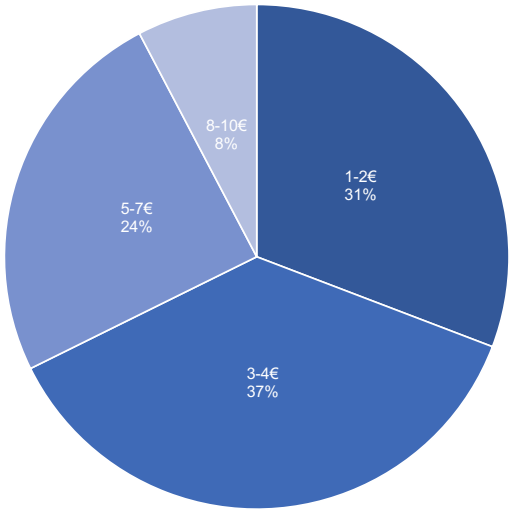
Loodava lahendusena arendatakse nutirakendus, mis oleks eestikeelne ning annaks kasutajale teaduspõhist tagasisidet lapse une parandamiseks. Loodav lahendus piirduks ainult une parandamiseks vajaliku informatsiooniga ning väldiks kõrvalisi lisafunktsionaalsusi, mis tihti lisavad kasutajakogemuse keerukust. Lisaks oleks lahendus kasutatav mitmel seadmel korraga, mis võimaldaks mõlemal vanemal andmeid sisestada.

Tulevikus on plaanis rakendusse lisada ka lapse soovituslik päevakava, mis võimaldaks kasutajal paremini planeerida lapse päevarežiimi. Nimetatud funktsionaalsus annaks kasutajale ülevaate, milline peaks olema antud vanuses lapse ideaalne päevakava ning võimaluse korral oleks kasutajal võimalik päevakava sündmuste kohta saada teavitusi.

Antud lõputöö kontekstis valmib prototüüp rakendusest, mille komponentideks on kliendipoolne hübriid-mobiilirakendus ning teenusepoolne rakendus, mis tagab kasutaja andmete varundamise ja sünkroniseerimise klientrakendusega.

2.3 Ärilise tasuvuse analüüs

Rakenduse loomise üheks kriteeriumiks on ka ettevõttele kompenseerida rakenduse arendamise, hooldamise ning majutusega seotud kulud ning võimaluse korral ka lisatulu teenimine, mis võimaldaks tulevikus investeerida parema nõustamiskvaliteedi loomisele. Selle tarvis viidi Unehaldja klientide seas läbi küsitlus (Lisa 2), mille üheks osaks oli ka uurida klientide valmisolekut loodava lahenduse eest maksta. Küsitlusest selgus, et tasulise rakenduse kasutamisega oleks nõus 80,5% vastanutest. Rahaliselt oleks 30,8% nõus maksma 1-2 € kuus, 36,9% 3-4 € kuus, 24,6% 5-7 € kuus ning 7,7% 8-10 € kuus (Joonis 2). Lisaks oleks 85,7% vastanutest nõus toetama antud lahenduse loomist ühisrahastusplatvormil.



Joonis 2. Igakuine potentsiaalne maksevalmidus loodava lahenduse eest küsitlusele vastanute hulgas.

Statistikaameti andmetel sünnib Eestis ligi 14000 last aastas. Arvestades Unehaldja kui kaubamärgi usaldusväärsuse kasvu vanemate seas (2020. aasta alguses tegutsemist alustanud ettevõte on aastaga kasvatanud Eesti-sisese käibe pea 50000 €-ni), sellist tüüpi rakenduste üleüldist kasvavat populaarsust sihtgrupis ning otsese eestikeelse konkurentsi puudumist, on võimalik prognoosida, et igakuiselt hakkab rakendust kasutama 40% lapse saanud vanematest, nendest tasulise versiooni peale lülituvad hinnanguliselt 50% uutest kasutajatest (võttes arvesse turu-uuringus väljendatud suurt valmidust rakenduse kasutamise eest maksta). Seega võiks igakuiselt liituda keskmiselt 233 uut tasulise versiooni kasutajat. Kui neist igaüks kasutab rakenduse tasulist versiooni keskmiselt 6 kuud, on rakenduse aastakäive hinnanguliselt 50 232€ (kuumakse 2,99€ kasutaja kohta).

3 Loodava rakenduse analüüs

3.1 Nõuete määramine

Funktsionaalsete ning mittefunktsionaalsete nõuete väljaselgitamiseks viidi läbi intervjuu Unehaldja peanõustajaga ning lisaks ka küsitlus Unehaldja klientide hulgas Google Forms abil (Lisa 2). Küsimustikule vastas 77 inimest, kellest 71,4% on üks laps, 23,4% on kaks last ning 5,2% vastanutest on 3 last. Ühelgi vastanutest ei olnud rohkem kui kolm last. Keskmine alla aastase lapse vanus vastanute hulgas on 6,5 kuud. Kõigist vastanutest oli 63,6% varem juba lapse une jälgimisega seotud mobiilirakendust kasutanud, millest populaarsemad on Huckleberry (61,2%) ning Baby Tracker (16,3%). Lisaks arvas 63,3% mobiilirakendust kasutanud vastanutest, et neil oli sellest rakendusest kasu. Kõige enam toodi välja, et varem kasutatud rakenduste puhul oli kõige rohkem kasu lapse une aegade märkimisest ning sellega seotud statistika nägemisest, sobivate uneaegade soovitamisest ning kalendrivaatest kuhu on kellaajaliselt kuvatud kõik ülesmärgitud tegevused. Negatiivsetest aspektidest toodi välja, et on liiga palju üleliigset funktsionaalsust ning kohati peab liiga palju klikke tegema, et soovitud tulemuseni jõuda.

Antud lahenduse kontekstis on üks persoona – tavakasutaja. Tavakasutaja on isik, kes kasutab lahendust oma lapse või laste uneaegade märkimiseks.

Nii intervjuu kui ka küsitluse järgselt loodi nimekiri loodava lahenduse funktsionaalsetest ja mittefunktsionaalsetest nõuetest. Siiski on mitmed planeeritavad funktsionaalsused jäetud antud töö skoobist välja. Antud töö lahendus kujutab endast prototüüpi, mis sisaldab endas ainult kõige olulisemaid funktsionaalsusi.

3.1.1 Funktsionaalsed nõuded

Loodavale lahendusele seatud funktsionaalsed nõuded on järgmised:

- Tavakasutajana saan kasutajat registreerida e-posti aadressi abil;
- Tavakasutajana saan sisse ja välja logida;
- Tavakasutajana saan sisestada lapse andmeid;
- Tavakasutajana saan sisestada mitut last;

- Tavakasutajana saan lapse andmeid muuta;
- Tavakasutajana saan last rakendusest kustutada koos kõigi seni kogutud andmetega;
- Tavakasutajana saan tõuketeavitused sisse/välja lülitada;
- Tavakasutajana saan sisestada uut sündmust kalendrivaatest;
- Tavakasutajana saan olemasolevaid sündmusi muuta ja kustutada;
- Tavakasutajana saan kalendris vaadata päeva ja nädalavaadet;
- Tavakasutajana saan vaadata sündmuste statistikat viimase kahe nädala lõikes.

3.1.2 Mittefunktsionaalsed nõuded

Loodavale lahendusele seatud mittefunktsionaalsed nõuded on järgmised:

- Tavakasutajana soovin, et rakendus vastab Eesti „Isikuandmete kaitse seadusele“ [10] ning Euroopa Liidu „Isikuandmete kaitse üldmäärusele“ [11];
- Tavakasutajana olen võimeline rakendust kasutama ka värvipimeduse korral;
- Tavakasutajana soovin rakendusest aru saada kui räägin vaid eesti keelt;
- Tavakasutajana soovin rakendust kasutada ka siis kui internetiühendus puudub (siiski on esmasel registreerimisel internetiühendus vajalik);
- Tavakasutajana soovin, et ei peaks rakendust avades iga kord uuesti sisse logima;
- Tavakasutajana soovin, et mul oleks võimalus rakendust kasutada mitmel seadmel korraga;
- Tavakasutajana soovin, et minu andmed oleksid varundatud juhuks kui minu seadmega midagi juhtub;
- Tavakasutajana soovin, et rakendus oleks kättesaadav nii Android kui ka iOS operatsioonisüsteemis.

3.2 Tehnoloogiate valik

Arendamistehnoloogiate valikul on lähtutud mittefunktsionaalsetest nõuetest, mis määravad, et rakendus peab olema kasutatav nii Android kui ka iOS operatsioonisüsteemidel ning rakendust saab kasutada mitmel seadmel korraga ehk andmeid on vaja sünkroniseerida mitme seadme vahel. Lisaks on oluline nõue ka see, et rakendus oleks kasutatav internetiühenduse puudumise korral.

Antud nõudeid arvesse võttes peab loodaval lahendusel olema klientrakendus, mis täidab tehnoloogiapinus andmete sisestamise rolli ning teenusrakendus, mis täidab andmete sünkroniseerimise ja varundamise rolli.

3.2.1 Teenusepoolne lahendus

Teenusepoolse lahenduse loomiseks on kaks võimalust – kasutada MBaaS pilvelahendust või luua teenusrakendus ise käsitsi. Esimese puhul on eeliseks see, et kogu teenusepoolne infrastruktuur on hallatud teenusepakkuja poolt ning kasutaja lahendada on vaid teenuse seadistamine. Lisaks pakuvad MBaaS lahendused tavaliselt ka kasutajate haldamist, automaatseid tõuketeavitusi, e-posti teavitusi jne. Suurem enamus MBaaS platvorme pakuvad andmete hoiustamiseks NoSQL andmebaasi, mis raskendab keerukama andmemudeli kasutamist. Tuntumad MBaaS lahendust pakkuvad platvormid on Firebase [12], AWS Amplify [13] ning Azure Mobile Apps [14]. MBaaS platvormi kasutamise puuduseks on suur sõltuvus platvormil pakutavatest lahendustest ning hiljem platvormi vahetamine võib osutuda väga kulukaks. Lisaks võib viga MBaaS lahenduste kasutamisel kaasa tuua suured kulud [15].

Teenusepoolset lahendust ise luues on võimalik detailideni kogu funktsionaalsus seadistada ning ei pea sõltuma kolmandatest osapooltest. Lisaks on rakendust võimalik integreerida väliste süsteemidega ning jooksvalt süsteemi uuendada. Ise loodud teenusrakendust on vajadusel kergem ka ühe majutamisteenuse alt teise alla viia.

Loodava lahenduse kontekstis otsustati luua teenusrakendus käsitsi, kuna MBaaS teenuste eelanalüüsil selgus, et antud lahendused toetavad vaid NoSQL andmebaasisüsteemi, mis loodava lahenduse kontekstis ei ole sobiv. Lisaks ei ole töö autoril varasemat kogemust NoSQL andmebaasidega.

Teenusepoolse lahendusena kasutatakse Microsoft ASP.NET Core 5 raamistikku [16], mis on platvormist sõltumatu vabavaraline avatud lähtekoodiga teenusraamistik ning mõeldud veebirakenduste loomiseks. ASP.NET Core raamistik võimaldab luua nii teenusepoolset kui ka kliendipoolset kihti, kuid antud töö raames kasutatakse raamistikku andmete vahetamiseks kliendi ning teenuse vahel [17]. Antud raamistik valiti põhjusel, et autoril on eelnev tööalane kogemus antud raamistiku kasutamisel.

Antud töö käigus luuakse ASP.NET Core raamistikus REST API, mis võimaldab sünkroniseerida andmeid klientrakenduse ja teenusrakenduse vahel. Teenusrakendus liidestatakse andmebaasiga kasutades objekt-relatsioonvastendus (ORM) süsteemi Entity Framework Core [18], mis võimaldab automaatset andmebaasi tabelite ja teenusraamistiku objektide vahelist vastendamist, luues andmebaasi ja teenuserakenduse vahele abstraktsiooni kihi [19].

3.2.2 Klientrakenduse tehnoloogia

Nutiseadmetele rakenduse loomiseks on mitu alternatiivi, nt platvormipõhine (*native*) rakendus, hübriid-mobiilirakendus, progressiivne veebirakendus (*PWA*) või adaptiivne veebileht [20]. Platvormipõhised rakendused pakuvad kõige sujuvamat kasutajakogemust ning ligipääsu kõigile seadme riistvaralistele komponentidele [21]. Antud töö lahenduse kontekstis tähendaks see kahe erineva lahenduse loomist paralleelselt, üks Androidi ning teine iOS seadmetele. See omakorda tähendab, et tuleb hallata kahte erinevat koodibaasi, tunda mitut erinevat programmeerimiskeelt ja arendamistehnoloogiat ning seada üles mitu pidevintegratsiooni lahendust. Progressiivsed veebirakendused võimaldavad tavaliste veebitehnoloogiate (HTML, CSS, JavaScript) abil luua rakendusi, mis töötavad nii tavalises veebilehitsejas kui ka nutiseadmetel. Progressiivsete veebirakenduste eeliseks on nende arendamise lihtsus ning sõltumatus platvormi iseärasustest. Antud rakenduste puuduseks on küllaltki suur aku kasutatavus, vähene ligipääs platvormipõhistele liidestele ning ka jõudlus jääb alla platvormipõhistele või hübriid-mobiilirakendustele [20]. Lisaks ei ole võimalik progressiivseid veebirakendusi lisada Androidi või iOS platvormide vastavatesse rakenduste poodidesse [22], mistõttu ei pruugi kasutajad neid üles leida. Progressiivsete veebirakenduste puudused on ka adaptiivsetel veebilehtedel. Lisaks ei võimalda adaptiivsed veebilehed ligipääsu platvormipõhistele funktsionaalsustele ning nende arendamine võib olla isegi keerulisem [20]. Hübriid-mobiilirakenduste puhul ühendatakse platvormipõhised ning veebitehnoloogiad [23].

Hübriid-mobiilirakenduste puhul saab korraga arendada mitmele platvormile kasutades sama programmeerimiskeelt ning jagada koodibaasi, mis vähendab arendamisele kuluvat aega. Olemas on ligipääs platvormispetsiifilistele funktsionaalsustele, mida saab eraldiseisvalt seadistada [20].

Kuna antud töö raames on vajadus arendada korraga nii Androidi kui ka iOS seadmetele ning rakendust peab olema võimalik alla laadida vastavate platvormide poodides, siis kõige sobivam on lahendus luua hübriid-mobiilirakendusena.

Interneti suurima arendajate kogukonna StackOverflow poolt läbi viidud küsitlusest selgub, et professionaalsete arendajate seas enim kasutatavad hübriid-mobiilirakenduste raamistikud on React Native, Flutter ning Xamarin [24]. Raamistike võrdluseks on välja valitud mõningad kriteeriumid, mis on olulised arendamise seisukohast (Tabel 1).

Tabel 1. Hübriid-mobiilirakenduste raamistike valimise kriteeriumid.

Kriteerium	Olulisus
Programmeerimiskeel	Arendaja kogemus antud keeles lihtsustab raamistiku õppimist ning kiirendab arendusprotsessi
Arenduskeskkonna seadistamise keerukus	Kui kiiresti on võimalik arendamisega alustada ning uusi arendajaid projekti kaasata
Produktiivsus	Head arendustööriistad aitavad kiiremini arendada ning tekib potentsiaalselt vähem vigu
Kogukonna suurus ja dokumentatsioon	Aitab suurema tõenäosusega leida lahendusi arendamisel ettetulevatele probleemidele ning kiirendab arendusprotsessi
Testimise lihtsus	Aitab vältida vigu
Rakenduse väljalaske suurus	Rakenduse suurus mõjutab otseselt installeerimiste arvu [25]
Pidevintegratsioon/pidevvalmidus	Lihtsustab uute versioonide väljalaset
Maksumus	Kallid ülalhoidmiskulud ei pruugi ennast majanduslikult ära tasuda

React Native on Facebooki poolt loodud ja 2015 aastal avaldatud avatud lähtekoodiga mobiilirakenduste arendamise raamistik [26]. React Native võimaldab arendada

rakendusi nii Android, iOS, macOS, UWP, kui ka mitmele teisele platvormile [27]. Programmeerimiskeelena on kasutusel JavaScript ning komponentide loomiseks kasutatakse XML-il põhinevat JSX süntaksit. Lisaks on kasutusel CSS-iga sarnanev *StyleSheet* süsteem, millega kasutajaliidest kujundatakse. JavaScripti asemel on võimalik kasutada ka tugevalt tüübitud TypeScript keelt. React Native arenduskeskkonda on võimalik üles seada suhteliselt kiirelt, eeldades, et arendamiseks kasutatakse füüsilisi seadmeid. Kui soovitakse kasutada Android emulaatorit on vaja lisaks paigaldada Android Studio ning iOS simulaatori kasutamiseks on vajalik macOS operatsioonisüsteem ning XCode [28]. Koodi kirjutamiseks sobivad näiteks vabavaraline Visual Studio Code või tasuline WebStorm. Mõlemal programmil on olemas spetsiaalne React Native-i tugi, mis lihtsustab ja kiirendab arendamist. React Native on antud hetkel kõige populaarsem hübriid-mobiilirakenduste loomise raamistik [24], mistõttu leidub rohkesti nii ametlikku kui ka mitteametlikku dokumentatsiooni. Testimine on React Native puhul võimalik nii ühik, integratsiooni kui ka kasutajaliidese komponentide testidena, k.a. *snapshot* testid, mis testivad kasutajaliidese komponentide korrektset paigutust ekraanil [29]. Lisaks on võimalik läbivtestimine, mille kaudu saab imiteerida kasutaja liikumist läbi rakenduse (nt nuppude vajutamised, teksti sisestamine jne). Nagu kõikide hübriid-mobiilirakenduste puhul, on ka React Native lõpliku rakenduse maht suhteliselt arvestatav. „*Hello, World!*“ rakenduse puhul ~23MB [30], kuid teatud tehnikaid kasutades on võimalik rakenduse suurust märgatavalt vähendada [31]. Pidev integratsioon koos pidevvalmidusega on võimalik üles seada kasutades spetsiaalseid tasulisi teenuseid nt Visual Studio App Center, Codemagic või Bitrise, mis võimaldavad rakenduste jagamist testgruppidega või saatmist otse Androidi või iOS platvormipõhistesse rakenduste poodidesse. Tasuta pidev integratsioon on võimalik üles seada ka GitHub Actions abil.

Flutter on Google poolt loodud vaba lähtekoodiga platvormiülese rakenduse arendamise platvorm, mida kasutatakse Android, iOS, Linux, Macintosh, Windows ning Google Fuchsia rakenduste ning nii klassikaliste kui ka progressiivsete veebirakenduste arenduseks. Flutteri esimene stabiilne versioon (Flutter 1.0) ilmus alles aastal 2018. Flutteri arendus toimub programmeerimiskeeles Dart, mis on samuti Google poolt loodud. Flutteri platvormil arendamine ei nõua spetsiaalset integreeritud arenduskeskkonda (IDE) ning kasutada on võimalik nt Android Studio, IntelliJ IDEA või Visual Studio Code. Paigaldamine on detailselt kirjutatud lahti dokumentatsioonis [32],

mille järgi on kerge arenduskeskkonda üles seada. Flutteri puuduseks võib pidada tema uudsust ning selle tõttu ka veel võrdlemisi väikest kogukonda. Seega ei pruugi leida piisavalt dokumentatsiooni, et lahendada arendamisel ette tulevaid keerukamaid platvormispetsiifilisi probleeme. Flutter toetab kõiki testimisfaase (ühik-, integratsiooni- ning kasutajaliidese testid). Flutteri testimise omapäraks on see, et on võimalik luua UI komponentide kohta kasutajaliidese teste, mille käitamise kiirus on võrreldav ühiktestidega. Toodanguvalmis rakenduse maht „*Hello, World!*“ rakenduse puhul on ~15MB, mida on võrreldes platvormipõhiste rakendustega üpris palju. Flutteri pidev integratsioon koos rakenduse pidevvalmidusega on toetatud mitme tasulise veebiteenuse poolt, nt Codemagic, Bitrise või Appcircle. Pidev integratsioon on võimalik üles seada ka GitHub Actions abil, kuid antud teenusel puudub pidevvalmidus, mille saab juurde lisada kasutades tööriista fastlane [33].

Xamarin on Microsofti poolt arendatud avatud lähtekoodiga platvorm, mis võimaldab luua rakendusi nii Androidi, iOS, macOS kui ka UWP seadmetele [34]. Programmeerimiskeeleks on C# ning kasutajaliidese disain toimub peamiselt läbi XAML failide, mis baseeruvad XML struktuuril. Saab kasutada teekide haldamise platvormi NuGet. On olemas võimalus kontrollida eraldi Androidi ning iOS rakenduse kasutajaliidese kihte, et saavutada täielik platvormipõhine kogemus. Täiesti platvormiülese lahenduse loomiseks kasutatakse alamraamistikku Xamarin.Forms, mis lisab projektile lisa abstraktsioonikihi [35]. Xamarin.Forms võimaldab kasutada ühist koodi nii Androidi kui ka iOS rakenduse puhul, kuid platvormispetsiifiliste funktsionaalsuste kasutamiseks tuleb teha eraldi muudatusi vastava platvormi kasutajaliidese kihis. Xamarin arendus vajab arenduskeskkonda Visual Studio, kuid mõninga hulga vähemate võimalustega saab arenduskeskkonnana kasutada ka JetBrains Rider-it või Visual Studio Code-i. Viimase seadistamiseks Xamariniga on aga vähe dokumentatsiooni. Kasutades Visual Studiot on Xamarini komponendid võimalik paigaldada kerge vaevaga. Probleemi võib põhjustada asjaolu, et Windowsi operatsioonisüsteemis arendamisel on iOS simulaatori kasutamiseks vaja ühendada Visual Studio macOS seadmega [36]. Lõplik rakenduse maht Androidile loodud rakenduse puhul on „*Hello, World!*“ projekti korral ~15MB [37]. Lisaks on hetkel toimumas Xamarini platvormi ühildamine Microsoft .NET 6 raamistikuga, mille käigus Xamarin.Forms asendub MAUI (Multi-platform App UI) raamistikuga [38]. Kuna Xamarin on juba suhteliselt vana platvorm (aastast 2011), on ka arendajate kogukond suur

ning dokumentatsiooni leiab rohkesti. Testimist saab läbi viia kasutades standardseid .NET raamistiku lahendusi ning olemas on ka spetsiaalsed Xamarini-põhised testimise liidesed (nt Xamarin.UITest [39]). Xamarin rakenduse pideva integratsiooni jaoks on Microsofti poolt loodud ühilduvus tasuliste Azure Pipelines ning Visual Studio App Center teenustega. Lisaks on toetatud Jenkins ning TeamCity pideva integratsiooni teenused [40].

Kõik analüüsitud raamistikud on avatud lähtekoodiga ning tasuta kasutatavad. Xamarini kasutamisel on peidetud kulu Visual Studio näol, kuna üle viieliikmelise arendustiimi korral tuleb soetada Visual Studio litsents [41]. Kõikide raamistike korral on kõige suurem potentsiaalne kulu pideva integratsiooni ning pidevvalmiduse saavutamisel. Võimalik on rakendusi ka arendusmasinas testida, kompileerida ning manuaalselt vastavatesse platvormipõhistesse poodidesse lisada, kuid see on ajamahukas ning võib põhjustada vigu.

Arvestades autori kogemust raamistikes kasutatavates programmeerimiskeeltes, siis tuleb välistada Flutteri kasutamine, kuna uue keele õppimine nõuab aega ning autoril on juba eelnev kogemus nii JavaScripti kui ka C# keelega. Xamarini puhul tuleb arvestades seda, et mõne aasta pärast lõpetatakse raamistiku toetamine Microsofti poolt MAUI kasuks [38]. Seega tuleks arvestada sellega, et loodav platvorm on tulevikus vaja uue raamistiku peale üle viia, mis eeldab lisaaega ja kulutusi. Kõigest eelnevast tulenevalt kasutatakse antud töö lahenduses React Native raamistikku.

3.2.3 Andmebaasi valik

Andmebaasi valik sõltub eelkõike salvestatavate andmete olemusest ning sellest, mida nende andmetega edasi soovitakse teha. Antud lahenduse kontekstis on oluline ka see, et andmebaasi tarkvara oleks tasuta. Eraldi on vaja analüüsida klientrakenduses ning veebiteenuses kasutatava andmebaasi valikut.

Loodava lahenduse puhul on tegemist relatsiooniliste andmetega, mis tähendab, et andmed on kindla struktuuriga. Ka NoSQL andmebaasides on võimalik kindla struktuuriga andmeid hoida, kuid mitte-relatsioonilised andmebaasid ei toeta SQL päringuid, mistõttu tuleb dokumentide-vahelised seosed käsitsi programmeerida [42]. NoSQL andmebaasid on mõeldud pigem struktureerimata andmete jaoks sellistes

rakendustes, mis vajavad väga kiiret reageerimisaega, näiteks sõnumivahetuse rakendused, või on tegemist väga mahukate andmetega [43].

Lähtudes sellest, et loodavas lahenduses kasutatakse kindla struktuuriga andmeid ning töö autor on juba tuttav SQL päringukeelega, siis eelistati loodavas lahenduses kasutada relatsioonilisi andmebaase.

Üks rakenduse loomise mittefunktsionaalne nõue oli, et klientrakendus peab töötama ka siis kui võrguühendus puudub. See eeldab andmete esmast salvestamist klientrakenduses ehk kasutaja mobiiliseadmel. Kõige sobivam lahendus selleks on SQLite andmebaasimootor, mis on sisse ehitatud kõikidesse Android ning iOS operatsioonisüsteemidesse [44]. SQLite on vabavaraline andmebaas, mis ei vaja seadistamist ega administreerimist [45], mis teeb sellega töötamise väga lihtsaks. Lisaks on SQLite mahult tagasihoidlik ning kasutab töö käigus vaid ~200kB mälu [45].

Sünkroniseerimise lihtsustamiseks kasutatakse SQLite andmebaasil põhinevat vabavaralist WatermelonDB andmebaasi raamistikku, mis on spetsiaalselt mõeldud React ning React Native rakenduste andmebaaside haldamiseks [46]. WatermelonDB raamistikku loomisel on juba algselt andmete sünkroniseerimisele mõeldud, mis lihtsustab oluliselt sünkroniseerimise funktsionaalsuse realiseerimist loodavas lahenduses.

Veebiteenuse andmebaasi valikul on lähtutud põhimõttest, et andmebaas peab olema vabavaraline, kuna tasulise andmebaasi litsentsi hind ei pruugi olla vastavuses rakenduse prognoositava sissetulekuga. Antud juhul on analüüsi võetud kolm vabavaralist andmebaasi, milleks on SQLite, MySQL ning PostgreSQL.

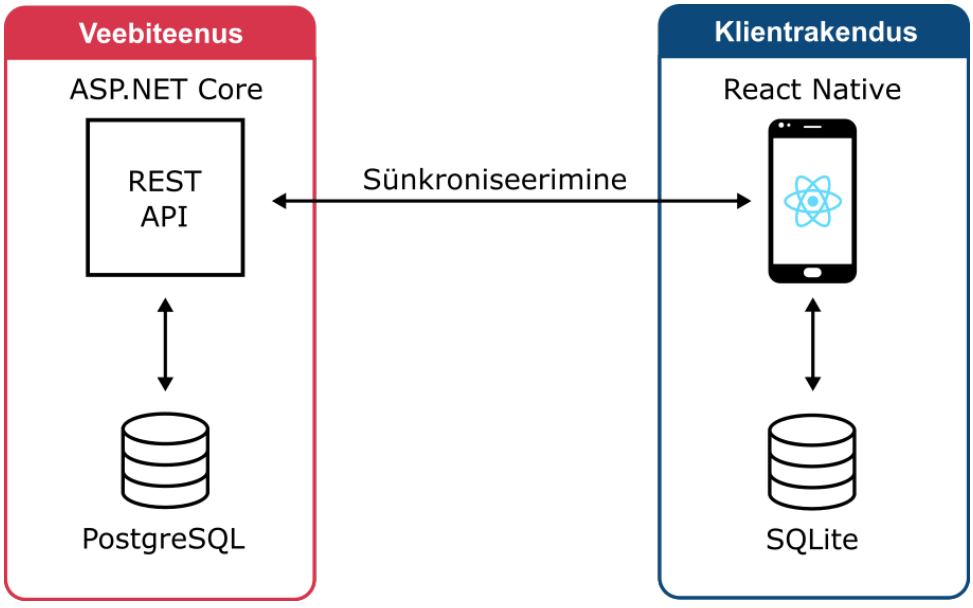
SQLite on kiire ja kompaktne andmebaas nagu ka eelnevalt mainitud. Siiski võib antud andmebaasiga tekkida probleeme, kui palju klientrakendusi korraga ühte andmebaasi kasutavad nagu see saab olema loodava lahenduse puhul [47].

MySQL ning PostgreSQL (tuntud ka kui Postgres) on ühed populaarsemad serveripõhised andmebaasisüsteemid [24]. Kui PostgreSQL toetab enamus SQL standardis olevaid erisusi, siis MySQL puhul on loobutud mitmest standardi erisusest (nt FULL JOIN, INTERSECT, EXCEPT) keskendudes andmebaasi kiirusele [48]. Siiski on leitud, et väiksemate kirjete arvu korral on nii SELECT, INSERT kui ka MAX laused

PostgreSQL andmebaasi korral kiiremad võrreldes MySQL-ga [49]. Kuigi mõlemad andmebaasid oleksid loodava lahenduse jaoks sobivad, siis valiti siiski PostgreSQL andmebaas, kuna see on täiesti vabavaralise litsentsiga [50] ning on väiksemate andmemahtude juures kiirem [49].

3.3 Rakenduse arhitektuur

Antud töö raames valminud lahenduse arhitektuuri põhikomponendid moodustavad serveripoolne veebiteenus ning kliendipoolne React Native klientrakendus (Joonis 3). Veebiteenuse komponendid on ASP.NET Core raamistikul paiknev rakendusliides (REST API), mis on omakorda ühendatud PostgreSQL andmebaasiga. Klientrakenduse moodustavad React Native raamistikul loodud mobiilirakendus, mis on ühendatud lokaalse SQLite andmebaasiga. Klientrakenduse ning veebiteenuse vahel toimub andmete sünkroniseerimine, et tagada andmete terviklus ka kasutaja nutiseadme rikke korral või seadme vahetuse puhul. Lisaks võimaldab andmete sünkroniseerimine kasutada rakendust mitmel seadmel korraga.

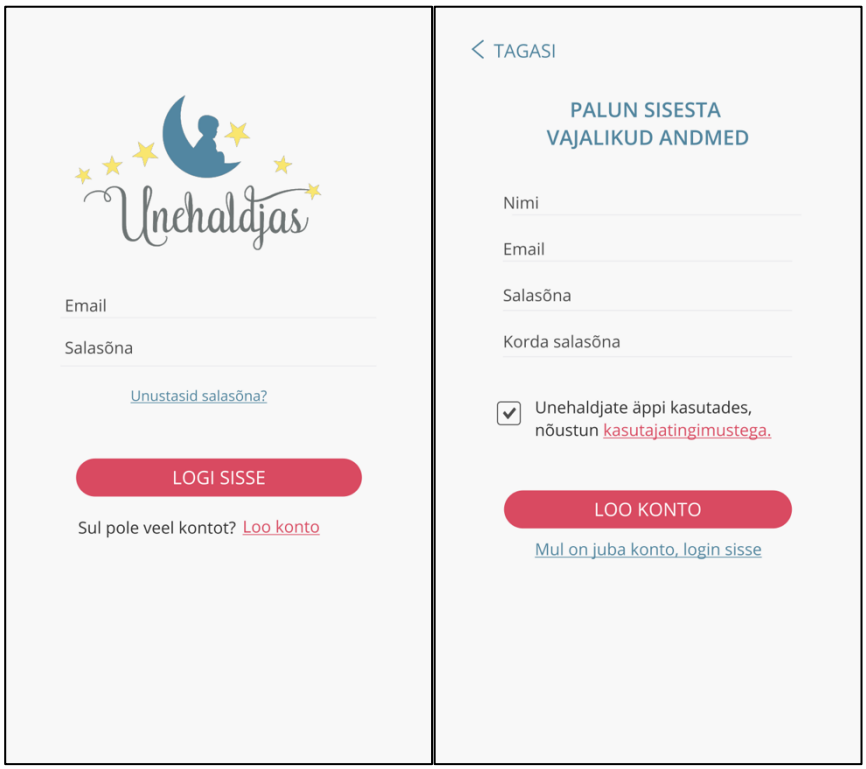


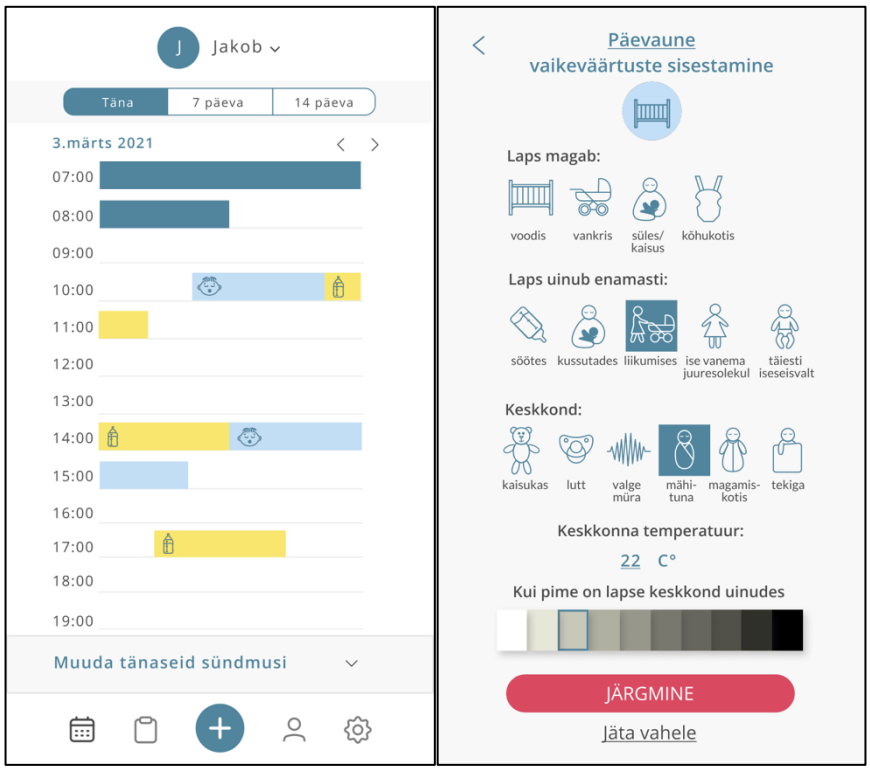
Joonis 3. Rakenduse arhitektuur.

3.4 Rakenduse disain

3.4.1 Kasutajaliidese disain

Rakenduse kasutajaliidese disain (Joonis 4) kujunes koostöös KuuKukk Disain disainistuudioga [51]. Töö autor oli eelkõike vastutav kasutajakogemuse teekonna loomisel ning täpsete juhiste andmisel disainerile seoses elementide paigutamisega. Kasutajaliidese disainil arvestati funktsionaalsete nõuetega, et kasutajaliides peab olema kasutatav ka värvipimedatel ning peab olema eesti keeles.

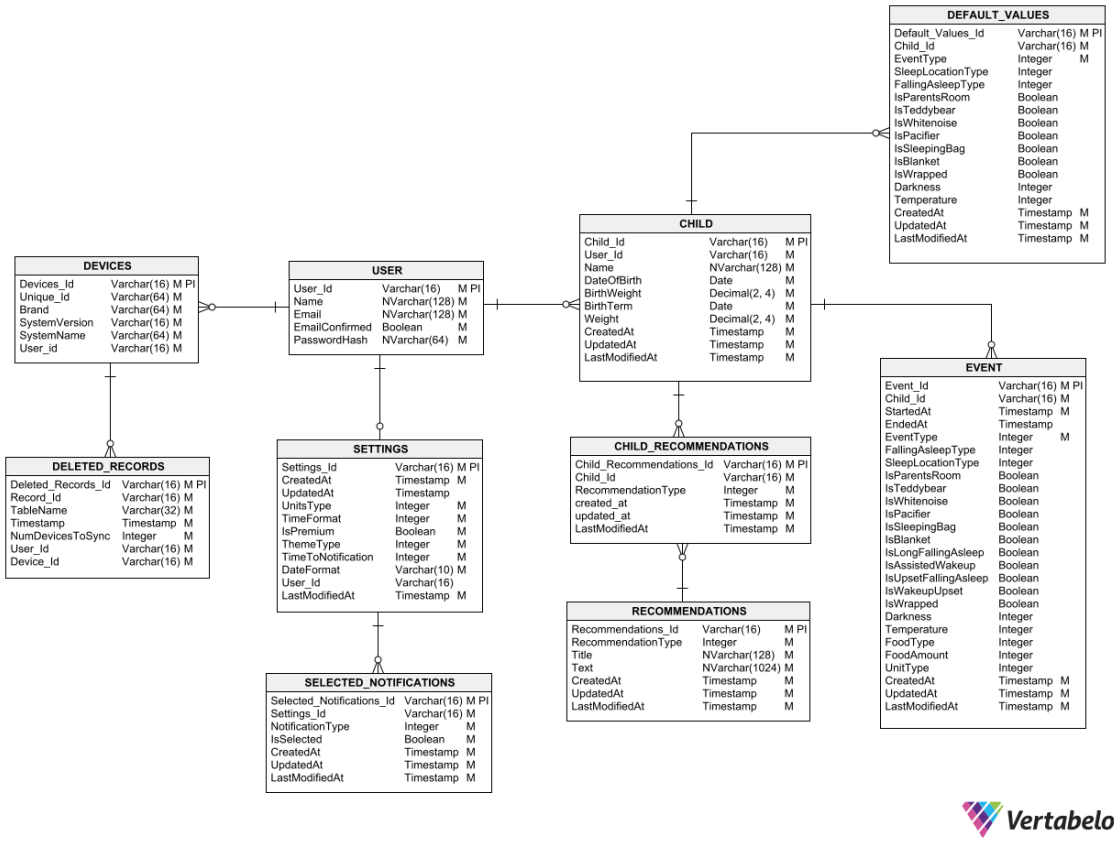




Joonis 4. Näited rakenduse esialgsest disainist - sisselogimine, kasutaja registreerimine, ühe päeva sündmused, vaikeväärtuste sisestamine.

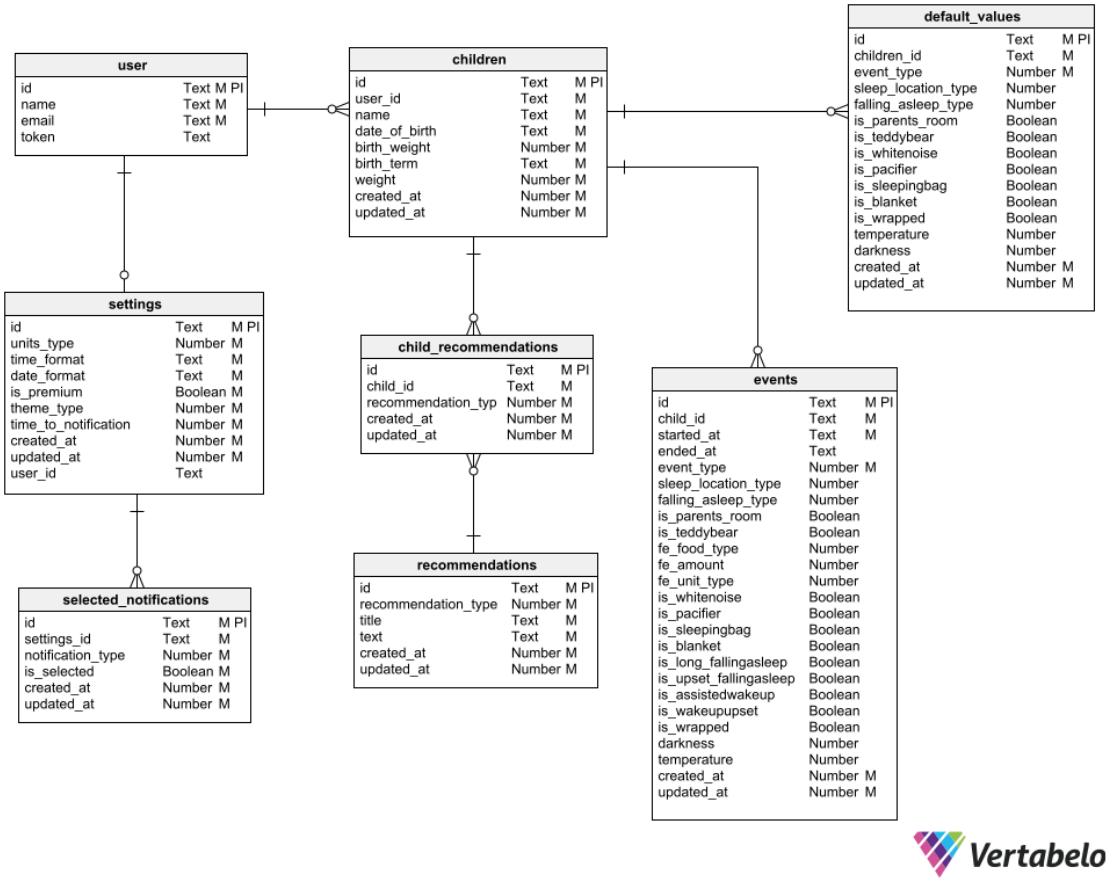
3.4.2 Andmebaasi disain

Veebirakenduse andmebaasi disaini loomiseks kasutati olemi-suhte diagrammi (Joonis 5, Joonis 6). Veebiteenuse peamine ülesanne on kasutajakonto haldamine ning andmete varundamine ja sünkroniseerimine. Kasutajakonto haldamiseks on loodud olem *User*, mida hallatakse läbi ASP.NET Core Identity kasutajate autentimise süsteemi [52]. *User* olemist on eraldatud olem *Settings* kuna *User* olemit ei sünkroniseerita. Lisaks salvestatakse andmebaasi ka kasutaja seadme andmed olemisse *Devices*, et oleks võimalik tuvastada mis seadet kasutaja hetkel kasutab ning oleks võimalik rakenduse vigade korral tuvastada kasutatud seade. Sünkroniseerimise tarvis on lisatud olem *Deleted_Records*, kuhu kantakse kõik kasutaja kustutatud kirjed juhul kui kasutajal on kasutuses mitu seadet. Kui kustutatud kirje on sünkroniseeritud, kustutatakse vastav kirje ka *Deleted_Records* tabelist. Lisaks on igale sünkroniseeritavale olemile lisatud atribuut *LastModifiedAt*, mis märgib kirje veebiteenuse andmebaasi lisamise aega. Oluline on lisada kirjele sisestamise aeg serveris, kuna klientrakenduse pool lisatud ajatempel ei pruugi erinevate seadmete vahel olla kooskõlas või muutub vahepeal kliendi ajatsoon.



Joonis 5. Veebirakenduse andmebaasi olemi-suhte diagramm. M – kohustuslik atribuut, PI – primaarvõti.

Klientrakenduse andmebaas sarnaneb suuresti veebiteenuse andmebaasile (Joonis 6). Kuna SQLite andmebaasil ei ole spetsiaalset kuupäeva ega kellaaja tüüpi, kasutatakse selle asemel Unix ajatemplit [53]. WatermelonDB raamistik kasutab primaarvõtmetena 16-kohalist juhuslikku sõne, mistõttu on see primaarvõtme tüüp kasutusel ka veebirakenduses. Sünkroniseerimise tarvis on igasse sünkroniseeritavasse olemisse lisatud *created_at* ning *updated_at* atribuudid.



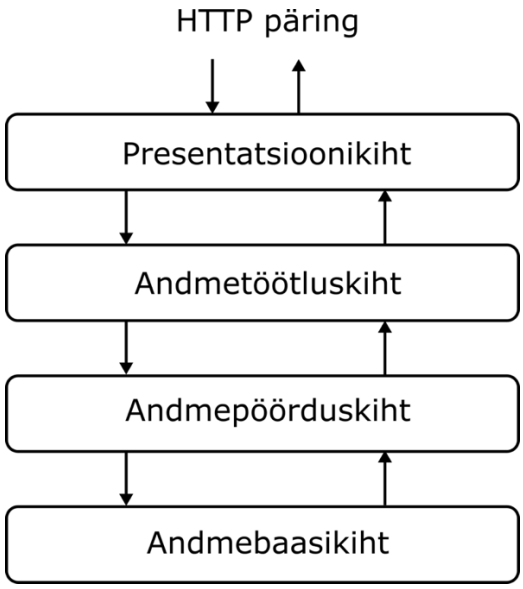
Joonis 6. Klientrakenduse olemi-suhte diagramm. M – kohustuslik atribuut, PI – primaarvõti.

4 Veebirakenduse arendus

4.1 Teenusepoolne lahendus

Teenusepoolse lahenduse eesmärgiks on vastata REST API päringutele. Päringud jagunevad kahte gruppi – kasutaja haldamisega seotud päringud ning andmete sünkroniseerimisega seotud päringud.

Veebiteenus on arendatud ASP.NET Core 5 raamistikus. Arendamisel on lähtutud kihilise arhitektuuri mustrist, mille kohaselt tuleb rakendus eraldada loogilisteks kihtideks, mis kõik etendavad mingit kindlat rolli [54]. Antud rakendus koosneb neljast kihist, milleks on presentatsioonikiht, andmetöötluskiht, andmepöörduskiht ning andmebaasikiht (Joonis 7). Presentatsioonikiht võtab vastu HTTP päringu, kontrollib päringu korrektsust ning saadab päringus sisalduvad andmed edasi andmetöötluskihile. Andmetöötluskiht muudab andmed andmebaasile vastavale kujule ning edastab andmepöörduskihile. Andmepöörduskiht tõlgib andmed SQL päringuteks ning edastab andmebaasikihile, kus toimub andmete sisestamine andmebaasi. Iga järgnev kiht edastab eelnevale kihile vastuse ülesande täitmise õnnestumise kohta. Kihilise arhitektuuri eeliseks on paindlikkus, mis tähendab, et igat kihti on võimalik vajadusel välja vahetada ilma teisi kihte muutmata. Kihtide sõltumatus saavutatakse kasutades sõltuvuste süstimist (*dependency injection*) [55].



Joonis 7. Veebiteenuse arhitektuur.

4.1.1 Presentatsioonikiht

Antud lahenduses on presentatsioonikihi ülesanneteks päringute vastuvõtmine, päringus olevate andmete õigsuse kontrollimine, sünkroniseerimise korral andmete edastamine andmetöötluskihile ning andmetöötluskihilt saadud töödeldud andmete tagastamine klientrakendusele koos staatuse koodiga.

Presentatsioonikiht koosneb REST API kontrolleritest, mis sisaldavad API otspunkte (Tabel 2). REST arhitektuurimudel kujutab endast juhiseid, millele rakendusliides peaks vastama. REST rakendusliides eksponeerib ressursse, mille poole pöördutakse kindlal URL-il oleva olekuta otspunkti kaudu ning sellele tehakse päringuid kasutades HTTP verbe, näiteks GET, POST, PUT jne [56]. REST rakendusliides tagastab staatuse koodi ning olenevalt otspunktist ka soovitud ressursi.

Tabel 2. Veebirakenduses kasutatavad API otspunktid.

Kontroller	Otspunkt	HTTP verb
api/v1/account	/login	POST
api/v1/account	/register	POST
api/v1/account	/changeemail	POST
api/v1/account	/changepassword	POST
api/v1/account	/delete	DELETE
api/v1/sync	/push	POST
api/v1/sync	/pull	POST

Antud lahenduses kasutatakse API versioonimist, mis võimaldab sõltumatult klientrakendusest otspunkte muuta. Versiooninumber lisatakse URLi koosseisu, et vältida vale versiooni poole pöördumist (Tabel 2).

Veebiteenuses võimaldatud API päringud jagunevad kahte gruppi – kasutaja haldamisega seotud päringud ning sünkroniseerimise päringud. Kasutaja haldamisega seotud otspunktid tagastavad vea tekkimisel lisaks staatuse koodile ka veateate, mida on võimalik kasutajale klientrakenduses kuvada (Joonis 8).

```
[HttpPost]
public async Task<IActionResult> Login(LoginDTO model)
{
    var appUser = await _userManager.FindByEmailAsync(model.Email);
    if (appUser == null)
    {
        return Unauthorized("Password or e-mail incorrect");
    }

    var result = await _signInManager
        .CheckPasswordSignInAsync(appUser, model.Password, false);
    if (result.Succeeded)
    {
        var jwt = await GenerateJwtWithValues(appUser);
        return Ok(new {token = jwt, appUserId = appUser.Id});
    }

    return Unauthorized("Password or e-mail incorrect");
}
```

Joonis 8. REST API sisselogimise otspunkti koodi näide.

Kõik rakendusliidese otspunktid käituvad asünkroonselt, mis võimaldab töödelda suure hulga samaaegseid päringuid väikese lõimefondiga [57], mis tähendab, et päringud ei pea üksteise järel ootama. Lisaks on kõik otspunktid välja arvatud „/register“ ning „/login“ turvatud, mis võimaldab neile ligipääsu vaid sisseloginud kasutajatel. Otspunktide tagastatavad staatuse koodid vastavad API disaini heale tavale [17].

4.1.2 Andmetöötluskiht

Andmetöötluskihi ülesandeks on sünkroniseerimisandmete teisendamine sõltuvalt sünkroniseerimise suunast. Andmete allalaadimise korral teisendab andmetöötluskiht andmebaasist tulevad andmed klientrakenduse jaoks vajalikku formaati. Vastupidiselt, andmete üleslaadimise korral teisendatakse päringuga tulnud andmed domeeniobjektideks ning edastatakse andmepöörduskihile. Andmete teisendamiseks kasutatakse vastendajaid (*mapper*).

Lisaks võib andmetöötluskihiga samale tasandile paigutada ka ASP.NET Core Identity [52] kasutajate haldamise mooduli, mis samuti andmepöörduskihist sõltub.

4.1.3 Andmepöörduskiht

Andmepöörduskihi peamiseks komponendiks on Entity Framework Core (EF Core) objekt-relatsioonvastendus süsteem [18]. EF Core vastutab kõige eest, mis on seotud andmebaasiga – andmebaasi skeemi loomine, migratsioonide haldamine, andmete andmebaasikihile edastamine, andmete pärimine, kustutamine jne. Andmepöörduskihi ülesandeks on kõik andmebaasikihile suunatud päringud tõlkida SQL lauseteks ning

andmebaasikihile edastada ning andmebaasikihilt tulnud vastus ja andmed tõlkida omakorda veebiteenuse raamistikule arusaadavale kujule. Antud rakenduses võtab andmepöörduskiht vastu andmed andmetöötluskihilt ning edastab SQL päringutena andmebaasikihile ning saadab andmebaasikihilt tulnud vastuse tagasi andmepöörduskihile.

4.1.4 Andmebaasikiht

Andmebaasi loomiseks veebiteenuses kasutatakse EF Core objekt-relatsioonvastendus süsteemi. EF Core puhul tuleb andmebaas luua *code-first* printsiibil ehk enne tuleb luua koodis klassid (domeeniobjektid), mis etendavad hiljem andmebaasis tabeleid. Andmebaasi skeem luuakse vastavalt loodud domeeniobjektidele. Kuna antud lahenduse puhul kasutatakse PostgreSQL andmebaasi, siis on vajalik kasutada ka vastavat draiverit [58].

Andmebaasi versioonihaldus on samuti EF Core raamistikku sisse ehitatud. Esmase migratsiooni loomisel genereerib EF Core migratsiooni klassid, kus on andmebaasi loomise käsud. Käske saab enne andmebaasi loomist veel käsitsi muuta. Iga järgnev migratsioon loob uued klassid minimaalse arvu käskudega, mis on vajalik muudatuste tegemiseks.

4.1.5 Teenuse turvalisus

REST API otspunktide turvamiseks tagastatakse sisseloginud kasutajale JSON Web Tokens (JWT) sõne. JWT võimaldab turvaliselt osapoolte vahel informatsiooni vahetada. JWT koosneb kolmest osast – päis, andmed (*payload*) ning signatuur [59]. Päises on signatuuri genereerimise algoritm, andmed sisaldavad endas informatsiooni kasutaja õiguste kohta ning signatuur on unikaalne väärtus, mis arvutatakse kasutades saladuseks olevat sõnet. JWT edastatakse base64 kodeeritud sõnena.

Loodud veebiteenuses tuleb pärast kasutaja registreerimist või sisselogimist ülejäänud REST API otspunktidele ligipääsemiseks kasutada veebiteenuse poolt väljastatud JWT sõnet. JWT sõne valideerimise eest ning REST API otspunktide turvamise eest vastutab ASP.NET Core Identity süsteem. JWT genereeritakse teenuses kasutades HMAC-SHA512 krüpteerimise algoritmi. Lisaks on ette nähtud suhtlus ainult läbi HTTPS protokolli.

4.2 Kliendipoolne lahendus

Klientrakendus loodi React Native raamistikule. React Native rakenduse arendamiseks on kaks võimalust – kasutades tööriistakomplekti Expo [60] või React Native käsurealiidest. Expo eeliseks on tema kasutamise lihtsus, mis tähendab, et ei ole vaja seadistada eraldi Androidi või iOS projekte ning arendatavat rakendust saab läbi Expo Go liidese otse füüsilisel seadmel jälgida. Siiski ei ole Expo sobilik, kui on tarvis arendatavas rakenduses kasutada platvormipõhiseid komponente või väliseid teeke, mis omakorda vajavad platvormipõhist seadistust. Nii on ka loodud lahenduse puhul. Kuna WatermelonDB ei toeta Expo kasutamist, siis arendati loodud lahendust kasutades React Native käsurealiidest.

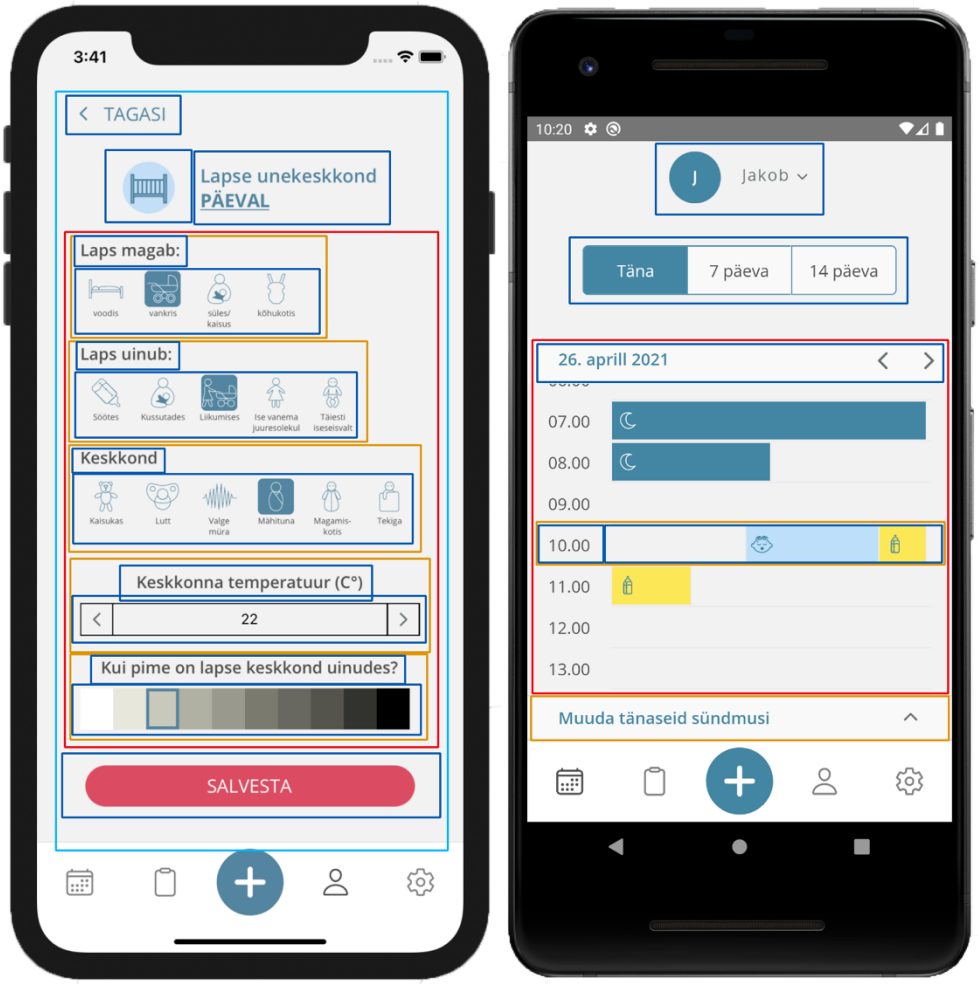
Arendamine toimus macOS operatsioonisüsteemil, kuna teistel operatsioonisüsteemidel ei ole võimalik React Native käsurealiidesel põhinevaid rakendusi iOS platvormile kompileerida [28]. Paigaldada tuli ka Android Studio ning XCode rakendused, mis võimaldasid kasutada vastavalt Android emulaatoreid ning iOS simulaatoreid.

Klientrakenduse arendamisel kasutati TypeScript keelt, mis võimaldab kasutada tugevalt tüübitud muutujaid ja funktsioone, millel on palju eeliseid võrreldes JavaScriptiga [61].

4.2.1 Kliendipoolse rakenduse arendus

React raamistik on üles ehitatud põhimõttel, et kasutajaliidest tuleb vaadelda kui hulka väiksemaid iseseisvaid komponente, mida saab vajadusel taaskasutada. Iga komponent käitub nagu JavaScripti funktsioon, mille sisendiks on muutuv hulk parameetreid ning mis tagastab React elemendi tuginedes sisestatud parameetritele [62].

Suure hulga komponentide loogiliseks korrastamiseks võeti eeskujuks aatomdisaini mudel [63]. Aatomdisaini mudeli järgi jagatakse komponendid nende suuruse järgi tasanditeks – aatomid, molekulid, organismid, mallid ning leheküljed. Iga järgnev tasand kasutab komponendi loomiseks eelneva tasandi komponente. Lõpuks valmib komponentidest lehekülg ehk vaade, mida kasutaja näeb (Joonis 9).



Joonis 9. Näited rakenduse komponentideks jaotamisest. Tumesinine - aatom, pruun - molekul, punane - organism, helesinine - mall.

Rakenduse oleku jälgimiseks kasutatakse kombinatsiooni asünkroonsest võti-väärtuspaari hoidlast (*AsyncStorage*) ning WatermelonDB andmebaasiraamistikust. Asünkroonne võti-väärtuspaari hoidla suhtleb otse operatsioonisüsteemiga, kasutades platvormi enda võimalusi. Võti-väärtuspaari hoidlat kasutatakse kasutaja sisselogimise andmete hoiustamiseks (kasutaja ID ning JWT sõne), mis tagab selle, et kasutaja ei pea pärast rakenduse sulgemist uuesti sisse logima. WatermelonDB kasutatakse andmebaasi olemite sidumiseks komponentidega, muutes komponendid „vaatlejateks“ [46]. See tähendab, et komponendiga on võimalik siduda andmebaas selliselt, et kirje uuendamise järel komponent samuti uuendatakse. Näiteks antud rakenduses ühe päeva kalendrivaates sündmuse kustutamisel uuendatakse kogu vaade ilma, et peaks käsitsi andmebaasi poole pöörduma.

4.2.2 Andmebaasi teostus klientrakenduses

Loodud lahenduse klientrakenduses kasutatakse WatermelonDB andmebaasiraamistikku [46]. WatermelonDB võib vaadelda kui laiendatud võimalustega ORM-i, mis lisaks andmebaasi skeemi loomisele, migratsioonide haldamisele ning andmebaasi päringute koostamisele võimaldab ka React komponente otse andmebaasiga siduda. See tähendab, et andmete lisamisel või muutmisel ei pea uuenduste kuvamiseks andmebaasipäringuid eraldi välja kutsuma. See vähendab oluliselt koodi mahtu, kuna programmeerija ei pea eraldi vaate uuendamiseks vajalikke andmebaasipäringuid kirjutama.

WatermelonDB kasutab sisemiselt SQLite andmebaasi, mis tähendab, et ei ole tarvis eraldi andmebaasi seadistusi teha, kuna SQLite on nii Androidi kui ka iOS operatsioonisüsteemidega kaasas [44].

4.2.3 Suhtlus veebiteenusega

Veebiteenusega suhtlemiseks kasutatakse Axios HTTP klienti [64]. Axiose eeliseks võrreldes JavaScripti enda *fetch* API-ga on see, et Axios võimaldab JSON andmete automaatset serialiseerimist ning paremat veahaldust.

Sünkroniseerimise tarvis pakub WatermelonDB spetsiaalset funktsiooni [65], mida on võimalik kasutada sünkroniseerimise päringu ülesseadmisel. Sünkroniseerimine teostatakse esmalt pärast kasutaja registreerimist ning esimese lapse sisestamist. Hilisem sünkroniseerimine toimub pärast igat tegevust, kas siis andmete lisamisel, uuendamisel või kustutamisel. Sünkroniseerimine teostatakse asünkroonselt rakenduse taustal. Vea korral sünkroniseerimist ei teostata, kuid rakenduse edasist kasutamist see ei takista. Näiteks kui sünkroniseerimise veebiteenus ei ole mitmeid päevi kättesaadav, saadetakse kõik sünkroniseerimata jäänud andmed korraga siis, kui veebiteenus uuesti kättesaadavaks muutub.

4.3 Sünkroniseerimine

Andmete sünkroniseerimine on tehnika, mis võimaldab säilitada andmete kooskõla ühes asukohas asuva allika ning teises asukohas asuva sihtseadme vahel ning vastupidi ning seeläbi ühtlustada andmed aja jooksul [66]. Antud lahenduse kontekstis on sünkroniseerimine kahesuunaline protsess, mis koosneb andmete allalaadimisest ning

üleslaadimisest. Sünkroniseerimise aktiivne pool on klientrakendus ning veebiteenus on passiivses rollis.

Loodud lahendus kasutab andmete sünkroniseerimisel andmebaasiraamistikku WatermelonDB [46]. WatermelonDB on välja sünkroniseerimise tarvis välja töötanud kindla protokolli, millele andmed ning ka päringud vastama peavad [65]. Serveripoolne andmebaas on alati tõe allikas. Igal sünkroniseeritaval andmebaasiolemil peab serveri poolel olema atribuut, mis kujutab endast kirje viimase muutmise ajatemplit. Ajatempli lisab server, kuna klientrakendusest tulevad ajatemplid ei ole usaldusväärsed. Klientrakenduse poolel lisatakse igale sünkroniseeritavale olemile atribuudid „lisamise aeg“ ning „viimase muutmise aeg“. Nende atribuutide järgi eraldatakse klientrakenduses kirjed, mis on lisatud või muudetud pärast viimase sünkroniseerimise aega.

Nii üleslaadimise kui allalaadimise korral peavad andmed vastama kindlale ettemääratud struktuurile (Joonis 10), kus loodud ning uuendatud kirjete korral edastatakse terve kirje ning kustutamise korral ainult kirje ID. Andmestruktuur eeldab, et kõik andmebaasiolemid on andmestruktuuris välja toodud ka siis, kui sünkroniseeritavaid andmeid ei ole. Kõik andmed sünkroniseeritakse kogu andmebaasi ulatuses. See tähendab, et kõik kahe sünkroniseerimise vahel loodud kirjed saadetakse korraga.

```
{
  projects: {
    created: [
      { id: 'aaaa', name: 'Foo', is_favorite: true },
      { id: 'bbbb', name: 'Bar', is_favorite: false },
    ],
    updated: [
      { id: 'ccc', name: 'Baz', is_favorite: true },
    ],
    deleted: ['ddd'],
  },
  tasks: {
    created: [],
    updated: [
      { id: 'tttt', name: 'Buy eggs' },
    ],
    deleted: [],
  },
  ...
}
```

Joonis 10. Illustreeriv näide sünkroniseeritavate andmete struktuurist JSON formaadis.

Andmete üleslaadimise korral saadetakse serverisse päring, milles on kaasas viimase sünkroniseerimise aeg ning sünkroniseeritavad muudatused (Joonis 10). Iga uuendatava kirje korral võrreldakse selle viimase sünkroniseerimise aega sellega, mis sisaldus päringus. Kui serveris asuval andmebaasikirjel on viimase muutmise aeg hilisem kui päringus sisalduv viimase sünkroniseerimise aeg, siis on tegemist konfliktiga ning sünkroniseerimine tuleb peatada. Vea korral tehakse klientrakenduses uuesti allalaadimise päring, mis konflikti lahendab. Igale kirjele, mis sisestatakse või uuendatakse, lisatakse serveris viimase muutmise aeg.

Andmete allalaadimise korral saadetakse serverile päring, kus on viimase sünkroniseerimise aeg. Vastusena saadab server tagasi kõik kirjed, millel on serveris viimase muutmise aeg hilisem kui viimase sünkroniseerimise aeg. Kui viimase sünkroniseerimise aega pole päringus märgitud, tähendab see esmast sünkroniseerimist ning serverist saadetakse klientrakendusele kõik olemasolevad andmed ning antud hetke ajatempel, mida järgmisel sünkroniseerimisel kasutatakse viimase sünkroniseerimise ajana.

5 Hinnang loodud lahendusele

Antud töö käigus valmis kasutatav rakenduse prototüüp. Eesmärgiks võetud funktsionaalsed ning mittefunktsionaalsed nõuded said lahenduses teostatud. Loodud lahenduse veebiteenuse pool valmis täielikult, kuid mobiilirakenduse osas tuleb veel mõningaid muudatusi ellu viia. Kuna töö autoril puudus täielikult varasem hübriid-mobiilirakenduste arendamise kogemus, tuli ette mitmeid tagasilööke, mis uue tehnoloogia õppimisega paratamatult kaasneb. Lisaks oli kohati takistavaks asjaoluks ka suhtlus kliendiga, kuna alati ei olnud võimalik koheselt tehtud tööle tagasisidet saada ning ettetulnud probleeme arutada. Seega tuli kohati palju tööd ümber teha.

Antud töö käigus sai pandud alus rakenduse edasiseks arenduseks ning võimalik on loodud prototüüp arendada täiemahuliseks rakenduseks. Edasises arenduses võiks kindlasti lisada keelte lisamise toe. Samuti lisada tume värvilaad, et rakenduse kasutamine öisel ajal oleks mugavam. Kindlasti tuleb üles seada ka pidev integratsioon ning pidevvalmidus, mis võimaldaks rakendust automaatselt nii App Store kui ka Play Store lisada ning rakendustele uuendusi paigaldada.

Lisaks eelmainitud funktsionaalsustele tuleb välja arendada ka tasulised funktsionaalsused. Selle tarvis tuleb luua reaalajaliste soovituste andmise süsteem, samuti iganädalaste nõuannete ning artiklite kuvamise funktsionaalsus, millega antud prototüübi loomisel juba alustati. Kuna rakenduse eel-analüüsil selgus, et kasutajate arv hakkaks olema piisavalt suur, siis tuleks kindlasti mõelda ka veebiteenuse jõudluse peale.

6 Kokkuvõte

Antud töö eesmärk oli luua hübriid-mobiilirakendus, mis pakuks Eesti lapsevanematele tuge beebide ja väikelaste une jälgimisel ning parandamisel, et laste uneprobleeme nii ennetada kui lahendada. Lahenduse skoop peamiselt hõlmas endas kasutaja registreerimist, lapse andmete sisestamist, lapse une- ja söötmisaegade logimist ning logimisandmete kuvamist graafikuna.

Töö käigus tehti funktsionaalsete ja mittefunktsionaalsete nõuete väljaselgitamiseks küsitlus potentsiaalsete rakenduse kasutajate hulgas ning teostati olemasolevate analoogiliste rakenduste analüüs. Lisaks analüüsiti potentsiaalseid arendusvahendeid ja raamistikke.

Töö käigus valmis nii ASP.NET Core veebiteenus kui ka hübriid-mobiilirakenduse prototüüp React Native raamistikus. Veebiteenuse roll loodud lahenduses on kasutaja konto haldamine ning andmete sünkroniseerimine. Klientrakendust on võimalik kasutada ka mitmel nutiseadmel korraga, kuna andmed sünkroniseeritakse perioodiliselt serveris asuva tagarakendusega.

Loodud lahendus on piisavalt jätkusuutlik, et sellest edasi täiemahuline rakendus arendada.

Kasutatud kirjandus

[1] Unehaldjas. https://unehaldjas.ee/ (01.02.2021)

[2] H. M. El Shakankiry. Sleep physiology and sleep disorders in childhood. *Nature and Science of Sleep*, 2011, 3, 101–114.

[3] C. C. Thiedke. Sleep Disorders and Sleep Problems in Childhood. *American Family Physician*, 2001, 63 (2), 277.

[4] J.-P. Chaput *et al.* Systematic review of the relationships between sleep duration and health indicators in school-aged children and youth. *Applied Physiology, Nutrition, and Metabolism = Physiologie Appliquee, Nutrition Et Metabolisme*, 2016, 41 (6), 266–282.

[5] M. Ghaedrahmati, A. Kazemi, G. Kheirabadi, A. Ebrahimi, and M. Bahrami. Postpartum depression risk factors: A narrative review. *Journal of Education and Health Promotion*, 2017, 6.

[6] S. M. Kerr, S. A. Jowett, and L. N. Smith. Preventing sleep problems in infants: a randomized controlled trial. *Journal of Advanced Nursing*, 1996, 24 (5), 938–942.

[7] Sleep training | Sleep regression | Sleep schedules. – Huckleberry. https://huckleberrycare.com/ (19.02.2021)

[8] Amila - Mobile apps for health and childcare. https://amila.io/ (21.02.2021)

[9] Baby Daybook - Newborn Tracker app. – Baby Daybook. https://babydaybook.app/ (21.02.2021)

[10] Isikuandmete kaitse seadus – Riigi Teataja. https://www.riigiteataja.ee/akt/104012019011 (15.02.2021)

[11] EUROOPA PARLAMENDI JA NÕUKOGU MÄÄRUS (EL) 2016/679 - füüsiliste isikute kaitse kohta isikuandmete töötlemisel ja selliste andmete vaba liikumise ning direktiivi 95/46/EÜ kehtetuks tunnistamise kohta (isikuandmete kaitse üldmäärus).

[12] Firebase. https://firebase.google.com/ (22.02.2021)

[13] Build Mobile & Web Apps Fast | AWS Amplify | Amazon Web Services. – Amazon Web Services, Inc. https://aws.amazon.com/amplify/ (22.02.2021)

[14] Mobile App Service | Microsoft Azure. https://azure.microsoft.com/en-us/services/app-service/mobile/ (22.02.2021)

[15] N. V. Contreras. How we spent 30k USD in Firebase in less than 72 hours | Hacker Noon. https://hackernoon.com/how-we-spent-30k-usd-in-firebase-in-less-than-72-hours-307490bd24d (22.02.2021)

[16] What is ASP.NET Core? A cross-platform web-development framework. – Microsoft. https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core (28.02.2021)

[17] Create Web APIs with ASP.NET Core. https://docs.microsoft.com/en-us/aspnet/core/web-api/ (28.02.2021)

[18] Overview of Entity Framework Core - EF Core. https://docs.microsoft.com/en-us/ef/core/ (28.02.2021)

[19] M. Lorenz, G. Hesse, J.-P. Rudloph, M. Uflacker, and H. Plattner. Object-Relational Mapping Reconsidered. *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017, 4877–4886.

[20] V. Saratchandran. Choosing the Right Mobile App Development Approach for Your Business. – Medium. https://medium.com/techies-toolkit/choosing-the-right-mobile-app-development-approach-for-your-business-204dbcc34093 (06.02.2021)

[21] S.-H. Lim. Experimental Comparison of Hybrid and Native Applications for Mobile Systems. *International Journal of Multimedia and Ubiquitous Engineering*, 2015, 10 (3), 1–12.

[22] Progressive Web App (PWA): what they are, pros and cons and the main examples on the market. – Medium. https://medium.com/iquii/progressive-web-app-pwa-what-they-are-pros-and-cons-and-the-main-examples-on-the-market-318f4538c670 (06.02.2021)

[23] What is a Hybrid Mobile App? – Telerik Blogs. https://www.telerik.com/blogs/what-is-a-hybrid-mobile-app- (06.02.2021)

[24] Stack Overflow Developer Survey 2020. – Stack Overflow. https://insights.stackoverflow.com/survey/2020/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2020 (04.02.2021)

[25] S. Tolomei. Shrinking APKs, growing installs. – Medium. https://medium.com/googleplaydev/shrinking-apks-growing-installs-5d3fcba23ce2 (06.02.2021)

[26] React Native: Bringing modern web techniques to mobile. – Facebook Engineering. https://engineering.fb.com/2015/03/26/android/react-native-bringing-modern-web-techniques-to-mobile/ (05.02.2021)

[27] Out-of-Tree Platforms - React Native. https://reactnative.dev/docs/out-of-tree-platforms (05.02.2021)

[28] Setting up the development environment - React Native. https://reactnative.dev/docs/environment-setup (07.02.2021)

[29] Testing - React Native. https://reactnative.dev/docs/testing-overview (07.02.2021)

[30] How we reduced our production apk size by 70% in React Native? - DEV Community. https://dev.to/srajesh636/how-we-reduced-our-production-apk-size-by-70-in-react-native-1lci (07.02.2021)

[31] A. Mohan. How I Reduced the Size of My React Native App by 86%. – Medium. https://medium.com/@aswinmohanme/how-i-reduced-the-size-of-my-react-native-app-by-86-27be72bba640 (07.02.2021)

[32] Install. https://flutter.dev/docs/get-started/install (05.02.2021)

[33] fastlane docs. https://docs.fastlane.tools/ (05.02.2021)

[34] Xamarin | Open-source mobile app platform for .NET. – Microsoft. https://dotnet.microsoft.com/apps/xamarin (07.02.2021)

[35] C. Petzold. *Creating Mobile Apps with Xamarin.Forms*, 1st ed. Redmond, Washington: Microsoft Press, 2016

[36] Installing Xamarin.iOS on Windows - Xamarin. https://docs.microsoft.com/en-us/xamarin/ios/get-started/installation/windows/ (07.02.2021)

[37] Application Package Size - Xamarin. https://docs.microsoft.com/en-us/xamarin/android/deploy-test/app-package-size (07.02.2021)

[38] Introducing .NET Multi-platform App UI. – .NET Blog. https://devblogs.microsoft.com/dotnet/introducing-net-multi-platform-app-ui/ (05.02.2021)

[39] King-of-Spades. Xamarin.UITest - Visual Studio App Center. https://docs.microsoft.com/en-us/appcenter/test-cloud/frameworks/uitest/ (07.02.2021)

[40] Continuous Integration with Xamarin - Xamarin. https://docs.microsoft.com/en-us/xamarin/tools/ci/ (07.02.2021)

[41] K. Selby. Microsoft Visual Studio 2019 Licensing. Microsoft Corporation

[42] N. Leavitt. Will NoSQL Databases Live Up to Their Promise?. *Computer*, 2010, 43 (2), 12–14.

[43] Relational vs. NoSQL data. https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data (28.02.2021)

[44] Most Widely Deployed SQL Database Engine. https://sqlite.org/mostdeployed.html (07.03.2021)

[45] S. Lee. Creating and Using Databases for Android Applications. *International Journal of Database Theory and Application*, 2012, 5 (2), 8.

[46] Check out the README - WatermelonDB documentation. https://nozbe.github.io/WatermelonDB/ (17.03.2021)

[47] Appropriate Uses For SQLite. https://www.sqlite.org/whentouse.html (08.03.2021)

[48] SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems. – DigitalOcean. https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems (08.03.2021)

[49] C. Asiminidis, G. Kokkonis, and S. Kontogiannis. Database Systems Performance Evaluation for IoT Applications. *SSRN Electronic Journal*, 2018.

[50] PostgreSQL: License. https://www.postgresql.org/about/licence/ (08.03.2021)

[51] Kuukukk Disain | Bränding, Graafiline disain, Veebilehed. https://kuukukkdisain.ee (17.03.2021)

[52] Rick-Anderson. Introduction to Identity on ASP.NET Core. https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity (02.04.2021)

[53] Datatypes In SQLite Version 3. https://www.sqlite.org/datatype3.html (17.03.2021)

[54] 1. Layered Architecture - Software Architecture Patterns [Book]. https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html (02.04.2021)

[55] Rick-Anderson. Dependency injection in ASP.NET Core. https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection (02.04.2021)

[56] RESTful Web services. – IBM Developer. https://developer.ibm.com/technologies/web-development/articles/ws-restful/ (02.04.2021)

[57] ASP.NET Core Performance Best Practices. https://docs.microsoft.com/en-us/aspnet/core/performance/performance-best-practices (02.04.2021)

[58] Npgsql - .NET Access to PostgreSQL | Npgsql Documentation. https://www.npgsql.org/index.html (02.04.2021)

[59] auth0.com. JWT.IO - JSON Web Tokens Introduction. http://jwt.io/ (04.02.2021)

[60] Introduction to Expo. – Expo Documentation. https://docs.expo.io/ (04.02.2021)

[61] Is TypeScript worth it? | BetterStack. https://betterstack.dev/blog/is-typescript-worth-it/ (04.04.2021)

[62] Components and Props – React. https://reactjs.org/docs/components-and-props.html (04.04.2021)

[63] B. Frost. Atomic Design. https://bradfrost.com/blog/post/atomic-web-design/ (04.04.2021)

[64] Axios. https://axios-http.com/ (04.04.2021)

[65] Sync - WatermelonDB documentation. https://nozbe.github.io/WatermelonDB/Advanced/Sync.html (04.03.2021)

[66] M. Faiz and U. Shanker. Data synchronization in distributed client-server applications. *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, 2016, 611–616.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Taavi Paal

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Hübriid-mobiilirakendus beebide ja väikelaste une jälgimiseks ja parandamiseks“, mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

17.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Klientide seas läbiviidud küsitluse ankeet

1. Mitu last sul on?
    a. 1
    b. 2
    c. 3
    d. 4
    e. > 4
2. Kui vanad su lapsed on? Mitme lapse puhul märgi vanused komadega. Alla 2-aastase lapse puhul märgi vanus kuudes (k) ning alla 4-kuuse imiku puhul nädalates (n). Nt (2n, 16k, 3a).
3. Kas oled varem kasutanud mõnd lapse unega seotud *trackerit* või muud mobiilirakendust, mis võimaldab lapse päevakavaga seotud sündmusi (nt uinaku algus ja lõpp, söötmised, mähkmevahetused, öised ärkamised) üles märkida ja jälgida?
    a. Jah
    b. Ei
4. Millist rakendust oled senini kasutanud oma lapse une ülesmärkimiseks? (Vali üks, mida kasutasid kõige kauem, kui oled proovinud mitut.)
    a. Huckleberry
    b. Glow: Baby tracker & Feeding, Diaper, Sleep log
    c. Baby Tracker
    d. Muu (täpsustusega)
5. Kui oled kasutanud mõnd muud rakendust, palun märgi siia selle nimi App Store-is või Google Play Store-is.
6. Millised võimalused meeldisid Sulle enda kasutatud rakenduse puhul enim? Millisest funktsionaalsusest oli Sulle selle rakenduse puhul kõige enam kasu?
7. Mis Sulle kasutatud rakenduse puhul ei meeldinud? Mis muutis kasutamise keeruliseks, tüütuks või mis oli Sinu arvates üleliigne?
8. Kas tunned, et rakenduse kasutamine aitas Sul lisaks logi pidamisele ka oma lapse und parandada?
    a. Jah
    b. Ei

9. Kui vastasid "jah", siis mil moel?
10. Minu unelmate nutirakendusel oleks võimalus üles märkida (vali kõik, mis kindlasti olemas võiks olla):
    a. Uinakute alguse ja lõpuaeg
    b. Söötmiste aeg, kestvus ja kogus
    c. Öised ärkamised ja nende kestvus
    d. Mähkmevahetused ja mähkme täituvus (nt kakane, pissine, kerge/täis/väga täis)
    e. Uinumisele kulunud aeg
    f. Uinumisviis (süles, vankris, rinnal, süües, jne)
11. Kui eelmises küsimuses ei olnud nimetatud kõike, mida Sa sooviks, lisa puuduolevad funktsioonid siia.
12. Minu unelmate nutirakendus annaks mulle lapse une alast nõu järgmistel teemadel:
    a. Eakohane ärkvelolekuaeg ehk kui kaua võib mu laps praeguses vanuses ärkvel olla?
    b. Eakohane uinakute graafik ehk kui palju uinakuid ja kui palju päevaund mu laps hetkel vajab?
    c. Eakohased soovitused uinumise lihtsustamiseks/kiirendamiseks
    d. Kuidas kohandada söötmiste graafikut parema une heaks
    e. Eakohase unerituaali loomine
    f. Päeva- ja ööune tasakaalu loomine
    g. Öiste ärkamiste vähendamine
    h. Und soodustava unekeskkonna loomine
    i. Info une arengu kohta vastavalt mu lapse vanusele
13. Kui eelmises loetelus oli mõni Sinu jaoks oluline teema puudu, märgi see siia.
14. Kas oleksid nõus sellise nutirakenduse eest maksma?
    a. Jah
    b. Ei
15. Kui jah, siis kui palju oleksid igakuiselt valmis maksma? (vali oma taluvuse ülemine piir ehk maksimaalne summa, mida oleksid nõus kulutama)
    a. 1-2 €/kuus
    b. 3-4 €/kuus
    c. 5-7 €/kuus

d. 8-10 €/kuus

16. Kui algataksime rakenduse väljatöötamise toetuseks Hooandja kampaania, kas oleksid nõus sellesse panustama?
    a. Jah
    b. Ei