OULUN AMMATTIKORKEAKOULU

BUDJETTISOVELLUKSEN OPTIMOINTI JA KÄYTTÖLIITTYMÄN SUUNNITTELU

Huttula Aatu & Junno Topias

Opinnäytetyö AMK

Kevät 2025

Tieto- ja viestintätekniikan tutkinto-ohjelma

Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Opinnäytetyö AMK
Tieto- ja viestintätekniikan tutkinto-ohjelma

Tekijä(t): Huttula Aatu, Junno Topias
Opinnäytetyön otsikko: Budjettisovelluksen optimointi ja käyttöliittymän suunnittelu
Työn ohjaaja(t): Lohiniva Meija
Työn valmistumislukukausi ja -vuosi: kevät 2025
Sivumäärä: 34

Opinnäytetyön aiheena oli React Native -sovelluksen optimointi ja käyttöliittymän suunnittelu. Työn tavoitteena oli syventää osaamista näistä aiheista sekä soveltaa opittua käytännössä kehittämällä suorituskykyinen ja käyttäjäystävällinen mobiilisovellus.

Tietoperusta koostui koulutuksen aikana opituista asioista sekä aiemmista projekteista. Lisäksi tietoa haettiin kirjallisuudesta, virallisista dokumentaatioista ja artikkeleista, joissa käsiteltiin React Nativelle tyypillisiä optimointitekniikoita ja käyttöliittymän suunnitteluperiaatteita. Optimointi toteutettiin käyttäen saatua tietoa ja testaamalla sen tehokkuutta. Käyttöliittymän suunnittelussa painotettiin selkeyttä, visuaalista ilmettä ja sujuvaa käyttökokemusta.

Työn tuloksena syntyi optimoitu ja käyttäjäystävällinen mobiilisovellus, joka vastasi asetettuja tavoitteita. Tulosten perusteella voitiin päätellä, että suorituskykyä onnistuttiin parantamaan ja käyttöliittymästä saatiin toimiva ja miellyttävä käyttää.

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author(s): Huttula Aatu, Junno Topias,
Title of thesis: Bujettisovelluksen optimointi ja käyttöliittymän suunnittelu
Supervisor(s): Lohiniva Meija
Term and year when the thesis was submitted: spring 2025
Number of pages: 34

This thesis focuses on optimizing a React Native application and designing user interfaces. The objective was to deepen our understanding of these topics and apply the acquired knowledge in practice by developing a high-performance and user-friendly mobile application.

The theoretical foundation was based on prior studies, coursework, and previous projects. Additional information was gathered from literature and articles related to React Native optimization techniques and UI design principles. The optimization process utilized commonly recommended methods to enhance performance and efficiency, while the user interface was designed with clarity and usability in mind

As a result, an optimized and user-friendly mobile application was developed. The findings indicate that the performance improvements were successful, and that the application provides a smooth and visually appealing user experience.

SISÄLLYS

SANASTO

käsite	selite
React Native	Ohjelmistokehys
API-pyyntö	Rajapintakutsu
IA	Informaatioarkkitehtuuri
UI	User-Interface, käyttöliittymä
UX	User-Experience, käyttäjäkokemus

1 JOHDANTO

Tämän työn aiheena on luoda budjettisovellus, joka auttaa käyttäjiä seuraamaan omaa talouttaan. Sovelluksen kohderyhmänä ovat kaikenikäiset ihmiset, jotka haluavat seurata menojaan ja suunnitella talouttaan paremmin. Budjetointi on tärkeä työkalu tulojen ja menojen hallinnassa, sillä se auttaa ylläpitämään taloudellista tasapainoa, saavuttamaan säästötavoitteita ja välttämään velkaantumista. Digitaalisten työkalujen avulla budjetointi voidaan tehdä entistä helpommaksi, koska budjetti voidaan muuntaa visuaalisempaan muotoon, tulot ja menot lasketaan automaattisesti, ja budjetti kulkeutuu sovelluksen muodossa aina mukana taskussa. Sovelluksen tarkoituksena on antaa käyttäjille yleiskuva omasta taloudestaan, seurata menoja ja tuloja, asettaa budjetteja eri luokille, hallita toistuvia laskuja ja menoja, sekä seurata säästöjä.

Toteutettava sovellus toimii Android- ja iOS-järjestelmillä varustetuilla älylaitteilla. Käyttäjän tunnistaminen ja tietojen tallennus tapahtuu pilvipohjaisen tietokannan avulla, jossa kirjautuminen tapahtuu sähköpostiosoitteen ja salasanan perusteella, mikä mahdollistaa budjetin tarkastelun eri laitteilla. Vaihtoehtoisesti sovellukseen voidaan kirjautua ilman käyttäjätietoja, tällöin budjettitiedot tallentuvat vain laitteelle paikallisesti, eikä tietoja voida käyttää muilla laitteilla. Sovellus luodaan käyttäen React Native-ohjelmistokehystä, joka toimii JavaScript-ohjelmointikielellä ja tietojen varastoimiseen käytetään Googlen Firebase kehitysalustaa.

Ensimmäinen teoriaosuus käsittelee optimointia, joka on keskeistä budjettisovellusten kehityksessä. Sovelluksen suorituskyky vaikuttaa suoraan käytettävyyteen—nopeasti latautuvat näkymät, sujuva tiedonkäsittely ja resurssitehokas ohjelmointi parantavat käyttäjäkokemusta. Budjettisovelluksissa saatetaan käsitellä laajoja tietomääriä, jos sovellusta käyttävä lisää runsaasti tietoa kuluistaan ja tuloistaan. Tehokkaat algoritmit ja kevyet tietorakenteet voivat tukea nopeaa hakua ja laskentaa. Hyvin optimoitu sovellus voi myös skaalautua paremmin ilman suorituskyvyn heikkenemistä.

Toinen teoriaosuus painottaa käyttöliittymän- ja käyttäjäkokemuksen suunnittelua. Sovelluksen tulisi tarjota selkeä ja helppokäyttöinen käyttöliittymä, jonka avulla käyttäjä hahmottaa taloudellisen tilanteensa. Talouden visualisointi, looginen navigointi ja käyttäjän tarpeisiin mukautuvat toiminnot vaikuttavat merkittävästi sovelluksen käytettävyyteen.

2 SOVELLUKSEN OPTIMOINTI

Tämä teoriaosuus käsittelee optimointia. Dokumentti käy ensin yleisellä tasolla läpi mitä optimointi tarkoittaa, jonka jälkeen aihe tarkentuu käsittelemään React Native projektiamme.

2.1 Optimointi ohjelmoinnissa

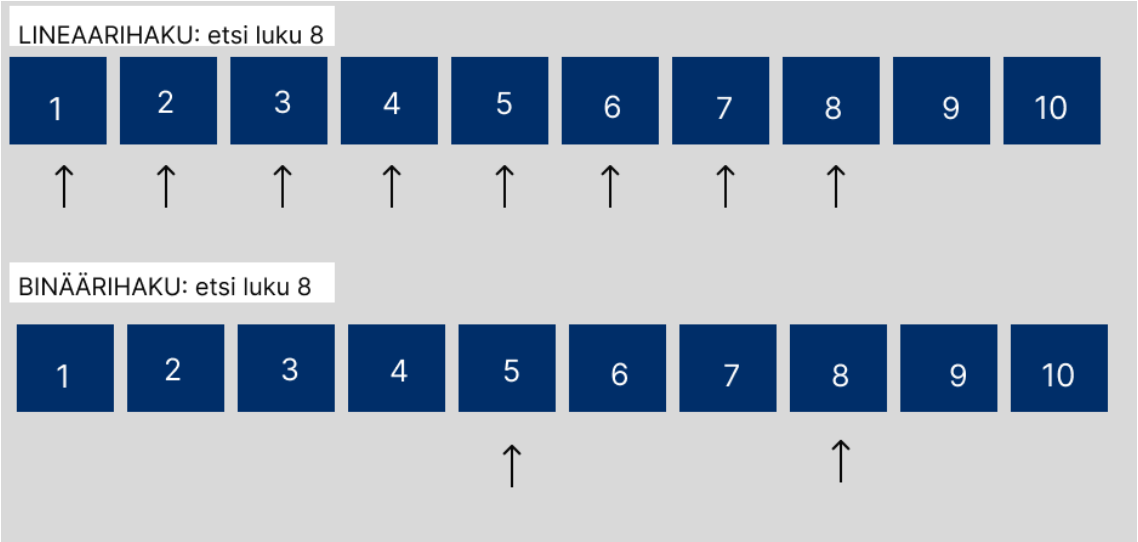
Optimointi ohjelmoinnin kontekstissa viittaa ohjelmointitapaan, jossa ohjelmistojärjestelmä toteutetaan siten, että sen suorituskyky ja tehokkuus paranevat. Optimoinnin tavoitteena on nopeuttaa ohjelman toimintaa ja vähentää sen resurssien, kuten muistin ja laskentatehon, kulutusta. Tämä saavutetaan yksinkertaistamalla ohjelman rakennetta ja vähentämällä suoritettavien tehtävien määrää. Optimoinnin keskiössä on kehittää ratkaisuja, jotka ovat mahdollisimman yksinkertaisia, resurssitehokkaita, nopeita ja suorituskykyisiä. (codeAcademy)

Ohjelman suorituskyvyn parantamiseksi on monia tapoja, joista yleisimpiä ovat koodin yksinkertaistaminen esimerkiksi budjettisovelluksessa kuukausittaisten kulujen laskentaa voidaan vähentää hyödyntämällä välimuistia, jolloin samaa tietoa ei tarvitse laskea uudelleen jokaisella tarkastelukerralla. Tämä voi vähentää laskennan kuormitusta ja nopeuttaa sovelluksen toimintaa. Turhien silmukoiden välttäminen, liiallisten kolmansien osapuolien kirjastojen vähentäminen ja ajon aikana tapahtuvien funktioiden uudelleen määrittäminen. Täytyy kuitenkin muistaa, ettei kaikkeen voida panostaa yhtä paljon. Joten kompromisseja parhaimman suorituskyvyn saavuttamiseen joudutaan tekemään. Tämä meinaa yleensä, että ohjelmoija joutuu panostamaan yhteen osa-alueeseen enemmän kuin toiseen. (codeAcademy)

Arkkitehtuurin valinta on myös yksi tekijä suorituskyvyn, resurssien käytön ja kuormituskestävyyden kannalta. Esimerkiksi verkkoviiveitä voidaan optimoida yhdistämällä API-pyyntöjä yhdeksi kutsuksi useiden erillisten pyyntöjen sijaan, mikä vähentää latenssia ja parantaa käyttäjäkokemusta. Tämä voidaan havainnollistaa konkreettisesti vertaamalla kahta erilaista lähestymistapaa:

erilliset API-pyynnöt, joissa jokainen pyyntö suoritetaan erikseen ja viivästyttää seuraavaa, verrattuna yhdistettyyn pyyntöön, joka hoitaa useat toiminnot kerralla. Yhdistetty pyyntö vähentää verkon kuormitusta ja parantaa suorituskykyä, koska se vaatii vähemmän yhteyksiä palvelimeen ja vähemmän käsittelyaikaa. Jos haetaan suorituskykyä ja nopeutta, voi ohjelmointikielen valinta myös olla kriittinen tekijä. Esimerkiksi matalamman tason kielet, kuten C, voivat tarjota enemmän hallintaa suorituskyvystä, kun taas korkean tason kielet, kuten Python, voivat nopeuttaa kehitystä, mutta niillä voi olla suurempia suorituskykyhaasteita. Näiden päätösten muuttaminen myöhemmin voi olla kallista ja vaatia merkittäviä muutoksia järjestelmän rakenteeseen. (codeAcademy)

Lähdekoodin optimoinnissa on ensin tunnistettava suorituskyvylliset ongelmat. Tähän voi käyttää erilaisia työkaluja kuten selaimeen rakennettu “DevTool” tai “CppCheck” joka on C++ kielen testityökalu. Ilman testausta ohjelmoija voi tahattomasti kirjoittaa itselleen lisää ongelmia. Algoritmien valinta voi merkittävästi vaikuttaa ohjelman ajonaikaiseen suorituskykyyn. Erityisesti silloin, kun esimerkiksi ohjelma käy läpi pitkän listan kokonaisuudessaan etsien tiettyä kohdetta, jonka voisi löytää huomattavasti nopeammin toisella lähestymistavalla. Tätä kutsutaan lineaariseksi hauksi. Parempi vaihtoehto voisi olla binäärihaku. Esimerkissä kuvassa 1 on havainnollistettu näiden kahden haun ero siten, että annetaan taulukko, jossa on kymmenen lukua. Tehtävänä on etsiä luku 8. Lineaarinen haku aloittaa tarkistamalla luvut järjestyksessä alkaen ensimmäisestä ja jatkaa, kunnes luku 8 löytyy. Sen sijaan binäärihaussa aloitetaan taulukon keskeltä. Verrataan keskellä olevaa lukua haluttuun lukuun ja tarkistetaan, onko se pienempi vai suurempi kuin etsittävä luku. Vertailun perusteella taulukko puolitetaan ja siirrytään siihen osaan, jossa mahdollisesti etsittävä luku sijaitsee. Tämä prosessi toistetaan, kunnes luku on löytynyt. Esimerkin mukaan lineaarisessa haussa tarvitaan 8 vaihetta, kun taas binäärihaussa riittää vain 2 vaihetta.



KUVA 1. Lineaari- ja binäärihaun ero.

Mutta on myös huomioitava data tyyppi ja minkälainen algoritmi sille soveltuu parhaiten. (medium)

Muistivuodot voivat myös olla merkittävä suorituskyvyn heikkenemisen syy, erityisesti pitkäkestoisissa sovelluksissa, kuten interaktiivisissa ohjelmissa. Muistivuoto tarkoittaa, että ohjelma varaa muistia, mutta ei vapauta sitä, mikä johtaa muistifragmentaatioon ja voi jopa estää järjestelmää käyttämästä muistia tehokkaasti. Tämä voi johtaa järjestelmän suorituskyvyn heikkenemiseen, ja suurilla tietomäärillä muistivuodot voivat estää ohjelman sujuvan toiminnan. (ibm)

Testaaminen oikeilla laitteilla varhaisessa vaiheessa on tärkeää, koska simulaattorit eivät kykene täysin mallintamaan todellisia laitteiden ja käyttöjärjestelmien suorituskykyerilaisuuksia. Erilaiset laitemallit, erityisesti vanhemmat tai vähemmän tehokkaat laitteet, voivat paljastaa suorituskykyongelmia, jotka eivät ilmene simulaattorissa. Säännöllinen testaaminen laitteilla koko kehityksen ajan mahdollistaa ongelmien tunnistamisen ja korjaamisen ennen kuin niistä tulee merkittäviä. Erityisesti alfa- ja beetatesteissä saadut käyttäjäraportit tarjoavat arvokasta tietoa sovelluksen suorituskyvyn parantamiseksi. (weblineGlobal)

2.2 React-Nativen Optimointi

React-Native on ohjelmistokehys, joka mahdollistaa yksinkertaisen ja joustavan tavan kirjoittaa sovelluksia niin Android- kuin IOS-käyttöjärjestelmille. Kuitenkin React Native ei ole täydellinen, ja kehittäjät voivat kohdata haasteita, kuten yhteensopivuusongelmia eri laitteiden välillä tai suorituskykyongelmia. Tällöin sovelluskehittäjien on usein löydettävä kiertoteitä tai ratkaisuja, jotka saattavat hidastaa kehitysprosessia.

Sovelluksen suorituskykyyn voivat vaikuttaa useat tekijät, kuten monimutkaiset käyttöliittymät ja animaatiot, tarpeettomat uudelleenrenderöinnit, optimoinnin puutteet, raskaat laskelmat sekä liiallinen datan haku. Monimutkainen käyttöliittymä ja huonosti toteutetut animaatiot hidastavat renderöintiä, kun taas tarpeettomat uudelleenrenderöinnit kuormittavat järjestelmää turhaan. Lisäksi tehottomat optimointikäytännöt, kuten suorituskykyä parantavien tekniikoiden laiminlyönti voi hidastaa sovelluksen toimintaa. Raskaat laskelmat komponenttien sisällä sekä liiallisen datan hakeminen lisäävät latausaikoja ja heikentävät käyttäjäkokemusta. (medium)

React Native -dokumentaatio tukee edellä mainittuja suorituskyvyn ongelmien ratkaisemista, mutta tuo esiin myös muita keskeisiä haasteita. Esimerkiksi sovelluksen suorittaminen kehitystilassa voi heikentää suorituskykyä, koska se on suunniteltu tukemaan virheiden ja varoitusten näyttämistä, mikä lisää prosessorin kuormitusta ja voi hidastaa sovelluksen toimintaa. Tällöin suorituskykytestit eivät anna tarkkaa kuvaa siitä, miten sovellus käyttäytyy todellisessa käyttöympäristössä. Jotta saadaan luotettavaa tietoa sovelluksen suorituskyvystä, on tärkeää testata sitä tuotantotilassa, jossa kehitystyökalut ja virheilmoitukset on poistettu. Konsolin tulostukset, kuten console.log-kutsut, voivat myös merkittävästi hidastaa sovelluksen toimintaa erityisesti tuotantokoodissa, joten niiden poistaminen on suositeltavaa.

Listojen käsittelyssä React Native dokumentaatio suosittelee vanhan ListView-komponentin sijaan FlatList- tai SectionList-komponentteja, jotka on optimoitu muistinkäytön ja suorituskyvyn parantamiseksi. Lisäksi getItemLayout-metodin käyttö voi parantaa suurten listojen renderöintitehokkuutta, koska se antaa etukäteen tiedon jokaisen listan kohteen koosta ja sijainnista. Tämä vähentää tarpeetonta laskentaa ja parantaa suorituskykyä erityisesti suurissa listoissa.

Animaatioiden suorituskykyä voidaan parantaa käyttämällä useNativeDriveriä mikä siirtää animaatiolaskennat natiivisäikeelle ja estää JavaScript-säikeen ylikuormittumisen. UI-säikeen kuormituksen vähentämiseksi voidaan hyödyntää ominaisuuksia, kuten shouldRasterizeIOS ja renderToHardwareTextureAndroid, mutta niiden käyttöä tulee harkita huolellisesti, sillä ne voivat lisätä muistinkäyttöä. (React Native Documentation)

3 KÄYTTÖLIITTYMÄSUUNNITTELU

Tässä teoriaosuudessa käsitellään UI-, ja UX-suunnittelun eroja, sekä niihin liittyviä yleisiä asioita. Osuudessa keskitytään keskeisiin periaatteisiin ja menetelmiin, joiden avulla pyritään luomaan käyttöliittymä, joka on miellyttävä ja visuaalisesti houkutteleva, sekä jättää käyttäjälle miellyttävän kokemuksen sovelluksen käytöstä.

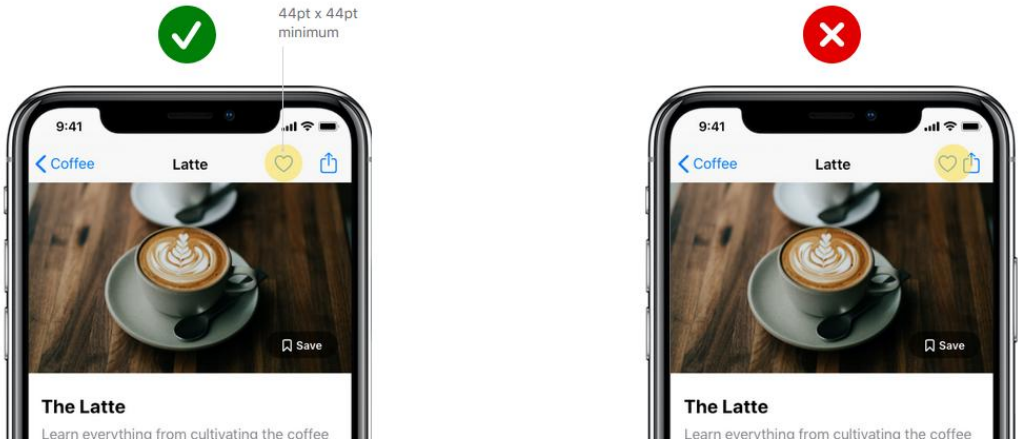
3.1 Käyttöliittymä

Käyttöliittymällä tarkoitetaan sovelluksen vuorovaikutteisuutta, ulkoasua ja tuntumaa. (Figma). Hyvä käyttöliittymä alkaa käyttäjäkunnan tuntemisesta. Suunnittelussa keskitytään käyttäjien tarpeisiin ja tavoitteisiin, sillä sovellusta käytetään tietyn päämärän saavuttamiseksi ja vuorovaikutus tapahtuu tietyllä tavalla (Brewer, Tidwell, Valencia, 2019, 1). Kohderyhmän ymmärtäminen helpottaa suunnittelijaa asettamaan sisältöä niin, että käyttäjä löytää etsimänsä nopeasti (Hannah, J, 2023). Jos suunnitellessa pystytään ennustamaan, mitä toimenpiteitä käyttäjä haluaa tehdä ensimmäisenä uudessa sovelluksessa, niiden suorittamisesta kannattaa tehdä erittäin helppoa. Kun käyttäjä kokee onnistumisen tunteen heti ensimmäisten sekuntien aikana, on todennäköistä, että hän jatkaa sovelluksen käyttöä, vaikka myöhemmin ilmenisi haasteita. Ohjelmisto toimii lopulta vain työkaluna, jonka avulla käyttäjien tavoitteet saavutetaan, ja niiden saavuttamista tukemalla mahdollisimman tehokkaasti voidaan lisätä käyttäjätyytyväisyyttä (Brewer, Tidwell, Valencia, 2019, 1).

Mobiilisovelluksia selataan usein kiireessä ja lyhyissä hetkissä. Siksi sisällön tulisi olla nopeasti ja vaivattomasti sisäistettävissä. Pitkien tekstikappaleiden ja perusteellisten selitysten sijaan suositaan selkeä rakenteisia lyhyitä kappaleita. Otsikot, väliotsikot, luettelot ja muut visuaaliset elementit parantavat luettavuutta. (Hannah, J, 2023).

Sovelluksen ulkoasun tulisi olla käyttäjälle selkeä. Tämän saavuttamiseksi on tehtävä tarkkaan harkittuja päätöksiä aina otsikoiden sijainnista tyhjän tilan määrään. Valittaessa värejä ja fontteja täytyy ottaa huomioon yhdenmukaisuus brändin kanssa, saavutettavuus ja toistuvuus sovelluksen muiden sivujen kanssa. (Figma)

Käyttäjä tarvitsee jatkuvaa palautetta sovellukselta varmistaakseen, että kaikki toimii kuten pitää (Brewer, Tidwell, Valencia, 2019, 1). Jatkuvalla palautteella tarkoitetaan, että käyttäjä saa välittömästi vahvistuksen toiminnostaan, esimerkiksi pudotusvalikko avautuu ja sulkeutuu painalluksesta. Sovelluksen klikattavien osien, kuten nappien ja pudotusvalikkojen tulisi tuntua käyttäjälle miellyttävältä (Figma). Mobiililaitteet ovat suunniteltu kosketustoimintoja varten. (Hannah, J, 2023). Interaktiivisten osien tulisi olla riittävän isoja painettaviksi, ja riittävän kaukana toisistaan välttäen vahinko painalluksia (kuva 2).



KUVA 2. Painikkeiden koko ja asettelu (Apple Developer).

Mobiililaitteiden näyttöjen koon takia sivuilla voidaan näyttää rajoitettu määrä sisältöä. On tärkeää käyttää hyvin kuvailevia opasteita kertomaan käyttäjille mistä löytää etsimänsä. Sillä mobiililaitteiden näytöillä on rajallisesti tilaa, on monesti tarpeellista piilottaa tai tiivistää tietoa näkymästä. Laajennettavat elementit ovat hyviä työkaluja piilottamaan asioita, jotka eivät ole käyttäjille välttämättömiä, mutta ne voidaan laajentaa tarvittaessa (Hannah, J, 2023).

Kun käyttää käyttöliittymiä jatkuvasti, tietyt toiminnot muuttuvat tavoiksi, eikä niitä tarvitse enää tietoisesti miettiä (Brewer, Tidwell, Valencia, 2019, 1). Tähän

voidaan käyttää esimerkkinä mobiilisovelluksessa navigointia, joka yleisesti tapahtuu näytön ylä- tai alaosassa sijaitsevasta valikosta.

Tumma teema parantaa käyttömukavuutta erityisesti hämärässä ja vähentää silmien rasitusta. Tummien värien käyttö sovelluksessa tuo hyötyä myös optimoinnin kannalta. Se voi pienentää virrankulutusta OLED-näytöillä, sillä OLED-näytöt ohjaavat jokaista pixeliä yksitellen antaen niille vain sen verran virtaa kuin ne tarvitsevat. Pixelit jotka käyttävät täysi mustaa (#000000) sammuvat OLED-näytöllä kokonaan säästäen eniten virtaa. (Nngroup).

KUVA 3. Kuvakaappaus Iltalehti sovelluksen etusivusta 13.02.2025

Kuvan 3 alareunassa näkyy mobiilisovelluksen footer-valikko, jota käytetään eri sivujen välillä navigointiin. Footer-valikkoa käytetään tärkeimpien ja useimmin käytettävien sivujen välillä navigointiin. Footerissa tulisi olla korkeintaan viisi sivua, muuten painikkeista joudutaan tekemään liian pieniä ja valikosta tulee epäselkeän näköinen ja epämiellyttävä käyttää (Babich, 2016). Tämän navigointitavan suosio mobiilisovelluksissa johtuu siitä, että useimmat käyttäjät voivat selata sovellusta muuttamatta otettaan laitteesta, mikä parantaa sovelluksen helppokäyttöisyyttä. Kuvassa 3 näkyy myös yläosassa sijaitseva navigointivalikko, tätä tapaa käytetään yleensä yhdessä jonkun muun navigointitavan kanssa. Tämä navigointitapa on lähes yhtä hyödyllinen kuin footer -navigointi, mutta suuremmilla näytöillä käyttäjän on usein vaihdettava otettaan tai käytettävä molempia käsiään yltääkseen painikkeisiin (Shaar, S 2024).

KUVA 4. Kuvakaappaus Iltalehti sovelluksen valikosta 13.02.2025

Iltalehti sovelluksessa alapalkin valikon avaaminen (kuva 3) tuo esiin lisää navigointivaihtoehtoja. Tässä valikossa näytetään ne sivut, joita käyttäjä tarvitsee harvemmin sovellusta käyttäessään. Tämän tyyliset valikot auttavat järjestämään sisältöä selkeästi ja vähentämään visuaalista kuormitusta (Shaar, S. 2024).

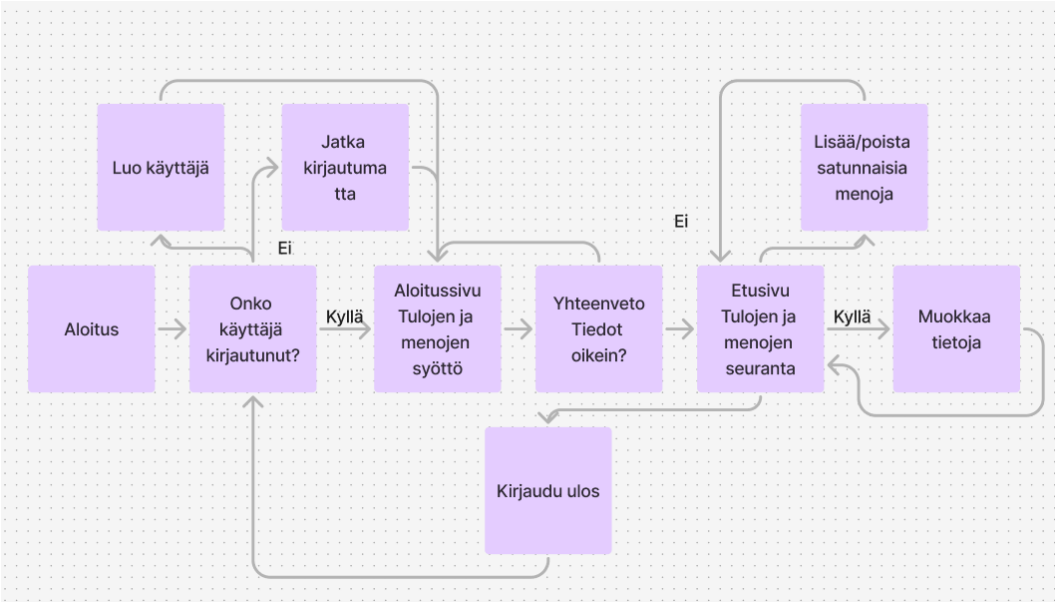
3.2 Käyttäjäkokemus

Käyttäjäkokemus tarkoittaa, miten käyttäjä kokee ja hyödyntää tuotetta. Se ei rajoitu pelkästään suoraan vuorovaikutukseen tuotteen kanssa, vaan kattaa myös sen, kuinka hyvin tuote tukee käyttäjän tehtävän tai tavoitteen suorittamista (productplan). Käyttäjäkokemus on ratkaiseva tekijä sovelluskehityksessä, sen tehtävänä on tehdä tuotteen käytöstä tehokasta ja nautinnollista. (Pham, A. 2024).

Vaikka käyttäjäkokemuksen ja käyttöliittymän kuvaukset kuulostavat samalta, ne ovat eri asioita. Käyttöliittymäsuunnittelussa keskitytään näkyviin elementteihin, joiden kanssa käyttäjä on vuorovaikutuksessa ja käyttäjäkokemus kattaa koko kokemuksen vuorovaikutuksen aikana (Pham, A. 2024). Käyttöliittymän suunnittelutapa vaikuttaa kokonaisvaltaiseen käyttäjäkokemukseen, joten käyttöliittymä on olennainen osa käyttäjäkokemusta (Vinney, C. 2025).

UX-suunnittelu (User Experience, eli käyttäjäkokemus -suunnittelu) keskittyy käyttäjän tarpeisiin ja käyttäytymisen ymmärtämiseen (Pham, A. 2024). Käyttäjäkokemustutkimuksilla, suunnittelija oppii mistä käyttäjä pitää, mitä ongelmia ja kipupisteitä he kohtaavat, ja miten he suhtautuvat sovelluksen käyttöön. Markkinaraon määrittämiseen voi olla hyödyllistä käyttää apuna SWOT-analyysia. Tutkimusten tulokset kootaan usein ostaja-, tai käyttäjäpersooniksi, jotka kuvaavat tarkasti kohdeyleisön eri tyyppejä. (figma)

Selvitettyään käyttäjän halut ja tavat, suunnittelija voi luoda IA:n, eli informaatioarkkitehtuurin tuotteelleen. Informaatioarkkitehtuuria käytetään visuaalisena pohjapiirroksena hahmottamaan olennaisen navigoinnin, sisältöhierarkkian, ominaisuudet ja vuorovaikutuksen. Yksi keskeinen IA työkalu on vuokaavio (kuva 5), jonka avulla kartoitetaan tärkeimmät käyttäjäpolut ja päätöspisteet. IA vuokaaviot auttavat tiimejä ymmärtämään silmäyksellä, miten tuotteen tulisi toimia ja missä on mahdollisesti aukkoja, jotka vaativat ominaisuuksia tai päivityksiä (figma).



KUVA 5. Budjettisovelluksen vuokaavio.

IA:n piirrettyä voidaan ruveta muuntamaan ideoista konkreettisia malleja, kuten luonnoksia ja prototyyppejä. Näitä malleja käytetään ideoiden testaamiseen, vaatimusten määrittämiseen, ja ominaisuuksien priorisoimiseen (figma).

UX-suunnittelijan työ ei ole ikinä täysin ohi. Sovellukset vaativat jatkuvasti päivityksiä pysyäkseen teknologian kehityksen mukana. Käyttäjien palautteesta ja back-end analytiikasta voi käydä ilmi, että joku prosessi sovelluksessa on liian hidas ja vaatii päivitystä (figma).

3.3 Käyttöliittymä ja käyttäjäkokemus käytännössä

Käytännön esimerkkinä voidaan käyttää oman asunnon suunnittelua ja käyttöä. Asunnon tapauksessa UI voisi tarkoittaa sisustusta, värejä, valaistusta, materiaaleja ja yksityiskohtia. Esimerkiksi olohuoneessa UI-suunnittelu näkyy sohvaryhmän värimaailmassa, seinien sävyissä sekä valaisimien muodosta ja materiaaleista. Sisustuksen, kuten myös UI:n tavoitteena on luoda visuaalisesti miellyttävä ympäristö, joka vetoaa käyttäjän aisteihin.

UX näkyy asunnon suunnittelussa esimerkiksi siinä, miten hyvin tilaa käytetään, ovatko toiminnot, kuten valokatkaisimet sijoitettu loogisesti ja kuinka mukavalta asuminen kokonaisuutena tuntuu. Keittiön tapauksessa UX-suunnittelu näkyy muun muassa näissä asioissa: ovatko kaapit sijoitettu ergonomisesti siten, että

niitä on helppo käyttää, onko työtasoilla riittävästi tilaa ruoanlaittoa varten, ja ovatko kodinkoneet ja pistorasiat sijoitettu järkeviin paikkoihin.

Yhteenveto

Vaikka asunto näyttäisi päällepäin hyvältä, se voi olla käytännössä hankalasti toteutettu, jos jääkaapin ovi ei avaudu täysin tai työtasot ovat liian matalalla. Kuten myös toimiva UX ilman visuaalista miellyttävyyttä voi tuntua tylsältä ja epämotivoivalta käyttää. Täydellinen suunnittelu huomioi sekä ulkonäön, että käyttökokemuksen, jotta asunto ei ainoastaan näytä hyvältä, vaan myös toimii hyvin arjessa.

4 TOTEUTUS

4.1 Projektimme optimointi

4.1.1 Alusta, ohjelmointi periaate ja kansio rakenne

Alustana käytettiin Expon versiota React-Nativesta sen helppouden ja ketterän kehityksen mahdollisuuksien takia. Ohjelmointikieleksi valittiin JavaScript, koska se on laajasti käytetty ja hyvin tuettu React-Native ja Expo dokumentaatioissa.

React-Native mahdollistaa modulaaristen ohjelmistokomponenttien kirjoittamisen, joita voidaan käyttää uudelleen. Tämä ohjelmointi periaate vähentää merkittävästi kirjoitettavan koodin määrää ja ohjelman kokoa.

Kirjoitettavat koodit jaoteltiin kansioihin komponentit ja palvelut. Komponentit sisältävät käyttöliittymä elementit, sekä niiden omat toiminnallisuudet. Kansiossa oli myös alakansio, jossa on pienempiä toteutuksia, joita käytettiin useammassa paikassa. Palvelut kansiossa on yleisiä toimintoja, kuten esim. navigointi eri komponenttien välillä, tarvittava konfiguraatio tietokanta yhteyden luomiseksi ja laskuri.

4.1.2 Kolmannen osapuolen kirjastot

Applikaation suorituskyvyn parantamiseksi päätettiin leikata kolmannen osapuolten kirjastojen käytöstä mahdollisuuksien mukaan. Esimerkiksi React-Native projekteissa paljon käytetty react-navigation/native-stack kirjasto korvattiin itse toteutetulla yksinkertaisemmalla navigaatiolla. Projektissa todettiin, että komponenttien väliset navigointitarpeet olivat riittävän vähäiset omalle toteutukselle. Ratkaisulla myös haluttiin säästää tallennustilaa ja parantaa sovelluksen resurssitehokkuutta. Joitain kirjastoja silti jouduttiin käyttämään, sillä osaaminen omaan toteutukseen oli puutteellinen. Näitä kirjastoja olivat:

- react-native-chart-kit, jonka avulla etusivulle voitiin lisätä vuokaavio.

- expo-vector-icons, joka tarjosi kattavan kirjaston erilaisia ikoneita käyttöliittymän toteutukseen.
- react-native-async-storage/async-storage, jonka avulla saadaan käyttäjän tili pysymään kirjautuneena sovellukseen. Tämä toteutetaan tallentamalla tilin autentikointitiedot välimuistiin rekisteröinnin tai kirjautumisen yhteydessä.

Omien taitojen ei katsottu riittävän tällaisten kirjastojen toteuttamiseen, minkä lisäksi oman ratkaisun kehittäminen olisi erittäin aikaa vievää ja työlästä. Kiireessä huolimattomasti toteutettu kirjasto voisi pahimmillaan heikentää suorituskykyä ja sisältää piilovirheitä, joita ei välttämättä havaittaisi.

4.1.3 Optimaalinen tapa esittää tietoa

Esimerkkiprojektissa on tarve renderöidä mahdollisia isompia listoja tietoa käyttäjän menoista, tuloista ja satunnaisista maksuista. Reactissa listojen renderöintiin on monia tapoja. Yksi yleisimmistä tavoista on käyttää map funktiota, joka iteroi listan jokaisen kohdan näkyviin käyttäjälle ottamatta huomioon onko listan kohta ruudussa vai ei. Tehokkaampi tapa on käyttää React-Nativen omaa FlatList komponenttia. FlatList komponentti renderöi käyttäjälle näkyviin listasta vain ne kohdat mitkä sattuvat olemaan näytöllä sillä hetkellä. Tätä kutsutaan laiskaksi lataukseksi (lazyloading). Komponentilla on myös sisään rakennettu rullaus ominaisuus, joten ohjelmoijan ei tarvitse itse tehdä erikseen ominaisuutta, jolla voi rullata isomman listan päästä päähän.

4.1.4 Modulaarisuus

Projektin toteutuksessa pyrittiin jakamaan pääsivujen sisältö pienemmiksi, itsenäisiksi komponenteiksi tarpeen mukaan. Tämä mahdollisti komponenttien uudelleenkäytön eri tilanteissa.

Esimerkiksi rekisteröinnin jälkeen käyttäjä ohjataan sivulle, jossa hänen tulee syöttää tietoja tuloistaan ja menoistaan. Jotta prosessi olisi selkeä ja skaalautuva, jokainen vaihe toteutettiin omana komponenttinaan. Näitä samoja komponentteja voitiin hyödyntää myös silloin, kun käyttäjä halusi myöhemmin muokata

antamiaan tietoja. Hajautetut komponentit sisälsivät oman tyylinsä, toiminto logiikkansa ja käyttöliittymän.

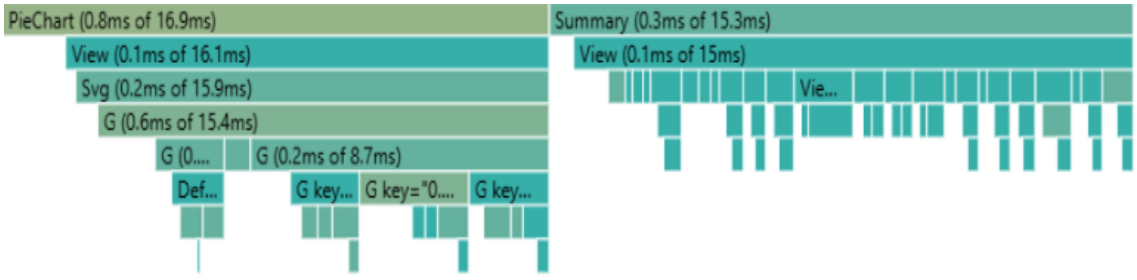
Toisenlaisia modulaarisia komponentteja olivat navigointipainikkeet, joilla käyttäjä siirtyy sovelluksen eri osiin. Sen sijaan, että jokainen nappi olisi toteutettu erikseen eri sivuille, luotiin erillinen tiedosto, jossa määriteltiin painikkeiden toiminnallisuus. Kutsuttaessa painikekomponenttia sille voitiin antaa nimi tai emoji, haluttu funktio sekä tyyli.

Modulaarinen toteutus tapa siis vähentää kirjoitettavan koodin määrää, sillä samaa osaa voidaan käyttää useassa kohdassa. Lisäksi koodin jatkokehittäminen ja muokattavuus paranevat, koska koko sovellus ei ole riippuvainen yhdestä yksittäisestä osastaan.

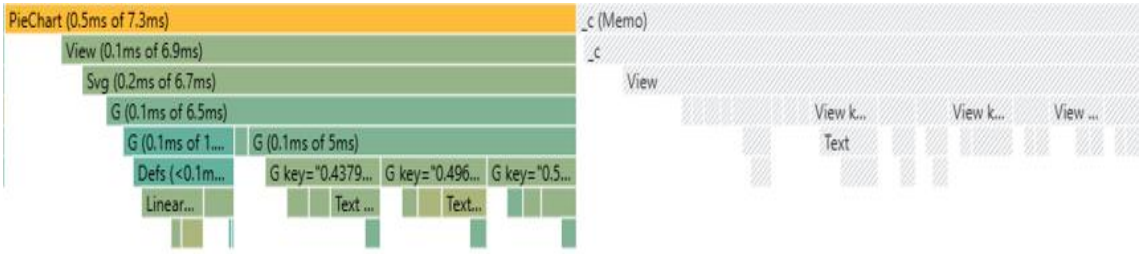
4.1.5 Optimoinnin testaus

React Native -sovelluksen suorituskykyä analysoitiin hyödyntämällä React Native DevToolsia, joka mahdollistaa sovelluksen ajonaikaisten tapahtumien, kuten uudelleenrenderöintien ja komponenttien latausaikojen, profiloinnin. Testauksen aikana tunnistettiin useita tekijöitä, jotka aiheuttivat tarpeettomia uudelleenrenderöintejä tai hidastivat komponenttien latautumista.

Profiloinnin tulokset (kuva 6 vasen lohko) osoittivat, että etusivulla olevan ympyrädiagrammin piirtäminen kesti 16.9 millisekuntia. Viiveen syynä oli se, että kaavio sai tietonsa funktiosta, joka suoritettiin aina komponentin uudelleen piirtyessä. Tämän optimointiin hyödynnettiin useMemoa jonka käyttöönotto puolitti piirtoajan 7.3 millisekuntiin (kuva 7 vasen lohko).



KUVA 6. Applikaation profiloinnin aikana saatuja tietoja



KUVA 7. UseMemon käyttöönotto.

Toinen merkittävä havainto liittyi yhteenvetokomponenttiin (kts. Kuvat 6 ja 7 oikea lohko), joka esittää käyttäjän tietoja. Komponentin alkuperäinen piirtoaika oli 15.3 millisekuntia. Kun koko komponentti sijoitettiin useMemo-hookin sisälle, piirtoaika saatiin poistettua kokonaan. Näiden optimointien avulla sovelluksen suorituskykyä parannettiin vähentämällä turhia uudelleenrenderöintejä ja nopeuttamalla komponenttien latautumista.

4.2 Käyttöliittymän toteutus

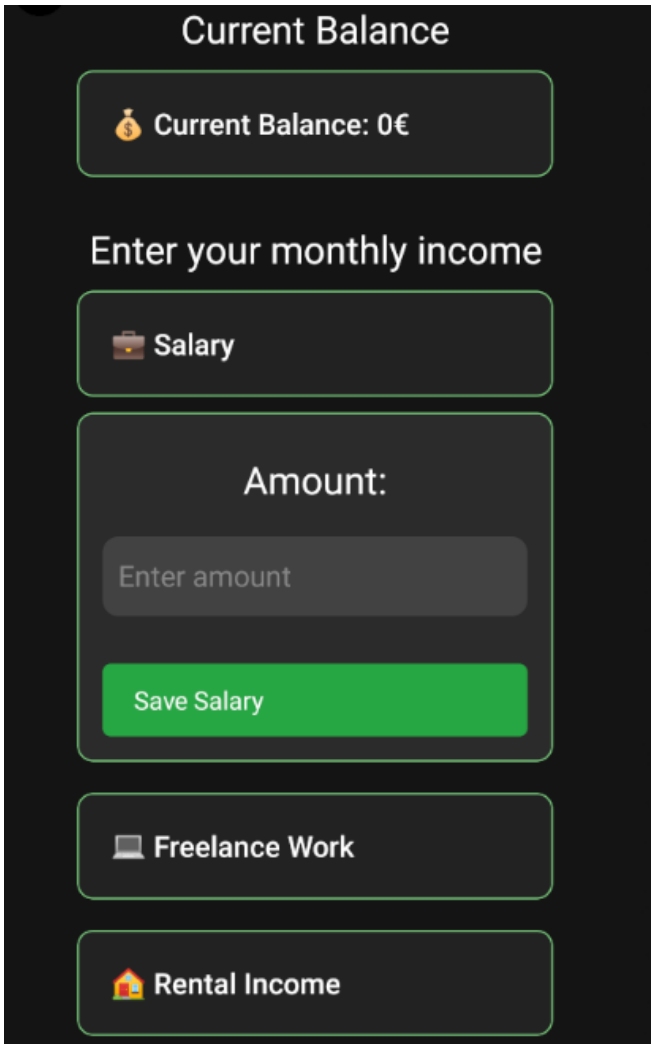
4.2.1 Sovelluksen värimaailma

Sovelluksessa päädyttiin käyttämään tummaa teemaa, koska se tarjosi sekä käyttömukavuutta että optimointiin liittyviä etuja. Kuten teoriaosuudessa sanottiin, tummien värien käyttö voi pienentää virrankulutusta etenkin OLED-näytöillä, mikä teki siitä houkuttelevan vaihtoehdon sovelluksen suunnittelussa. Näiden tekijöiden perusteella tumma teema valittiin sovellukseen.

4.2.2 Interaktiivisuus

Käyttöliittymässä on huomioitu selkeä ja sujuva vuorovaikutus. Toiminnallisia elementtejä sovelluksessa ovat napit ja pudotusvalikot, jotka helpottavat navigointia ja tietojen syöttämistä.

Nappeja käytetään tiedon tallentamiseen ja siirtymiseen sivujen välillä. Niiden koko ja sijoittelu on suunniteltu niin, että ne ovat helposti tunnistetavissa ja vaivattomia käyttää. Ne ovat riittävän isoja ja erillä toisistaan vähentäen virhepainallusten mahdollisuutta.

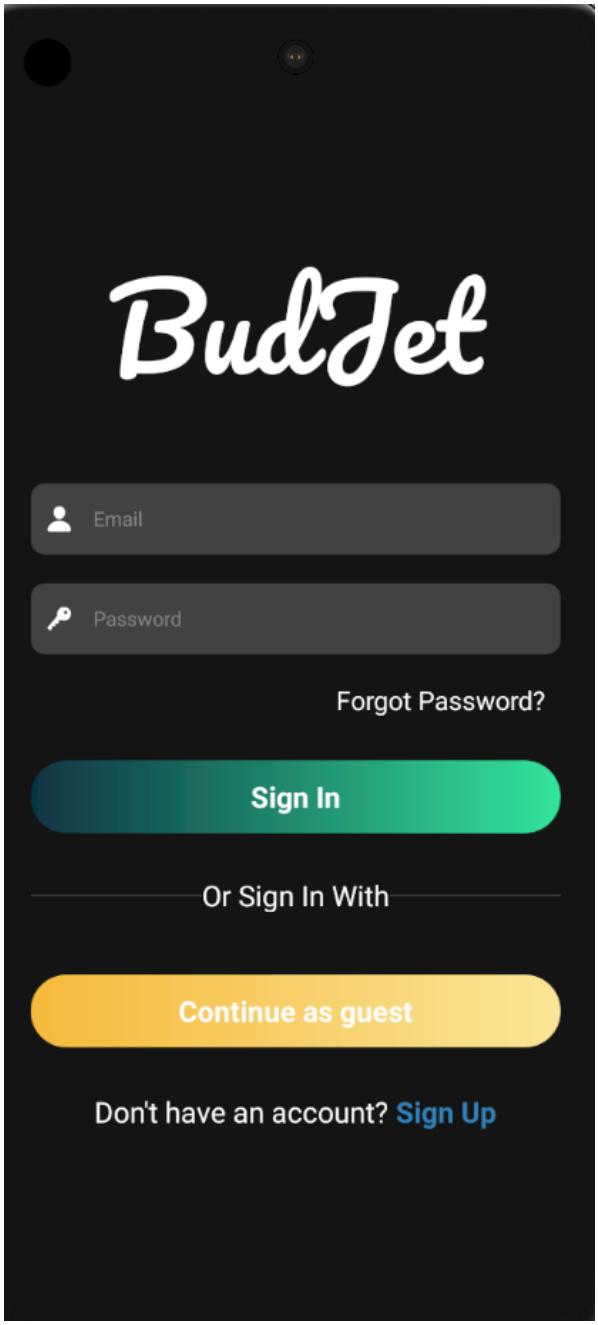


KUVA 8. Pudotusvalikot

Pudotusvalikot (kuva 8) auttavat näytön tilahallinnassa. Ne estävät turhan tiedon näyttämistä ja pitävät käyttöliittymän selkeänä. Valikot avautuvat ja sulkeutuvat sujuvasti käyttäjän syötteestä.

4.2.3 Fontit

Sovelluksessa käytetään järjestelmän oletusfonttia, joka helpottaa luettavuutta ja takaa yhteensopivuuden eri laitteilla. Oletusfontti parantaa myös käyttöliittymän sujuvuutta, sillä se mukautuu käyttäjän asetuksiin, sekä tukee eri kieliä ja merkistöjä ilman erillistä säätöä.

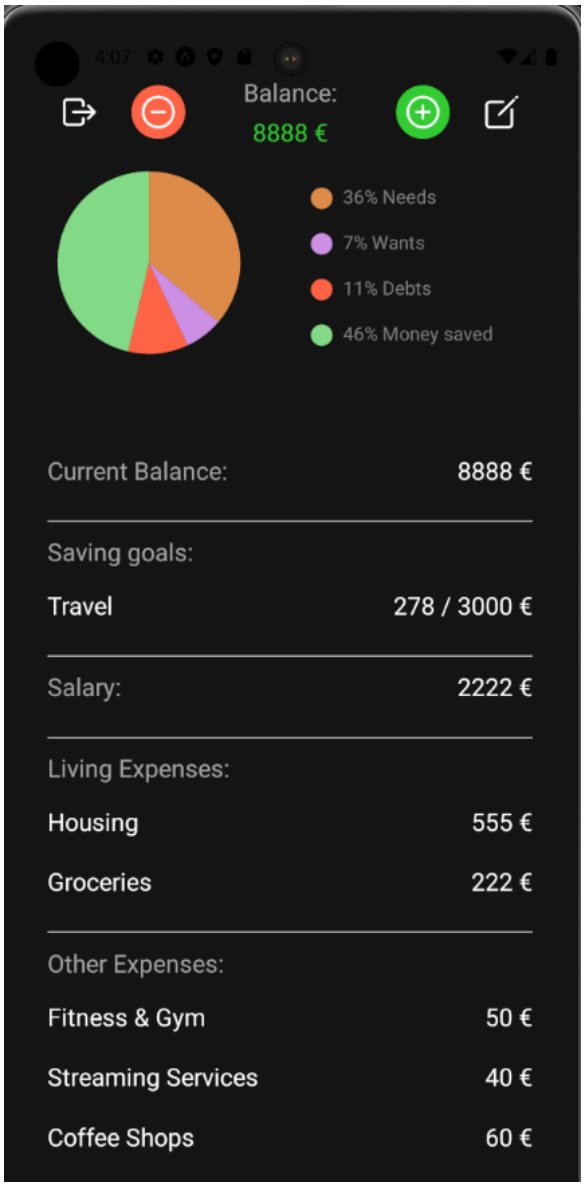


KUVA 9. Kirjautumissivu

Koristeellisempaa Pacifico-fonttia käytetään ainoastaan sovelluksen otsikossa kirjautumissivulla (kuva 9). Sans-Serif tyylinen fontti antaa otsikolle tunnistettavan ja persoonallisen ilmeen.

4.2.4 Asettelu

Käyttöliittymä on aseteltu selkeäksi ja loogiseksi, jotta tarvittavat toiminnot löytyvät helposti. Elementit on sijoitettu johdonmukaisesti, ja näkymät jaettu selkeisiin osiin helpottaen hahmottamista ja navigointia. Yläreunan (kuva 10) vasemmanpuoleinen painike kirjaa käyttäjän ulos sovelluksesta, oikeassa laidassa sijaitsevasta painikkeesta siirrytään tietojen muokkaustilaan. Näiden välissä olevista - ja + -napeista voidaan manuaalisesti muokata tämänhetkistä saldoa yllättävien menojen ja tulojen varalta.



KUVA 10. Sovelluksen päänäkymä.

Ympyräkaavio (kuva 10) koostuu käyttäjän tuloista, siihen on laskettu käyttäjän syöttämistä tiedoista budjetti tarpeisiin (needs, oranssi), haluihin (wants, lila), velkoihin (debts, punainen), ja näiden jälkeen säästetty raha (money saved, vihreä). Tarpeellisiin menoihin kuuluu kaikki asuinmenot kuten vuokra, ruoka, laskut. Haluihin kuuluu muut menot, näitä voi olla esimerkiksi suoratoistopalveluiden tilaukset, harrastukset ja muut menot, jotka eivät ole välttämättömiä elämiseen. Punaisella näkyvä velkojen osuus kaaviossa on lainojen takaisinmaksun yhteenlaskettu summa kuukaudessa. Säästetty raha on mitä tuloista jää, kun siitä vähennetään tarpeet, halut, ja velat.

Kaavion jälkeen listataan vielä tulot, menot ja säästötavoitteet taulukossa. Tämä selkeä esitystapa auttaa hahmottamaan taloudellisen tilanteen paremmin ja seuraamaan tavoitteiden edistymistä, mikä tukee Hannahin (2023) mainitsemaa periaatetta siitä, että mobiilisovelluksia käytetään usein lyhyissä hetkissä, ja tiedon tulisi olla helposti sisäistettävässä muodossa.

Tilankäytössä on keskitytty tasapainoon, jossa sisältö pysyy helposti luettavana mutta käyttöliittymä ei tunnu ahtaalta. Käyttöliittymän selkeyttä tukevat suunnittelupäätöksen, kuten riittävät marginaalit ja loogisesti sijoitetut elementit, mikä on linjassa Figman (2024) ohjeiden kanssa sovellusten ulkoasun selkeydestä. Näin varmistetaan, että käyttäjä löytää etsimänsä nopeasti, kuten Hannah (2023) korostaa kohderyhmän ymmärtämisen merkityksestä.

5 POHDINTA

Raportin aiheena oli budjettisovelluksen optimointi ja käyttöliittymäsuunnittelu. Tarkastelimme kuinka nämä kaksi asiaa vaikuttavat sovelluksen toimivuuteen ja käyttökokemukseen.

Ensimmäisenä osa-alueena oli tarkoitus syventää osaamista React Native -sovellusten optimoinnista, sekä soveltaa opittua käytännössä. Työn aikana kohdattiin haasteita suorituskyvyn parantamisessa, mutta aihetta opiskelemalla sovelluksen tehokkuutta onnistuttiin parantamaan. Lopputuloksena syntyi optimoitu mobiilisovellus, joka vastasi asetettuja tavoitteita. Jatkokehityksen kannalta sovelluksen testaaminen laajemmalla käyttäjäryhmällä ja syvällisempi analyysi eri optimointitekniikoiden vaikutuksesta voisivat tarjota arvokasta lisätietoa suorituskyvyn mahdollisista kyykkäyksistä.

Käyttöliittymäsuunnittelu oli toinen tärkeä osa-alue. Sen tarkoituksena oli tehdä sovelluksesta mahdollisimman helppokäyttöinen ja intuitiivinen. Sovelluksen käyttöliittymän suunnittelussa otettiin huomioon hyvän käyttöliittymäsuunnittelun periaatteet, kuten selkeä navigointi, tietojen visualisointi ja yksinkertaiset syöttö- ja muokkausvaihtoehdot. Käyttöliittymä suunniteltiin niin, että se on helposti omaksuttavissa ja tarjoaa käyttäjille kaikki tarvittavat työkalut talouden hallintaan ilman turhia monimutkaisuuksia.

Raportin laatiminen antoi syvemmän ymmärryksen siitä, kuinka tärkeitä optimointi ja käyttöliittymä suunnittelu ovat sovelluskehityksen kannalta sekä kuinka ne vaikuttavat toisiinsa. Hyvin optimoitu sovellus parantaa käyttökokemusta, ja hyvä käyttöliittymä voi tehdä optimoinnin vaikutukset käyttäjälle näkyviksi.

Jatkokehitysideoita sovellukselle voisi olla tuki eri kielille ja valuutoille. Pidemmälle kehitetyssä sovelluksessa voisi myös nähdä budjetit kuukausittain, ja aiemmalta kuukaudelta yli jäänyt raha tuotaisiin seuraavan kuukauden budjettiin mukaan. Säästetty raha voitaisiin myös automaattisesti jakaa

säästötavoitteiden välillä. Sovellus voisi auttaa luomaan strategioita lainan maksuun.

Yhteenvetona voidaan todeta, että optimointi ja käyttöliittymäsuunnittelu ovat molemmat keskeisiä tekijöitä sovelluskehityksessä. Tämä raportti on auttanut ymmärtämään, kuinka tärkeää on tasapainottaa molempia tekijöitä ja kuinka niiden yhdistelmä parantaa sovelluksen toimivuutta ja käyttäjäkokemusta.

LÄHTEET

1. Caupolican D. 2023. Programming Optimization, Codeacademy. Hakupäivä: 02/11/2024. https://www.codecademy.com/resources/docs/general/programming-optimization

2. Bogatinov L. 2023. PROGRAMMING: A Guide to Code Optimization Techniques. Medium. Hakupäivä: 27/01/2025. https://medium.com/@bogatinov.leonardo/programming-a-guide-to-code-optimization-techniques-e0babca0d49b

3. IBM. 2025. Memory-leaking programs. IBM. Hakupäivä: 27/01/2025 https://www.ibm.com/docs/en/aix/7.2?topic=performance -memory-leaking-programs

4. Dias A. 2024. Mobile App Performance Optimization Strategies: A Detail Guide. Hakupäivä 13/02/2025 https://www.weblineglobal.com/blog/app-performance-optimization-strategies/

5. Oliveira D. 2023. Optimizing performance in React Native: Tips and techniques. Thoughtbot. Hakupäivä: 17/01/2025. https://thoughtbot.com/blog/optimizing-performance-in-react-native-tips-and-techniques

6. React O. 2024. How to improve React Native performance? Medium. Hakupäivä 20/01/2025 https://medium.com/@onix_react/how-to-improve-react-native-performance-adab7347e78c

7.React Native. 2025. Performance. Hakupäivä 20/01/2025 https://reactnative.dev/docs/performance#common-sources-of-performance-problems

8. Msowski. 2023. Ace Your React Native Interview with these Questions. Medium. Hakupäivä 28/07/2023 https://medium.com/@msowski/ace-your-react-native-interview-with-these-questions-fe917e924285

9. Tidwell, J., Brewer, C., Valencia, A. 2019. Designing Interfaces 3rd Edition, O'reilly Media, Inc. Hakupäivä: 31/10/2024. https://learning.oreilly.com/library/view/designing-interfaces-3rd/9781492051954/

10. Hannah, J. 2023. Designing for mobile: 5 best practices for UI designers. Hakupäivä: 14/02/2025. https://www.uxdesigninstitute.com/blog/designing-for-mobile-best-practices/

11. Apple Developer: UI Design Dos and Don'ts. Hakupäivä: 14/02/2025. https://developer.apple.com/design/tips/

12. Figma, UI vs UX: What's the difference between UI and UX. Hakupäivä: 22/01/2025. https://www.figma.com/resource-library/difference-between-ui-and-ux

13. Shaar, S. 2024. Mobile navigation: patterns and examples. Justinmind. Hakupäivä: 13/02/2025. https://www.justinmind.com/blog/mobile-navigation/

14. Babich, N. 2016. The Golden Rules of Bottom Navigation Design. Smashing Magazine. Hakupäivä: 12/02/2025. https://www.smashingmagazine.com/2016/11/the-golden-rules-of-mobile-navigation-design/

15. Kohler, T., Zhang, A. 2023. Dark Mode: How Users Think About It and Issues. Hakupäivä: 01/02/2025. Click or tap here to enter text.https://www.nngroup.com/articles/dark-mode-users-issues/

16. ProductPlan, User Experience. Hakupäivä: 22/02/2025. https://www.productplan.com/glossary/user-experience/

17. Pham, A. 2024. SAVYCOM: The Importance Of User Experience In Software Development. Hakupäivä: 22/02/2025. https://savvycomsoftware.com/blog/user-experience-in-software-development/#

18. Vinney, C. 2025. UX Design Institute: What is UX design? Hakupäivä: 22/02/2025. https://www.uxdesigninstitute.com/blog/what-does-ux-stand-for/#What_is_user_experience_UX_design