Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2019

Toni Leivo

LAPSILLE SUUNNATTU TEHTÄVÄTAULU IPAD-SOVELLUKSENA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikan koulutus

2019 | 44 sivua

Toni Leivo

LAPSILLE SUUNNATTU TEHTÄVÄTAULU IPAD-SOVELLUKSENA

Tämän opinnäytetyön tarkoituksena on kehittää lapsille ja heidän vanhemmilleen sovellus, joka korvaa paperille kirjattavan tehtävätaulun digitaalisena versiona. Sovellus on kehitetty Applen iPad-laitteistolle.

Sovellus tehtiin helpottamaan vanhempia kirjaamaan lapsien tehtävien tuloksia digitaaliseen ympäristöön ja saada lapset motivoitumaan tehtävistä, joita vanhemmat antavat lapsille.

Sovelluksen kehittämiseen käytettiin Xcode-sovellusta ja Swift-ohjelmointikieltä. Sovelluksessa käytettiin Core data -tietojen tallennusjärjestelmää ja MVVM-ohjelmointimallia. Sen ulkoasu toteutetaan App Storessa olevien muiden lapsille suunnattujen sovellusten kaltaisesti.

Sovellus onnistui, ja nyt vanhemmat sekä tarhat ja päiväkodit voivat ottaa käyttöönsä tämän uuden sovelluksen lapsien kehittymisen edesauttamiseksi.

Sovelluksen ansiosta vanhemmat saavat kirjattua vain yhden napin painalluksella lapselle annetun tehtävän tehdyksi. Tämä nopeuttaa vanhempia kirjaamaan tulokset ja paperilappuja ei ole enää joka paikassa.

ASIASANAT:

iOS, Swift, Xcode, Apple, Core data, MVVM, iPad

BACHELOR´S | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme in Information and Communications Technology

2019 | 44 pages

Toni Leivo

IPAD TASK BOARD APPLICATION FOR CHILDRENS

The objective of this thesis was to develop an application for children and their parents to replace the paper written task board with a digital version of it. The application was developed for Apple iPad devices.

The application has been created to help parents to write the children's task results in a digital environment and make the children motivated to complete the assigned tasks.

The application was developed with the Xcode program and the Swift programming language. The application used Core Data as data storage and the MVVM design pattern. The appearance of the application was designed to be similar to other children-focused applications in the App Store"?

The result was a completed application and now parents and day care centers can use this new application to monitor their children's progress.

Because of this application, parents can log tasks that are given to children with only one button click. Parents can speedily keep a log of their children's results with no need for paper anymore.

KEYWORDS:

iOS, Swift, Xcode, Apple, Core data, MVVM, iPad

SISÄLTÖ

KÄYTETYT LYHENTEET

Apple	Yhtiö, joka valmistaa IT-alan tuotteita ja käyttöjärjestelmiä
App Store	Applen kehittämä sovellus, joka toimii sovelluksien sovelluskauppana
iOS	Applen kehittämä käyttöjärjestelmä
iOS Simulator	Xcoden sisältämä työkalu iOS-sovelluksien simulointiin
IPad	Applen kosketusnäytöllinen taulutietokone
Interface Builder	Xcode-ohjelmiston mukana tuleva käyttöliittymän rakennustyökalu
MVVM	Model-View-ViewModel (malli-näkymä-näkymä malli) ohjelmistoarkkitehtuurityyli
Navigation controller	Xcoden ominaisuus, view controllerin yläreunaan tulee yläpalkki, jonka avulla pystyy suunnistamaan sovelluksen eri näkymissä ja pystyy lisäämään nappeja yläpalkkiin
Storyboard	Xcoden ominaisuus mihin saa lisättyä käyttöliittymän ulkoasun elementtejä
Swift	Xcode:ssa käytettävä ohjelmointi kieli sovelluksien tekemiseen
Tab bar controller	Xcoden ominaisuus, joka lisää view controllerin alareunaan alapalkin. Tästä valikosta pystyy siirtymään eri näkymiin
View Controller	Xcoden ominaisuus, johon lisätään yhden näkymän ulkoasu
Xcode	Applen kehittämä ohjelmointiympäristö

1 JOHDANTO

Tämän opinnäytetyön lähtökohtana on kehittää lapsille ja heidän vanhemmilleen sovellus, joka korvaa paperille kirjattavan tehtävätaulun digitaalisena versiona. Puhelinsovellus tuo tehtävälistan nykypäivään, ja se toteutetaan Applen iPad-laitteille.
Sovellus on tehty lapsien motivoinnin edistämiseksi. Lapsen huoltaja antaa lapselle tehtävän ja lapsen toteutettua tehtävän, huoltaja merkitsee sovellukseen tehtävän tehdyksi. Suoritetuista tehtävistä lapsi saa pisteitä. Jokainen lapselle kertynyt piste kerääntyy myös tulostaulukkoon, josta lapsi myöhemmin pystyy tarkastelemaan omaa aktiivisuuttaan tehtävien suorituksessa. Lisäksi lapset samassa perheessä tai päiväkotiryhmissä voivat kisata keskenään siitä, kuka on aktiivisin suorittamaan tehtäviään.
Opinnäytetyön alussa käydään läpi käytettyjä tekniikoita, työkaluja ja teoriaa niistä. Tämän jälkeen siirrytään sovelluksen suunnitteluun, toteutukseen ja viimeistelyyn. Lopuksi esitellään sovelluksen ja jatkokehityssuunnitelmat.
Opinnäytetyössä käsitellään kokonaisuudessaan sovelluksen kehityksen kulkua suunnittelusta julkaisuun asti.

2 SOVELLUKSEN KEHITYSPROSESSI

2.1 Projektisuunnittelu

Opinnäytetyössäni haluan luoda digitaalisen mahdollisuuden korvata nykyisen manuaalisen tavan kirjoittaa lapsen tehtäviä taululle tai paperille. Digitaalinen ratkaisu on opinnäytetyössäni iPad-sovellus. Tämän avulla on mahdollista luoda jopa ison päiväkotiryhmän kaikki tehtävätaulut yhteen paikkaan. Päiväkodin vetäjien ja vanhempien työ helpottuisi, eikä tarvitsisi käyttää enää kynää ja paperia tehtävien kirjaamiseen.
Ongelman ratkaisua tehdessä tulen ottamaan huomioon lapsen motivointiin liittyvät ongelmat ja hyödyt, sovellukseen tuotavan ulkoasun ja tehtävätaulun helppokäyttöisyyden. Sovellus tullaan itsessään toteuttamaan Applen ohjelmointiympäristössä, käyttäen luvussa 3 esiteltyjä teknologioita.
Opinnäytetyön lopputuloksena tulisi olla käyttövalmis sovellus, johon pystyy lisäämään lapsia, tehtäviä ja tauluja. Tauluihin tulisi pystyä kirjaamaan tehtävät tehdyksi. Jokaisesta kirjatusta merkinnästä kertyy lapselle piste, joka tulee näkyviin pistetauluun. Pistetaulussa lapset pystyvät vertaamaan toisiaan, kilpailemaan keskenään ja katsomaan, kuka on ahkerin.

Lapsen motivoinnin keinojen selvitys

Opinnäytetyössäni toteutettavan sovelluksen päätarkoitus on edesauttaa lapsen motivaation pitämistä vanhempien antamiin tehtäviin ja tätä kautta myös kehittämään lasta tehtävien laadulla.
Lapsen keskittyminen ei yleensä ole parhaimmillaan, kun pitää tehdä lapsen mielestä tylsiä tehtäviä ilman mitään hyötyä. Lähdin ratkaisemaan tätä ongelmaa tutkimalla ensin, mitkä ovat hyviä tapoja motivoida lasta.
Sovellukseeni otan pääpiirteiksi seuraavat asiat. (Lapsen motivointi.)

1. Ota lapsen mielipide huomioon.
2. Tee yhdessä lapsen kanssa.
3. Aseta lapselle odotuksia ja kannustusta ja auta häntä tekemään hyvin se mitä hän tekee.
4. Huomaa lapsen onnistuminen.
5. Auta lasta huomaaman oma edistymisensä.
6. Huolehdi sinun ja lapsesi välisestä suhteesta.

Sovelluksessa lisättäessä tehtävä on toivottavaa, että vanhempi ottaa myös lapsen mielipiteen huomioon tehtävää luodessa ja mahdollisesti myös luo nämä lapselle annettavat tehtävät yhdessä. Näin lapsella on enemmän motivaatiota tehdä annettu tehtävä valmiiksi. Sovelluksessa on pisteytyslista, joka kannustaa ja huomioi lapsen valmiiksi saatuja tehtäviä. Lapsi voi myös kilpailla tämän avulla muiden lasten kanssa ja motivoituu. Sovellusta käyttäessä lapsen pitää olla vanhempiin kontaktissa, ja se edesauttaa yhteisiä suhteita.

Tulen ottamaan kaikki edellä mainitut piirteet huomioon sovellusta luodessani. Näin mahdollistan parhaimman mahdollisen lopputuloksen toteutettavalle sovellukselle.

2.2 Projektinhallinta

Projektinhallinta on tärkeää projektia tehdessä. Tähän vaikuttavat monet eri asiat ja yksi näistä asioista on projektinhallintamenetelmät. Suosituimmat menetelmät ovat vesiputousmalli ja ketterät menetelmät.
Vesiputousmallilla tarkoitetaan, että projektin tavoitteet ja aikataulu ovat selkeät alusta lähtien ja projekti etenee selkeässä järjestyksessä. Vesiputousmallissa hyvinä puolina on tarkka suunta, missä järjestyksessä projektia viedään eteenpäin. Huonoina puolina on, että kaikki pitää suunnitella ennen projektin aloitusta ja mahdollisiin konflikteihin pitää varautua etukäteen. Tämä menetelmä sopii täydellisesti minun opinnäytetyöni luonteeseen ja käytin tätä menetelmää luodessani sovellusta. Minulla oli tarkka visio siitä mitä rupean sovelluksessa ratkaisemaan, miten ulkoasun ja ongelmat tulen toteuttamaan. Tehtävät etenivät suorassa järjestyksessä. (Projektinhallinta menetelmät.)
Ketterät menetelmät on toinen suosittu projektinhallintamalli. Ketterillä menetelmillä tarkoitetaan, että projektia ei suunnitella etukäteen, vaan ruvetaan tekemään ja katsotaan mihin suuntaan projekti etenee. Hyvinä puolina tässä menetelmässä on, että voi muuttaa esimerkiksi ulkoasua tarpeiden mukaan myöhemmässä vaiheessa, jos projektin kulku etenee siihen suuntaan. Huonona puolena on, ettei aikataulua voi määritellä etukäteen. Tämä menetelmä ei sopinut minun opinnäytetyöhöni, koska minulla oli tarkka lopputulos valmiiksi ajateltu.

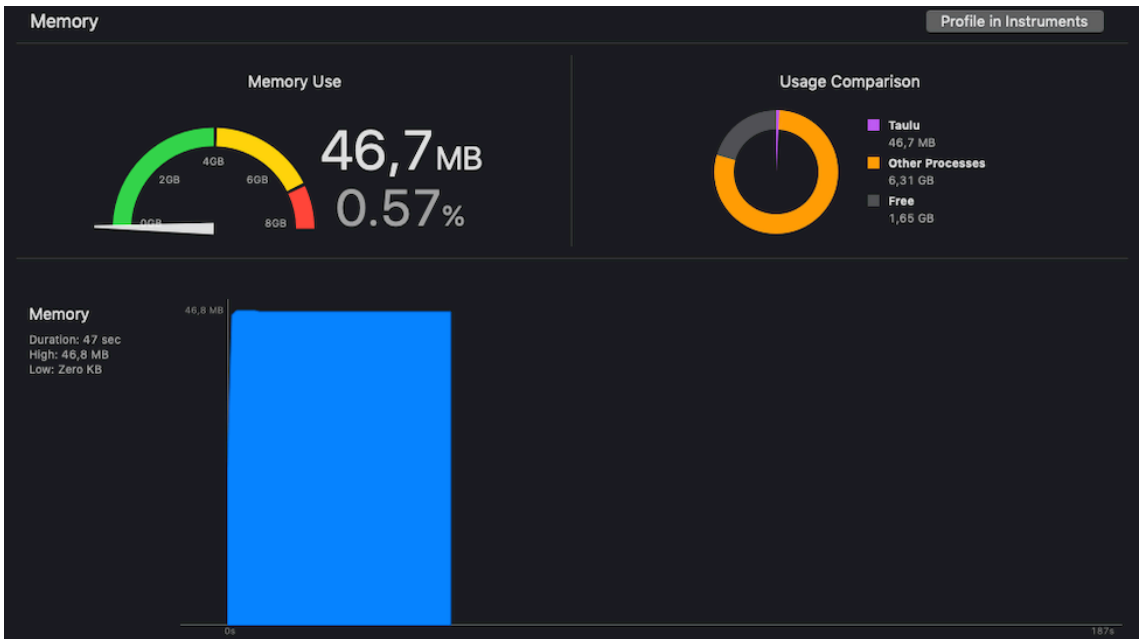
3 SOVELLUKSESSA KÄYTETYT TEKNOLOGIAT

3.1 iOS-käyttöjärjestelmä

Projektissa tuotettava sovellus toteutetaan iOS 12 -käyttöjärjestelmäversiolle ja iPad-laitteelle (6. sukupolvi). Valitsin iOS 12 -käyttöjärjestelmän sovelluksel leni sen tarjoamien etujen takia. iOS 12 on tällä hetkellä viimeisin Applen julkaisema käyttöjärjestelmäversio, ja se tarjoaa parhaimmat käyttöedut sovellukseni tekemiseen. IPad-laitteen valitsin sen ominaisuuksien ja hinnan takia. IPad on huomattavasti halvempi malli kuin iPad Pro. Tästä syystä sovellustani olisi helpompi markkinoida ja saada isomman asiakaskunta.
Apple julkaisi ensimmäisen iOS-käyttöjärjestelmä version vuonna 2007. Kolmena ensimmäisen vuonna Apple julkaisi käyttöjärjestelmän iPhone OS nimisenä, mutta vuonna 2012, kun iPad tuli markkinoille, Apple nimesi käyttöjärjestelmän iOS-nimellä. Tästä vuodesta lähtien Applen käyttöjärjestelmän nimi on ollut iOS. (iOS-versions 2019.)
Vuodesta 2007 vuoteen 2019 Apple on julkaissut yhteensä 12 eri iOS-käyttöjärjestelmä versiota. Vuosiin on mahtunut monta merkittävää muutosta sovelluksen kehityksessä. Esimerkiksi vuonna 2011 Apple julkaisi App Storen iPhone OS 2.0 -versiossa. Vuonna 2012 Apple julkaisi videoiden nauhoitus mahdollisuuden kamera -sovelluksen avulla. Vuonna 2013 Apple julkaisi FaceTime palvelun puhelimessa. Tämä mahdollistaa kahden ihmisen välisen keskustelun videopuheluna. Vuonna 2015 Apple julkaisi oman karttasovelluksensa. Vuonna 2018, syyskuussa Apple julkaisi Screen Time sovelluksen, jonka avulla iPhonen käyttäjä pystyi seuraamaan omaa käyttöaikaansa monella tavalla. (iOS-versions 2019.)

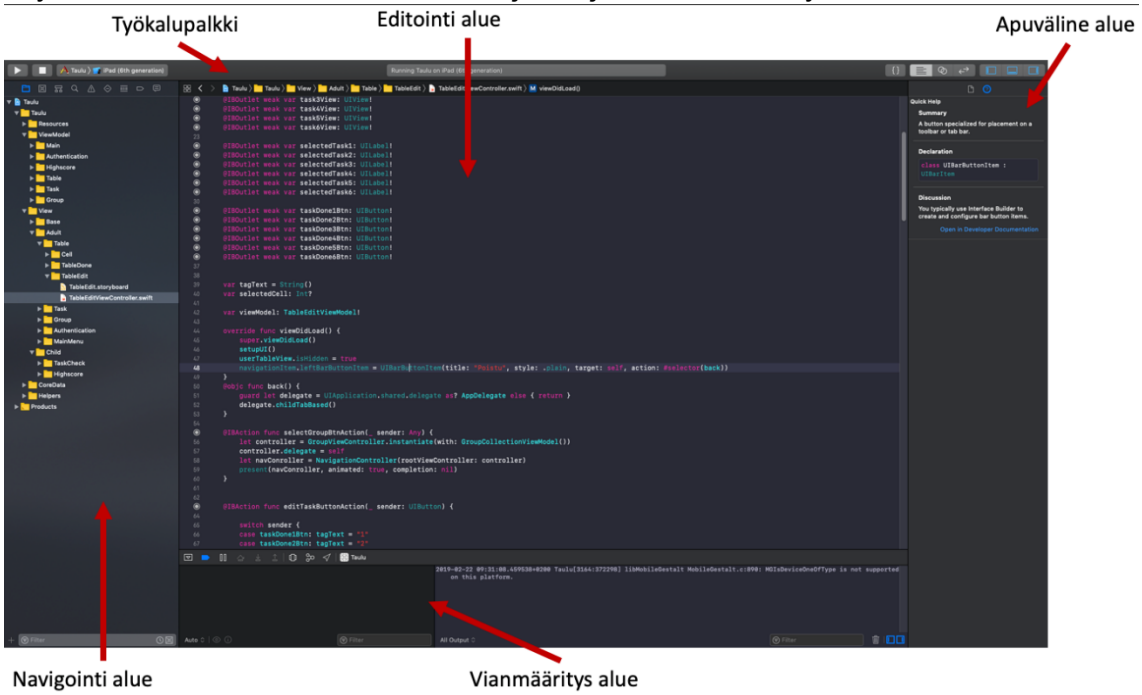
3.2 Xcode-sovellus

Projektini tekemisessä käytän Xcode 10 -sovellusta. Xcode-sovellus on Applen tarjoama ilmainen ohjelmointiympäristö iOS- ja Mac OS x -sovellusten kehittämistä varten. Kehittämisen lisäksi Xcode-sovelluksessa voi muokata lähdekoodia, paikallistamaan lähdekoodissa olevia virheitä ja korjaamaan näitä virheitä. Xcode-sovelluksella pystyy myös analysoimaan sovelluksen suorituskykyä ja simuloida sovelluksen toimintaa iOS-simulaattorilla.



Kuva 1. Esimerkki Xcoden suorituskykynäkymästä.

Xcode 10 -sovellus on ladattavissa ilmaisena App Storessa. Sovelluksen vaatimuksena on tietokone, jossa on Mac OS 10.13.6 -käyttöjärjestelmä. Xcoden käyttäminen koostuu sovelluksessa jaettujen osioiden käytöstä.



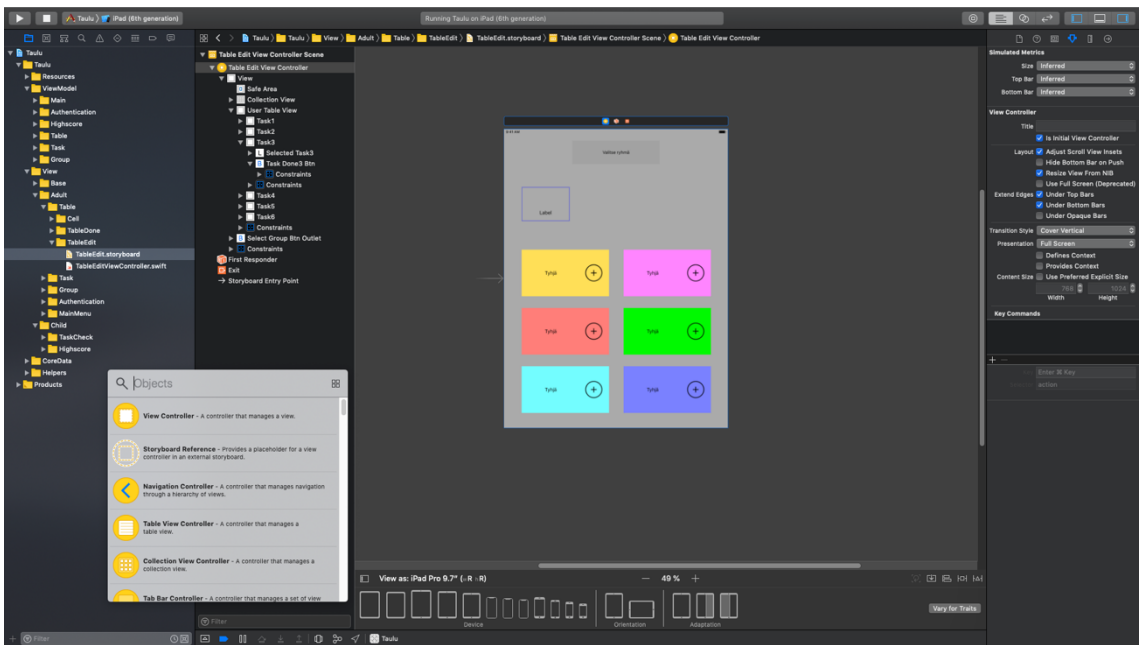
Kuva 2. Xcoden osiot.

Työkalupalkista ohjataan iOS-simulaattoria. Voidaan valita iPhone- tai iPad-versio, jossa simuloidaan projektia. Editointialue on paikka, johon tulee kaikki ohjelmoinnissa tuotettava koodi. Apuvälinealue on usein ohjelmoidessa käytetty alue. Sieltä näet muuttujan arvon. Storyboardissa pystyt asettamaan ulkoasuelementeille arvoja ja muokkaamaan näiden ominaisuuksia. Navigointialueella on kaikki sovellukseen lisätyt tiedostot. Niitä voi järjestää kätevästi kansioi-

den avulla. Storyboadilta voi myös hakea kaikkea, mitä on ohjelmoitu. Jos sovelluksessa on jokin virhe, kun yrittää käynnistää simulaattorin, niin se tulee myös tähän alueelle. Vianmääritys alue näyttää kaikki sovelluksessa olevat muuttujat samalla, kun projekti on simulaattorissa. Sovelluksen voi pysäyttää ja tarkastella muuttujia tarkemmin pysäytyskohdasta riippuen. (Xcode 2019.)

3.2.1 Interface Builder -suunnittelutyökalu

Interface Builder on Xcodessa oleva ulkoasun suunnittelutyökalu, jonka avulla pystyy tekemään ulkoasun kokonaan ilman ohjelmointia. Tämä suunnittelutyökalu toimii yksinkertaisella nappaa ja pudota -menetelmällä, eli valikosta otetaan elementti, pudotetaan se ulkoasun päälle ja se yhdistyy dynaamisesti ulkoasuun.



Kuva 3. Interface builderin näkymä.

Ulkoasun voi tehdä, joko storyboardiin tai xib-tiedostotyyppiin. Yhteen storyboardiin pystyy lisäämään monta eri view controlleria ja näkemään kokonaisuuden tällä tavalla. Xib-tiedostoihin pystyy lisäämään ainoastaan yhden ulkoasu elementin, joka myöhemmin pitää yhdistää view controlleriin joko ohjelmoiden tai interface builderin avulla. (Interface builder 2019.)

3.2.2 iOS Simulator

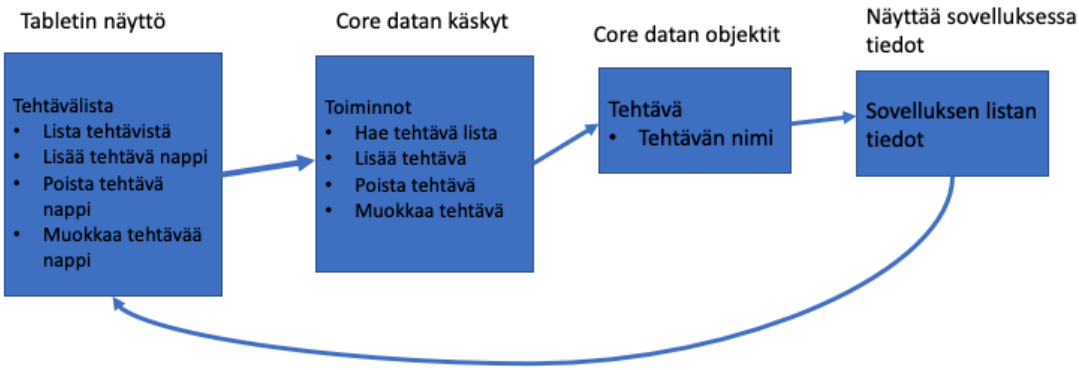
Kehitettäessä sovellusta pitää sitä testata jatkuvasti ja tämä onnistuu hyvin iOS Simulatorilla. iOS Simulator käynnistyy Xcoden työkalupalkin Run-painiketta painettaessa. Ensin sovellus rakentaa kaikki viimeisimmät muutokset ja sen jälkeen käynnistää sovelluksen simulaattorissa. Simulaattorin avulla pystyy tarkistamaan ja tarkkailemaan sovelluksen suorituskykyä ja mahdollisia ohjelmointi-

virheitä. Sovellus näkyy simulaattorissa samanlaisena, kuin se näkyisi iOS-laitteellakin. iOS Simulator mahdollistaa myös sovelluksen testauksen ulkoisella laitteella. (iOS-simulator 2018).

Kuva 4. iPad (6-sukupolvi) iOS simulaattorin aloitusnäkymä.

3.2.3 Core Data -lisäosa

Core Data on Xcoden tarjoama tiedostojen tallennusmekanismi. Tämän lisäosan avulla ohjelmoija pystyy asettamaan sovelluksen sisällä lisättävien tiedostojen tallentumisen suoraan sovellusta käyttävän käyttäjän puhelimen omaan muistiin.



Kuva 5. Core datan toimintatapa.

Core data mahdollistaa tiedon haun, lisäyksen, muokkauksen ja sen poiston. Valitsin Core datan sovellukseeni, koska siihen ei tarvita ulkoisia lisäosia ja se toimii ilman Internettiä. Core dataan tallennetut objectit on myös helposti siirrettävissä ulkoiselle palvelimelle, jos myöhemmin tulee tarve laajentaa sovellusta. (Core data 2019)

3.3 Swift-ohjelmointikieli

Swift on Applen kehittämä ohjelmointikieli, joka julkaistiin vuonna 2014. Swift-ohjelmointikieltä ennen Applella oli käytössä Objective-C ohjelmointikieli, joka on kehitetty vuonna 1980. (Swift 2019.)

Valitsin Swift ohjelmointikielen, koska Swift on paljon uudempi ja modernimpi verrattuna Objecitve-C:n. Swift vaikuttaa myös huomattavasti helpommalta opittavalta kuin Objective-C ohjelmointikieli. (Swift 2019.)

4 SOVELLUKSEN TOTEUTUS

4.1 Kartoitus

Lähdin aluksi liikkeelle miettimällä, mitä ominaisuuksia ja käyttökokemuksia tulisi olla sovelluksessa, joka on suunniteltu lapsille. Aloitin tämän tutkimalla App Storessa tällä hetkellä olevia lapsille suunnattuja sovelluksia.
Ensimmäinen asia, jonka huomasin, oli että lähestulkoon kaikki jo olemassa olevat sovellukset ovat todella värikkäitä. Painikkeet ovat värikkäitä ja eri värisiä, taustakuvat ovat todella värikkäitä ja erivärisiä ja animaatiohahmot ovat todella värikkäitä ja erivärisiä niiden tarkoitus oli myös selvästi hauskuuttaa käyttäjää ja pitää mielenkiintoa yllä.
Toisena huomasin, että sovellukset ovat todella simppeleitä ja mahdollisimman helppokäyttöiseksi tehty. Painikkeet olivat helpoissa paikoissa ja helposti painettaviksi tehty, valikot olivat helposti löydettävissä ja tosi selkeitä, kaikki teksti oli tosi selkeää ja helppolukuista ja myöskin kuvakkeet olivat tehty selkeiksi.
Kolmantena huomasin, että suurimmassa osassa sovelluksista on jokin mikä herättää käyttäjän mielenkiinnon ja pitää sitä yllä. Osassa sovelluksista löysin tilastotauluominaisuuden, joka kerää käyttäjäkunnan pisteitä ja näyttää kaikkien sovellusta käyttävien käyttäjien pisteet samassa paikassa ja se kenellä on eniten pisteitä, pääse korkeimmalla sijoituksella. Joissakin sovelluksissa taas on ominaisuus, jonka tarkoituksena on kasvattaa jotakin tuotetta tai kerätä jotakin tavaraa. Tämä luo mielenkiinnon ja päivittäisen aktiivisuuden sillä, että käyttäjä käy päivittäin tekemässä samat rutiinit sovelluksessa.
Neljäntenä huomasin, että suurimmassa osassa sovelluksista oli jokin tarkoitus kehittää, opettaa tai edesauttaa lasta elämässä. Joissakin sovelluksissa kehitettiin lapsen matemaattisia taitoja. Käyttäjän piti laskea jokin lasku ja täten pääsi etenemään sovelluksessa seuraavalle tasolle. Joissakin sovelluksista kehitettiin ohjelmointitaitoja. Käyttäjän piti valita oikea painike mikä vastaisi ohjelmointikoodissa komentoa ja täten käyttäjä oppii ohjelmoinnin perusajatuksen. Joissakin sovelluksista kehitettiin luovuutta. Käyttäjän pitää värittää erilaisia elementtejä tai hahmoja. Kuunnella musiikkia ja tehdä sitä itse. Nämä asiat olivat mielestäni tärkeimpiä asioita, joita lapsille tehtävässä sovelluksessa tulee ottaa huomioon.

4.2 Suunnittelu

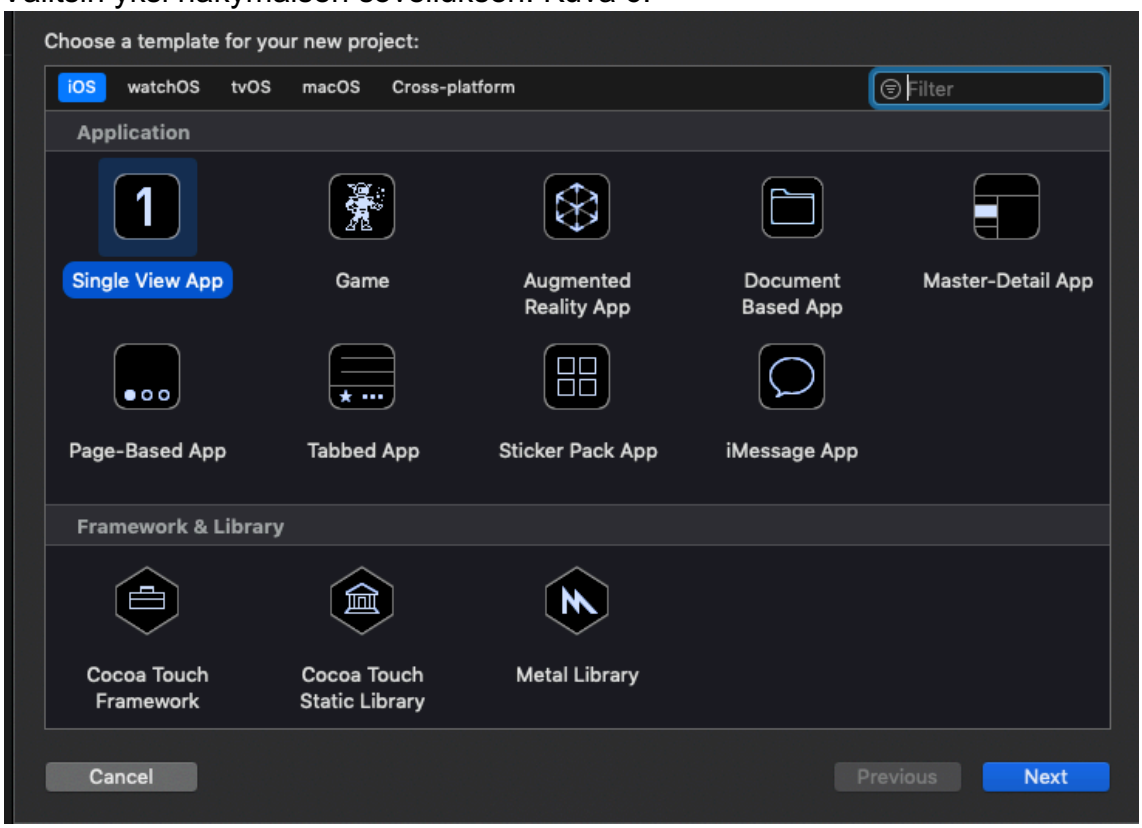
Lähdin miettimään omaa sovellustani ja mietin, mitä kaikkia ominaisuuksia lapsille suunnatuista sovelluksista voisin yhdistää omaan sovellukseeni. Päätin tarvitsemani seuraavat asiat sovellukseeni:

- jokin mikä koukuttaa lasta käyttämään sitä
- jokin mikä kehittää lasta elämässä tai edesauttaa lapsen elämistä
- värikkään ja siistin ulkoasun
- helppokäyttöiset toiminnot

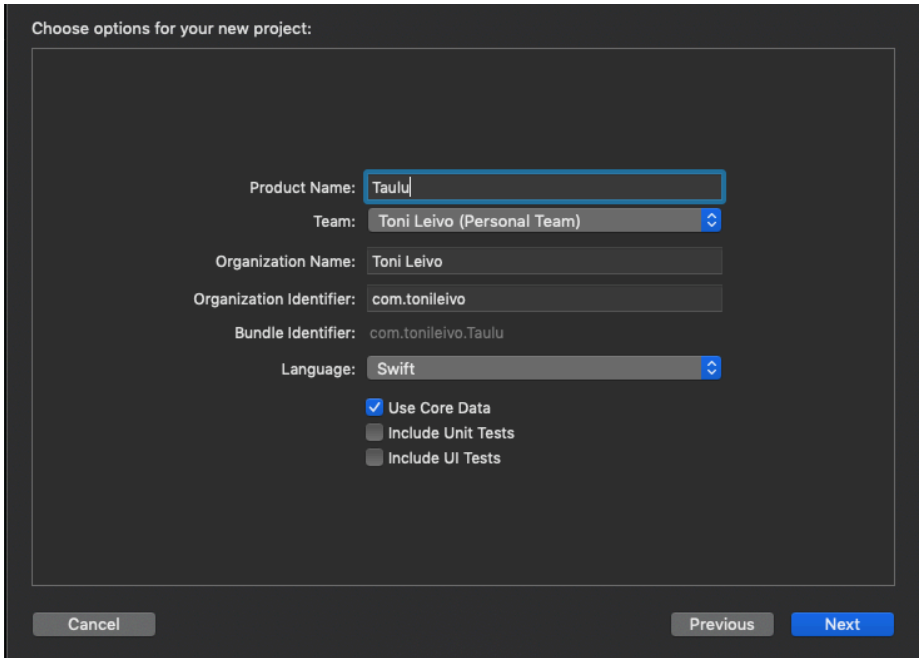
Sovelluksessa tulee myös olla pin-koodi, jolla saa suojattua, että lapsi ei pääse merkitsemään ylläpidon asettamia tehtäviä tehdyksi.
Sovelluksen kuluksi suunnittelin, että aluksi käyttäjä asettaa pin-koodin, tämän jälkeen tulee valikko ja sieltä valitaan, onko lapsi vai ylläpito. Kun valitsee lapsen, käyttäjä näkee jonkin näköisen tilastotaulukon tehtävien kuluista ja oman tämänhetkisen tilanteen. Ylläpidon puolella taas olisi paikka, jossa luodaan ryhmä, henkilö ja tehtävä. Kun nämä on luotu, ne yhdistetään tauluun ja henkilöön. Vain ylläpidolla on mahdollisuus merkitä tehtävä tehdyksi. Tehtävien merkintä nollaantuu joka päivä.
Sovelluksen teeman väriksi tulee sininen ja koitan käyttää värejä mahdollisimman paljon. Kuvakkeet tulee olla selkeät ja simppelit. Myös sovelluksen käyttö pitää olla mahdollisimman selkeää ja helppoa.
Sovelluksen tekemisessä käytän Xcode-käyttöympäristöä ja Swift-ohjelmointikieltä. Tulen tekemään sovelluksen tabletille. Käytän ominaisuuksina storyboardia ja MVVM-arkkitehtuuria.

4.3 Toteutus

Aluksi asensin MacBook pro 13” -kannettavaan tietokoneeseen Xcode-sovelluksen. Sovelluksen asennuksen jälkeen käynnistin Xcode-sovelluksen ja loin siellä uuden Xcode-projektin.
Valitsin yksi näkymäisen sovelluksen. Kuva 6.

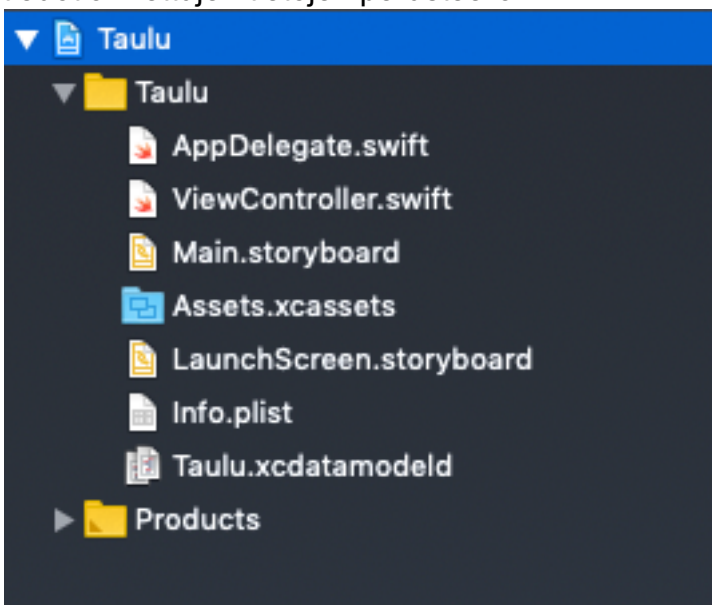


Kuva 6. Yksinäkymäinen sovelluksen valinta.
Annoin sovellukselleni nimen, valitsin ohjelmointikieleksi Swift ja rastitin kohdan ”käytä Core Data”.



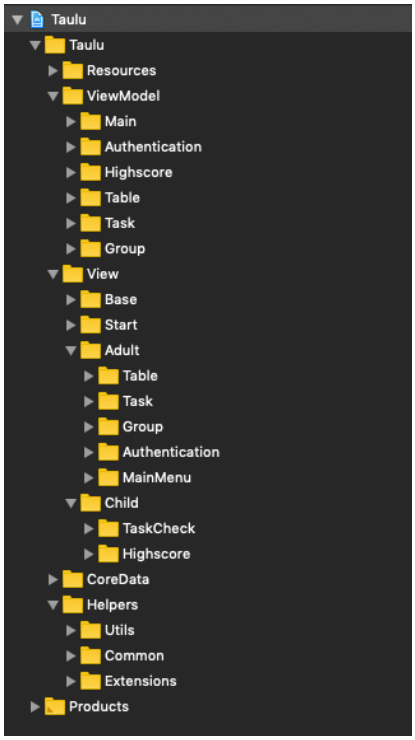
Kuva 7. Sovelluksen tietojen luonti.

Valitsin tallennuspaikan ja näin loin sovelluksen. Sovellukseen ilmestyi aloitustiedot annettujen tietojen perusteella.



Kuva 8. Sovelluksen aloitus tiedot luonnin jälkeen.

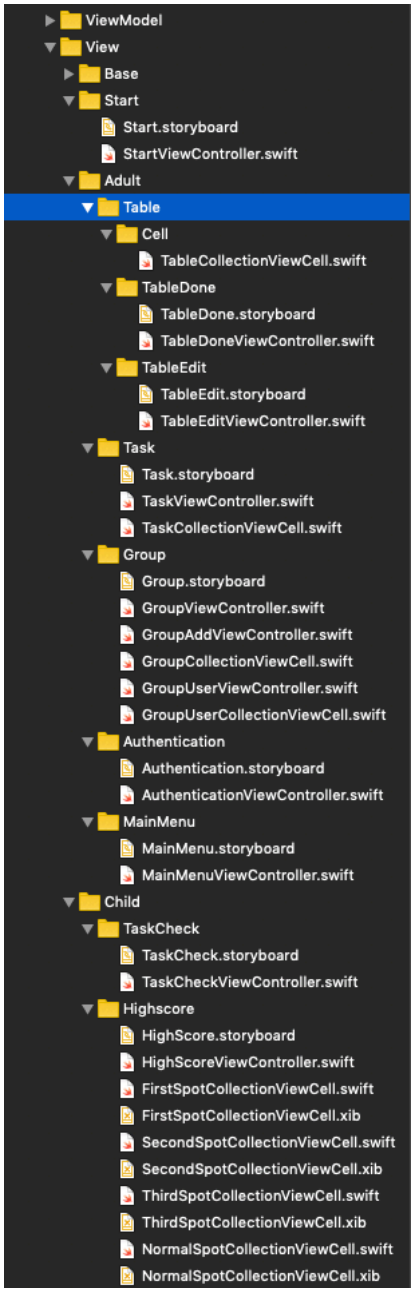
Aloitin luomalla kansioita sovellukseeni suunnitelmani pohjalta. Kansioita on hyvä luoda, jotta pysyy järjestys sovellusta tehdessä tai myöhemmin sitä tutkiessa.



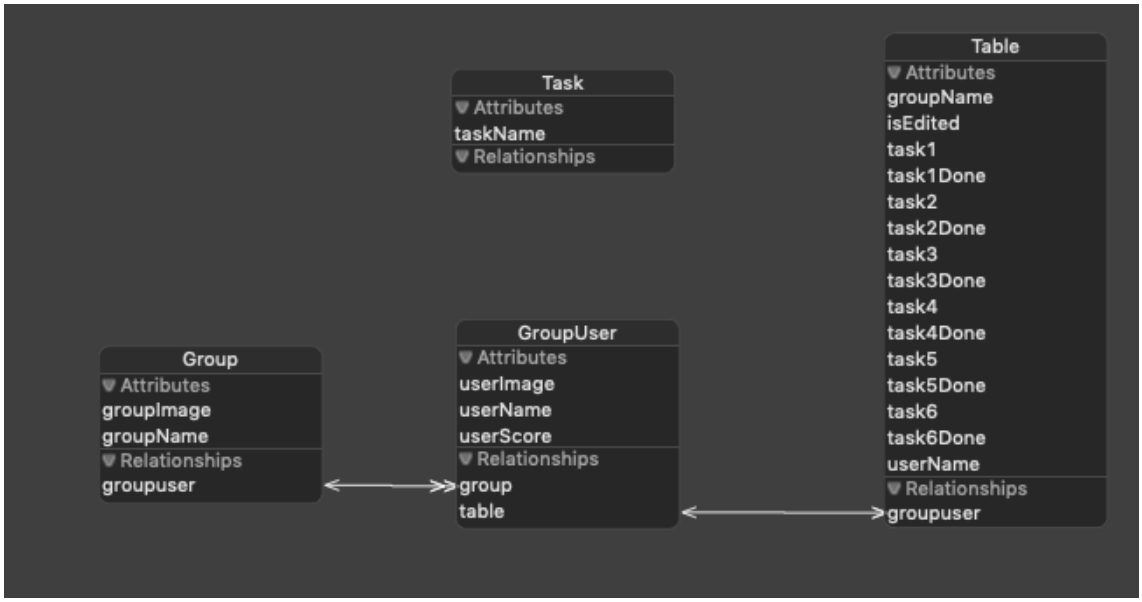
Kuva 9. Kuvassa on kaikki kansiot sovelluksen tekemistä varten.

Kansiot luotuani tein storyboardin jokaiseen kansioon ja nimesin ne myös kansion tyylisesti. Kaikki storyboard-tiedostotyypit päättyvät .storyboard-muotoon. Loin jokaiselle kansiolle oman storyboardin. Tämän avulla storyboard ei kasva liian suureksi ja sitä on jälkeenpäin helpompi hahmottaa.

Storyboardeiin tulee yksi tai useampi view controller -ulkoasuelementti. Jokaiselle view controller -elementille loin myös oman luokkatiedoston. Luokkatiedostot päättyvät Xcodessa aina .swift-tiedostopäätteeseen.

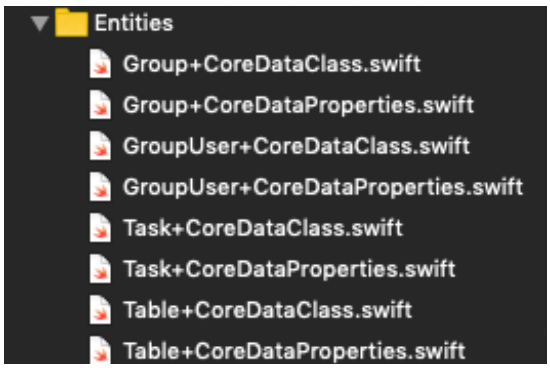


Kuva 10. Kuvassa näkyvät kaikki luodut storyboardit ja niille view controllerit. Seuraavaksi järjestin Core datani kuntoon eli loin kansiorakenteen. Loin myös tarvittavat objektikokonaisuudet ja asetan niille arvot, joita käytin sovelluksessa.



Kuva 11. Kuvassa näkyy kaikki Core datan objektit ja kokonaisuudet.

Kun loin nämä, käytin Xcoden editoria ja loin jokaiselle core data kokonaisuudelle oman luokkatiedoston.



Kuva 12. Kuvassa on ohjelmointi luokat Core datan objekteille

Aloin tekemään sovellusta MVVM-ohjelmointiarkkitehtuurimallilla. Tein jokaiselle view controller -luokalle myös view model -luokan. Näissä luokissa on data, joka tulee Core datasta. Otin sen datan view controlleriin tästä välikappaleesta, view modelista.

Kuva 13. Kuvassa näkyy MVVM-toimintatapa.

Ohjelmoin myös navigaation controllerin ja tab bar controllerin ilman interface builderia. Se helpotti minua hieman, ettei minun tarvinnut luoda jokaiseen storyboardiin erikseen navigation controlleria, kun käytän monta eri storyboardia. Tällä tavalla pystyin helposti navigoimaan monen eri storyboardin välillä.

```swift
import UIKit

class TabBarController: UITabBarController, UITabBarControllerDelegate {

    init(_ tabs: [UIViewController]) {
        super.init(nibName: nil, bundle: nil)
        self.viewControllers = tabs
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        setupViews()
    }

    // MARK - UITabBarControllerDelegate

    func tabBarController(_ tabBarController: UITabBarController, didSelect viewController: UIViewController) {
        viewController.tabBarItem.badgeValue = nil

    }

    // MARK - Override this method in child classes.

    func setupViews() {
        delegate = self

        view.backgroundColor = Constants.Color.background

        let selectedColor = Constants.Color.tabBarSelected
        let unselectedColor = Constants.Color.tabBarForeground
        let titleFont = Constants.Font.tabBar


        UITabBarItem.appearance().setTitleTextAttributes([.foregroundColor: unselectedColor], for: .normal)
        UITabBarItem.appearance().setTitleTextAttributes([.foregroundColor: selectedColor], for: .selected)
        UITabBarItem.appearance().setTitleTextAttributes([.font: titleFont], for: .normal)
        UITabBarItem.appearance().setTitleTextAttributes([.font: titleFont], for: .selected)


    }
}
```

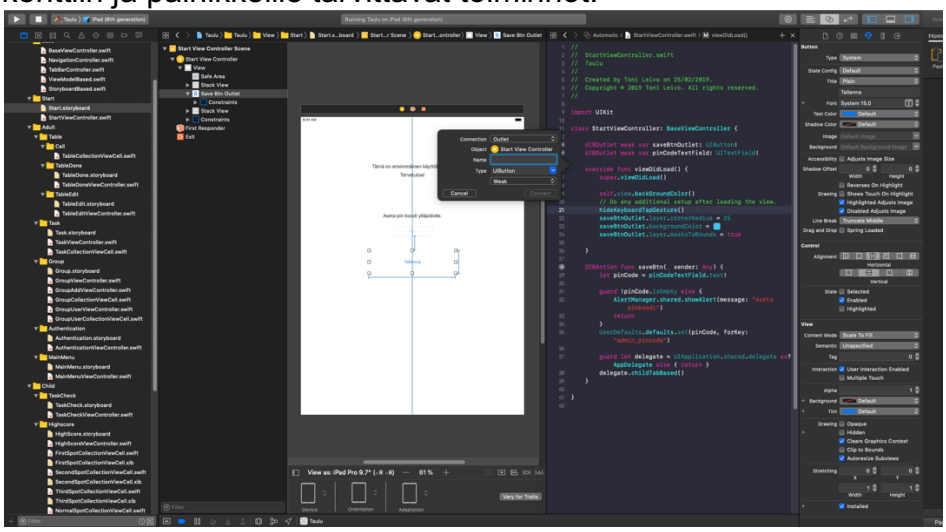
Kuva 14. Tab bar controllerin teko ohjelmoiden.

Asetin aloitusikkunaksi lapsen toimintoihin tarkoitetun tab bar controllerin. Pistin siihen myös tarkistuksen, onko ylläpito asettanut pin-koodia vai ei. Jos ylläpito ei ole asettanut pin-koodia, aukeaa StartViewController, TaskCheckViewController sijaan. StartViewControllerissa ylläpito asettaa sovellukseen pin-koodin.

```swift
import UIKit
import CoreData

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        childTabBased()
        UserDefaults.lastAccessDate = Date()
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        _ = Persistense.shared.saveContext()
    }


    func setupRootViewController(_ vc: UIViewController? = nil) {
        if window == nil {
            window = UIWindow(frame: UIScreen.main.bounds)
        }
        if let rootVc = vc {
            window?.rootViewController = rootVc
        }
        window?.makeKeyAndVisible()
    }

    func checkIfPinIsSet() -> Bool {
        guard UserDefaults.defaults.value(forKey: "admin_pincode") != nil else {
            startCheck()
            return false
        }
        return true
    }

    private func startCheck() {
        let startViewController = UIStoryboard(name: "Start", bundle: nil).instantiateViewController(withIdentifier: "StartViewController")
        setupRootViewController(startViewController)
    }

    public func childTabBased() {
        guard checkIfPinIsSet() else { return }
        setupRootViewController(TabBarController.createTabBased())
    }

    public func adultTabBased() {
        setupRootViewController(TabBarController.createAdultTabBased())
    }

}
```

Kuva 15. Sovelluksen AppDelegate

Tein ulkoasua jokaiseen view controlleriin. Kun ulkoasu on tehty, yhdistän ulkoasu elementit view controllerin luokkatiedostoon. Lisään tarvittavat otsikot tekstikenttiin ja painikkeille tarvittavat toiminnot.



Kuva 16. Painikkeen liittäminen ohjelmointikoodiin.

Asetin sovelluksen tarkastamaan jokaisella käynnistyskerralla, onko käyttäjä avannut sovelluksen tänään vai ei. Jos ei ole avannut sovellusta tänään, sovellus tyhjentää automaattisesti käyttäjän viime päivän merkinnät sovelluksen muistista pois. Näin uusi päivä alkaa ja ylläpito voi merkitä jälleen päivän tehtävät tehdyiksi.

```swift
import Foundation

extension UserDefaults {

    static let defaults = UserDefaults.standard

    static var lastAccessDate: Date? {
        get {
            return defaults.object(forKey: "lastAccessDate") as? Date
        }
        set {
            guard let newValue = newValue else { return }
            guard let lastAccessDate = lastAccessDate else {
                defaults.set(newValue, forKey: "lastAccessDate")
                return
            }
            if !Calendar.current.isDateInToday(lastAccessDate) {
                deleteTaskDone()
                UserDefaults.defaults.removeObject(forKey: "lastAccessDate")

            }
            defaults.set(newValue, forKey: "lastAccessDate")
        }
    }

    static func deleteTaskDone() {
        let tables = Persistense.shared.context.fetchObjects(Table.self, sortBy: nil, predicate: nil)
        for table in tables {
            table.task1Done = false
            table.task2Done = false
            table.task3Done = false
            table.task4Done = false
            table.task5Done = false
            table.task6Done = false
            table.saveContext()
        }
    }
}
```

Kuva 17. Funktio jonka avulla sovellus nollaa tehdyt tehtävät päivittäin.

4.4 Viimeistely

Ulkoasu

Lisäsin kaikkiin tekstikenttiin varmistuksen, että käyttäjä on kirjoittanut tekstiä tai muuten sovellus avaa uuden ikkunan ja kertoo käyttäjän jättäneen tekstikenttiä täyttämättä. Käyttäjä ei tämän varmistuksen takia pysty jatkamaan eteenpäin ilman, että täyttää tekstikenttää. Tämä varmistaa, ettei sovellukseen pysty tallentamaan tyhjiä tiedostoja.

```swift
import UIKit

class AuthenticationViewController: BaseViewController, StoryboardBased, ViewModelBased {

    @IBOutlet weak var pinCodeTextField: UITextField!
    @IBOutlet weak var loginBtnOutlet: UIButton!

    var viewModel: AuthenticationViewModel!


    override func viewDidLoad() {
        super.viewDidLoad()
        hideKeyboardTapGesture()
        loginBtnOutlet.layer.cornerRadius = 25
        loginBtnOutlet.backgroundColor =
        loginBtnOutlet.layer.masksToBounds = true
    }

    func initViewModel() {

    }

    @IBAction func authenticationBtn(_ sender: Any) {
        let pinCode = pinCodeTextField.text!
        let userDefaultsPin = UserDefaults.defaults.value(forKey: "admin_pincode") as! String
        guard !pinCode.isEmpty else {
            AlertManager.shared.showAlert(message: "Aseta pin koodi")
            return
        }
        guard pinCode == userDefaultsPin else {
            AlertManager.shared.showAlert(message: "Pin koodi on väärin")
            return
        }
        guard let delegate = UIApplication.shared.delegate as? AppDelegate else { return }
        delegate.adultTabBased()
    }

}

```

Kuva 18. Koodia jossa näkyy miten varmistin, ettei tekstikenttä ole tyhjä.
Viimeistelin kaikki kuvakkeet ja painikkeet oikeaan kokoon. Lisäsin painikkeille reunat ja värit. Listoille tein myös funktion, joka arpoo aina ikkunan avattaessa uuden värin jokaiselle listan jäsenelle. Tämä ominaisuus tuo lapselle hieman hauskuutta sovelluksessa.

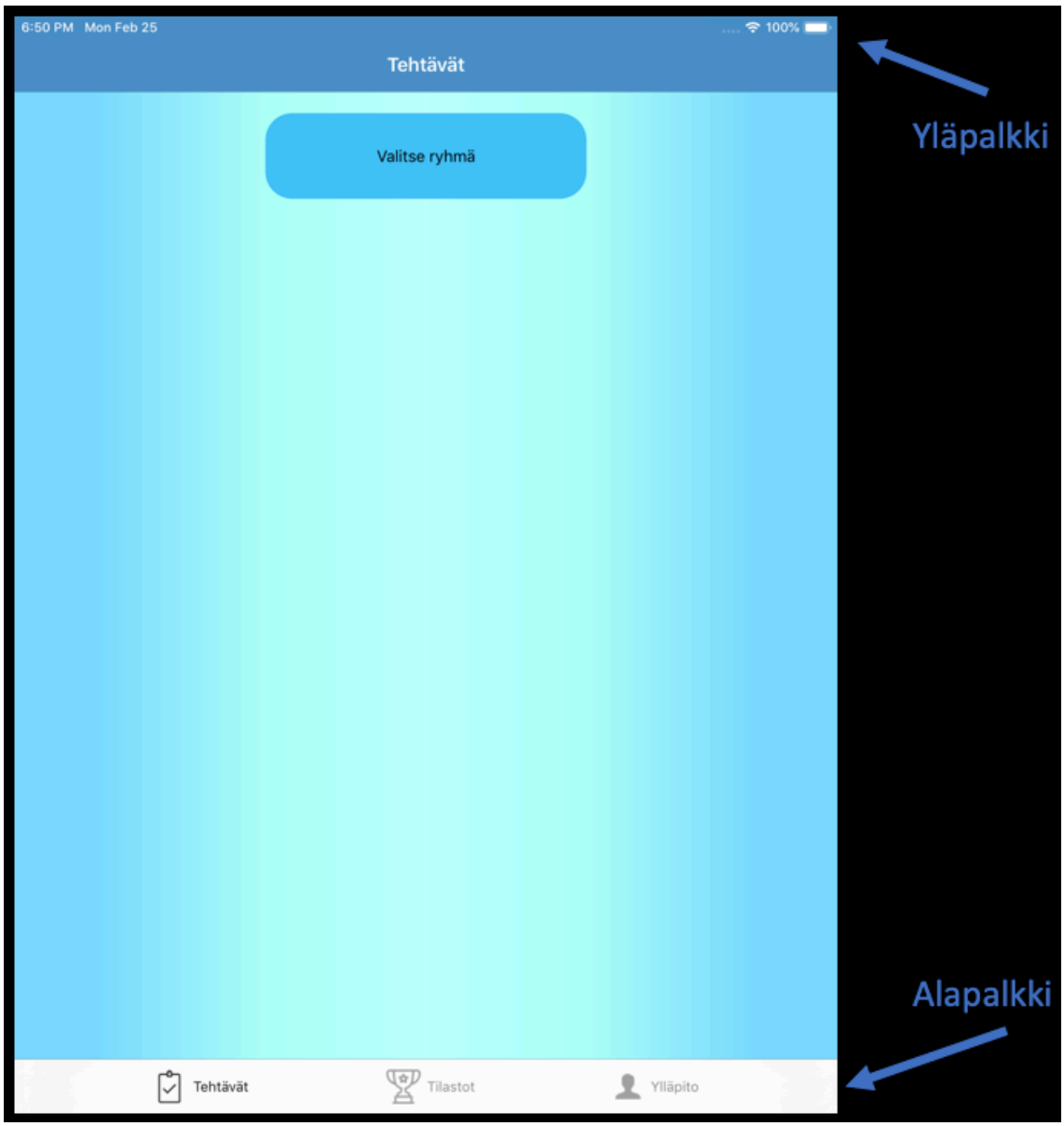
Testaus

Sovellusta testattiin koko kehityksen ajan Xcoden omalla simulaattorilla. Sovellusta testattiin 9.7”:n, 10.5”:n, 11”:n ja 12.9”:n simulaattoriversoilla. Sovelluksen ulkoasu oli yhteensopiva kaikissa testatuissa versoissa.
Pääasiallinen testaus tapahtui iPad (6.sukupolvi) versiolla. Tässä iPad versiossa on 9.7”:n näyttö. Sovellus oli viikon mittaisessa päivittäisessä käytössä ilman ongelmia. Sovellus ei sulkeutunut testauksien aikana, eikä muitakaan ongelmia havaittu.

5 SOVELLUKSEN TOIMINNALLISUUS

5.1 Lapsen osuus

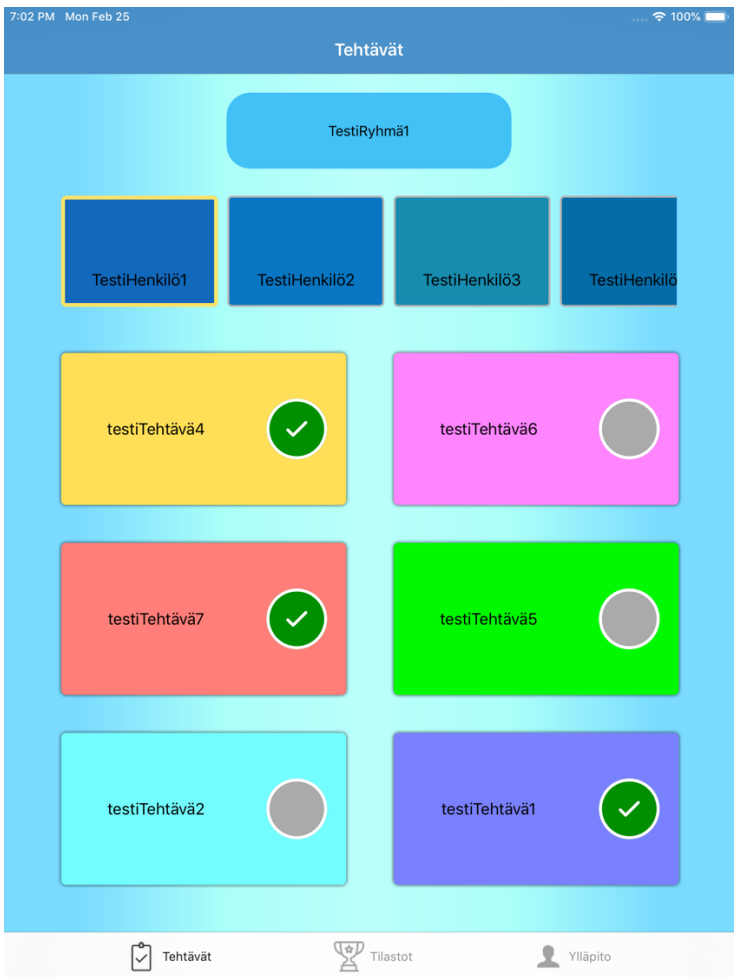
Kun käyttäjä avaa sovelluksen ensimmäisenä näytölle aukeaa Tehtävät-valikko. Yläpalkissa on otsikko riippuen alapalkin valinnasta. Alapalkissa on kolme valinta painiketta: ”Tehtävät”, ”Tilastot” ja ”Ylläpito”. Näillä painikkeilla on myös kuvake.



Kuva 19. Kuvassa näkyy navigation bar (Yläpalkki) ja tab bar (Alapalkki).

Tehtävät

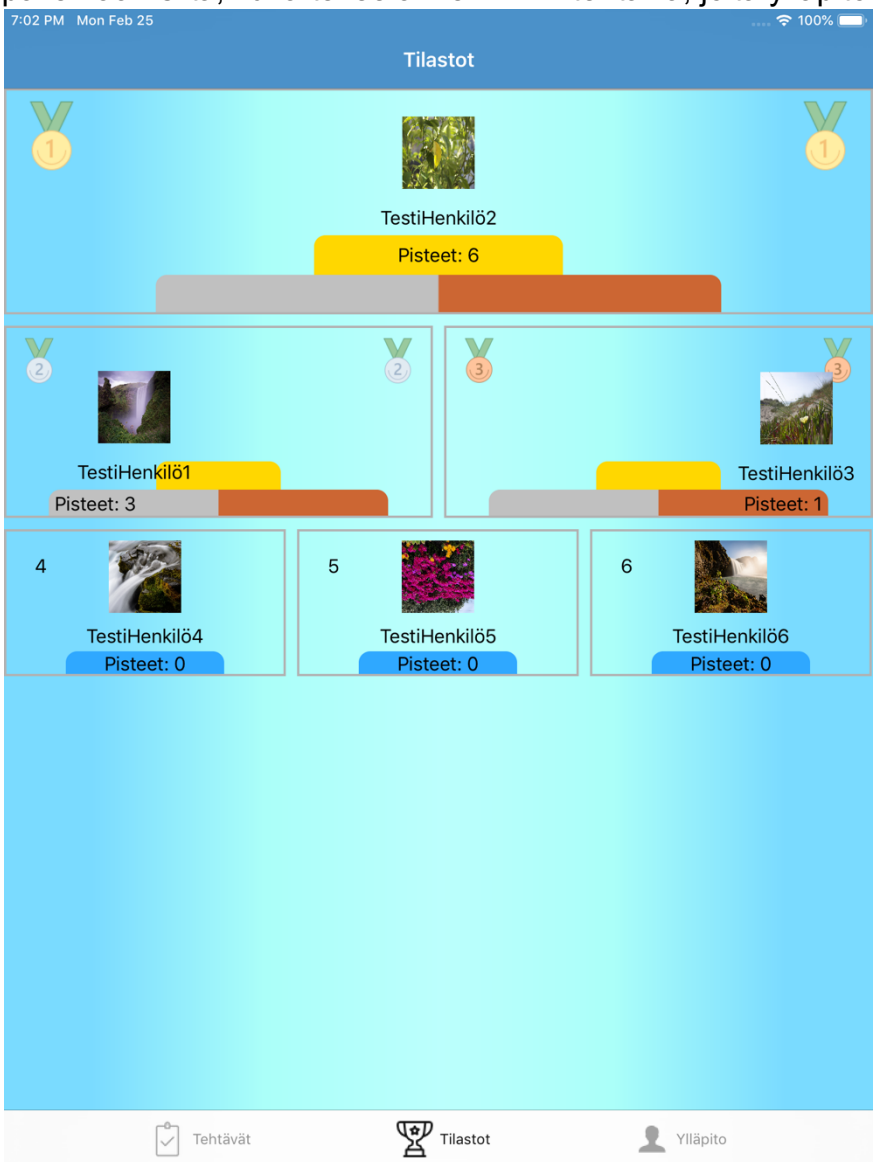
Alapalkin Tehtävät-painike avaa uuden näkymän. Tästä näkymästä lapsi pystyy tarkastelemaan omaa tilannetta tehtävien etenemisen suhteen ja katsomaan onko hän tehnyt tehtävää vai ei.
Tehtävä painikkeen ikkunassa on ”Valitse ryhmä” -painike. Siitä käyttäjä siirtyy ikkunaan, jossa on listattu kaikki ryhmät, mitä ylläpito on lisännyt aikaisemmin. Lapsen pitää valita ryhmä mihin hän itse kuuluu painamalla ryhmää listassa. Kun lapsi painaa jotakin ryhmää, siirtyy hän takaisin edelliseen ikkunaan. ”Valitun ryhmän” painikkeen otsikoksi vaihtuu lapsen valitseman ryhmän nimi. Samaan aikaan painikkeen alapuolelle ilmestyy lista ryhmään kuuluvista lapsista. Kun lapsi valitsee jonkin lapsen nimen listasta painamalla tätä, listan alapuolelle ilmestyy taulu. Taulussa näkyy tehtävät mitä ylläpito on lisännyt valitulle lapselle ja tehtävän oikealla puolella näkyy, kuvake onko henkilö tehnyt kyseistä tehtävää vai ei.



Kuva 20. Lapsen näkyvyys omista tehtävistä.

Tilastot

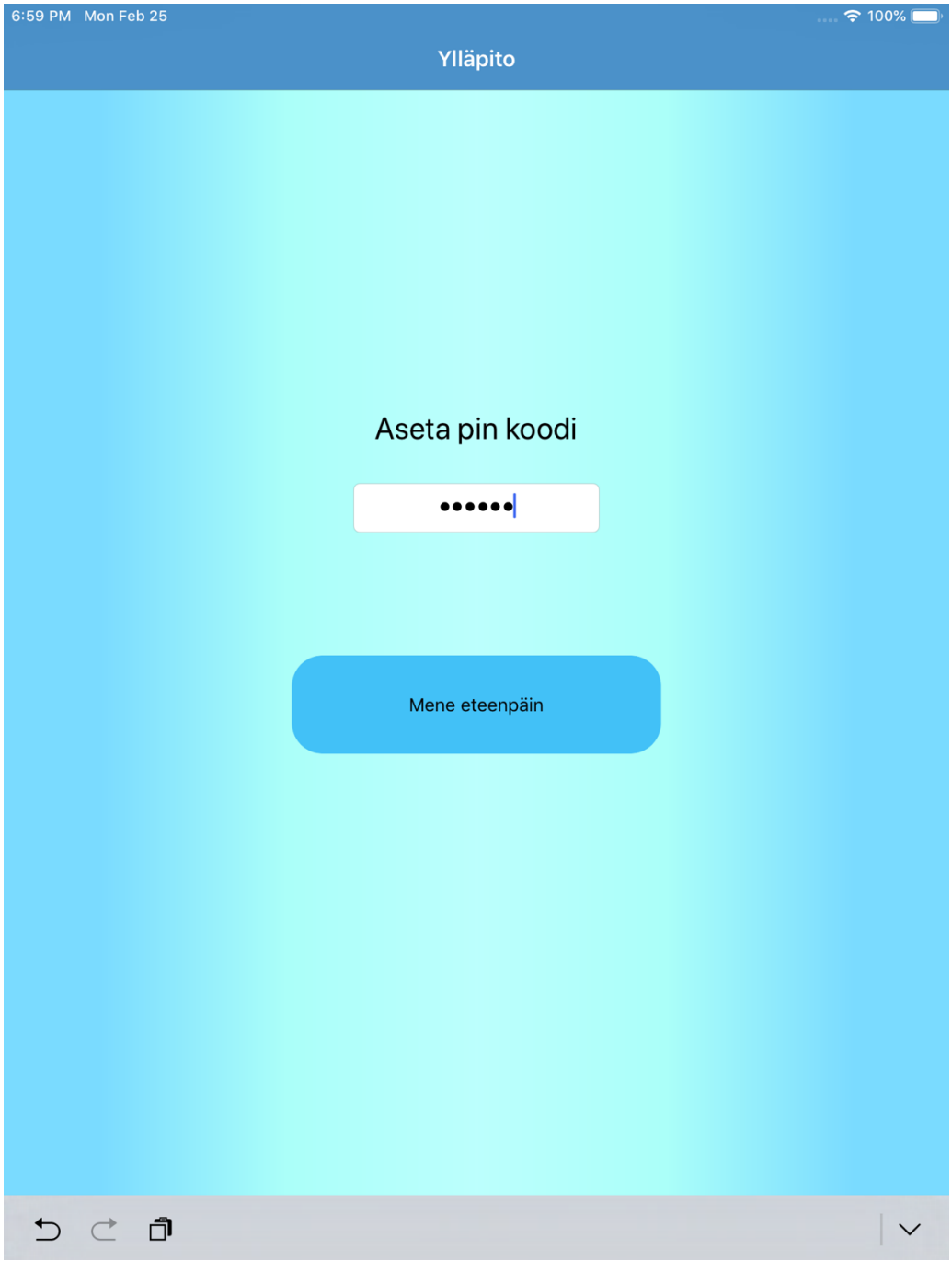
Tilastot-painike avaa näkymän, josta lapsi pystyy tarkastelemaan omaa kokonaistilannettaan. Tilastoissa näkyy jokaisen sovellukseen luodun lapsen toteutuneiden tehtävien pisteet listattuna suurimmasta pienimpään. Lapset pystyvät kilpailemaan siitä, kuka tekee ahkerimmin tehtäviä, joita ylläpito määrää lapselle.



Kuva 21. Sovellukseen lisättyjen henkilöiden pistemäärä tilastoina.

Ylläpito

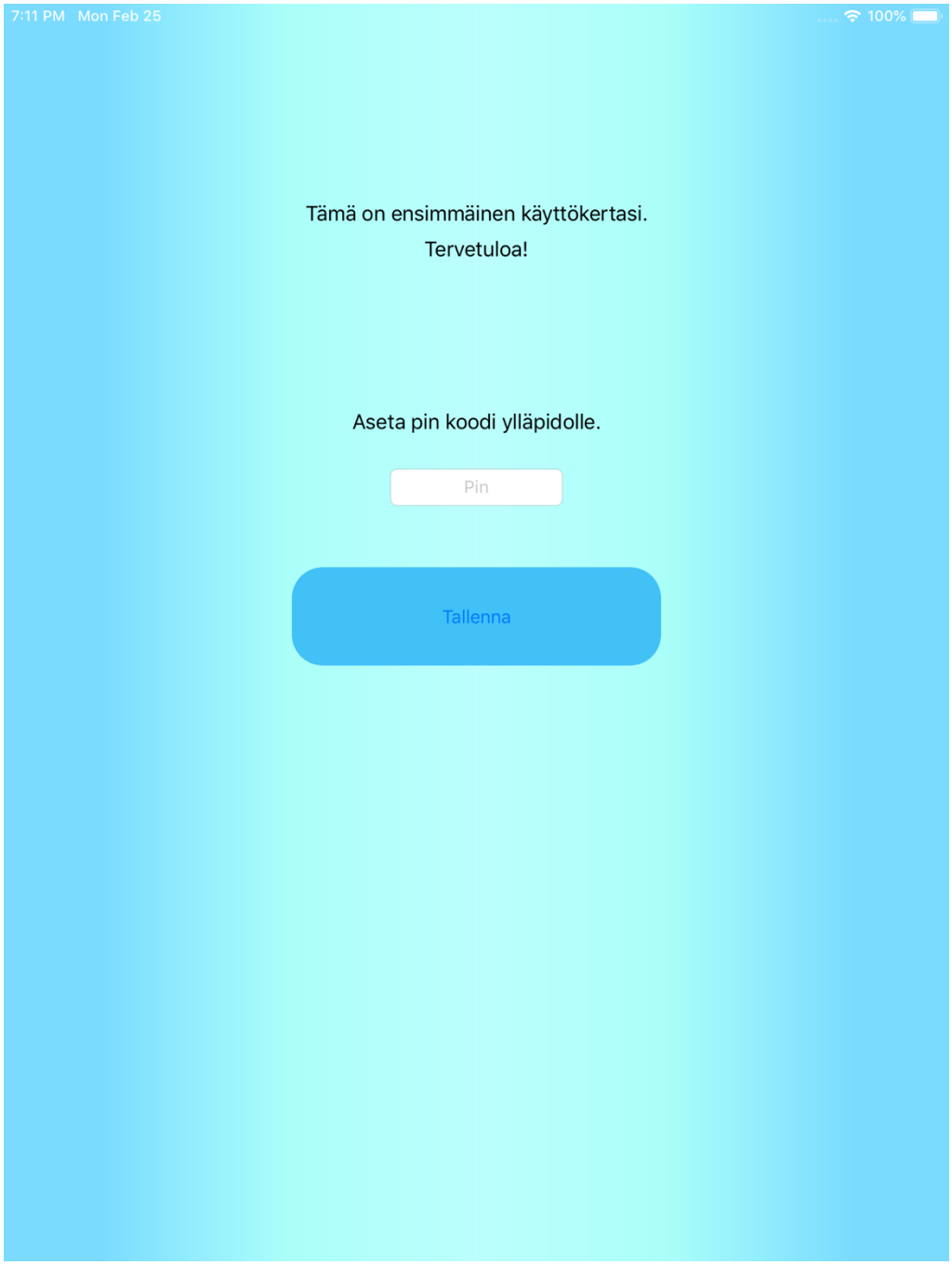
Ylläpito-painike avaa näkymän, jossa kysytään ylläpidon asettamaa pin-koodia. Jos pin-koodi ei ole oikea, tulee ilmoitus väärästä pin-koodista, jos pin-koodi on oikea, niin ylläpito pääsee sovelluksessa ylläpidon puolelle.



Kuva 22. Näkyvyys kun kirjautuu Ylläpidon puolelle.

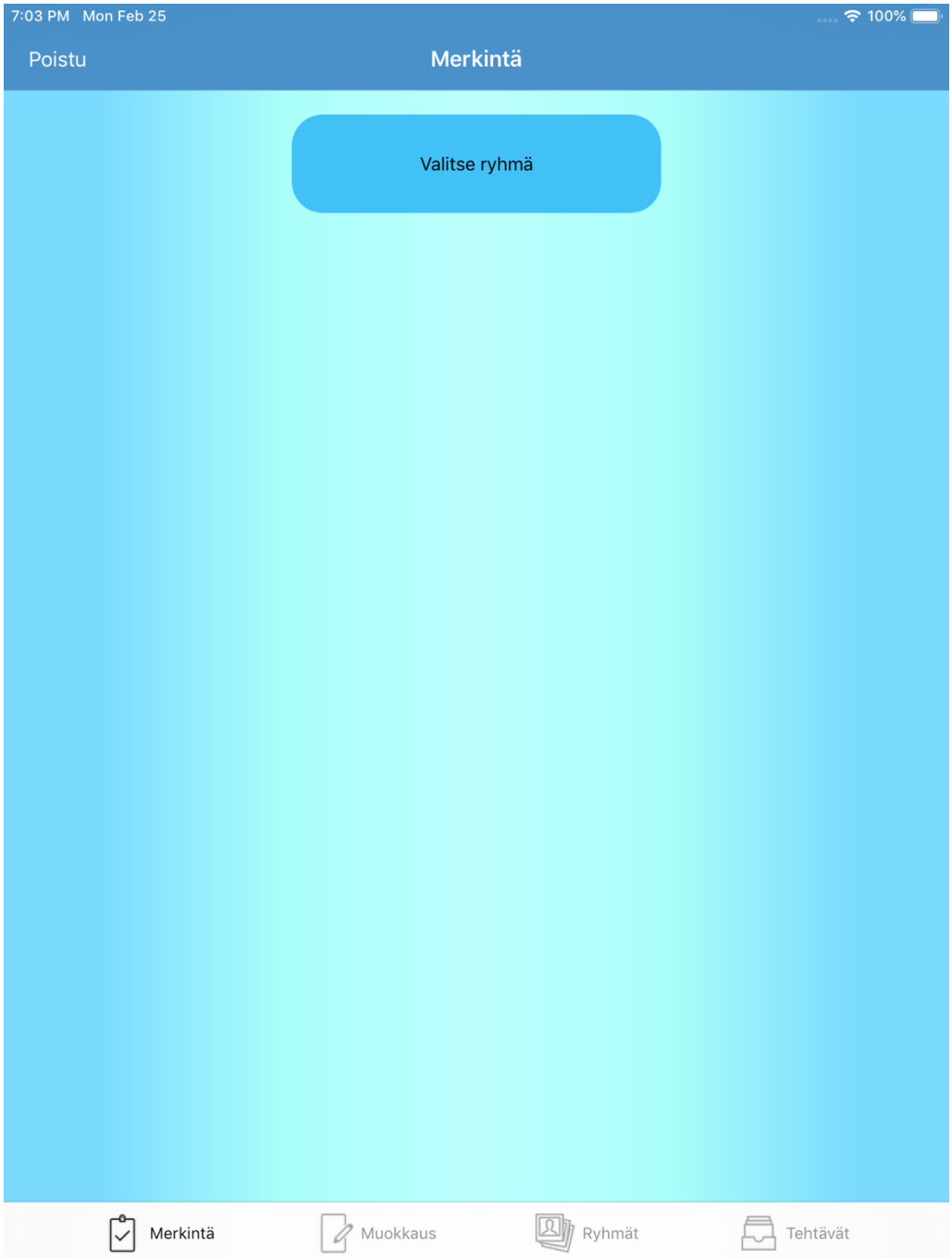
5.2 Ylläpidon osuus

Sovelluksen ensi lataamisen jälkeen aukeaa ikkuna. Täällä ikkunassa ylläpito lisää pin-koodin, jonka avulla hän myöhemmin kirjautuu sisälle sovelluksen ylläpidon puolelle.



Kuva 23. Pin-koodin lisäys ensimmäisellä käynnistyskerralla.

Ylläpidon tarvitsee tunnustautua joka kerta ennen kuin hän pääsee muokkaamaan ylläpidolle kuuluvia osia sovelluksesta. Tunnistautumisen jälkeen sovelluksessa aukeaa Merkintä-valikko. Yläpalkissa on ”Poistu” painike ja otsikko riippuen alapalkin tekstistä. Poistu painikkeesta ylläpito kirjautuu ulos ja kun hän haluaa uudestaan mennä ylläpidon puollelle, joutuu hän kirjautumaan uudelleen. Alapalkissa on neljä valinta painiketta: ”Merkintä”, ”Muokkaus” ja ”Ryhmät” ja ”Tehtävät”. Näillä painikkeilla on myös kuvakkeet.

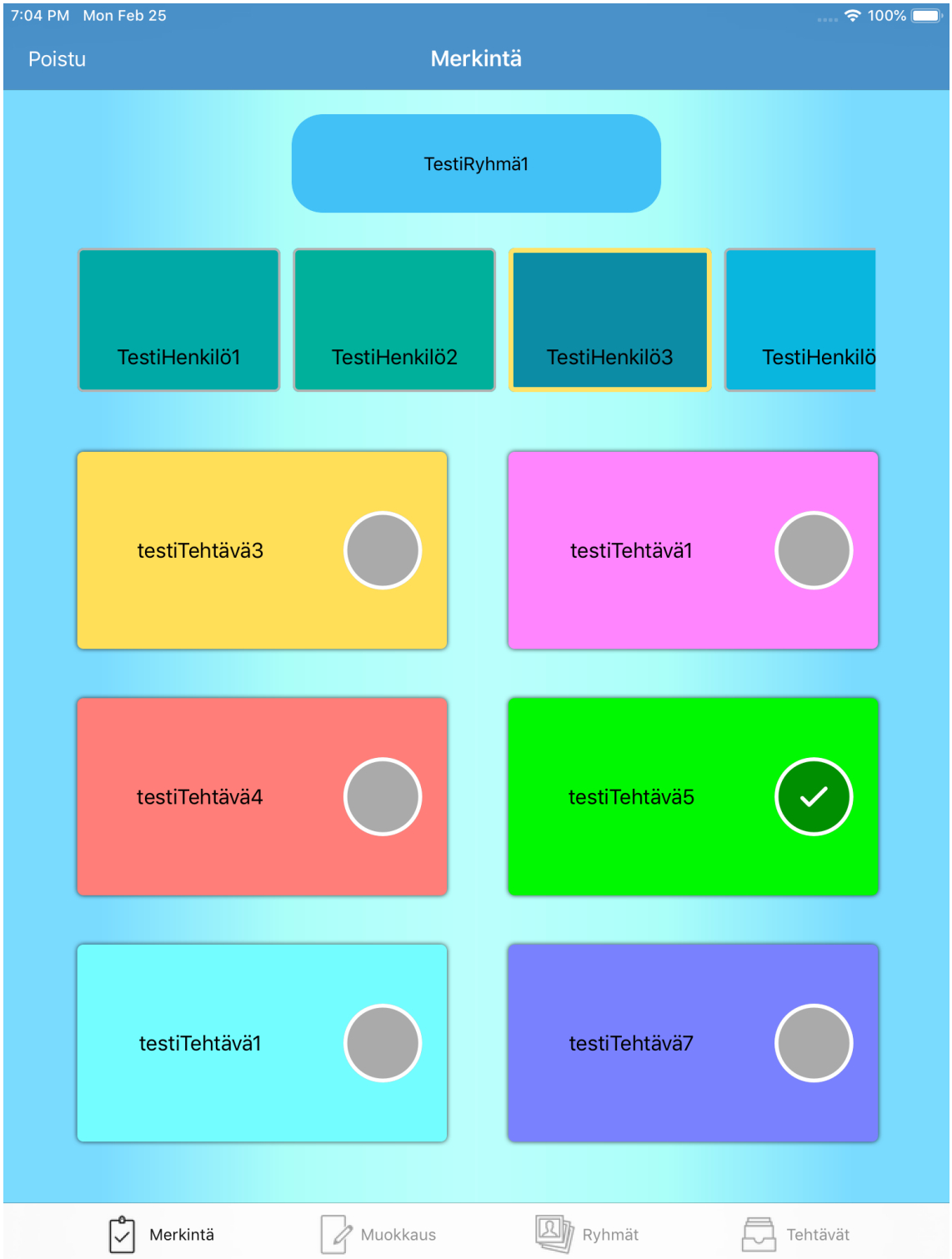


Kuva 24. Kuva aikuisen alku näkymästä.

Merkintä

Merkintä-painike avaa ikkunan, josta ylläpito pystyy tarkastelemaan kaikkien lapsien tilannetta tehtävien etenemisen suhteen ja katsomaan ovatko he tehneet tehtäviään vai ei. Ylläpito merkitsee täältä myös valitulle lapselle tehtävän tehdyksi, jos lapsi on sen tehnyt.
Merkintä-painikkeen ikkunassa on ”Valitse ryhmä” painike. Siitä käyttäjä siirtyy ikkunaan, jossa on listattu kaikki ryhmät, joita ylläpito on lisännyt aikaisemmin.

Ylläpidon pitää valita ryhmä, jota hän haluaa muokata painamalla ryhmää listassa. Kun ylläpito painaa jotakin ryhmää, siirtyy hän takaisin edelliseen ikkunaan. ”Valitun ryhmän” painikkeen otsikoksi vaihtuu ylläpidon valitseman ryhmän nimi. Samaan aikaan painikkeen alapuolelle ilmestyy lista ryhmään kuuluvista lapsista. Kun ylläpito valitsee lapsen nimen listasta painamalla tätä, listan alapuolelle ilmestyy taulu. Taulussa näkyy tehtävät, jotka ylläpito on lisännyt valitulle lapselle. Tehtävän oikealla puolella näkyy, kuvake onko lapsi tehnyt kyseistä tehtävää vai ei. Kuvaketta painamalla ylläpito pystyy vaihtamaan lapsen tehtävän arvon tekemättömästä tehdyksi.

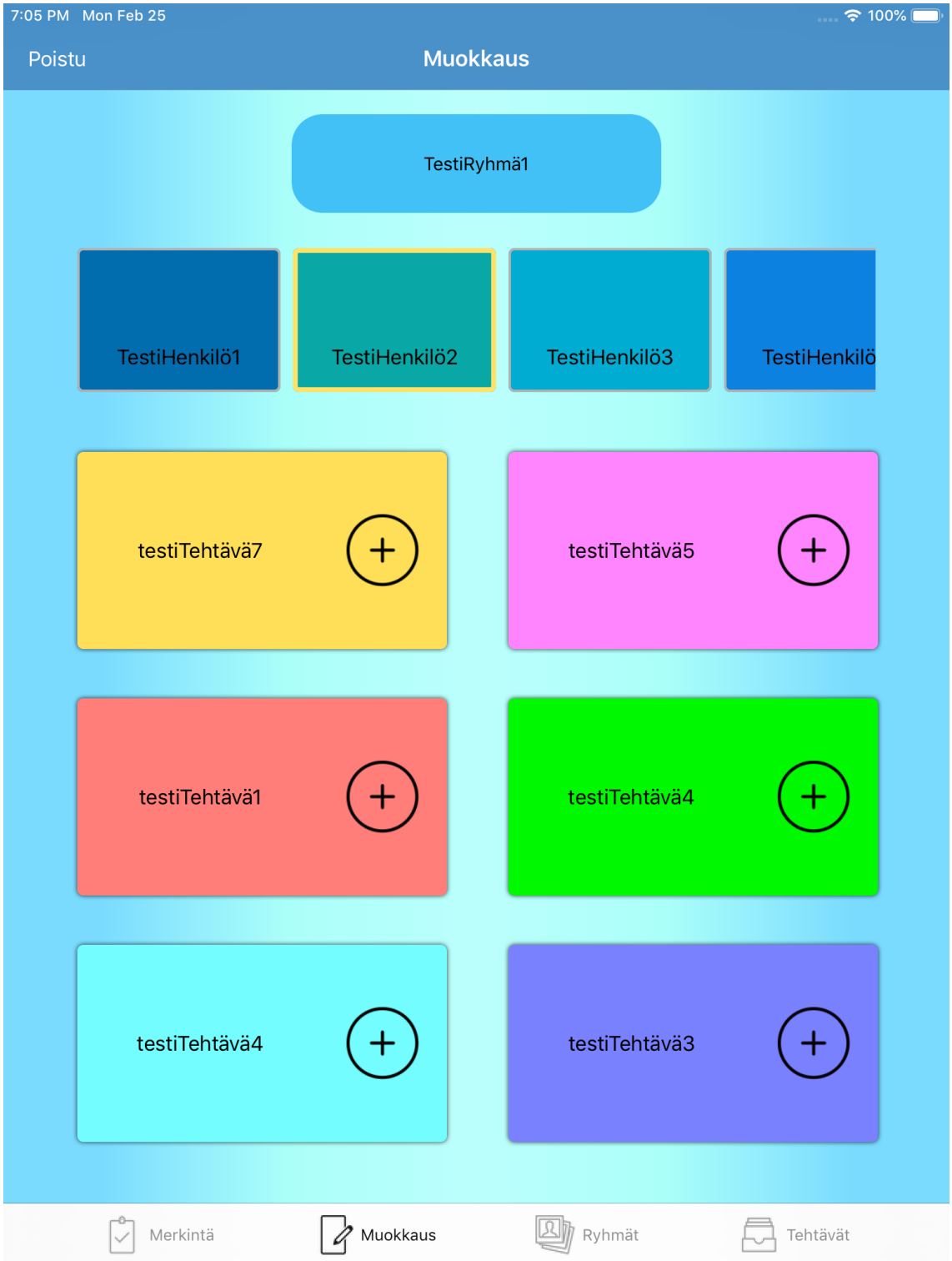


Kuva 25. Merkintä sivu, kun on valinnut ryhmän ja henkilön.

Muokkaus

Kun ylläpito haluaa muokata sovelluksen tehtäviä ja tauluja se tehdään asetus-sivun kautta Muokkaus-painikkeella. Painikkeen ikkunassa on ”Valitse ryhmä” -

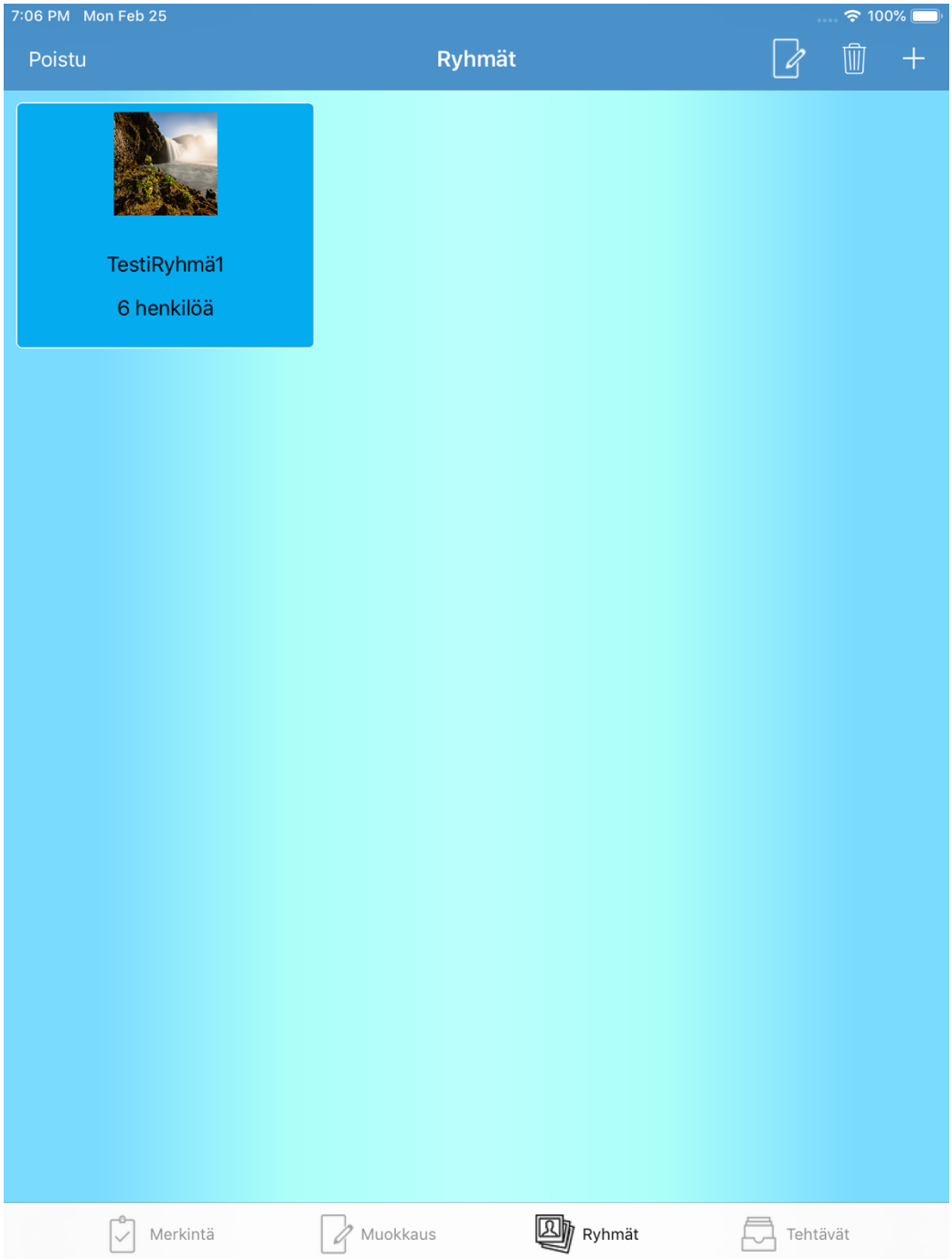
painike. Valitse ryhmä -painikkeesta käyttäjä siirtyy ikkunaan, jossa on listattu kaikki ryhmät, joita ylläpito on lisännyt aikaisemmin. Ylläpidon pitää valita ryhmä, jota hän haluaa muokata painamalla ryhmää listassa. Kun ylläpito painaa jotakin ryhmää, siirtyy hän takaisin edelliseen ikkunaan. ”Valitun ryhmän” painikkeen otsikoksi vaihtuu ylläpidon valitseman ryhmän nimi.
Samaan aikaan napin alapuolelle ilmestyy lista ryhmään kuuluvista lapsista. Kun ylläpito valitsee lapsen nimen listasta painamalla tätä, listan alapuolelle ilmestyy taulu. Taulussa näkyy lapsen tehtävät. Jos ylläpito on aikaisemmin lisännyt lapselle tehtävän, se tulee tauluun siihen kohtaan mihin se on lisätty, jos ei ole niin tähän kohtaan tulee teksti ”Ei vielä tehtävää”.
Ylläpito pystyy lisäämään tehtävän tai vaihtamaan tehtävää painamalla kuvaketta tehtävän oikealta puolelta. Kuvakkeen painamisesta aukeaa uusi ikkuna. Ikkunassa on listattu kaikki lisätyt tehtävät ja ylläpidon pitää valita yksi tehtävä painamalla tehtävää listassa. Painamisen jälkeen ylläpito palaa edelliseen ikkunaan ja tehtävän tekstiksi muuttuu valittu tehtävä.



Kuva 26. Muokkaus-ikkuna, ryhmän ja henkilön valitsemisen jälkeen.

Ryhmät

Kun ylläpito haluaa luoda uusia ryhmiä tai muokata ryhmän tietoja tai jäseniä, avataan Ryhmät-välilehti alapalkista.



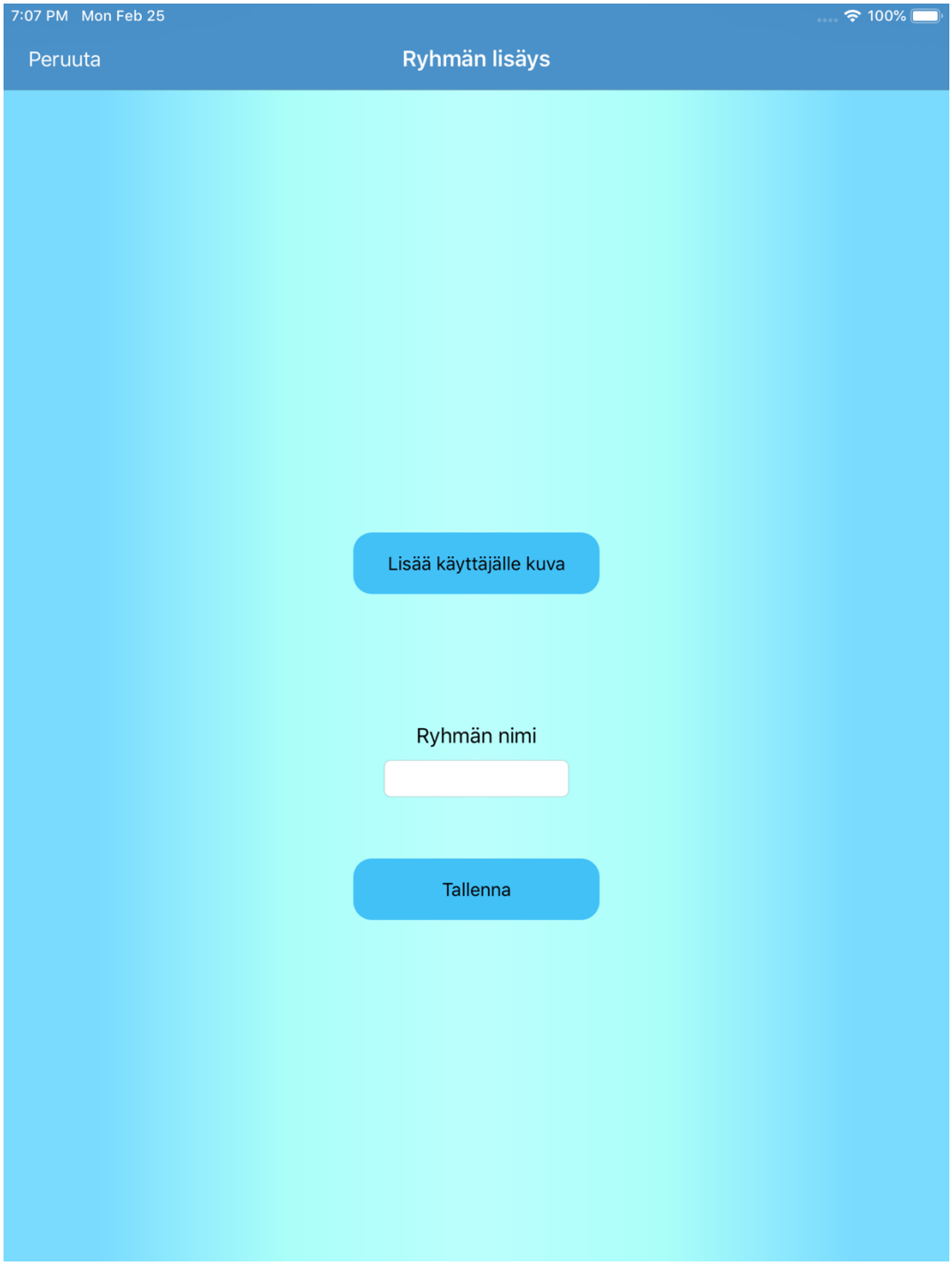
Kuva 27. Ryhmien etusivu.

Ryhmät-ikkunassa muokkaa-painiketta painaessa käynnistyy muokkaustila ja listassa olevat ryhmät alkavat heilua ja niiden vasempaan yläreunaan ilmestyy muokkaa-kuvake. Kun muokkaustila on päällä ja painaa listassa olevaa ryhmää, aukeaa uusi ikkuna, jossa on ryhmän kuva ja ryhmän nimi. Täällä pystyy vaihtamaan ryhmän tietoja.

Ryhmät ikkunassa poista-painiketta painaessa listassa olevat ryhmät alkavat heilua ja niiden vasempaan yläreunaan ilmestyy poista kuvake. Kun poista-tila on päällä ja painaa listassa olevaa ryhmää aukeaa ikkuna, jossa kysytään, haluatko varmasti poistaa kyseisen ryhmän. Jos ylläpito valitsee kyllä, niin ryhmä, sekä ryhmään luodut henkilöt poistuvat kokonaan.

Kuva 28. Näkymä poistaessa ryhmiä.

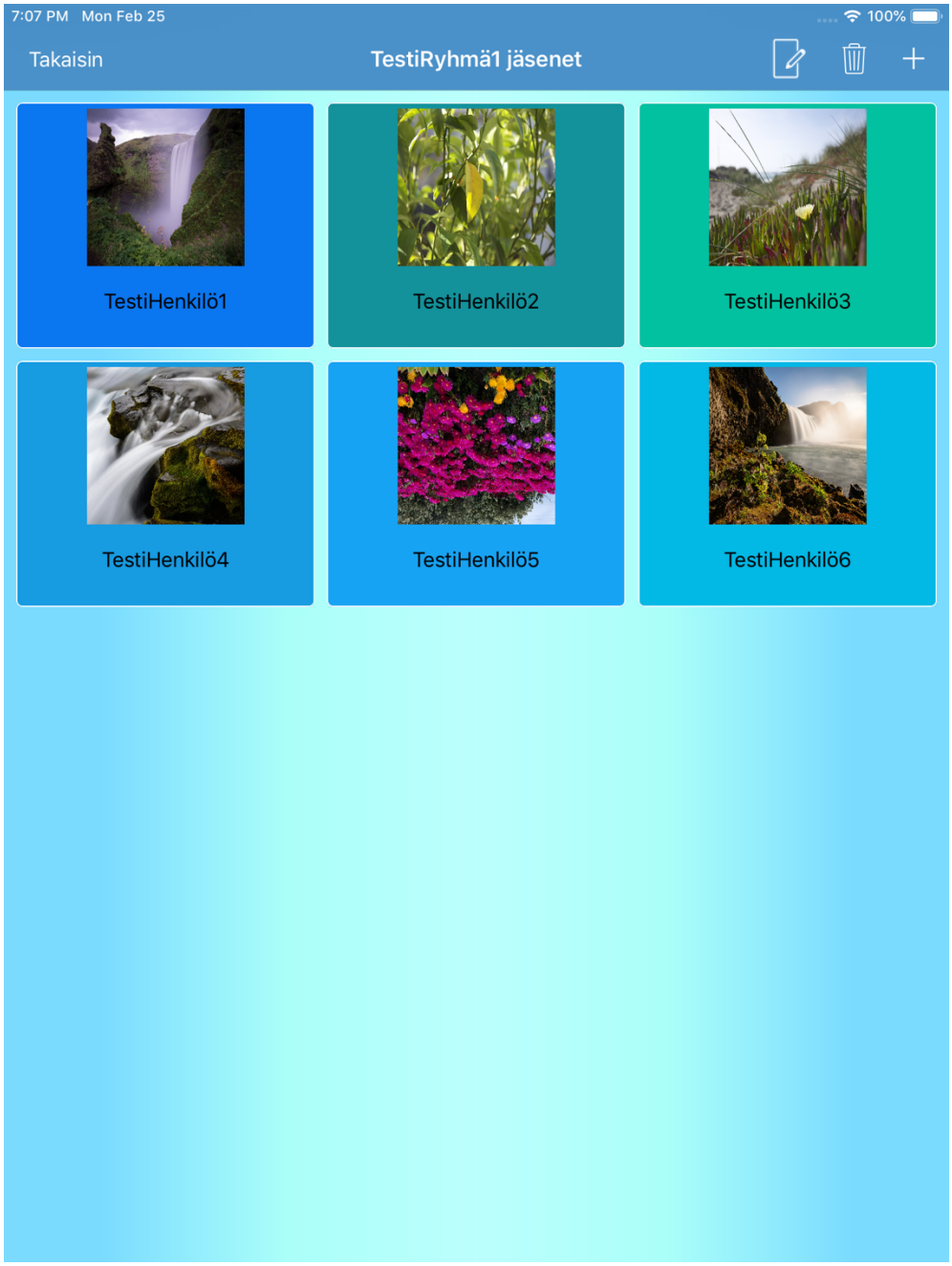
Ryhmät ikkunassa lisää-painiketta painaessa käyttäjä siirtyy ikkunaan, jossa lisätään ryhmälle kuva ja ryhmän nimi. Ikkunassa on Lisää käyttäjälle kuva- painike, tekstikenttä ja tallenna-painike. Ryhmää ei pysty lisäämään ilman, että käyttäjä on lisännyt kuvan ja kirjoittanut tekstikenttään ryhmän nimen. Kun tiedot on täytetty ja painaa tallenna-painiketta, käyttäjä siirtyy edelliseen ikkunaan ja uusi ryhmä ilmestyy listalle.



Kuva 29. Ryhmää lisättäessä oleva ikkuna.

Henkilöt

Henkilöitä voidaan lisätä ryhmiin myös Ryhmät-näkymässä. Tällöin käyttäjän pitää painaa Ryhmät-näkymässä olevalla listalla olevaa ryhmää, johon hän haluaa lisätä henkilön. Jokaisessa ryhmässä on omat henkilöt. Yläpalkissa on lisäksi myös oikealla puolella painikkeet: ”muokkaa”, ”poista” ja ”lisää”.



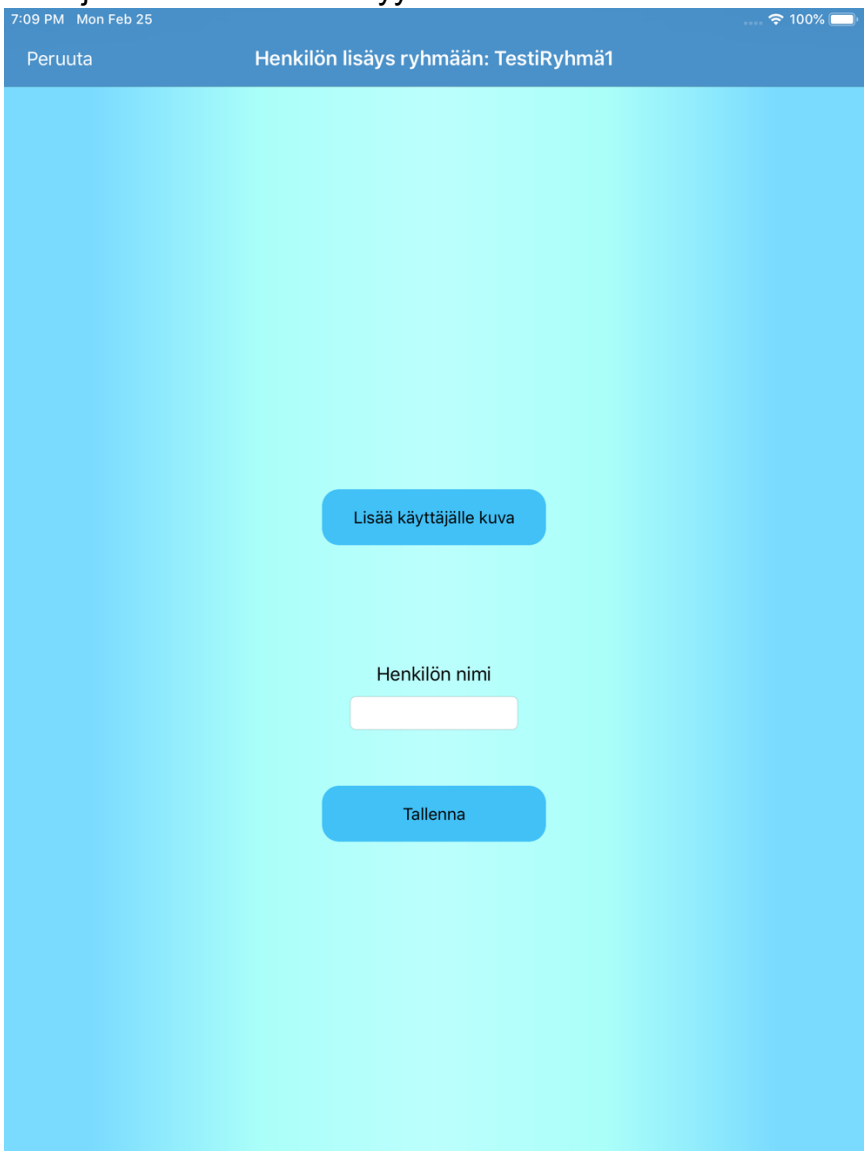
Kuva 30. Ikkuna jossa näkyy kaikki henkilöt.
Kun henkilöitä muokataan muokkaus-painikkeesta, listassa olevat henkilöt alkavat heilua ja niiden vasempaan yläreunaan ilmestyy muokkaa-painike. Kun muokkaustila on päällä ja painaa listassa olevaa henkilöä, aukeaa uusi ikkuna, jossa on henkilön kuva.

Kuva 31. Näkymä, kun on painanut muokkaus painiketta.

Henkilöt ikkunassa poista-painiketta painaessa alkaa poista tila ja listassa olevat henkilöt alkavat heilua ja niiden vasempaan yläreunaan ilmestyy, poista kuvake. Kun poista-tila on päällä ja painaa listassa olevaa henkilöä aukeaa ikkuna, jossa kysytään, haluatko varmasti poistaa kyseisen henkilön. Jos ylläpito valitsee kyllä, niin henkilö poistuvat kokonaan.

Henkilöt ikkunassa lisää painiketta painaessa käyttäjä siirtyy ikkunaan, jossa lisätään henkilölle kuva ja henkilön nimi. Ikkunassa on Lisää käyttäjälle kuva -painike, tekstikenttä ja tallenna-painike. Henkilöä ei pysty lisäämään ilman, että

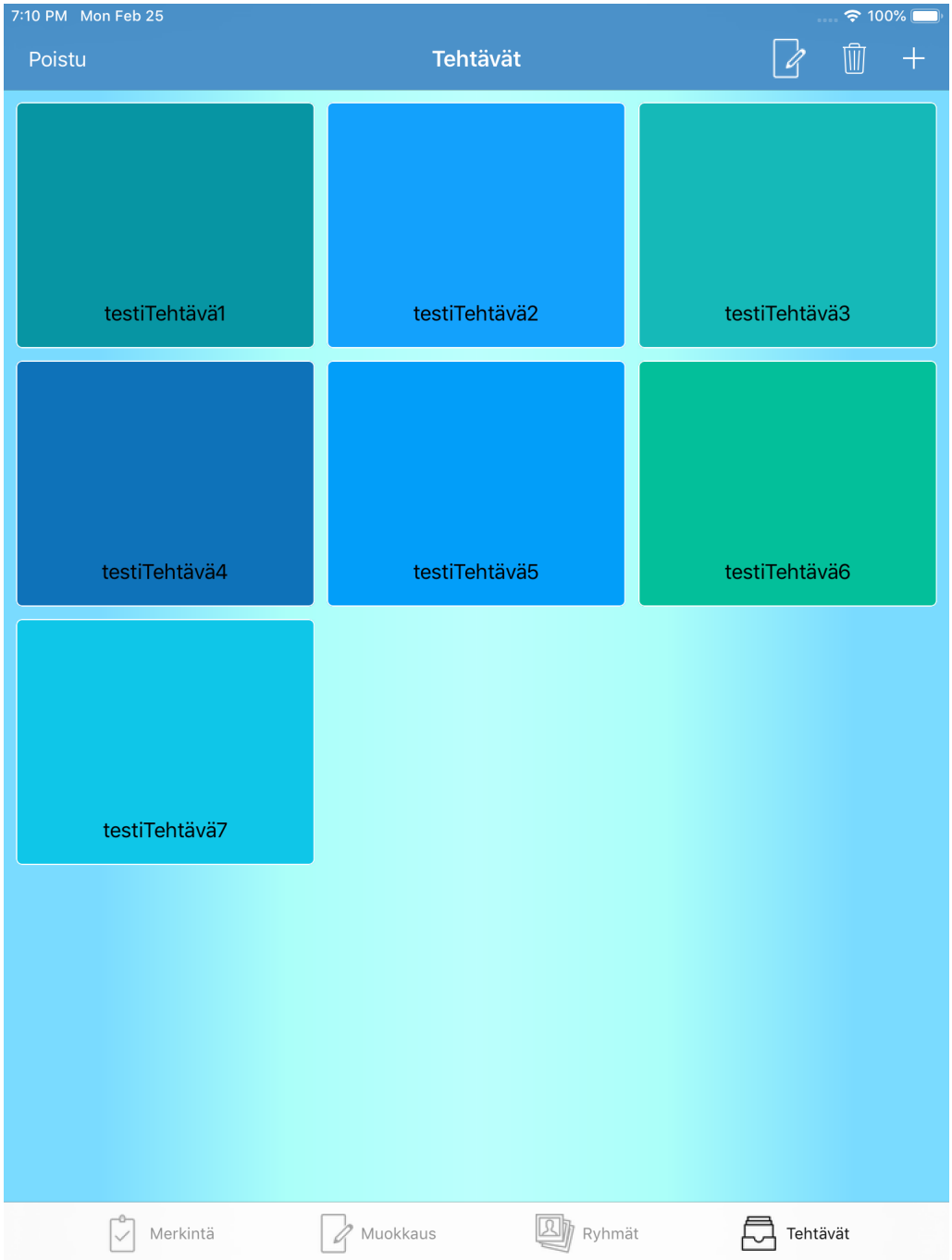
käyttäjä on lisännyt kuvan ja kirjoittanut tekstikenttään henkilön nimen. Kun tiedot on täytetty ja painaa tallenna painiketta, käyttäjä pomppaa edelliseen ikkunaan ja uusi henkilö ilmestyy listalle.



Kuva 32. Henkilöä lisättäessä oleva ikkuna.

Tehtävät

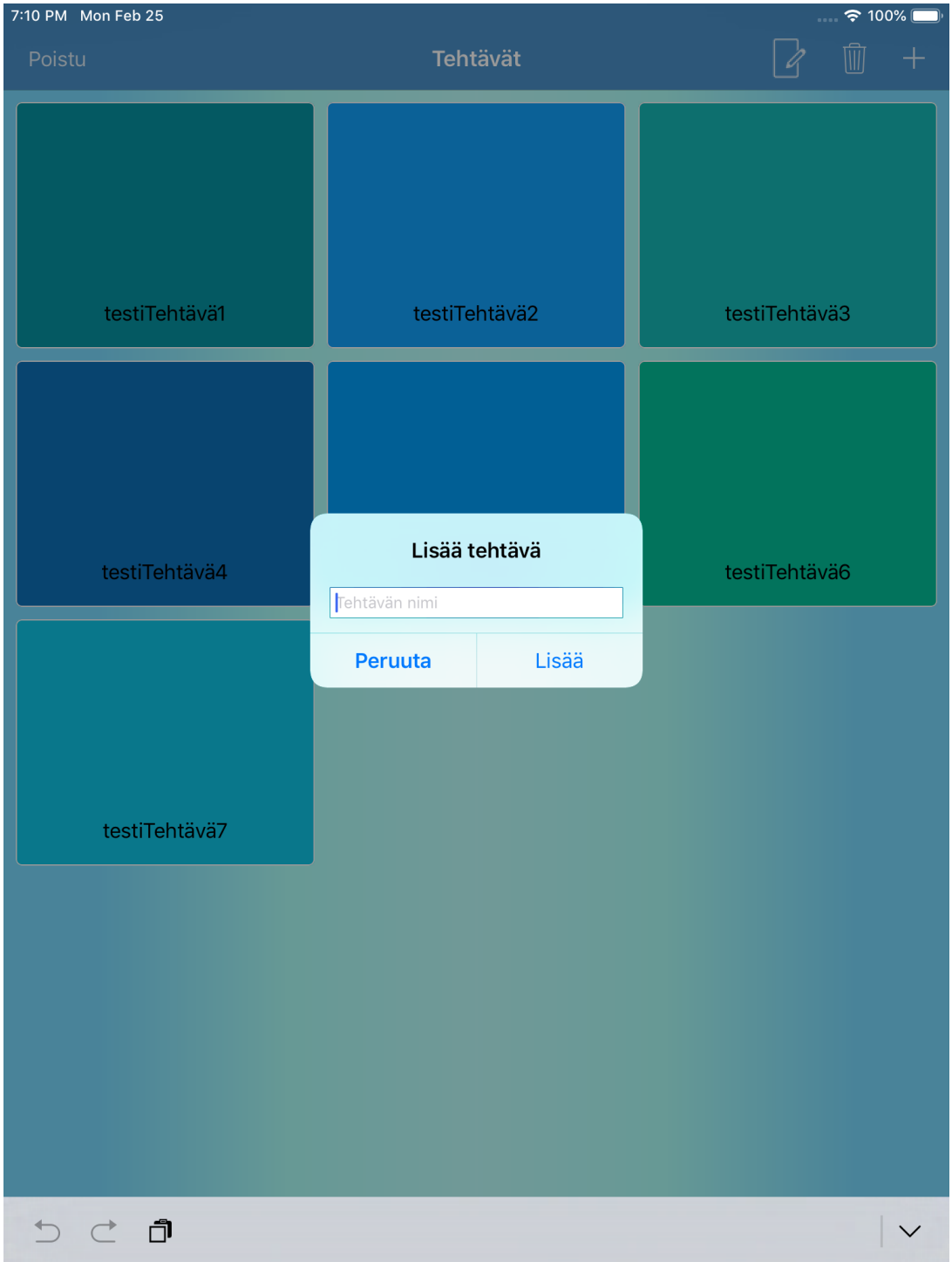
Kun henkilöille halutaan lisätä tehtäviä painetaan Tehtävät-painiketta Täältä näkymästä ylläpito pystyy lisäämään tehtäviä, joita myöhemmin lisätään tehtävä tauluun. Yläpalkissa on lisäksi myös oikealla puolella napit: ”muokkaa”, ”poista” ja ”lisää”.



Kuva 33. Tehtävät ikkunasta näkymä.

Tehtävät ikkunassa voidaan myös muokata tehtäviä ryhmien ja henkilöiden tapaan. Muokkaa-painiketta painaessa listassa olevat tehtävät alkavat heilua ja niiden vasemmasta yläreunasta pääsee muokkaamaan tehtävien nimiä. Tehtävät-ikkunassa poista painiketta painaessa niin ikään voidaan poistaa tehtäviä. Sovellus varmistaa aina poistot käyttäjältä.

Tehtäviä lisättäessä käsketään käyttäjää lisäämään tehtävälle nimi. Näkymässä on tekstikenttä, Lisää- ja Peruuta-painike. Peruuta-painikkeesta käyttäjä pystyy keskeyttämään tehtävän lisäyksen. Lisää-painikkeesta käyttäjä pystyy lisäämään tehtävän, jos hän on kirjoittanut tekstikenttään tekstiä. Kun käyttäjä painaa Lisää-painiketta tehtävä ilmestyy listalle.



Kuva 34. Ikkuna tehtävän lisäyksessä.

6 JATKOKEHITYS

Sovelluksen mahdollisia jatkokehityskohteita voi olla esim. pin-koodin muokkaamismahdollisuus, tietokannan lisääminen ja sovelluksen sisäisiä ostoja. Tällä hetkellä ainut tapa vaihtaa tai poistaa pin-koodi on, sovelluksen poistaminen ja uudelleen asennus. Kun käyttäjä poistaa ja asentaa sovelluksen uudelleen, niin kaikki sovellukseen lisätyt tiedot poistuvat. Tämän pystyisi korjaamaan tietokannan lisäämisellä sovellukseen. Sovellus tallentaisi silloin kaikki käyttäjän tallentamat tiedot ulkoiseen palveluun, riippuen siitä minkä valitsisin tietokannaksi. Tämä myös mahdollistaisi sovelluksen laajemman käytön. Sovelluksessa voisi olla kaikkien käyttäjien yhteinen tulostaulu. Sovellukseen voisi myös lisätä sovelluksen sisäisiä ostoja, joiden avulla tienaisin rahaa sovelluksella. Esimerkiksi, voisi rajoittaa, sitä kuinka monta ryhmää käyttäjä voisi tehdä. Ostamalla lisäoikeuden käyttäjä pystyisi luomaan enemmän ryhmiä.
Sovellusta ei ole julkaistu App Storessa. Valmis sovellus tulisi julkaista App Storeen kaikkien käyttäjien ladattavaksi. Julkaisu on maksullista ja siihen tarvitaan Apple Developer -lisenssi. Jos hankin lisenssin, voin julkaista sovelluksen. Julkaisun jälkeen voisin testata sovellusta isommalla kohderyhmällä ja esimerkiksi päiväkotien olisi helppo osallistua testaukseen.

7 YHTEENVETO

Opinnäytetyön tavoitteena oli luoda Applen iPad-tabletille sovellus, jonka tarkoituksena olisi auttaa lapsia pitämään mielenkiinto aikuisen antamia tehtäviä kohtaan. Ominaisuuksina on myös lasten välinen kilpailu, joka motivoi lasta tekemään enemmän annettuja tehtäviä päästäkseen tulostaulussa ensimmäiseksi. Opinnäytetyö toteutettiin Xcode-sovelluksella ja Swift-ohjelmointikielellä.
Opinnäytetyö eteni todella hyvin ja opin todella paljon uusia asioita tehdessäni opinnäytetyötäni. Sovelluksen ensimmäinen ulkoasun suunnitelma oli epäkäytännöllinen verrattuna valmiiseen tulokseen. Ulkoasun eri kehitysvaiheiden jälkeen saatiin aikaiseksi toimiva UI, jossa kaikki suunnitellut toiminnallisuudet saatiin onnistumaan. Kaikki toiminnallisuudet jotka olin suunnitellut toteutettavaksi opinnäytetyössäni, onnistuivat loistavasti.
Aikaisemmat kokemukset ja osaaminen Swift-ohjelmointikielestä tukivat hyvin sovelluksen kehittämistä. Uudet asiat, joita opinnäytetyössä olin suunnitellut opettelevani ja käyttäväni, olivat MVVM-ohjelmointimalli, Core data -tallennusjärjestelmä, puhelimen kamera- ja kuvakirjasto sekä Tab bar controller. Nämä asiat onnistuivat mielestäni erinomaisesti. Core datassa oli hieman aluksi vaikeuksia henkilöiden ryhmään lisäämisen osalta, mutta tämäkin onnistui hyvin pienien lisätutkimuksien avulla.
Olen todella tyytyväinen lopputulokseen ja uskon tämän sovelluksen tulevan auttamaan lapsia tekemään tehtäviä, joita vanhemmat antavat.

LÄHTEET

Core data 2019. Framework Core data. Viitattu: Helmikuu 2019 https://developer.apple.com/documentation/coredata
iOS-versions 2019. The History of iOS, from Version 1.0 to 12.0. Viitattu: Helmikuu 2019 https://www.lifewire.com/ios-versions-4147730
Xcode 2019. Xcode 10 overview. Viitattu: Helmikuu 2019. https://developer.apple.com/xcode/
Interface builder 2019. Interface Builder Built-In. Viitattu: Helmikuu 2019 https://developer.apple.com/xcode/interface-builder/
iOS-simulator 2018. Getting Started in Simulator. Viitattu: Helmikuu 2019. https://developer.apple.com/library/archive/documentation/IDEs/Conceptual/iOS_Simulator_Guide/GettingStartedwithiOSSimulator/GettingStartedwithiOSSimulator.html
Lapsen motivointi. 10 tapaa tukea lapsen sisäistä motivaatiota. Viitattu: Maaliskuu 2019. https://hyvinkasvatettu.com/2017/06/23/10-tapaa-tukea-lapsen-sisaista-motivaatiota/
Projektinhallinta menetelmät. 6 yleistä menetelmää projektityöhön (sis. Agile, Waterfall ja Kanban). Viitattu: Maaliskuu 2019 https://www.agendium.com/post/agile-waterfall-kanban-6-projektinhallintamenetelmaa
Swift 2019. Swift documentation. Viitattu: Helmikuu 2019 https://swift.org/documentation/