
# BenchCoach

Ville Korhonen

Haaga-Helia ammattikorkeakoulu

Tietojenkäsittelyn tradenomitutkinto

Toiminnallinen raportti

2024

## Tiivistelmä

**Tekijä(t)**
Ville Korhonen

**Tutkinto**
Tradenomi.

**Opinnäytetyön nimi**
BenchCoach

**Sivu- ja liitesivumäärä**
37 + 15

Tämän toiminnallisen opinnäytetyön ideana on luoda mobiilisovellus. Sovelluksen tarkoitus on helpottaa juniorijääkiekkovalmentajien työmäärää. Toiminnallinen osa pitää sisällään mobiilisovelluksen suunnittelun sekä luomisen eli koodaamisen. Lopputuloksena sovellusta käyttävä valmentaja voi tallettaa luomansa joukkueen pelaajille itse valitsemiaan tilastoitavia tapahtumia otteluista sekä koko joukkuetta koskevia, itse valitsemiaan tapahtumia. Nämä tilastoidut asiat, sekä joukkue että pelaajat, tallentuvat Firebase Realtime database- tietokantaan. Back endin täydentää Firebasen tarjoama Authentication, mikä todentaa käyttäjän ja varmistaa, että oikea henkilö kirjautuu ja pääsee käsiksi omiin tietoihinsa. Itse sovelluksen käyttöliittymä on luotu Expo-kehitysympäristössä käyttäen React Nativea.

Opinnäytetyön kirjallisessa osassa käydään läpi valmistuneen sovelluksen vaiheita. Tietoperustassa taas tutustutaan mobiiliohjelmointiin, back endiin sekä käyttöliittymään. Tietoperustan on tarkoitus auttaa lukijaa ymmärtämään, mitä sovelluksen läpikäynnissä kerrotaan raportin lopussa.

Toiminnallisen työn lopputulos on tarkoitus julkaista suljettuna testiversiona Google Play-kaupassa. Testaajina tulee toimimaan Ilveksen juniorivalmentajia.

**Asiasanat**
Mobiiliohjelmointi, Firebase, React Native, Expo

## Sisällys

# 1 Johdanto

Aiheeni on tabletilla tai muulla mobiililaitteella käytettävä sovellus, jossa käyttäjä voi tallentaa joukkueen, sen pelaajat sekä pelaajien tilastoja jääkiekko-ottelun aikana tai ottelun jälkeen otteluvideota katsottaessa. Päädyin tähän aiheeseen, sillä olen itse ollut jääkiekon parissa koko elämäni. Viimeiset kaksi vuotta olen ollut apuvalmentajana Ilveksen U18-joukkueessa. Osana rooliani on ollut tilastointi ottelutapahtumista, sillä Jääkiekkoliitto ei vielä tässä ikäluokassa tarjoa henkilökohtaisista tilastoista muuta kuin pelaajien maalit, syötöt, tehopisteet sekä jäähyminuutit. Monet seurat kuitenkin haluaavat tilastoida enemmän ja kerätä erilaista dataa ottelutapahtumista, niin Ilveskin. Päädyimme tilastoimaan tehotilastoa, eli ketkä pelaajista ovat jäällä, kun teemme maalin tai vastustaja tekee maalin. Lisäksi tilastoimme myös aloitukset, voitetut sekä hävityt. Niistä laskin aloitusprosentin. Tilastoimme vielä kiekonriistoja sekä laukausten blokkaamisia. Tein tämän kaiken A4-paperille jokaisessa ottelussa, niin, että kirjasin joukkueen kokoonpanon ja jokaisen pelaajan kohdalle tukkimiehenkirjanpidolla kyseiset tapahtumat. Ottelun jälkeen siirsin tilastot manuaalisesti Excel-tiedostoon. Muita mahdollisia tilastoinnin kohteita voisi olla vaikkapa laukaukset, miten monta laukausta pelaaja ampuu ja montako niistä meni maalivahdille asti. Mietin monesti jo ennen ottelua, kun kirjasin kokoonpanoa ja valmistelin tilastopohjan, voisiko tämän tehdä digitaalisesti. Ainainen kirjoittaminen ja tilastoiden hidas siirtäminen voisi tapahtua lähes itsestään sovelluksella, mihin tarvitsisi vain painaa nappeja ottelun aikana. Joukkueellamme on käytössä tabletti, mihin sovelluksen voisi ladata. Sovelluksen avulla valmentajan työmäärä pienenisi, sillä hitaat välivaiheet tilastoinnissa jäisivät pois. Valmentajille jäisi enemmän aikaa pelaajan kohtaamiseen sekä opettamiseen, mikä on kuitenkin tärkeimpiä valmentajan tehtäviä.

Juniorijääkiekko on mielestäni muuttunut lähemmäksi ammattilaismaailmaa. Pelaajat ovat urheilullisempia, joukkueet ovat paremmin valmennettuja sekä seurojen tarjoamat resurssit ovat parempia. Vaatimustaso on noussut paljon siitä, kun itse olin juniorijääkiekkoilija. Pelaajat myös vaihtelevat herkemmin seuraa hakiessaan parasta väylää uralleen sekä vastoinkäymisiä kohdatessaan. Tällaisissa vastoinkäymisissä monesti on kyse pelaajan sekä pelaajan vanhempien ja muiden pelaajaan liittyvien sidosryhmien pettymyksestä pelaajan omaan rooliin joukkueen sisällä. Pelaajan, vanhempien tai pelaajan ympärillä olevien sidosryhmien kritiikki kohdistuu yleensä tällaisissa tilanteissa valmentajaan. Laadukas tilastointi antaa valmentajalle työkalun, mikä voisi auttaa pelaajia sekä heidän sidosryhmiään ymmärtämään valmentajien päätöksiä. Ammattimaisuus on myös läsnä, kun pelaajia arvostellaan seuran sisällä sekä tehdään ranking-listoja. Tämä tilastointi auttaisi myös pelaajien rankkaamisessa sekä antaisi valmentajille pelaajien kanssa pidettäviin kehityspalavereihin hieman kättäpitempää.

Näen, että urheilussa ja etenkin jääkiekossa on vielä paljon kehitettävää digitalisaation näkökulmasta. Tämä juniorijääkiekkoilun tilastointi on yksi asia. Datan analysointi on vasta tulossa jääkiekkomaailmaan ja vielä harvalla joukkueella on oma data-analyytikko. Wisehockey tilastoi Suomen Liigan ottelut ja tarjoaa paljon dataa seuroille sekä kuluttajille, mutta juniorijääkiekossa on vielä paljon saavutettavaa tilastoinnin ja datan analysoinnin kautta.

Sovelluksen rakentamiseen käytin Expo- kehitysympäristöä sekä ohjelmointiin React Nativea. Nämä ovat tuttuja itselleni mobiiliohjelmoinnin kurssilta. Tietokannaksi valitsin Firebase Realtime Database-tietokannan.

# 2 Mobiiliohjelmointi

## 2.1 Mobiililaite

Mobiililaite on tavallisesti kooltaan sellainen, jonka voi ottaa helposti mukaan ja siitä löytyy arkielämää helpottavia ohjelmia, mutta myös viihdekäyttöön liittyviä ominaisuuksia, kuten radio tai peli. (Mikkonen 2004, 2.)

Tänä päivänä yleisimpiä mobiililaitteita ovat älypuhelimet, tabletit, älykellot ja e-kirjan lukijat. Käsite on melko laaja. Mobiililaitteeksi voidaan lukea sellaiset laitteet, joissa on joko wifi-yhteys tai mobiilidata. Niiden akun kesto on useampi tunti. Niihin voi syöttää tietoja sekä tarkastella erilaisia tietoja. Niihin voi ladata ohjelmia ja erilaisia ominaisuuksia. Ne ovat langattomia sekä kooltaan sellaisia, että niitä voidaan helposti kuljettaa mukana. (Lifewire 2022.)

Varmasti yleisin mobiililaite on älypuhelin. Tämä ensimmäisten kännyköiden jälkeläinen löytyy melkein jokaisen aikuisen taskusta. Siinä, missä tavallisella kännykällä pystyi soittamaan ja vastaanottamaan puheluita, lähettämään ja vastaanottamaan tekstiviestejä, voi älypuhelimella tehdä lähes mitä vaan. Älypuhelimissa on lähes tietokoneen verran tehoa, ja sillä otetut valokuvat vastaavat laadukkaan digikameran tasoa. Ihminen pärjää periaatteessa pelkällä älypuhelimella nykypäivän yhteiskunnassa, eikä välttämättä tarvitse tietokonetta. Toinen merkittävä ja yleinen mobiililaite on tabletti. Niitä löytyy monessa eri koossa ja näin niitä voi hyödyntää erilaisissa toiminnoissa. Pienimmät tabletit eivät ole paljon puhelinta isompia. Tabletit toimivat samalla tavalla kuin älypuhelimet, kosketusnäytöllä. Lisäksi niihin voidaan ostaa lisäosaksi näppäimistö, mikä auttaa pitämään ruudun paremmin näkyvissä sekä nopeuttaa hieman kirjoitusta. (Lifewire 2022.)

Aivan viimeisinä vuosina markkinoille on tullut paljon erilaisia älykelloja ja rannekkeita. Monelta valmistajalta on tullut omat versionsa. Kellot sekä rannekkeet paritetaan yleensä älypuhelimen kanssa ja niiden välillä pystyy jakamaan erilaista dataa. Älykelloihin on suunniteltu myös omia sovelluksia ja niiden toiminnot muistuttavat älypuhelimen sekä tabletin toimintaa. (Lifewire 2022.)

Älykelloista on myös lapsille oma versionsa, joka toimii kuin puhelin. Onko virallinen nimi sitten kellopuhelin, mutta laite on kuin ranteessa oleva puhelin. Sitä hallitaan vanhempien älypuhelimessa olevasta sovelluksesta. Sieltä määritetään, kenen kanssa lapsen on mahdollista olla yhteydessä viestien tai puheluiden muodossa. Kellopuhelimen kautta ei ole mahdollista päästä sosiaaliseen mediaan tai internetiin. Sovelluksen kautta näkee lapsen sijainnin sekä voi asettaa lapselle oman alueen. Kun lapsi on poistunut alueelta, tulee vanhempien puhelimen sovellukseen ilmoitus. (Mutsimedia 2023.)

## 2.2 Mobiilisovellus

Mobiilisovellus on ohjelma, joka on suunniteltu toimimaan nimenomaan jollakin mobiililaitteella, kuten vaikkapa älypuhelimella, älykellolla tai tabletilla. Mobiilisovellus käyttää laitteen toimintoja ja ominaisuuksia yleensä hyödykseen, kuten vaikkapa kameraa tai paikannusta. Mobiilisovelluksen asentaa yleensä käyttäjä itse tai laitteen valmistaja on ladannut valmiiksi sovelluksia. Käyttäjä voi valita ladattavan sovelluksen sovelluskaupasta. Nämä kaupat ovat yleensä kolmannen osapuolen hallinnoimia (Google Play ja Apple app store). Jotta kehittäjä saa sovelluksen kauppaan käyttäjien ulottuville, kulkee se tietyn protokollan läpi ja monesti sovelluksen julkaisusta kaupassa joutuu maksamaan pienen summan. (Salz & Moranz 2013.)

### 2.2.1 Natiivisovellus

Mobiilisovelluksen voi kehittää natiivina, eli se on rakennettu vain yhdelle tietylle käyttöjärjestelmälle, kuten Android tai iOS. (Bautomo.)

Tämä tarkoittaa, että sovelluksen koodi tulee tehdä kahdesti. Toinen Androidille ja toinen iOS:lle. Hyviä puolia natiivisovelluksessa on se, että sen voi luoda hyvin tarkasti tietylle laitteelle ja tietylle käyttöjärjestelmälle. Tällöin saadaan maksimoitua laitteen tarjoamat ominaisuudet sekä samalla säästettyä laitteen tehoja. Päivitettäessä natiivisovellusta tulee päivitys tehdä molemmille alustoille erikseen, mikä lisää työmäärää sekä kustannuksia. (Tecinspire 2022.)

### 2.2.2 Hybridisovellus

Sovelluksen voi koodata myös hybridisovelluksena, jolloin se toimii useammalla käyttöjärjestelmällä. Hybridisovelluksen kehittäminen säästää resursseja, kun se toimii koodauksen jälkeen usealla käyttöjärjestelmällä, kun taas natiivisovellus täytyy koodata jokaiselle käyttöjärjestelmälle uudestaan. (Bautomo.)

Hybridisovellusta ei helpolla pysty erottamaan natiivisovelluksesta, mutta niiden teknologioissa löytyy eroavaisuuksia. Hybridisovelluksessa on vain yksi koodipohja ja se paketoidaan natiivikuoreen. Tämä mahdollistaa sovelluksen käyttää eri käyttöjärjestelmien tuomia ominaisuuksia laitteissa. Kuten hybridisovelluksen luomisessa, myös sen päivittämisessä tarvitsee työstää vain yhtä koodia. (Techinspire 2022.)

### 2.2.3 Progressive Web App - PWA

Hybridisovelluksen ja natiivisovelluksen lisäksi on mahdollista tehdä Progressive web app. Tällainen websovellus toimii selaimessa, eikä sitä ole pakko ladata sovelluskaupoista. Sovellus toimii kuten verkkosivut. Tällaisen sovelluksen voi halutessaan asettaa GooglePlay-kauppaan asiakkaille

ladattavaksi sovellukseksi. Tällöin ladataan vain pikakuvake, mikä säästää laitteen muistia ja resursseja. PWA-sovellus pyörii kaikilla laitteilla, mukaan lukien Mac- sekä Windows- koneiden selaimet. Tällaiset PWA-sovellukset päivittyvät siten, että käyttäjän ei tarvitse itse tehdä mitään. Erityisesti silloin kun halutaan maksimoida käyttäjämäärä, on tällainen PWA-sovellus hyvä, koska sitä voidaan käyttää sekä sovelluksena että selaimessa. (Techinspire 2022.)

## 2.3 Käyttöjärjestelmä

Mobiililaitteiden käyttöjärjestelminä suosituimpia ovat tällä hetkellä Linux-käyttöjärjestelmästä polveutuva Android sekä Applen hallinnoima iOS. Erona näissä on se, että Android on avoimen lähdekoodin versio ja iOS-järjestelmää voi muokata vain Apple. (Kotimikro 2022.)

### 2.3.1 Android

Android on yleisin käyttöjärjestelmä mobiililaitteille. Ensimmäinen versio Androidista julkaistiin vuonna 2008. Androidissa on kokonaan avoin lähdekoodin kehitysalusta, minkä vuoksi se on täysin muunneltavissa ja on jopa mahdollista tehdä täysin oma versionsa Androidista. (Harju 2013.)

Monet laitevalmistajat tekevät oman version Androidista erottuakseen paremmin toisistaan. (Kotimikro 2022.)

### 2.3.2 iOS

iOS on Applen luoma mobiilikäyttöjärjestelmä ja sitä käytetään esimerkiksi Applen iPhonessa ja iPadissa. iOS oli alkuaikoinaan iPhone OS ja vasta vuonna 2011 se muuttui iOS-nimiseksi. Noin joka viides mobiililaitteen käyttäjä käyttää Applen tarjoamaan iOS. Tämä on vuoden 2023 tilasto, se on korkein käyttäjämäärä sitten edellisen huipun vuodelta 2022. (Techtarget 2024.)

## 2.4 Kehitysympäristö

Kehitysympäristö voidaan määritellä siten, että se käsittää joukon erilaisia menettelytapoja sekä työkaluja, joilla voidaan kehittää, korjata virheitä ja testata suunniteltua ohjelmaa tai sovellusta. Ohjelmistokehityksessä kehitysympäristö kattaa joukon prosesseja sekä työkaluja, ja ne ovat tärkeässä roolissa, kun ohjelmaa ja sen koodia luodaan. Silloin tällöin tähän termiin viitataan, kun puhutaan integroidusta kehitysympäristöstä (IDE). IDE on ohjelmistokehitystyökalu, mitä ohjelmoija käyttää koodaamiseen, testaamiseen ja virheenkorjaukseen. Kehitysympäristö-termillä osoitetaan koko ympäristöä, kattaen kehitys-, esivaihe- sekä tuotantopalvelimet. IDE-termillä taas tarkoitetaan paikallista sovellusta, mitä käytetään koodaamiseen. Näillä kahdella termillä on paljon päällekkäisyyksiä. (Techopedia 2016.)

Eclipse IDE oli ensimmäinen Androidille kehitelty kehitysympäristö. (Harju 2013.)

### 2.4.1 Expo-kehitysympäristö

Expo on kehitysympäristö, missä voi kehittää mobiilisovelluksia. Expo on avoimen lähdekoodin yhteisö. Expon avulla ohjelmoija voi koodata, päivittää, lähettää sekä seurata sovelluksen toimintaa sen elinkaaren aikana. Voit käyttää valmiita kirjastoja tai omaa koodia. (Expo.)

Expo kehitysympäristössä koodatut sovellukset ovat React Native-ohjelmointikielellä tehtyjä sovelluksia. (Expo Docs.)

### 2.4.2 Node.js

Node.js on avoimen lähdekoodin ja eri alustojen välinen JavaScript-ajoympäristö. Se mahdollistaa JavaScript suorittamisen palvelimessa. Node.js on erittäin suorituskykyinen, sillä se käyttää V8 JavaScript-moottoria. Samaa moottoria käyttää myös Google Chrome. Node.js on suunniteltu tapahtumapohjaiseksi, sillä tarkoitetaan sitä, että se käsittelee sisään tulevat pyynnöt tapahtumina ja operoi käyttäen asynkronista I/O-mallia. Kun se suorittaa I/O-toiminnon, kuten vaikkapa lukee verkosta tai hakee tietoa tietokannasta, se ei tuhlaa suorittimen resursseja odottamiseen. Tämä mahdollistaa Node.js:n käsittelemään tuhansia samanaikaisia yhteyksiä yhdellä palvelimella niin, että virhekoodeja ei synny. (Nodejs.)

## 2.5 Ohjelmointikielet

Ohjelmointikieli on ohjelmoijan käyttämä tietokonekieli, jolla hän luo ja kehittää mobiilisovelluksia, tietokonesovelluksia tai vaikkapa verkkosivuja. (Tieturi.)

Ohjelmointikielet ovat joukko ohjeita, mitkä mahdollistavat ihmisen ja tietokoneen kommunikoinnin keskenään. Niin kuin kielillä, mitä käytämme puhuessamme ihmisten kesken, on myös ohjelmointikielillä oma syntaksi, rakenne, sanasto sekä slangi ja lyhenteet. Ohjelmointikielet mahdollistavat ihmisen innovoida, automatisoida sekä käyttää mielikuvitusta erilaisten ideoiden toteuttamiseen. Ohjelmointikielet jaetaan yleensä kahteen ryhmään, paradigmapohjaisiin sekä sovelluspohjaisiin kieliin. Ohjelmointikielet ovat käytössä kaikilla osa-alueilla ohjelmistokehityksen elinkaaren aikana. (GitHub.)

### 2.5.1 Paradigmapohjaiset kielet

Nämä paradigmapohjaiset kielet määritellään niiden ohjelmointiparadigman mukaan, joka edustaa tietynlaista ohjelmointityyliä. Imperatiiviset kielet ovat suunniteltu siten, että kuvaavat toimenpiteiden järjestystä ongelman ratkaisemiseksi. Ne kuvaavat, miten ohjelma toimii käyttäen komentoja,

jotka muuttavat ohjelman tilaa. Tällaisia kieliä ovat esimerkiksi C, C++ ja Java. Paradigmapohjaisia kieliä ovat myös funktionaaliset kielet, kuten Haskell tai Scala. Nämä toimivat siten, että ilmaisevat matemaattisten funktioiden avulla. Funktiot ovat tärkeässä osassa ja ne välttävät tilan muuttumista sekä muokattavia tietoja. Kolmas paradigmapohjainen osa on merkintäkielet. Ne on suunniteltu lisäämään metadataa tekstiin ja niitä käytetään usein tekstin muotoiluun ja esittämiseen verkkokehityksessä. Tällaisia kieliä ovat esimerkiksi HTML ja XML. (GitHub.)

### 2.5.2 Sovelluspohjaiset kielet

Sovelluspohjaiset kielet luokitellaan siten, mikä niiden käyttötarkoitus on. Verkkokehityksen kielet ovat tärkeässä osassa interaktiivisten ja visuaalisesti houkuttelevien verkkosivujen ja -sovellusten luomisessa. Esimerkiksi JavaScript tai PHP. Mobiilisovellusten kehityksen kielet, kuten Kotlin ja Swift, ovat räätälöity mobiilialustojen sovellusten rakentamiseen. Datatieteen kielet ovat suunniteltu tilastolliseen analyysiin, koneoppimisen ja datan käsittelyyn. Tästä hyvänä esimerkkinä toimii Python ja SQL. (GitHub.)

### 2.5.3 Ohjelmointikielien toiminta

Kehittäjä kirjoittaa ohjeita, mitä he haluavat tapahtuvan ohjelmointikielellä käyttäen tekstieditoria tai integroitua kehitysympäristöä. Kun ohjelmoija on saanut koodin valmiiksi, kone joko kääntää tai tulkkaa koodin, riippuen, millä ohjelmointikielellä koodi on kirjoitettu. Käännettäviä kieliä on esimerkiksi C++ ja tulkattavia kieliä Python. Käännettävässä kielessä koodi muutetaan binäärikoodiksi ennen kuin se suoritetaan. Tulkattavassa kielessä koodi käännetään ajon aikana. Tietokone suorittaa käännetyn tai tulkatun koodin noudattaen koodarin kirjoittamaa koodia, mikä saa aikaan halutun lopputuloksen. Koodia kannattaa käydä läpi, sillä on toivottavaa, että koodi on laadultaan hyvää. Joskus suorituksen aikana saattaa tapahtua virheitä, mitkä saattavat vaihdella loogisista virheistä koodissa odottamattomiin ongelmiin. Virheiden tunnistaminen ja korjaaminen on tarkkaa työtä ja saattaa olla aikaa vievää. Virhekorjaus on erittäin tärkeä osa ohjelmistokehitystä. Siihen on olemassa erilaisia tekniikoita sekä työkaluja. (GitHub.)

### 2.5.4 JavaScript

JavaScriptiä käytetään yleisesti määrittämään web-sovellusten toiminnallisuutta sekä käyttäytymistä. JavaScript toimii verkkoselaimessa, mahdollistaen koodarin määrittää, kuinka verkkosivut toimivat käyttäjän kanssa eli millainen vuorovaikutus heillä on. Sillä on suuri rooli asiakaspuolen skriptaamisessa, se mahdollistaa dynaamiset ja interaktiiviset käyttäjäkokemukset verkkosivuilla. (GitHub.)

### 2.5.5 React Native

Mobiiliohjelmoinnissa yksi yleisimmistä ohjelmointikielistä on JavaScript pohjainen framework - React Native Sen etuna on ohjelmoijien mielestä se, että sillä pystyy koodaamaan sovelluksia sekä Androidille että iOS:lle samanaikaisesti. (Netguru.)

React Native on siitä hieno framework, että se on avointa lähdekoodia sekä ilmainen. Valittavana on todella paljon erilaisia avoimen lähdekoodin JavaScript-paketteja. Niiden käyttäminen onnistuu asentamalla ne omaan projektiin. Tämä taas mahdollistaa vauhdikkaan kehitystyön ja projektin etenemisen. (Fraktio blogi.)

React Native on siis JavaScript-pohjainen framework, mikä on ottanut erityisesti jalansijan mobiiliohjelmoinnin parista. Eikä syyttä, sillä React Nativella voidaan koodata ohjelmaa samanaikaisesti Androidille ja iOS:lle, samassa koodipohjassa. React Native on avartanut monen koodarin repertuaaria web-ohjelmoinnista mobiilisovelluksien kehittämiseen. React on hyvin samantapainen kuin React Native. React Native on hieman tuoreempi ja samalailla sen ovat kehittäneet Facebookin insinöörit. Siinä missä React Nativea käytetään mobiilisovelluksien koodaamiseen, käytetään Reactia verkkosivujen rakentamiseen. (Netguru.)

# 3 Back End

Back Endillä tarkoitetaan ohjelmistokehityksessä sitä, mikä tapahtuu piilossa käyttäjältä. Eli verkkosivujen tai sovelluksen tietojen siirtoa, vastaanottoa, käsittelyä sekä erilaisten lomakkeiden ja tiedostojen käsittelyä, kirjautumista ja salasanan tarkistamista, käyttäjän varmistamista. Lisäksi tietokantoihin liittyvät yhteydet kuuluvat back endin piiriin. Kaikki palvelimella tapahtuva on siis back endiä. (Bautamo.)

## 3.1 Tietokanta

Tietokannat ovat datan ja tiedon järjestettyjä kokonaisuuksia, tavallisesti tietokannat tallennetaan tietokonejärjestelmään. DBMS eli Database management system on tietokannan hallintajärjestelmä ja sillä hallitaan tietokantoja. Tietokantajärjestelmään eli tietokantaan kuuluu data, DBMS sekä sovellus. (Oracle.)

Erilaisia tietokantatyyppejä on useita ja yleisimmät ovat relaatiotietokantoja. Lisäksi on myös hajautettuja tietokantoja, pilvitietokantoja sekä NoSQL-tietokantoja. (Geeksforgeeks.)

### 3.1.1 Relaatiotietokanta

Relaatiotietokantoihin tieto ja data on tallennettu taulukoihin, mitkä sisältävät sarakkeita sekä rivejä. Tällöin niitä on helpompi käsitellä. Taulukoidun tiedon hyötynä on se, että niitä voi helpommin muokata, hallita, päivittää, käyttää, valvoa sekä uudelleen järjestää. Relaatiotietokantojen yleistyminen alkoi 1980-luvulla. Ne ovat hyvin joustavia ja tehokkaita, kun puhutaan jäsennellyn tiedon käyttämisestä. (Oracle.)

Relaatiotietokannoista haetaan, lisätään, poistetaan sekä päivitetään tietoa erilaisilla SQL-kyselyillä. Nämä ovat yleensä lyhyitä, muutaman rivin mittaisia koodipätkiä. (IBM.)

### 3.1.2 NoSQL-tietokanta

NoSQL-tietokannat eroavat relaatiotietokannoista siinä, että tiedon ei tarvitse olla rakenteeltaan tietynlaista, vaan se voi olla tallennettuna jäsentelemättömänä tai osittain jäsenneltynä tietona. NoSQL-tietokannat ovat nostaneet suosiotaan sitä mukaan, kun sovelluksista on tullut monimutkaisempia ja vaativampia. (Oracle.)

### 3.1.3 Pilvitietokanta

Pilvitietokannat ovat tietokantoja, missä tieto on tallennettu jäsenneltynä tai jäsentelemättömänä. Ne voivat olla joko SQL- tai NoSQL-tietokantoja. Tämä tieto on tallennettuna yksityisessä,

julkisessa tai hybridipilvialustassa. Pilvitietokannoista löytyy kahta eri tyyppiä: perinteisiä sekä palveluina tarjottavia. Perinteinen pilvipalvelu-malli toimii hyvin samanlailla kuin tavallinen tietokanta. Perinteisessä pilvipalvelussa organisaatio ostaa tilaa pilvipalvelun tarjoajalta virtuaalikoneesta, sinne tietokanta ja tiedot sijoitetaan. Tällaisessa palveluina tarjottavassa pilvitietokannassa, palveluntarjoaja on vastuussa tietokantojen hallinnoimisesta sekä ylläpidosta. Tämä helpottaa siinä mielessä, että esimerkiksi päivitykset, varmuuskopiointi, tietoturva ovat ulkoistettu, mikä jättää organisaatiolle aikaa muuhun työntekoon. Tätä mallia kutsutaan DBaaS-malliksi. (Oracle.)

Skaalautuvuus on yksi pilvitietokantojen merkittävimmistä eduista. Tänä päivänä yritykset käyttävät yhä enemmän ja enemmän pilvitietokantoja. (Geeksforgeeks cloud database.)

## 3.2 Firebase

Firebase on backend palvelualusta, mikä on luotu vuonna 2011. Google osti sen tekijöiltään vuonna 2014. Firebase tarjoaa erilaisia palveluita ja työkaluja koodareille, ketkä kehittävät web- tai mobiiliohjelmia, erilaisia tietokantaratkaisuja, ratkaisuja kirjautumiseen sekä todentamiseen. Esimerkiksi suosittu Duolingo käyttää Firebasen palveluita. (Lido.)

### 3.2.1 Authentication

Firebase tarjoaa todentamisen palvelua eli authenticationia. Melkein kaikki sovellukset tarvitsevat tietoa käyttäjästä, jotta käyttäjän on turvallista käyttää tuotetta alustalla kuin alustalla. Authentication varmistaa sen, että käyttäjä pääsee käsiksi omiin tietoihinsa. Firebase authentication tarjoaa valmiita käyttöliittymäkirjastoja kehittäjän avuksi tunnistamaan käyttäjät. Se tukee tunnistautumista puhelinnumerolla, salasanalla tai mm. Googlen-tilillä. (Firebase.)

### 3.2.2 Realtime database

Realtime database on Firebasen tarjoama tietokanta, se on pilvipohjainen NoSQL-tietokanta. Data on tallennettu JSON-muodossa ja tietokanta toimii, vaikka sovellus olisi offline-tilassa. Se synkronoi datan kaikkien käyttäjien kesken reaaliajassa. Tämä mahdollistaa kaikkien käyttäjien saada uusinta dataa tietokannasta kaiken aikaa. Realtime database tarjoaa mahdollisuuden asettaa omat säännöt siitä, kuka pääsee käsiksi dataan, sekä kuka voi lisätä dataa tietokantaan. Firebasen authentikationin lisäämällä sovellukseen, voi kehittäjä säätää tietokannan käyttäjiä, miten data on saatavilla sovellukseen kirjautujille. (Firebase database.)

# 4 Käyttöliittymä

Käyttöliittymällä tarkoitetaan sitä osaa ohjelmistosta tai sovelluksesta, minkä kautta käyttäjä on yhteydessä laitteeseen tai tietokoneeseen. Englannin kielessä käyttöliittymästä puhutaan user interface:na eli UI:nä. UI:n avulla käyttäjä siis käyttää sovellusta, eli antaa erilaisia komentoja sovellukselle, syöttää erilaisia tietoja sekä vastaanottaa erilaisia tietoja sovellukselta. Käyttöliittymä voivat olla hyvinkin luovia tekeleitä, ne sisältävät erilaisia elementtejä. Yleensä sovelluksissa on esimerkiksi erilasia painikkeita, ikoneita, valikkoja ja tekstinsyöttökenttiä. Oikeastaan laite, millä sovellusta on tarkoitus käyttää, määrittelee, mitä käyttöliittymä voi pitää sisällään. Kun sovellusta suunnitellaan, on syytä paneutua siihen, että UI olisi mahdollisimman helppokäyttöinen, tehokas ja käyttäjäystävällinen. On hyvä ottaa huomioon, mihin sovellusta ollaan käyttämässä ja millaisia tarpeita käyttäjillä on sovellukselle. On toivottavaa, että UI olisi sellainen, että kommunikointi käyttäjän sekä järjestelmän välillä olisi mahdollisimman tehokasta ja miellyttävää. (Bautomo.)

## 4.1 Käyttäjäkokemus

Käyttäjäkokemus eli UX. UX lyhenne tulee englannin kielen sanoista user experience. Sillä tarkoitetaan käyttöliittymäelementtien kanssa tapahtuvaa vuorovaikutusta. Millaista on käyttää sovellusta tai palvelua? Onnistuuko halutut toiminnot ja toiminnallisuudet helposti vai vaikeasti käyttäjän näkökulmasta? Tämä kaikki riippuu siitä, millaisiin ratkaisuihin käyttöliittymän suunnittelijat ovat päätyneet. Kokonaiskäyttäjäkokemus syntyy tunteesta, mikä käyttäjälle jää ja siihen vaikuttaa kaikki pienimmätkin ratkaisut sovelluksen käyttämisen kannalta. Värit, tekstinkoko, napin painalluksen seuraukset sekä toimintojen loogisuus vaikuttavat tähän tunteeseen. (Timehouse, 2022.)

## 4.2 UI-suunnittelu

Käyttöliittymä suunnittelulla tarkoitetaan sovelluksen näkyvän osan suunnittelua. Se on eri asia kuin käyttäjäkokemuksen suunnittelu. Käyttäjäkokemuksesta puhutaan user experience:nä eli UX:nä. Käyttöliittymäsuunnittelusta puhutaan UI-suunnitteluna. Se on siis visuaalisen osan suunnittelua sekä kuinka interaktiivinen sovellus on. Se koskee sitä, minkälaisia painikkeita sovellus pitää sisällään. Painikkeiden muotoja sekä värejä, tekstisyöttökenttien tyylejä tai vaikkapa animaatioita. UI-suunnittelun tarkoitus on tehdä sovelluksen ulkonäöstä mahdollisimman houkutteleva ja myyvä. Tarkoitus on saada sovelluksen ulkonäkö ja kokonaisuus yhtenäiseksi, selkeäksi sekä eheäksi. Onnistuessaan UI-suunnittelu voi auttaa sovellusta menestymään, kun pahimmillaan epäselvä ja huonosti suunniteltu käyttöliittymä johtaa sovelluksen hylkäämiseen käyttäjän toimesta. Puhumattakaan mainehaitasta sovellusta ja sen tekijöitä kohtaan. (Timehouse, 2022.)

### 4.3 UX-suunnittelu

Käyttäjäkokemuksen suunnittelu nivoutuu hyvin paljon yhteen käyttöliittymäsuunnittelun kanssa. UX-suunnittelun tarkoitus on miettiä, kuinka saada sovellus tai ohjelma mahdollisimman käyttäjäystävälliseksi juuri kyseiseen tarkoitukseen. Tulee miettiä ja suunnitella, että sovellus on kokemuksena helppokäyttöinen ja selkeä, sekä käyttäjäpolku on suunniteltu siten, että se sopii juuri tähän tarkoitukseen. On myös syytä paneutua ongelmatilanteisiin ja kuinka ne voidaan tunnistaa käyttäjätutkimuksien avulla. (Timehouse, 2022.)

# 5 Oma sovellus - BenchCoach

## 5.1 Lähtökohta

Digitalisaatio on löytänyt tiensä myös juniorivalmennukseen, joskin vieläkin paljon käytetään kynää ja paperia. Sovelluksen lähtökohtana oli siis luoda vanhanaikaiseen toimintamalliin uusi, modernimpi versio, jolla saisi juniorivalmentajan työmäärää helpotettua sekä tiettyjä prosesseja nopeutettua; tilastoinnin muuttaminen manuaalisesta kynällä kirjoittamisesta siihen, että nappeja painettaessa kaikki tapahtuu lähes itsestään. Kohderyhmä, jolle valmis sovellus olisi omiaan, ovat juniorijääkiekkovalmentajat. Myös pelaajien vanhemmat voivat ottaa käyttöön sovelluksen seuratakseen oman jälkikasvunsa otteita tilastoinnin näkökulmasta. Omakohtainen kokemus asiasta ajoi eteenpäin ja olinkin suunnitellut mielessäni ominaisuuksia sovellukselle paljon ennen kuin aloitin tämän opinnäytetyön. Tavoite on julkaista elokuussa 2024 testiversio Google Play:n kautta. Tavoitteena on myös kehittää tuotos, joka voisi auttaa itseäni myös työllistymään.

### 5.1.1 Tavoitellut toiminnallisuudet

Aikaisemmin tämä otteluista tilastointi oli tapahtunut siten, että A4-paperille merkittiin kokoonpano ja jokaisen pelaajan alle merkittiin kynällä tukkimiehen kirjanpidolla niitä asioita, joita olimme sopineet tarkkailevamme. Tämän jälkeen siirsin nämä tiedot Excel-tiedostoon, missä tein vielä yhteenlaskuja, jotta sain koko kauden yhteenvedon pidettyä ajan tasalla. Tämä oli siis henkilökohtaisen tilastoinnin toimintamalli. Samalla tavalla koko joukkuetta koskevia tilastoja tehtiin, tosin nämä katsottiin nauhalta otteluiden jälkeen.

Toiminnallisuuksia, joita sovelluksen tulisi pitää sisällä, olisivat mahdollisuus kirjata dataa ylös, tallentaa dataa sekä hakea dataa, jotta sitä voi tutkia ja analysoida. Tämä tarkoittaa oman sovellukseni kohdalla sitä, että pitää pystyä tallentamaan tietokantaan joukkue sekä pelaajat, lisätä henkilökohtaisia tilastoja, joita voi valita tallennettavaksi, voida ladata joukkue, mistä voi valita kokoonpanon otteluun ja hakea pelaajien tilastoja niin ottelukohtaisesti kuin koko kauden osalta. Matkan aikana halusin vielä ominaisuuden, jolla voi jakaa näitä tilastoja Excel-tiedostona. Kaikki tämä tulisi olla käyttäjäkohtaista, joten rekisteröinti ja kirjautuminen tulisivat olla myös osa sovellusta. Lopputulos on onnistunut, mikäli nämä toiminnallisuudet onnistuvat ja sovellus toimii ilman virheitä sekä ristiriitoja.

### 5.1.2 Valitut työkalut

Työkaluina ja toimintamalleina luotan koulun kursseilta tutuiksi tulleisiin prosesseihin. Sovellus luotiin Expo-kehitysympäristössä. Syy tuttuihin toimintamalleihin nojaamiseen on selvä, sillä aikaa halusin säästää mahdollisimman paljon itse sovelluksen koodaamiseen ja todennäköisten ongelmien

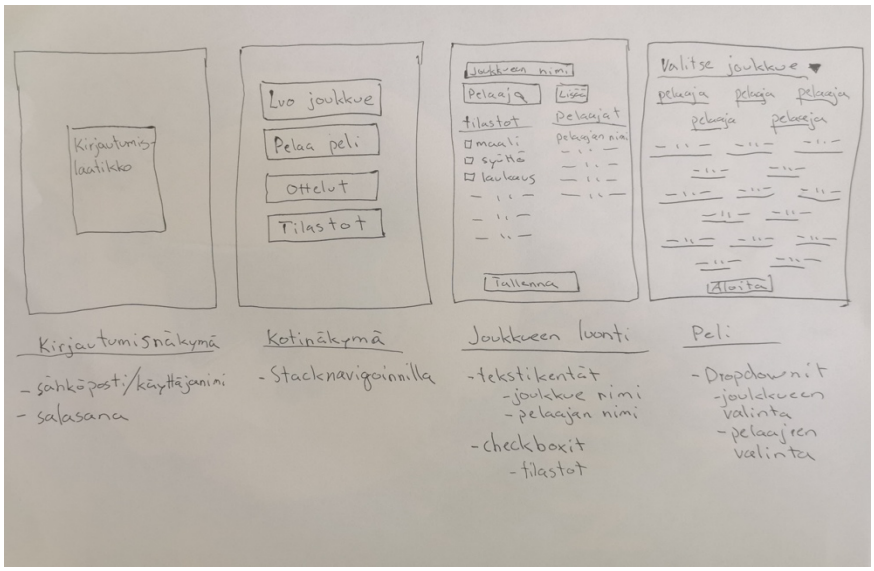
ratkomiseen. Tämä kehitysympäristö on erittäin tuttu jo mobiiliohjelmoinnin kurssilta. Käytin omaa puhelintani Expo Go-sovelluksen kautta ja latasin Android Studion kautta Android tabletin emulaattorin. Näin pystyin näkemään, miltä sovellus näytti tabletin ruudulla sekä puhelimen näytöllä testatessani sovellukseni toimivuutta. Editorina käytin Visual Studio Codea. Tätä olen käyttänyt koko ajan, kun olen koodannut. Lisäksi opiskelijana sain GitHub Copilotin veloituksetta käyttööni lataamalla sen GitHub:n sivuilta. Lisäksi käytin myös toista tekoälyä, sillä ongelmien ratkaisussa käytin OpenAi:n chat gpt:tä. Tällä tavoin nopeutin työhön käytettyä aikaa. Ohjelmointikielenä käytin React Nativea, sillä sen mahdollistama koodaaminen sekä Androidille että iOS:lle samaan koodipohjaan, oli merkittävä syy valintaani. Tämä helpottaa julkaisua Applelle, mikäli koen, että sovellukselle olisi tarvetta silläkin alustalla.

### 5.2 Suunnitelma

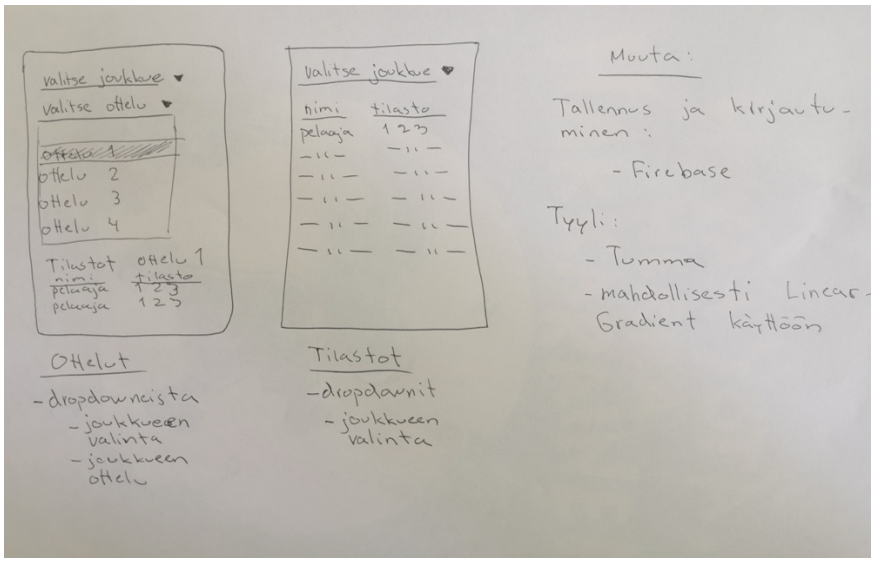
Valmensin juniorijääkiekkojoukkuetta pelikaudet 2022–2023 sekä 2023–2024. Valmennusryhmässä yhtenä osana rooliani oli tilastoida ottelun tapahtumia otteluiden aikana. Itselläni oli siis hyvin vahva ajatus sovelluksen vaatimuksista ja siitä, millaiseen käyttöön se tulisi. Näiden kokemuksien pohjalta aloin suunnitella sovellusta. Isoimmat mietteeni olivat sovelluksen helppokäyttöisyys sekä eri ikäisten joukkueiden tarpeet tyydyttävä lopputulos. Väriteemaltaan olen mieltynyt tumman teeman kannattajaksi. Kieleksi valitsin englannin, siihen on kaksikin erilaista syytä. Ensimmäinen on se, että mielestäni melko moni IT-alan töistä sekä opinnoista tapahtuu englanniksi. Toinen syy oli se, että tämä mahdollistaa suuremman käyttäjämäärän. Tunnen ihmisiä kiekkomaailmasta ympäri Eurooppaa, joten saatan tarjota tätä myös kotimaan rajojen ulkopuolelle testiin.

#### 5.2.1 Rautalankamalli

Rautalankamalli tehtynä paperille kynällä. Siihen on hahmoteltu hyvin suuripiirteisesti, miltä sovellus tulee näyttämään ja minkälaisia ominaisuuksia siinä mahdollisesti käytetään. Tämä oli lähtökohta, mistä aloin rakentaa sovellusta. Oma epävarmuus osaamisestani oli osasyy siihen, etten hirveän tarkkaa suunnitelmaa uskaltanut rakentaa. Halusin ennemmin tilanteen, missä sovellusta kasvatetaan, mikäli osaaminen sen suo, kuin että joudun tinkimään monesta ominaisuudesta osaamiseni takia.




Kuva 1. Rautalankamalli osa 1




Kuva 2. Rautalankamalli osa 2

### 5.3 Vaihe 1

Tämä kattaa sovelluksen alustuksen ja luomisen Expo-kehitysympäristössä sekä Firebasessa. Lisäksi ensimmäiseen vaiheeseen kuului myös kirjautumisen luominen käyttöliittymään sekä kotinäkymän luominen. Lisäksi tein visuaalisia ratkaisuja jo tässä vaiheessa.

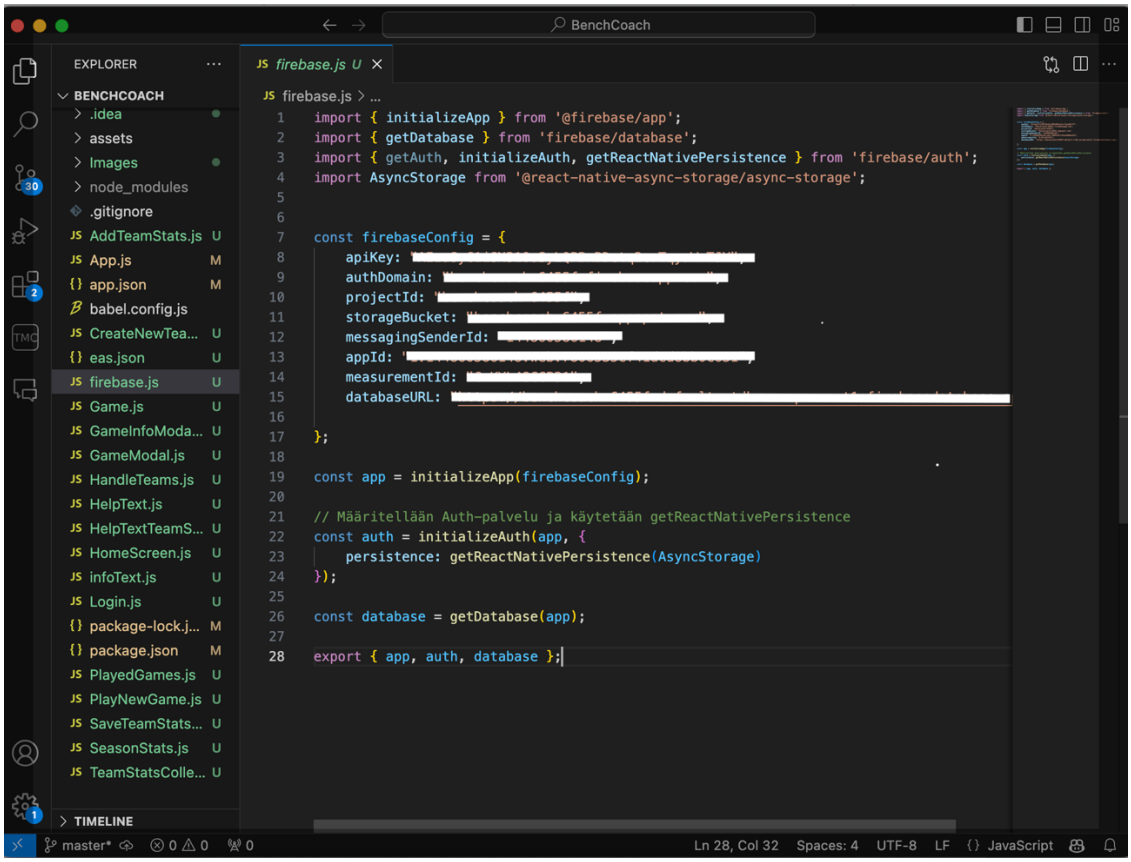
#### 5.3.1 Alustus

Expo-ohjelma alustetaan komentorivillä, se vaatii toimiakseen Node.js ja Expo Cli:n Nämä olen ladannut jo aikaisempien projektien aikana. Seuraavaksi alustin sovelluksen komennolla:

```
villekorhonen@Ville-MBP ~ % npx create-expo-app BenchCoach
```

Kuva 3. Komentorivin kuvankaappaus alustuksesta.

Tämä komento luo sovelluksen ja itse sovelluksen koodaaminen voi alkaa. Seuraavaksi loin sovelluksen backendin pohjan Firebasessa. Firebase consolessa luodaan uusi projekti, joka nimetään. Tässä oli ensimmäinen tiukempi paikka, sillä Firebase ei ollut itselleni täysin tuttu. Yhden projektin olin siellä tehnyt. Ohjeet ovat tosin melko hyvät. Valitsin Firebasen, koska se oli osittain tuttu, enkä halunnut käyttää aikaani täysin uusien menetelmien opetteluun. Sen tarjoama backend tietokantoineen sekä todennuksineen vakuuttivat minut. Halusin päästä projektissani eteenpäin. Firebasella on kolme eri projektivaihtoehtoa: Android, iOS sekä Web. Valitsin näistä Web-projektin, sillä itselläni oli halu tehdä hybridisovellus. Lisäksi tämä web-malli mahdollistaa integroinnin missä tahansa web-kehitysympäristössä. Tietokannaksi valitsin Realtime databasen, joka on pilvipohjainen NoSQL-tietokanta. Sinne tallennettava data on JSON-muodossa ja se sykronoi tiedot reaaliajassa, mikä teki itseeni vaikutuksen. Koin, että se sopi tälle omalle sovellukselleni hyvin, sillä otteluiden aikana viimeisimmät tilastotiedot tulee olla saatavilla heti. Lisäksi koin, että olisi ollut liian vaivalloista rakentaa reaalitietokanta tähän omaan tarkoitukseeni yksin.



Kuva 4. Firebase.js

### 5.3.2 Sisäänkirjautuminen ja kotinäkymä

Kun alustus oli valmis, siirryin rakentamaan itse sovellusta. Ensimmäinen asia oli kirjautumisen luominen ja testaaminen, että yhteydet Firebasen ja sovellukseni välillä toimivat. Kirjautuminen on suunniteltu niin, että käyttäjä kirjaa sähköpostinsa ja määrittää itselleen salasanan. Tämän jälkeen käyttäjän on käytävä klikkaamassa Firebasen lähettämää varmennuslinkkiä omassa sähköpostissaan. Vasta tämän jälkeen käyttäjä voi kirjautua sisään. Firebase luo käyttäjälle oman userID:n, mikä toimii tietokannassa käyttäjän ID:nä, minkä alle hänen tallentamansa data asettuu. Firebasella on myös toiminto, mikäli käyttäjä on unohtanut salasanansa. Rakentelin sen kirjautumisen yhteyteen ja syöttämällä sähköpostinsa käyttäjä saa viestin sähköpostiin ja siellä olevan linkin kautta voi asettaa uuden salasanan. Sisäänkirjautumisen lisäksi tyylittelin melko pitkälle jo ulkoasua. Teinkin oman logon sovellukselle ja asetin sen taustalle sovellukseen. Säilytin tumman teeman, sillä mielestäni se on tyylikäs. Lisäksi käytin React Native Elements-kirjaston Input- sekä Button-komponentteja tyylittelemään tekstinsyöttökenttää ja nappeja.


Kuva 5. Sisäänkirjautumisnäkymä.

Kotinäkymän suunnittelin mahdollisimman selkeäksi sekä yksinkertaiseksi. Olin alkuun suunnitellut jättäväni tuon sisäänkirjautumisen takana olevan logon myös tähän taustalle, mutta mielestäni näkymä oli liian sekava. UI:n näkökulmasta lisäsin tähän sivulle LinearGradient-komponentin, mikä tekee sen, että taustavärin saa muuttumaan yhdestä väristä toiseen suoraviivaisesti. Tämä komponentti on ”expo-linear-gradient”-paketista. Lisäksi halusin, että sovellusta voi käyttää sekä tabletilla että puhelimella, joten lisäsin ”react-native-responsive-fontsize” -kirjastosta RFValuen. Se mahdollisti responsiivisen fonttikoon määrittelyn eri näyttökokojen mukaan. Lisäksi käytin prosentteja painikkeiden koon määrittelyyn. Tässä vaiheessa, kun olin päässyt vauhtiin, koodasin juurikansion sellaiseksi, että siihen tarvitsee tehdä vain lisäyksiä. Navigaatioksi olin valinnut StackNavigationin. Mielestäni se soveltui paremmin omaan ohjelmaani kuin TabNavigation, enkä halunnut alavalikkoa. Nämä navigaation käytettävät komponentit löytyvät ”React-Navigation”-kirjastosta. Navigaation lisäksi rakentelin headerin, minkä jätin pois sisäänkirjautumisesta, mutta muissa osin sovellusta se on näkyvillä. Header sekä ikonit ovat samasta React Native-kirjastosta kuin sisäänkirjautumissivun elementit. Kun mietin UI:ta, halusin siihen jotakin, mikä viittaa vanhaan tapaan toimia valmentamisessa ja tilastoinnissa. Keksin, että lisään tussimaisen fontin. Tällaisen löysin ”Permanent-Marker_400Regular”:sta. Se on fontti Expo Google Fonts-kirjastosta, mitä pystyy käyttämään ”useFonts”-hookin avulla.


Kuva 6. Kotinäkymä.

## 5.4 Vaihe 2

Kun olin saanut luotua sovelluksen juuren, yhdistettyä sovelluksen Firebaseen sekä ensimmäiset tärkeät komponentit, oli aika paneutua siihen, miten sovelluksen toiminnot tulisivat toimimaan. Tämä vaihe kattoi komponentit, mitkä mahdollistavat joukkueen luonnin, pelaajien tallentamisen, haluttujen tilastojen valinnan sekä itse ottelun luomisen.

### 5.4.1 Joukkueen, pelaajien sekä tilastojen luonti

Tämä vaihe oli ensimmäinen, mikä vaati aikaa sekä suurta pohtimista. Nimenomaan se, että miten luon tietokannan, miten kaikki tieto tallennetaan. Niin kuin sanoin, Firebasen tietokannat eivät ole itselleni niin tuttuja, niin jouduin miettimään paljon rakennetta, miten kaikki tieto järjestetään, niin ettei tietokantaan tule isoja ristiriitoja. Lähtökohtana oli se, että jokainen sovelluksen käyttäjä saa oman solmun tai haaran. Käyttäjällä on oma userID, minkä alle hänen tallentamansa data tulee. Tämän jälkeen tietokantaan käyttäjän alle tallentuu joukkue. Jokaiselle joukkueelle tallennetaan joukkueen nimi, pelaajat, tilastot sekä ottelut. Jokaisella joukkueella on teamID ja jokaisella pelaajalla on oma playerID. Näin pystymme hakemaan tiedot yksilöllisen avaimen myötä. Lisäksi joukkueella on pelaajien henkilökohtaiset tilastot merkattuna erikseen, omaan statistics haaraan. Pelaajat-haarassa on pelaajien nimi, tilastot, sekä playerID. Games-haarassa ottelut tallennetaan yksilöllisen avaimen alle ja ne kattavat kokoonpanon, siten että jokainen hyökkäysketju ja puolustajapari ovat omina haaroinaan. Näissä hyökkääjien ja puolustajien omissa haaroissa on tallennettuna playerID sekä tilastot eli stats. Lisäksi ottelun alle tallennetaan päivämäärä, vastustaja sekä pelipaikka. Eli onko ottelu koti- vai vierasottelu.


Kuva 7. Tietokannan rakenne kuvakaappauksella.

```
{
  "users": {
    "WUw3cQ08KceyeNu5Y1FQOA2XaLI3": {
      "userID": "WUw3cQ08KceyeNu5Y1FQOA2XaLI3"
    }
  },
  "teams": {
    "-O5CkS-Sr2WUTq9Bpbd3": {
      "games": {
        "-O5CIGv5PAqJTGYyOwRv": {
          "date": "2024-08-27T08:19:00.000Z",
          "defenders1": [
            {
              "playerID": "-O5CkLG7itioMwvv8_W7",
              "stats": { //Valitut henkilökohtaiset tilastot
              }
            },
            {
              "playerID": "-O5CkJ1Ei90gFZcLU9vu",
              "stats": { //Valitut henkilökohtaiset tilastot
              }
            }
          ],
          "forwards1": [
            {
              "playerID": "-O5CkGQgH5VetWNToQy2",
              "stats": { //Valitut henkilökohtaiset tilastot
              }
            },
            {
              "playerID": "-O5CkEeoFKuCKYdt_Ijh",
              "stats": { //Valitut henkilökohtaiset tilastot
              }
            },
            {
              "playerID": "-O5CkDQag8LmfkEfNl0i",
              "stats": { //Valitut henkilökohtaiset tilastot
              }
            }
          ],
          "isAway": false,
          "isHome": true,
          "isPlayoffGame": false,
          "isPreseasonGame": true,
          "isRegularSeasonGame": false,
          "opponent": "Hifk",
          "team": {
            "name": "Ilves",
            "teamID": "-O5CkS-Sr2WUTq9Bpbd3"
          }
        },
        "name": "Ilves",
        "players": {
          "-O5CkDQag8LmfkEfNl0i": {
            "name": "Ville Korhonen",
            "playerID": "-O5CkDQag8LmfkEfNl0i",
            "stats": {//Valitut henkilökohtaiset tilastot
            }
          },
          "-O5CkEeoFKuCKYdt_Ijh": {
            "name": "Sami Sandell",
            "playerID": "-O5CkEeoFKuCKYdt_Ijh",
            "stats": {//Valitut henkilökohtaiset tilastot
            }
          },
          "-O5CkGQgH5VetWNToQy2": {
            "name": "Tommi Huhtala",
            "playerID": "-O5CkGQgH5VetWNToQy2",
            "stats": {//Valitut henkilökohtaiset tilastot
            }
          },
          "-O5CkJ1Ei90gFZcLU9vu": {
            "name": "Arto Tukio",
            "playerID": "-O5CkJ1Ei90gFZcLU9vu",
            "stats": {//Valitut henkilökohtaiset tilastot
            }
          },
          "-O5CkLG7itioMwvv8_W7": {
            "name": "Teemu Kesä",
            "playerID": "-O5CkLG7itioMwvv8_W7",
            "stats": {//Valitut henkilökohtaiset tilastot}
          }
        },
        "statistics": { // valitut henkilökohtaiset tilastot
        },
        "teamID": "-O5CkS-Sr2WUTq9Bpbd3"
      }
    }
  }
}
```

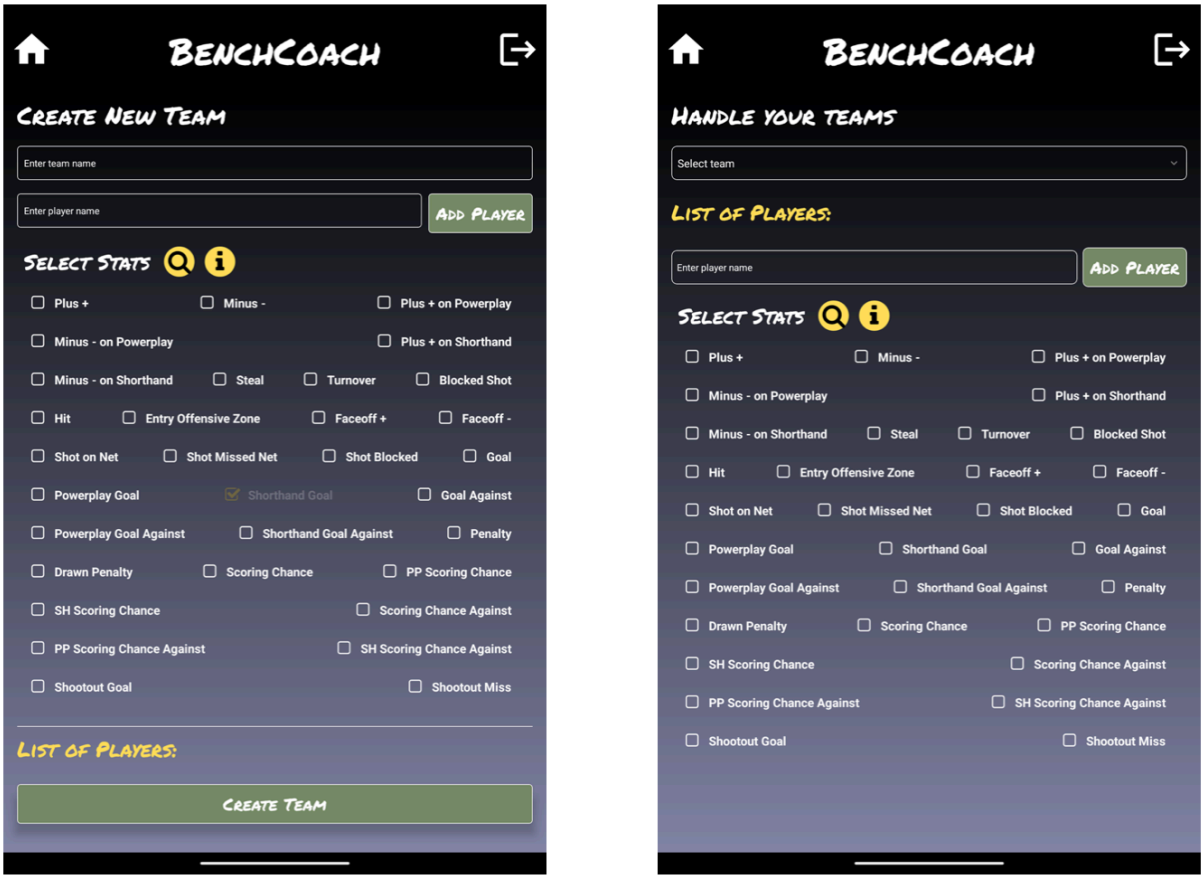

Kuva 8. Esimerkkikuva tietokannasta JSON-muodossa

Yllä olevassa kuvassa (Kuva 8) on esimerkki tietokannan rakenteesta erään käyttäjän osalta. Helpoiten tietokanta on selitettynä niin, että on käyttäjä. Käyttäjällä on joukkue. Joukkueella on ottelut, pelaajat sekä tilastot. Kriittisillä asioilla, kuten käyttäjällä, otteluilla ja pelaajilla on kaikilla identifioiva avain eli ID. Tietokannan säännöt ovat sellaiset, että vain varmennettu käyttäjä voi lukea ja kirjoittaa tietokantaan. Tämän kanssa meni ylivoimaisesti eniten aikaa ja välillä tuntui turhauttavalta, kun sovelluksen tekeminen ei edennyt päiviin. Jatkuvaa testaamista, että mikä toimii. Varmasti tässä on turhaakin tavaraa tallentunut tietokantaan, mutta taitojeni rajallisuus ei mahdollistanut muuta.

Joukkue, pelaajat sekä henkilökohtaiset tilastot valitaan yhden näkymän alta. Tämä näkymä on Create new team. Siinä annetaan joukkueelle nimi. Pelaajien nimet kirjoitetaan yksitellen ja ne lisätään listalle painamalla add player-nappia. Tämä ei vielä lisää pelaajaa tietokantaan vaan erilliselle listalle. Kun kaikki joukkueen pelaajat ovat listalla, valitaan tilastoitavat henkilökohtaiset tilastot, mitä halutaan pelaajille tallentaa. Tämä tapahtuu painamalla haluttua tilasto-checkboxia. Kun joukkue on valmis, painetaan create team-nappia ja joukkue tallentuu tietokantaan käyttäjän alle siten, että kaikille pelaajille tulee ne samat tilastot, jotka käyttäjä on juuri valinnut. Sovellus luo joukkueelle yksilöllisen ID:n sekä pelaajille omat playerID:t. Tilastojen valinnassa yritin pitää mielessä, että käyttäjiä voisi olla mahdollisimman laajalla skaalalla. Näin ollen monenlaisia tilastoja on valittavana. Koitin pitää ulkoasun yksinkertaisena ja lisäsin ohjeita käyttäjälle helpottaakseni käyttäjäkokemusta. Tilastot halusin määrittää jo tässä vaiheessa, eikä käyttäjä voi luoda esimerkiksi omia tilastoja tai tilastonimiä. Tämä johtuu siitä, että halusin luoda mahdollisimman paljon valmiita laskutoimituksia, mitkä mahdollistavat paremman käyttäjäkokemuksen. Tästä esimerkkinä, vaikka aloitukset, voitetut sekä hävityt. Näin pystyn tarjoamaan käyttäjille aloituksista voittoprosentin. Keltainen ympyrä, missä on suurennuslasi-ikoni, näyttää tilastoitavien asioiden selityksen. Toinen keltainen ympyrä, missä on i-ikoni, kertoo ohjein, miten toimia kyseisessä näkymässä.

Yhden pelikauden aikana tapahtuu yleensä muutoksia joukkueiden kokoonpanoissa. Halusin varautua tähän mahdollisuuteen ja siksi olikin luonnollista luoda näkymä, jossa on mahdollista tehdä muutoksia. Tämän näkymän nimi on Handle your teams. Komponentti on lähes identtinen joukkueen luomiseen tarkoitetun komponentin kanssa. Tässä näkymässä pystyy siis lisäämään uuden pelaajan joukkueeseen tai poistamaan. Lisäksi tässä näkymässä lisätään uudelle pelaajalle tilastoitavat asiat. Komponentti toimii siten, että alasvetovalikosta valitaan joukkue, jonka kokoonpanoa halutaan muokata. Haluttu joukkue tulee listana näkyviin. Yksittäisen pelaajan voi poistaa painamalla roskakori-ikonia nimen perästä. Virhetilanteiden varalta sovellus kysyy alert-viestin, onko käyttäjä varma pelaajan poistamisesta. Tässä komponentissa pelaajan lisääminen tapahtuu siten, että kun nimi on kirjoitettu syöttökenttään, tulee tilastoitavat asiat valita ennen kuin painetaan add player-nappia, sillä napin painallus lisää pelaajan tietokantaan saman tien. Sovellus ilmoittaa alert-viestillä, mikäli tilastoitavia asioita ei ole valittu. Samat keltaiset ympyrän muotoiset napit auttavat

käyttäjää myös tässä näkymässä. Yksi ominaisuus, jonka olisin halunnut, on että, kun joukkue on valittu, olisivat ne valintaruudut tilastojen osalta painettuina, mitkä on joukkueelle valittu jo luontivaiheessa. Tämä olisi järkevää ja poistaisi käyttäjältä väärien tilastojen asettamista uusille pelaajille. Käyttäjäkokemus olisi tällöin parempi. Mutta en saanut sitä tehtyä ja halusin siirtyä eteenpäin. Tämä on varmasti yksi kehityskohde tuleviin päivityksiin.




Kuva 9. Joukkueen luonti ja joukkueen hallinta.

### 5.4.2 Ottelun luominen

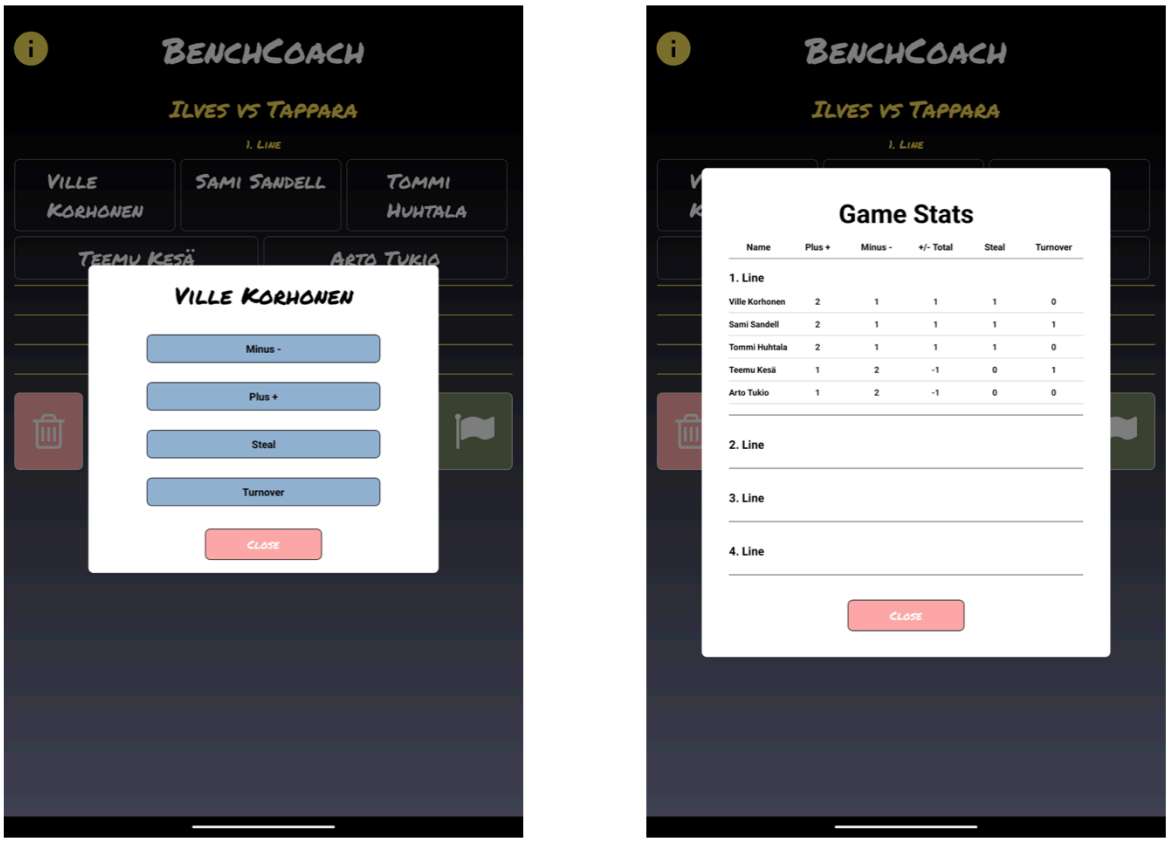
Ottelun luonti ja ottelusta tapahtuvien tilastojen kerääminen oli siis yksi suurimmista syistä tämän sovelluksen kehittämiseen. Digitaalinen tapa kerätä tilastot yhteen paikkaan, mistä niitä olisi mahdollista tutkia sekä jakaa eteenpäin. Ajatuksena oli se, että tilastoja pystyisi keräämään pelin aikana vaihtoaitiossa ja sen takia halusin, että käyttäjä voisi toimia mahdollisimman vaivattomasti. Se ei tietenkään poista sitä, etteikö sovellusta voisi käyttää myöhemmin ottelutallennetta katsoessa. Yleensä valmentajilla on kokoonpano, josta he näkevät kentälliset. Tätä käytin itsekin lähtökohtana. Tästä syystä myös tietokannan rakenne on sellainen, että sieltä löytyy omat kohdat sekä hyökkäysketjuille että puolustajapareille. Näin ollen ne oli helppo saada myös käyttöliittymään

esille. Lisäksi tämä tukee mahdollista jatkokehitystä mahdollisimman monipuolisen tilastoanalysoinnin puolesta, sillä ottelua voidaan analysoida myös ketjujen osalta tai puolustajaparien osalta. Lisäksi ottelun luonnissa valitaan vastustaja sekä päivämäärä. Ottelun paikka oli myös sellainen asia, minkä halusin mukaan. On tyypillistä, että joillakin pelaajilla on suuriakin eroja suorittamisessa verrattaessa koti- ja vierasotteluita. Tämä valinta mahdollistaa otteluiden suodattamisen jatkokehityksessä siten, että tilastot voidaan jakaa sekä pelkästään kotiotteluihin tai vierasotteluihin. Koska oma osaaminen on vielä kehittymässä, en saanut tätä ratkaisua vielä toimimaan. Muutenkin tässä kohtaa jouduin tietokannan kanssa hieman ongelmiin. En saanut millään toimimaan käyttöliittymässä sitä, että olisin hakenut tilastoitavat tiedot players-haarasta ja asettanut ne nollana aina uuteen otteluun, sillä halusin tallentaa koko kauden tilastot tuohon players-haaran tilastoihin. Tästä syystä mietin asiaa melko paljon maalaisjärjellä, kuinka saisin puhtaan tilaston otteluun pelaajan kohdalle, enkä niinkään teknillisesti. Keksin, että tallennan tilastoitavat asiat tietokantaan omaan statistics-haaraan nolla-arvoilla, mistä ne aina haetaan ja asetetaan jokaisen pelaajan kohdalle uuteen otteluun. Komponentti toimii siten, että ensin valitaan joukkue alasvetovalikosta. Kun haluttu joukkue on painettu, ilmestyy ruudulle kokoonpanon pohja. Painamalla pelipaikkaa, aukeaa ruudulle modal, missä on lista valitun joukkueen pelaajista. Painamalla halutun pelaajan nimeä, asettuu kyseinen pelaaja kokoonpanoon sille paikalle. Virhetilanteissa sovellus antaa alert-viestin, mikäli valittu pelaaja on jo ottelun kokoonpanossa. Kun kaikki pelaajat ovat valittu, kirjoitetaan tekstisyöttökenttään vastustajan nimi. Sitten valitaan Datetimepickeriä käyttävästä kalenterista päivämäärä ottelulle. Tässä oletuspäivämääränä on aina nykyhetki. Lopuksi valitaan koti- tai vierasottelu checkboxia painamalla. Kun käyttäjä painaa create game-nappia, ottelu tallentuu tietokantaan games osioon. Mikäli tietokannassa ei ole vielä yhtään ottelua joukkueen kohdalla, se luo tämän games-haaran sinne. Tässäkin olen koittanut ennakoida käyttäjän mahdollisia virheitä alert-ilmoitusten muodossa. Mikäli käyttäjä ei muista kirjoittaa vastustajaa tai valita, onko ottelu kotona vai vieraissa, tulee ruudulle ilmoitus.


Kuva 10. Ottelun luonti sekä tilastojen tallennus.

Kun ottelu on luotu, siirrytään uuteen näkymään. Siellä kokoonpanon pelaajat toimivat nappeina. Nimeä painamalla modal aukeaa ja sieltä tulee esiin joukkueelle valitut henkilökohtaiset tilastot. Nämä tilastot toimivat siten, että aina painettaessa haluttua tilastoa, kasvaa se yhdellä. Näkymän alareunassa ovat muut toiminnot. Sieltä löytyy nappi, jolla voi poistaa ottelun tietokannasta. Toinen nappi aukaisee toisenlaisen modalin, mikä näyttää sen hetken tilanteen pelaajien tilastoista tästä ottelusta. Tämä on hyödyllinen, mikäli seurataan vaikkapa keskushyökkääjien aloituksia ottelun aikana. Näin voidaan laittaa tärkeään aloitukseen sellainen, joka on ollut voitollinen ottelussa. Kolmas nappi on peruutusnappi, jolla voi peruuttaa viimeisimmän lisäyksen, mikäli käyttäjä huomaa, että painoi väärää tilastoa tai pelaajaa. Tällä napilla pystyy poistamaan vaikka kaikki lisäykset, sillä logiikka on poistaa aina viimeisin lisäys. En keksinyt tähän mitään parempaa toimintamallia, mutta jatkokehityksessä tämä voisi olla kehityskohde. Viimeisellä, neljännellä napilla ottelu lopetetaan. Tässä kohtaa käytin sitä maalaisjärkeä, mistä mainitsin. Eli, vaikka tilastot tallentuvat tietokantaan reaaliajassa ottelun kohdalle, halusin ne myös players-haaraan. Näin ollen taidoillani saisin ne sieltä helposti käyttöön, kun halutaan nähdä tilastot koko kauden osalta. Kun tuota ottelun lopettavaa nappia painetaan, lisää se jokaisen pelaajan kohdalla oleviin tilastoihin ottelun sekä ottelulle yhden arvon lisää. Tämän jälkeen lisätään ottelun tilastot pelaajan kohdalla oleviin tilastoihin.

Lopuksi siirrytään kotinäytölle. Tämä tulee olemaan sellainen asia, minkä haluan muuttaa sellaiseksi, että tilastot vain tallennetaan otteluun ja sieltä niitä voidaan hakea analysoitavaksi. Isoin syy tähän on se, että vaikka tämä oma tapani toimii, niin se on liian vaivalloinen. Sen huomaa jo siinä, kun testattaessa koko joukkueella, kesti melko kauan lopettaa ottelu, kun sovellus alkoi laskea taustalla tilastoja kahdellekymmenelle pelaajalle. Myös ottelunäkymässä on alert-ilmoituksia kriittisiin toimintoihin liittyen. Alert-viesti tulee näytölle, kun käyttäjä painaa joko ottelun poistavaa roskakorinappia tai ottelun päättävää lippunappia.




Kuva 11. Kaksi erilaista modalia ottelunäkymässä. Tilastojen tallennus ja ottelun tilastot koottuna.

### 5.5 Vaihe 3

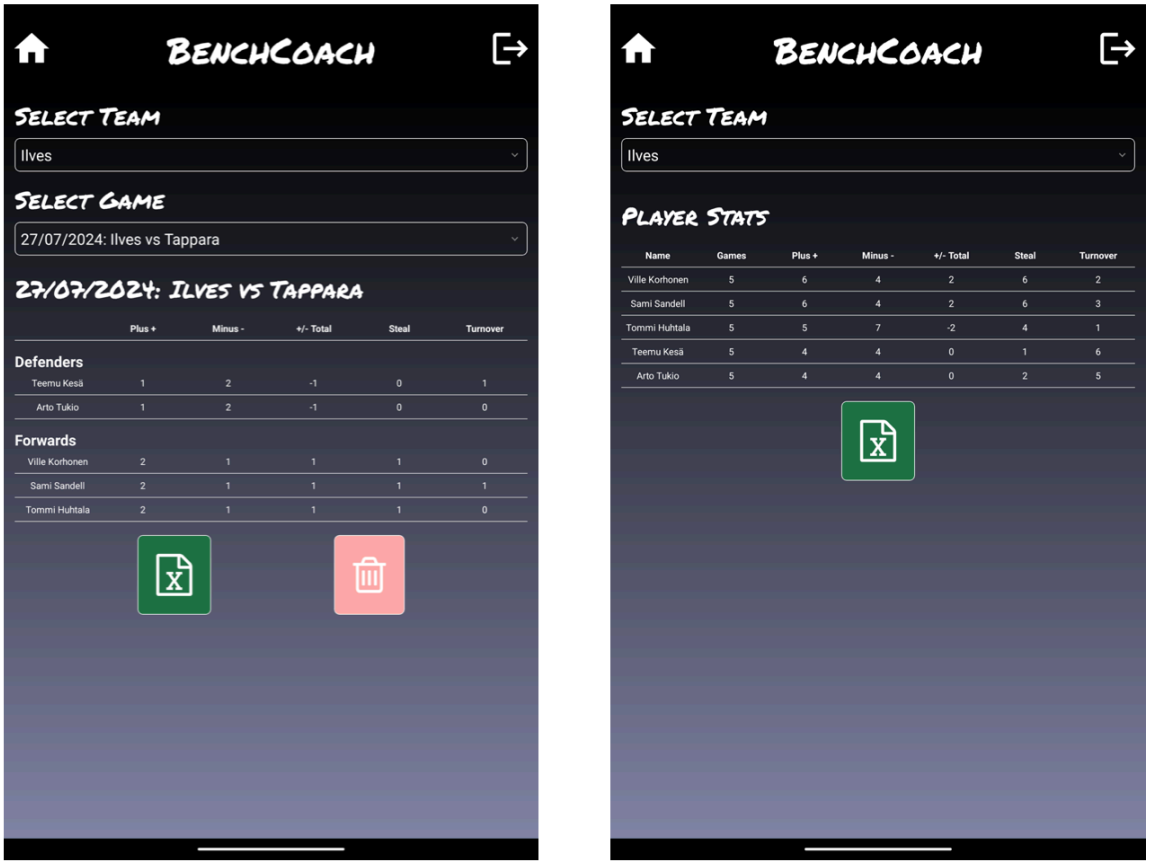
Kun joukkueen, pelaajien sekä otteluiden luominen onnistui, oli aika siirtyä siihen, kuinka tallennettu data näytetään. Kolmas vaihe paneutui siihen, että saadaan näytettyä sekä yksittäisen ottelun tilastot että koko kauden tilastot. Tämän lisäksi tulisi pystyä jakamaan näitä koosteita. Rakensin

molemmille oman osion. Tämänkin voisi varmasti tehdä samaan näkymään, mikä yksinkertaistaisi sovellusta.

### 5.5.1 Played Games ja Season Stats

Nämä molemmat osat sovelluksesta näyttävät melko samalta. Molemmissa näkymissä valitaan ensin oma joukkue alasvetovalikosta, minkä jälkeen dataa tulee esiin. Tosin ottelukohtaisessa tilastonäkymässä tulee vielä valita alasvetovalikosta ottelu, jonka tilastoja aikoo analysoida. Sitten ruudulle ilmestyy ottelun tilastot. Ottelun voi joko poistaa tai jakaa Excel-tiedostona. Poistaminen tapahtuu painamalla roskakorinappia ja Excel-tiedostona jakaminen tapahtuu painamalla vihreää nappia, missä on Excel-ikoni. Played games-komponentin kanssa ei ollut suurempaa harmia, sillä tiedot olivat ottelun kohdalla ja oli melko yksinkertaista noutaa tiedot. Suurimmat kompastumiseni koin, kun halusin saada käyttäjäkohtaisesti tilastot näkyviin sekä käyttöliittymään että Excel-tiedostoon. Missä järjestyksessä tilastojen otsikot ovat, on melko tärkeää. Sitä kautta pystyy nopeammin analysoimaan tilastoja ja ne ylipäätään käyvät paremmin järkeen. Ja nimenomaan, ettei siellä ole turhia tilastoja, mitä käyttäjä ei ole valinnut joukkueen luontivaiheessa. Tässä jouduin turvautumaan if- ja else-lauseisiin, mikä saattoi monistuneena tehdä koodista hieman kömpelöä.

Season stats-komponentti on siis lähes samanlainen toiminnoiltaan kuin tuo Played games. Tässäkin valitaan alasvetovalikosta oma joukkue, minkä jälkeen ruudulle tulee valitun joukkueen pelaajien kaikkien otteluiden tilastot yhteen laskettuina. Myös tämän näkymän tiedot saa jaettuna Excel-tiedostona. Isoin ero on siis siinä, mistä osasta tietokantaa data haetaan. Tämä hakee players-haarasta, kun taas Played games-komponentti hakee games-haarasta. Tämä tapa toimii, mutta ei siis ole ideaali. Ensimmäinen ongelmallinen asia, minkä huomasin tässä, oli se, kun käyttäjä poistaa ottelun. Tällaisessa tilanteessa kun ottelun poistaa, niin jouduin tekemään monimutkaisen funktion, mikä poistaa samalla players-haarasta jokaisen pelaajan kohdalta ottelun tilastojen verran arvoja sekä vähentää ottelumäärän yhdellä. Paljon helpompi olisi ollut tehdä molemmat komponentit niin, että tiedot haetaan games-haarasta. Kaikki poistotoimet sekä datan haku olisi ollut huomattavasti helpompaa ja yksinkertaisempaa. Toinen kehitysidea on se, että rakentaisi yhden näkymän, jossa kaikki nämä ottelut sekä koko kauden tilastot ovat saatavilla. Se olisi käyttäjäystävällisempää. Lisäksi mahdollinen koti- sekä vierasotteluiden tilastojen suodattaminen omiin listoihinsa olisi hieno lisä. Miksei myös se, että voisi olla valinta harjoitus-, runkosarja- ja pudotuspeliotteluille. Ja näitäkin pystyisi tässä tilastojen koonnissa hyödyntämään.



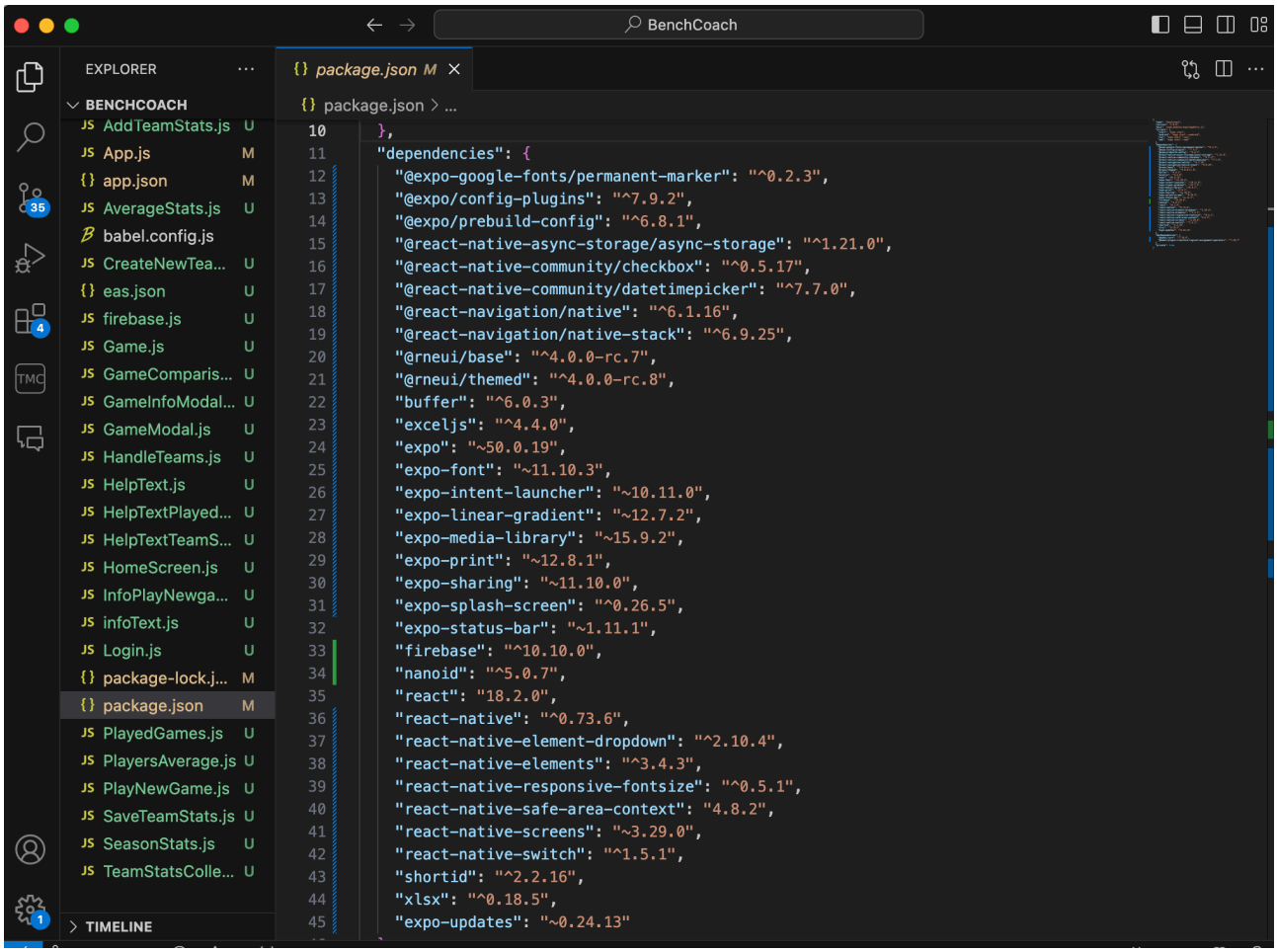

Kuva 12. Played games ja Season stats näkymät.

### 5.5.2 Joukkuetilastot

Vihoviimeinen ominaisuus, minkä halusin tämän opinnäytetyön puitteissa tehdä, oli oma osuus koko joukkueen tilastoille. Tällä tarkoitetaan yleensä joukkueen luomia maalipaikkoja otteluissa. Kokonaisuudessaan logiikka ja koodi ovat melko samanlaiset kuin muussa osassa sovellusta. Valitaan joukkue ja joukkueelle valitaan joukkuetta koskevat tilastoitavat asiat. Tämä tapahtuu, kun painetaan kotinäkymästä create team stats- nappia. Tämä näkymä toimii siten, että jälleen valitaan oma joukkue alasvetovalikosta. Tämän jälkeen esiin tulevat joukkueelle valittavana olevat tilastot. Tässäkin näkymässä on käyttäjää helpottamassa keltaiset ympyränmuotoiset napit. Suurennuslasi-ikoninen nappi näyttää, mitä tilastot tarkoittavat ja i-ikoninen nappi kertoo ohjeet, kuinka toimia. Tilastot esitetään samanlailla kuin pelaajien henkilökohtaiset tilastot eli checkboxeina. Kun käyttäjä on painanut halutut tilasto-checkboxit, tallennetaan ne tietokantaan painamalla save stats-nappia. Tilastoitavat asiat tallentuvat tietokantaan joukkueen alle omaksi teamStatistics-haaraksi. Tästä haarasta ne haetaan ja asetetaan halutun ottelun kohdalle nolla-arvoilla. Mikäli käyttäjä haluaa poistaa joukkueeltaan koko joukkueen tilastot ja mahdollisesti vaihtaa tilastoitavia asioita, voi hän painaa delete stats-nappia.

Kun halutaan tallentaa valittuja tilastoja, tulee siirtyä toiseen näkymään. Tässä kohtaa on täytynyt luoda jo ottelu henkilökohtaisia tilastoja varten, jotta haluttu ottelu löytyy listalta. Sen jälkeen tallennetaan halutut tilastot. Tilastot tallentuvat ottelun alle omaksi teamStatistics-osaksi. Tähän osioon pääsee, kun painaa kotinäkymässä save team stats-nappia. Alasvetovalikosta valitaan ensin joukkue ja tämän jälkeen ilmestyy ottelut sisältävä alasvetovalikko. Tästä valitaan haluttu ottelu. Tämän jälkeen painetaan esiin tulevasta create team statistics-nappia. Näin otteluun on luotu joukkuetilastot. Nyt käyttäjä voi tallentaa aiemmin koko joukkueelle valitsemiaan tilastoja, mitkä näkyvät nappeina ruudulla. Jokainen painallus lisää yhden arvon kyseiselle tilastolle. Viimeinen joukkuetilastoja koskeva asia on tilastojen koonti ja jakaminen. Tässä osassa on jo hieman sitä toimintamallia, mitä ajattelin henkilökohtaisiin tilastoihin. Eli sekä ottelukohtaiset että koko kauden tilastot löytyvät samasta kohtaa sovellusta. Lisäksi voi suodattaa koti- sekä vierasotteluiden tilastot omiksi kokonaisuuksiksi. Tässäkin kohtaa sovellusta ongelmia oli, kuinka renderöidä tieto käyttäjänkohtaisesti sekä sovellukseen että jaettavalle Excel-tiedostolle. Erilaisia sääntöjä erityyppisille tilastoille täytyi tehdä if- ja else-lauseiden muodossa, jotta jokainen käyttäjä saisi uniikin käyttäjäkokemuksen. Tähän osaan pääsee kotinäkymästä painamalla team stats-nappia. Alasvetovalikosta valitaan oma joukkue. Sen jälkeen käyttäjällä on mahdollisuus valita uudesta alasvetovalikosta yksittäisen ottelun joukkuetilastot, total statistics- nappia painamalla kaikkien otteluiden yhteenlasketut joukkuetilastot tai home/away statistics-nappia painamalla kaikkien koti- sekä vierasotteluiden yhteenlasketut joukkuetilastot. Kaikki nämä tilastot on mahdollista jakaa omina Excel-tiedostoinaan painamalla vihreitä nappeja. Kehityskohteena tähän osaan sovellusta on se, miten tieto näytetään sovelluksessa sekä jaettavassa tiedostossa. Tällä hetkellä se on vähän niin kuin lätkäisty siihen. Paremmalla rakenteella käyttäjäkokemus paranee ja tilastojen analysointi nopeutuu. Lisäksi, jos lisään erityyppisten otteluiden valintamahdollisuuden, täytyy voida suodattaa harjoitusottelun tilastot omaksi kokonaisuudeksi, runkosarjaottelut omaksi sekä pudotuspelit omaksi kokonaisuudeksi.


Kuva 13. Joukkuetilastojen näkymiä.




Kuva 14. Kuvakaappaus sovelluksessa käytetyistä kirjastoista

## 5.6 Julkaisu

Kun olin tyytyväinen sovelluksen ominaisuuksiin, oli aika paneutua julkaisuun. Itselleni tämä oli täysin uusi asia. Olin siis jo suunnitteluvaiheessa päättänyt, että julkaisu tulee tapahtumaan ainakin alkuun vain Googlen Play-kaupan kautta. Tätä varten on luotava kehittäjätili Googlelle. Kehittäjän tulee todentaa itsensä Googlelle. He vaativat tarkkoja tietoja henkilöstä, kuten henkilönumeron sekä kuvan passista. Tämän lisäksi täytyi jotenkin todentaa osoite. Lisäksi täytyy maksaa kertaluontoinen 25 dollarin maksu, minkä jälkeen voit alkaa julkaisemaan omia sovelluksia. Kun kehittäjä on todennettu, kirjaudutaan Google Play Consoleen, missä tuo julkaisu tapahtuu. Oma kehittäjänimeni on ViKo.

Julkaisuun tarvitaan Bundle file, jolla tarkoitetaan tiedostoa, johon on niputettu JavaScript-koodia yhdeksi tiedostoksi. Tiedosto sisältää kaiken tarvittavan JavaScript-koodin, mitä sovellus tarvitsee toimiakseen. Voidakseen luoda tiedoston, on tarvinnut ensin kirjautua omalle expo-tilille. Tämän tiedoston saa luotua komennolla:

```
villekorhonen@Ville-MBP BenchCoach % eas build --profile production
```

Kuva 15. Komentorivin kuvakaappaus komennosta luoda bundle file.

Tämän tiedoston lisäksi tarvitaan app icon ja banner. App icon on se pieni kuvake, joka tulee näkymään käyttäjälle oman laitteen ruudulla. Tätä painamalla sovellus käynnistyy. Banner on myös kuva ja se tulee Google Play-kauppaan markkinoimaan sovellusta. Näiden lisäksi tarvitaan kolme näyttökuvaa. Lyhyt sekä pidempi kuvaus sovelluksesta. Lisäksi tarvitaan vielä tietosuojaseloste, mikä takaa läpinäkyvyyttä, rakentaa käyttäjien luottamusta sekä informoi käyttäjiä oikeuksista. Nämä kannattaa haalia kasaan ennen kuin aloittaa julkaisuprosessin Google Play Consolessa. Näiden jälkeen Google Play Consolessa oli hyvät ohjeet kuinka edetään sovelluksen julkaisun kanssa. Googlella on liuta vaiheita, mitä tulee käydä läpi ennen kuin voi julkaista sovelluksen. Näistä kriittisin on testaus; sovellusta ei voi julkaista, ellei sitä ole testattu. Tämä koskee 13.11.2023 jälkeen luotuja käyttäjätilejä. Googlella on useampikin erilainen testivaihtoehto. Sisäinen testi on nopea tapa testata pienellä testiryhmällä sovelluksen toimivuutta. Tämä on vapaaehtoinen vaihe. Suljettu testi on siis pakollinen vaihe minulle uutena kehittäjänä. Tarvitsen vähintään kaksikymmentä testaajaa suorittamaan testin. Lisäksi on mahdollista suorittaa avoin testi. Tämä tarkoittaa sitä, että kehittäjä asettaa sovelluksensa Google Play-kauppaan, mistä kuka tahansa voi ladata testiversion ja testata sovellusta sekä antaa palautetta. Avoin testaus on käytettävissä, kun kehittäjällä on pääsylupa tuotantoon.

Tällä hetkellä menen sisäisen testauksen vaiheessa. Halusin testata alkuun vain muutamalla käyttäjällä sovellusta. Huomaan myös, että on melko suuri kynnys tuoda esille oma tekele. Jotenkin sitä miettii, että onko tämä tarpeeksi hyvä, että tämän voi näyttää koko maailmalle. Sisäisessä testissä on mukana Ilveksen U20-joukkueen valmentajat. Lisäksi Ilveksen U18-joukkueen tulisi aloittaa testaaminen pian. Olenkin jo saanut testaajilta parannusehdotuksia, joita olen alkanut päivittämään sovellukseen. Mutta ne eivät ole osa tätä opinnäytetyötä. Tavoite oli siis saada sovellus suljettuun testiin, mutta tässä olen hieman epäonnistunut ja olenkin edennyt siis hieman pidemmän kaavan kautta. Suljettu testi vaatii siis 20 testaajaa ja tämä saattaa osoittautua ongelmalliseksi.

# 6 Pohdinta

Sovelluksen lähtökohtana oli siis henkilökohtainen kokemus jääkiekkovalmentajan tekemästä tilastoinnista ja nimenomaan sen vanhanaikaisuudesta. Tavoitteena oli nopeuttaa prosessia, millä tilastot saadaan talteen, saadaan pidettyä tallessa sekä pystytään helposti esittämään muille. Näin niitä voidaan analysoida paremmin ja tehokkaammin. Samalla pystyisin kehittämään osaamistani ohjelmoinnin osalta sekä edesauttamaan työllistymistäni.

## 6.1 Sovelluksen onnistuminen

Miten tässä onnistuin? Mielestäni hyvin. Sovellus on toiminut testikäytössä, vaikka otanta on hyvin lyhyt. Nyt on mahdollista tallentaa ottelun aikana tilastot suoraan tietokantaan ilman välivaiheita kynän ja paperin kanssa. Nyt sovellus laskee yhteen koko kauden tilastot sekä yksittäisiä tilastoja otteluissa, kuten plusmiinus tai aloitusprosentti. Nämä kaikki oli aikaisemmin itse laskettava ja vaivalloisesti klikkailtava Excel-tiedostoon. Nappia painamalla on mahdollista saada tilastoja tulostusvalmiiksi ilman, että täytyy itse luoda taulukoita. Hieman tarvitsee vielä itse hienosäätää Excel-tiedostoa, mutta vain tyylien osalta.

Käyttäjäkokemuksen kannalta en ole vielä saanut palautetta, mutta oman kokeilun kautta voin todeta, että hieman selkeämmän kokonaisuuden olisi voinut vielä tehdä. Paljon on samankaltaisuutta eri tilastojen suhteen. Käyttäjä voi mennä sekaisin, kun on esimerkiksi mahdollista painaa samasta modalista Rush Chance- tai Rush Chance Against-nappia. Tällöin tulee varmasti virhepainalluksia. Tähän voisi mahdollisesti jaotella tietyt tilastonapit eri kohtiin tai erivärisiksi. Lisäksi kotinäkymä voisi olla jotenkin selkeämpi vielä, sillä käyttäjällä saattaa mennä sekaisin, mistä löytyy otteluiden henkilökohtaiset tilastot, koko kauden henkilökohtaiset tilastot sekä joukkueen tilastot. Lisäksi symboleilla toteutetut napit voivat olla hieman vaikeita ymmärtää. Itselle ne ovat tietysti kristallin kirkkaat, mutta uudella käyttäjällä voi olla haasteita niiden kanssa.

Tietokannan rakenteen suunnittelussa sekä onnistuin että epäonnistuin. Onnistuminen tulee siitä, että ylipäätään sain tämän tietokannan toimimaan. Data tallentuu ja sen saa esille sieltä. Se ei missään nimessä ollut itsestäänselvä juttu, sillä nyt oltiin ensimmäistä kertaa asialla. Epäonnistuminen tulee siitä, että tietokanta ei ole optimaalisesti suunniteltu ja siellä on mielestäni turhaa dataa. Lisäksi sitä on moneen kertaan muutettu jo luomisvaiheessa, mikä vei paljon aikaa työskentelyltä. Selkeämpi suunnitelma tietokannalle, vaikka tosin sekään ei välttämättä olisi auttanut. Ja joudun vielä todennäköisesti hienosäätämään tietokantaa, jotta se olisi järkevämpi ja toimisi paremmin. Tässä on vielä paljon kehityttävää omalla kohdalla.

### 6.2 Jatkokehitys

Sovellusta on mahdollista kehittää jatkossa ja raportissakin on mainittu jo monta kehityskohdetta. Jatkokehityksen kannalta olisi ollut parempi tehdä komponenteista hieman pienempiä, pilkkoa toiminnot hieman pienempiin osiin. Lisäksi komponenttien nimeämisessä olisi voinut käyttää paremmin järkeä, sillä nyt osa komponenteista on liian samannimisiä. Itse pysyn kärryillä, mistä mikäkin asia löytyy, mutta mahdollinen apuri, ei välttämättä löydä oikeaa komponenttia helposti. Juniorivalmentajilta tulikin siis jo toiveita sovellukselle. Erilaisia tarkempia tilastoitavia asioita oli toivelistalla, lisäksi mahdollisuus tallentaa hyökkäysketjun maalipaikat otteluista, laskea keskiarvoa kauden otteluista sekä verrata ottelun tilastoja koko kauden keskiarvoon. Tähän keskiarvojen vertailuun oli vielä toiveena värikoodeja, mitkä kertovat kuinka pelaaja on suoriutunut ottelussa verrattuna aikaisempiin otteluihin. Itse ideoin erilaisen ottelun, sillä kaudella pelataan yleensä harjoitusottelu, runkosarjaottelu sekä pudotuspeliottelu. Tästä mainitsinkin jo aiemmin. Haluaisin kehitellä koko kauden tilastoihin mahdollisuuden, millä voisi laskea haluamiensa otteluiden tilastot yhteen. Monesti valmentajat haluavat pilkkoa kauden pienempiin osiin, kuten viiden ottelun jaksoihin. Sellainen otteluraportti olisi hieno, mikä näyttäisi kaikki yhdessä valitussa ottelussa tapahtuneet asiat yhdellä A4-kokoisella sivulla, kuten henkilökohtaiset tilastot, ketjutilastot, joukkueen tilastot ja mahdolliset vertailut keskiarvoon. Yksi asia on ehdottomasti sellainen ominaisuus, että henkilökohtaisia tilastoja voisi käydä merkkaamassa myöhemminkin. Eli ottelunäkymään pääsisi palaamaan uudestaan. Nyt se ei ole vielä mahdollista, vaan henkilökohtaiset tilastot tulee kirjata yhdeltä istumalta. Tällä hetkellä ei ole myöskään mahdollista tehdä tilastointia maalivahtien osalta sovelluksessani, joten olisiko tähän mahdollista yhdistää tämä ominaisuus.

### 6.3 Tekoäly apurina

Tekoäly on ollut viimeisen reilun vuoden suuri puheenaihe niin mediassa kuin koulussakin. Sen käyttö on lisääntynyt valtavasti. Koulussa olen huomannut, että melkein kaikki sitä käyttävät ja nyt myös monet työpaikkailmoitukset kysyvät, kuinka onnistuu tekoälyn hyödyntäminen työskentelyssä. Tekoäly toimi myös minun apunani tässä laajassa projektissa koodauksen osalta sekä ongelmien ratkaisussa. Niin kuin totesin jo aikaisemmin, käytän GitHub:n copilottia koodatessani Visual Studio Codessa. Tämä nopeutti hieman sellaisia komponentteja, missä oli paljon toisteisuutta. En ainakaan vielä osaa sanoa, onko tuo muuten mullistava työkalu. Chat gpt:tä käytin myös koodin rakentamisessa. Tämän koen olevan parempi apuväline. Rehellisesti sanottuna en usko, että näin vaativaa sovellusta olisin saanut tehtyä yksin tällä aikataululla ilman tätä tekoälyä. Luinkin tuossa yhden uutisen, missä hieman kritisoitiin tekoälyä, koska se estää opiskelijaa oppimasta. Olen hieman samaa mieltä, on helppo kääntyä chat gpt:n puoleen ja saada vastauksia. Tämä laiskistaa ja saattaa estää asioiden sisäistämistä. Mutta toisaalta siltä voi myös saada oppia, parhaassa

tapauksessa se oli itselleni kuin opiskelutoveri, keneltä sai vinkkejä ongelmakohdissa. Tietty kriittisyys tulee tosin pitää mielessä silloin kun käyttää tekoälyä. Oli monta ongelmaa, mihin chat gpt tarjosi ihan puuta heinää. Näissä ongelmissa lopullinen ratkaisu lähti kuitenkin itseltäni ja sitä kautta ongelmat ratkesivat. Eikä tekoäly pysty tekemään kaikkea, ei ummikko pysty monimutkaista sovellusta rakentamaan, mikäli hänellä ei ole mitään hajua ohjelmoinnista käyttäen vain Chat gpt:tä. Uskon kuitenkin, että tekoäly on tullut jäädäkseen ja tulevaisuudessa sitä käytetään yhä enemmän ohjelmoinnissa. Ihmisen se tarvitsee tekemään luovat päätökset.

### 6.4 Sovelluksen hyödyllisyys

Katson tätä itse värilasien läpi, mutta mikäli itselläni olisi ollut tällainen sovellus omalla valmennusuralla käytössä kahden edellisen vuoden aikana, olisin ehdottomasti käyttänyt. Tämän avulla olisin säästänyt aikaani ja mahdollisesti voinut auttaa nuoria enemmän. Tässä vaiheessa testiryhmä ei ole vielä käyttänyt sovellusta riittävän paljon, jotta he voisivat sanoa hyödyllisyydestä. Ainakin valmentajien innostuneisuus sovellusta esitellessä on ollut suurta. Saattaa olla tosin, että innostuneisuus on ollut enemmän ihmettelyä, että voiko olla tosiaan niin, että minä olen tehnyt sen. Nyt kun olen paneutunut tilastointiin jääkiekon puolelta, on itselleni selvinnyt, että on joitakin palveluita, minkä kautta on mahdollista saada tilastoja. Yksi on otteluvideoiden pilkkomiseen käytettävä palvelu, joka laskee samalla joitakin tilastoja. Toinen on sivusto, joka pilkkoo pelaajien vaihtoja otteluista ja tilastoi kentällä tapahtuvia asioita. Näin ollen oman sovellukseni hyödyllisyys saattaa olla hieman pienempi kuin olin ajatellut ja sopivia käyttäjiä vähemmän.

### 6.5 Opinnäytetyön onnistumien

Olen ehdottomasti tyytyväinen opinnäytetyöhöni ainakin tässä vaiheessa, kun arviota ei ole vielä tullut. Luulen, että olen tyytyväinen siinäkin vaiheessa, kun arvio tulee, mikäli työ on hyväksytty. Muuten, kun miettii tätä projektia, niin lopputulos sovelluksen osalta on toimiva. Kaikki ne ominaisuudet, joita lähdin hakemaan, toimivat. Raportti on mielestäni itseni näköinen. Aikataulussa en pysynyt, sillä tämän tuli olla valmis elokuun 2024 alussa. Tässä jäin jälkeen, sillä nyt syyskuussa 2024 viimeistelen työtäni. Suurin syy on siinä, että heinäkuu oli perheellä lomaa ja oli hyvin vaikea keksiä syy, miksi en olisi perheeni kanssa lomaillut vaan tehnyt tätä työtä. Muuten aikatauluttaminen onnistui hyvin ja olin todella kurinalainen sekä ahkera. Olisin voinut suunnitella työnjaon hieman tasaisemmin, nyt tein paljon enemmän itse sovellusta kuin tätä raporttia. Viikon olisi voinut jakaa vaikka siten, että kolme tai neljä päivää koodaamista ja yksi tai kaksi päivää raportin tekoa. Viikonloput pyrin pitämään vapaana, sillä henkinen jaksaminen on erittäin tärkeää. Hieman tuli varoittelua ohjaajaltani, että malttaisin rajata työni selkeästi sekä pysyisin tässä päätöksessä. Tässä hieman epäonnistuin, sillä vaikka rajasin mielestäni selkeästi ison kuvan, niin monessa kohtaa tein vielä hieman pidemmälle ja enemmän kuin olin suunnitellut.

Se lähtötaso, mistä lähdin tähän opinnäytetyöhön, oli aika paljon matalampi kuin nyt. Mielestäni olen mennyt mobiiliohjelmoinnissa eteenpäin ja oppinut paljon uusia asioita. Etenkin Firebase on nyt paljon tutumpi. Erilaiset kirjastot React Nativella ovat myös laajemmin tiedossani sekä osaan enemmän erilasia tapoja toteuttaa sovelluksen toimintoja. Ohjelmointitaitoni ovat ehdottomasti menneet eteenpäin sekä ymmärrys mobiiliohjelmoinnista avartunut. Ymmärrän paremmin sovelluksen luomisesta sekä sen jatkuvasta kehittämisestä.

## Lähteet

Bautomo. Bautomon sanastoa. Luettavissa: https://bautomo.com/sanasto/ Luettu 17.5.2024

Expo. Luettavissa: https://expo.dev Luettu 30.7.2024

Expo Docs. Reference. Luettavissa: https://docs.expo.dev/versions/latest/ Luettu 30.7.2024

Firebase. Firebase authentication. Luettavissa: https://firebase.google.com/docs/auth Luettu 7.8.2024

Firebase database. Firebase Realtime database. Luettavissa: https://firebase.google.com/docs/database Luettu 7.8.2024

Fraktio blogi. React Native: kaikki mitä olet aina halunnut kysyä. Luettavissa: https://www.fraktio.fi/blogi/react-native-kaikki-mita-olet-aina-halunnut-kysya Luettu 25.9.2024

Geeksforgeeks, 2024. What is database? Luettavissa: https://www.geeksforgeeks.org/what-is-database/ Luettu 25.9.2024

Geeksforgeeks cloud database, 2024. How to design a cloud based database? Luettavissa: https://www.geeksforgeeks.org/how-to-design-a-cloud-based-database/ Luettu 25.9.2024

GitHub. What is a programming language? Luettavissa: https://github.com/resources/articles/software-development/what-is-a-programming-language Luettu 15.11.2024

Harju, J. 2013. Android-ohjelmoinnin perusteet. Books of Demand GmbH. Helsinki.

IBM. What is a relational database? Luettavissa: https://www.ibm.com/topics/relational-databases Luettu 25.9.2024

Kotimikro, 2022. Mikä Android on? Kaikki matkapuhelinten ja tablettien käyttöjärjestelmästä. Luettavissa: https://kotimikro.fi/oheislaitteet/kayttojarjestelma/android/mika-android-on-kaikki-matkapuhelinten-ja-tablettien-kayttojarjestelmasta Luettu 17.5.2024

Lido. What is Google Firebase? Everything you need to know in 2024. Luettavissa: https://www.lido.app/firebase/what-is-google-firebase Luettu 5.8.2024

Lifewire, 2022. What is a mobile device? Luettavissa: https://www.lifewire.com/what-is-a-mobile-device-2373355. Luettu 16.5.2024.

Mikkonen, T. 2004. Mobiiliohjelmointi. Talentum. Helsinki.

Mutsimedia, 2023. Lasten kellopuhelimet testissä – mikä laite kannattaa hankkia? Luettavissa: https://mutsimedia.fi/lapsen-kanssa/lasten-kellopuhelimet-testissa-mika-laite-kannattaa-hankkia/. Luettu 17.5.2024.

Netguru. What is React Native? Complex guide for 2024. Luettavissa: https://www.net-guru.com/glossary/react-native#what-is-react-native Luettu 30.7.2024

Nodejs. Introduction to Node.js. Luettavissa: https://nodejs.org/en/learn/getting-started/introduc-tion-to-nodejs Luettu 6.8.2024

Oracle. Mitä tietokanta tarkoittaa? Luettavissa: https://www.oracle.com/fi/database/what-is-data-base/ Luettu 18.6.2024

Salz, P. A. & Moranz, J. 2013. The everything. Guide to mobile apps. Adams media. Avon, Mas-sachusetts, USA.

Tecinspire, 2022. Miten valitsen uuden sovelluksen teknologian? Luettavissa: https://tecinspire.com/miten-valitsen-uuden-sovelluksen-teknologian/ Luettu 18.6.2024

Techopedia, High-Level Language, 2024. High-Level Language (HLL) Luettavissa: https://www.techopedia.com/definition/3925/high-level-language-hll Luettu 25.9.2024

Techopedia, 2016. Development Enviroment. Luettavissa: https://www.techopedia.com/defini-tion/16376/development-environment Luettu 30.7.2024

Techtarget, 2024. Definition – Apple iOS. Luettavissa: https://www.techtarget.com/searchmobi-lecomputing/definition/iOS Luettu 29.7.2024

Tieturi. Ohjelmistokielet. Luettavissa: https://www.tieturi.fi/koulutusala/ohjelmistokehitys/ohjelmoin-tikielet/ Luettu 30.7.2024

Timehouse, 2022. Mitä sinun tulee tietää UX- ja UI-suunnittelusta kehittäessäsi verkkosivustoa. Lu-ettavissa: https://www.timehouse.fi/ux-ui-suunnittelu/ Luettu 7.8.2024

# Liitteet

## Liite 1. BenchCoach ohjeet

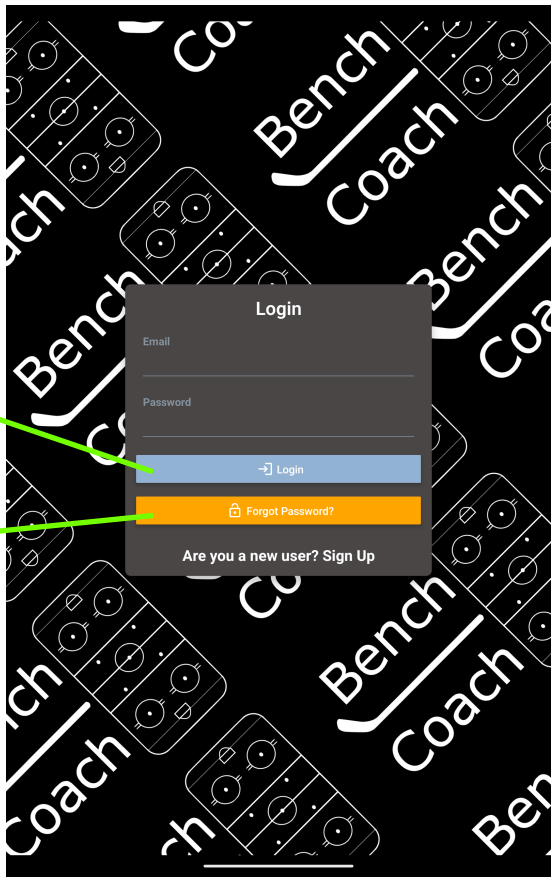
# BenchCoach ohjeet

Aloitusnäkymä, jossa luodaan käyttäjätili. Kirjoita sähköpostiosoitteesi ja salasana. Tämän jälkeen paina sign up-nappia. Ohjelma lähettää antamaasi sähköpostiin viestin, jossa on varmennuslinkki, jota tulee käydä painamassa. Tämän jälkeen menee muutamia minuutteja, että syöttämäsi tiedot toimivat ja voit kirjautua sisään.

Kun olet luonut tunnukset ja painanut sähköpostiin tullutta linkkiä tai omaat jo tunnukset, vaihda näkymä sisäänkirjautumiseen painamalla Login -tekstiä.


Kun olet jo varmentanut sähköpostisi, syötä sähköpostisi ja salasanasi niille tarkoitettuihin kenttiin ja paina login-nappia. Sinun tulisi päästä seuraavaan tilaan, jossa on ”Welcome (käyttäjän email)” tervehdys.
”Start coaching”-nappia painamalla pääset varsinaiselle kotisivulle ja ”logout”-nappia painamalla pääset kirjautumaan ulos sovelluksesta.

Jos olet unohtanut salasanasi, Forgot password-nappia painamalla voit asettaa uuden salasanan. Kirjoita sähköpostiosoitteesi ja ohjelma lähettää antamaasi sähköpostiin viestin, jossa voit asettaa uuden salasanan.



Luo uusi joukkue ja lisää pelaajat sekä tilastoitavat henkilökohtaiset tilastot. Tämä kohta tulee tehdä ensin, jotta sovellusta voi käyttää.

Joukkueen hallinta. Tästä pääset lisäämään ja poistamaan joukkueesi pelaajia. Lisäksi voit poistaa oman joukkueesi.

Ottelun luonti. Täällä valitaan ottelun kokoonpano, liisätään vastustaja, päivämäärä sekä koti/vieras. Myöhemmässä vaiheessa pääset myös tallentamaan henkilökohtaisia tilastoja sekä seuraamaan ottelun aikana tilastoja.

Jokaisen ottelun henkilökohtaiset tilastot löytyvät täältä.

Koko kauden henkilökohtaiset tilastot löytyvät täältä.

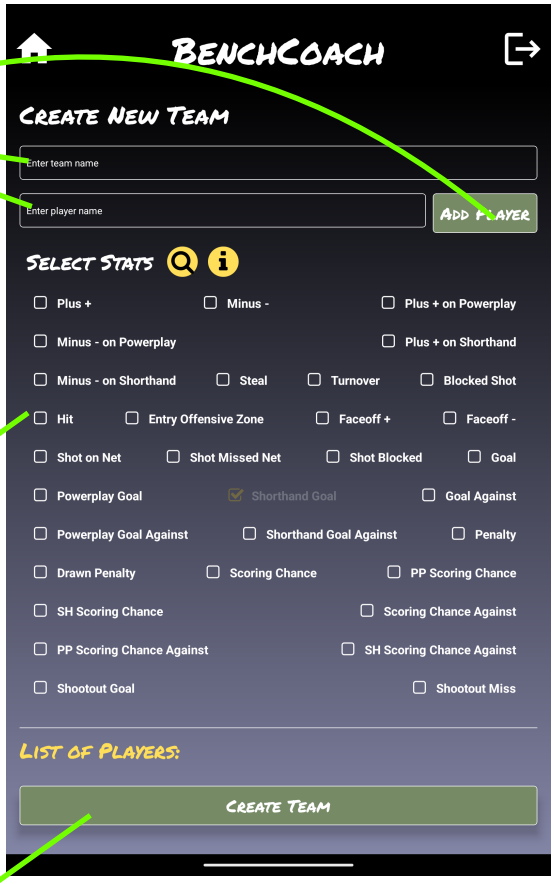

Täällä valitaan tilastot, joita halutaan tallentaa **koko joukkueelle,** kuten maalipaikat. Tämän voi tehdä vasta, kun on luonut joukkueen ”Create new team”-osiossa.

Tästä pääsee tutkimaan joukkueen tilastoja otteluista sekä koko kauden osalta. Voit myös tutkia pelkästään koti- tai vieras-otteluiden tilastoja.

Täällä tallennat joukkueelle valittuja tilastoja, mutta sinun täytyy luoda ottelu ensin ”Create new game”-osiossa sekä valita tilastoitavat asiat ”Create team stats”-osiossa.

# Create new team

Aloita kirjoittamalla joukkueesi nimi. Sitten kirjoita jokaisen joukkueesi pelaajan nimi ja paina ”Add player”-nappia. Tämä tehdään yksi pelaaja kerrallaan. Pelaajan nimi ilmestyy ruudun alareunaan ”List of Players"-kohdan alle. Listalla pelaajan nimen vieressä on roskakori, jota painamalla pelaajan voi poistaa ennen kuin joukkue tallennetaan tietokantaan.




Tilastoitavat henkilökohtaiset tilastot valitaan klikkaamalla haluttua tilastoa. Joukkueen kaikille pelaajille luodaan samat henkilökohtaiset tilastot.

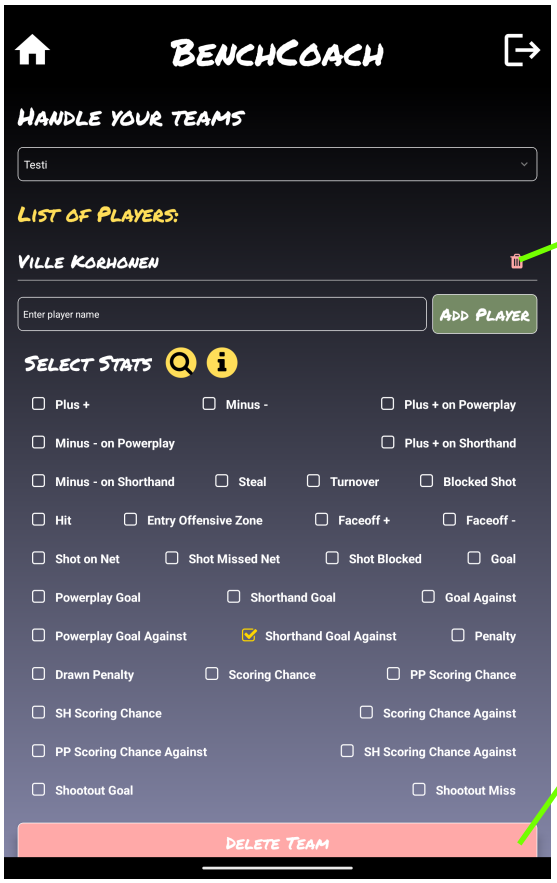
Viimeisenä asiana painetaan ”Create team”-nappia. Tämä tallentaa joukkueen, pelaajat sekä pelaajille tallennettavat henkilökohtaiset tilastot tietokantaan.

# Handle your teams

Valitse joukkueesi alasvetovalikosta ja joukkueen pelaajat ilmestyvät ”List of players”-kohdan alle.

Kirjoita lisättävän pelaajan nimi sekä valitse henkilökohtaiset tilastot, jotka haluat pelaajalle tallentaa.

Pelaaja tallentuu joukkueeseen painamalla ”Add player”-nappia.

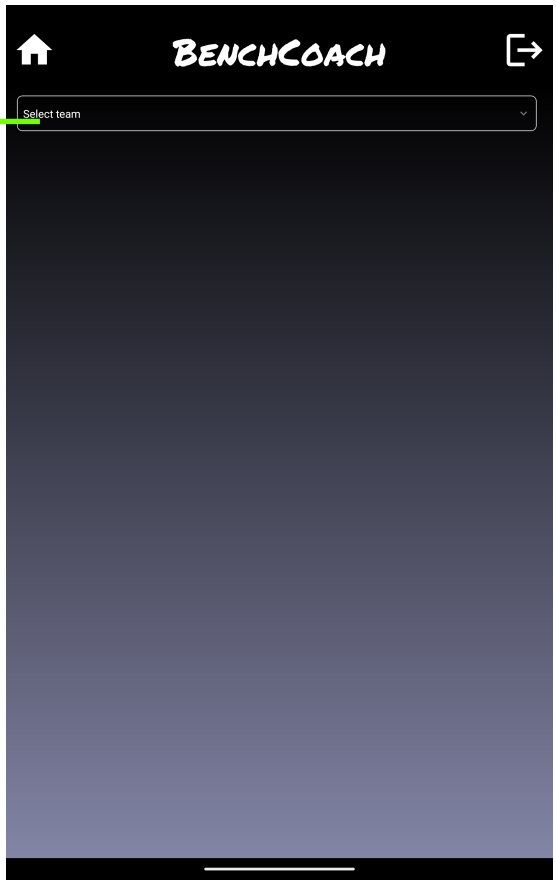




Jos haluat poistaa yksittäisen pelaajan joukkueesta, voit tehdä sen painamalla roskakoria. Tällöin pelaaja poistuu tietokannasta ja kaikki kyseiseltä pelaajalta tallennettu tieto häviää.

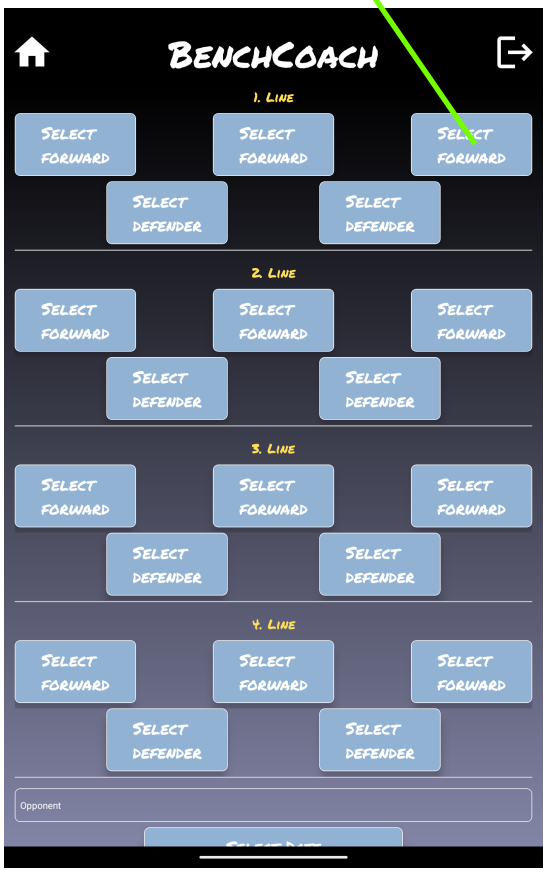
Jos jostakin syystä haluat poistaa koko joukkueen, paina ”Delete team”-nappia. Tällöin kaikki tieto joukkueesta häviää tietokannasta.

# Create new game

Valitse joukkueesi alasvetovalikosta ja pääset valitsemaan kokoonpanon otteluun.



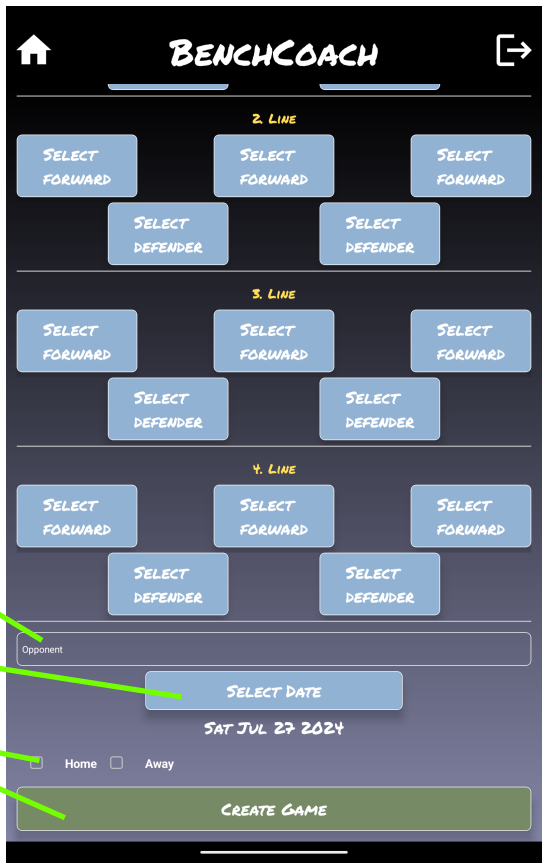

Painamalla pelipaikkaa, aukeaa lista pelaajista ruudulle ja painamalla halutun pelaajan nimeä asetat pelaajan kokoonpanoon kyseiselle pelipaikalle.

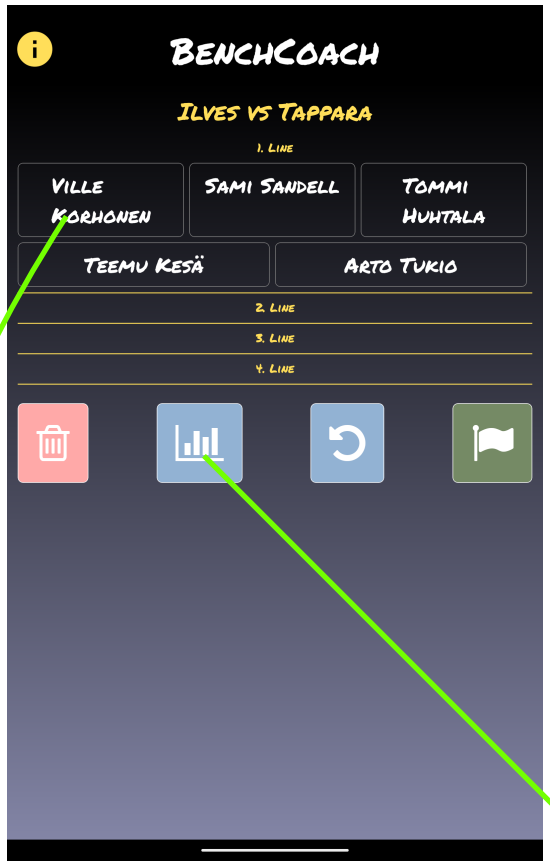



Kirjoita vastustajan nimi

Valitse päivämäärä

Valitse koti- tai vierasottelu ja paina ”Create game”-nappia.



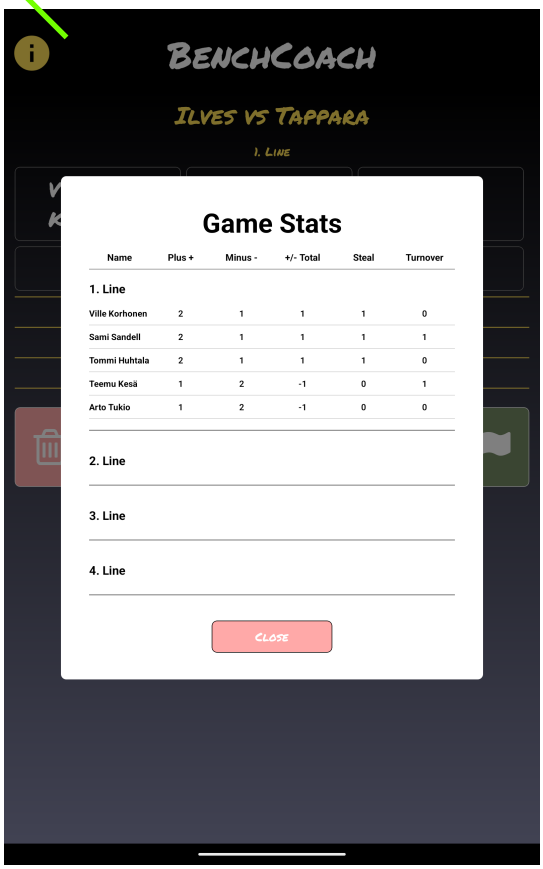



Roskakori-nappi poistaa ottelun, mikäli käyttäjä ei sitä halua tallentaa.

Mikäli tilastojen tallentamisessa tapahtuu virhe, voi tätä nappia painamalla poistaa viimeisimmän lisäyksen pelaajan tilastoihin.

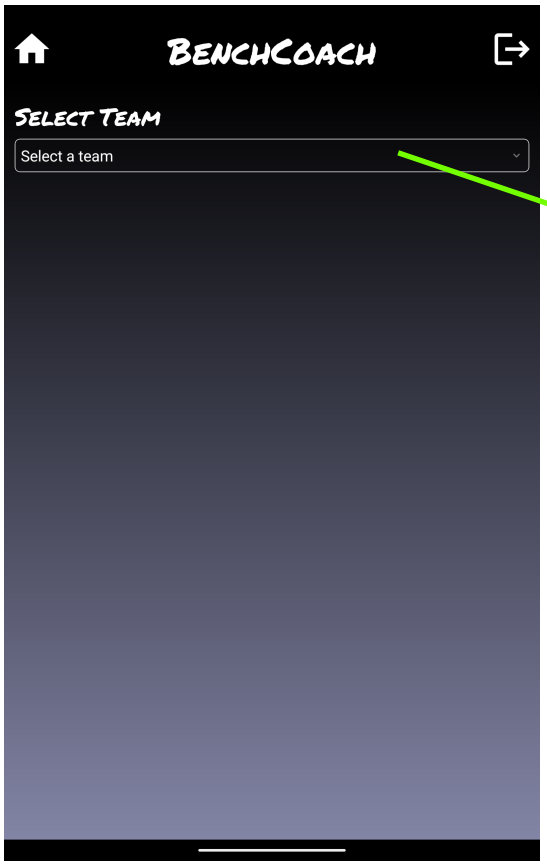
Tätä lippua painettaessa ottelu päätetään ja tilastot tallennetaan tietokantaan niin ottelun kohdalle kuin koko kauden tilastoihin.


Kun painat kokoonpanossa olevaa pelaajaa, aukeaa pieni ikkuna, josta löytyvät henkilökohtaiset tilastot, jotka käyttäjä on valinnut ”Create new team”-osiossa. Painamalla haluttua tilastoa lisätään se ottelun tilastoihin.

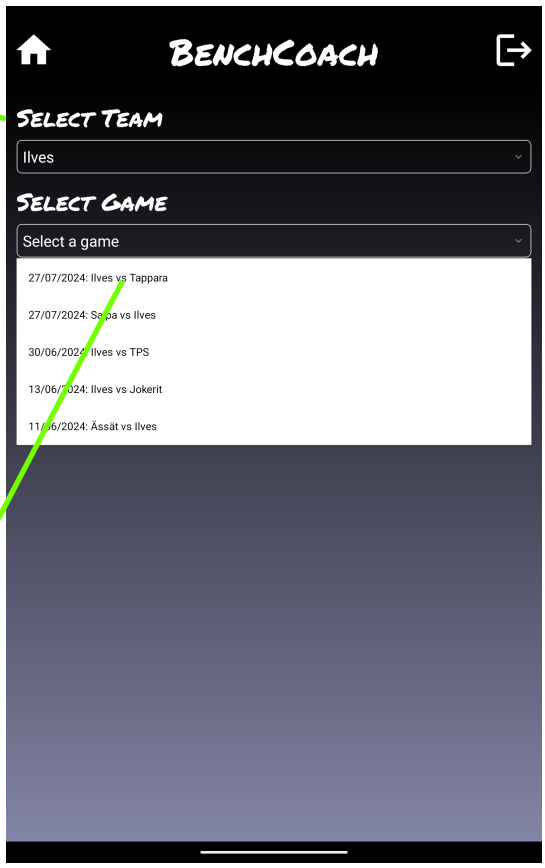


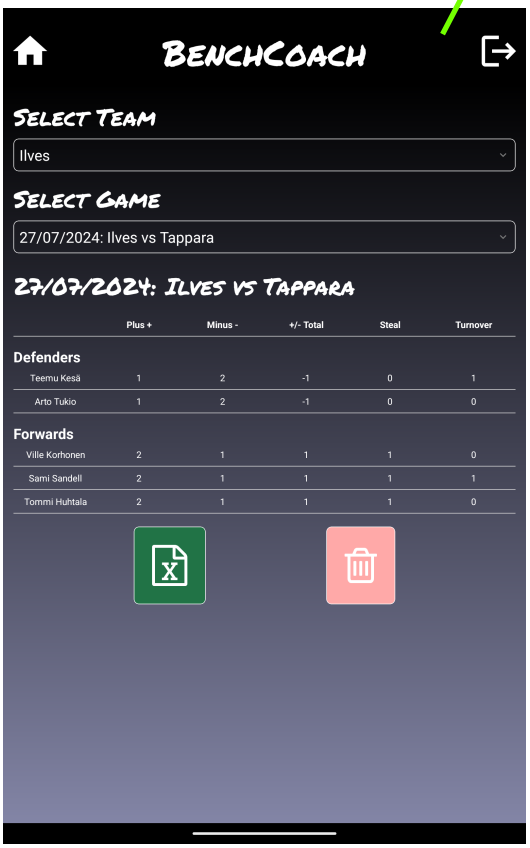

Painamalla kaavion kuvaa aukeaa ottelussa tallennetut tilastot näkyviin. Näitä voi tarkastella pelin aikana.

# Played Games




Valitse joukkue alasvetovalikosta, minkä jälkeen aukeaa kyseisen joukkueen ottelut toisesta alasvetovalikosta.



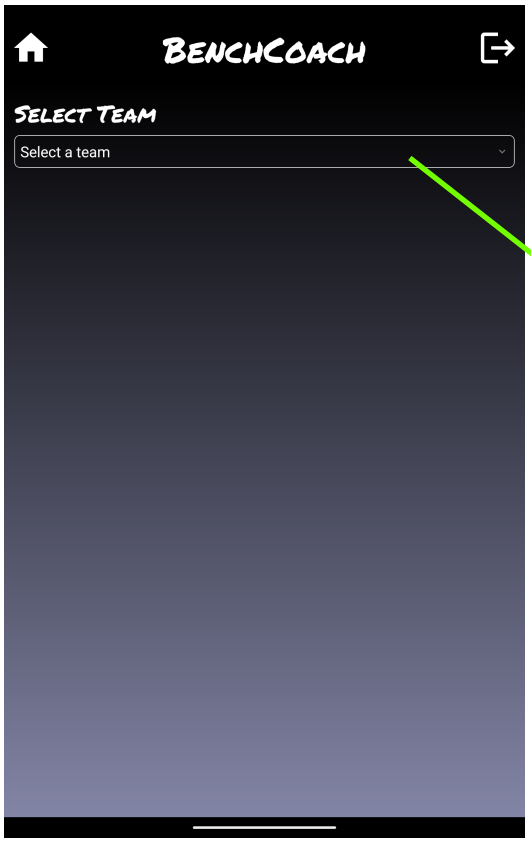




Painamalla haluttua ottelua aukeaa ottelussa tallennetut tilastot tarkasteltavaksi.

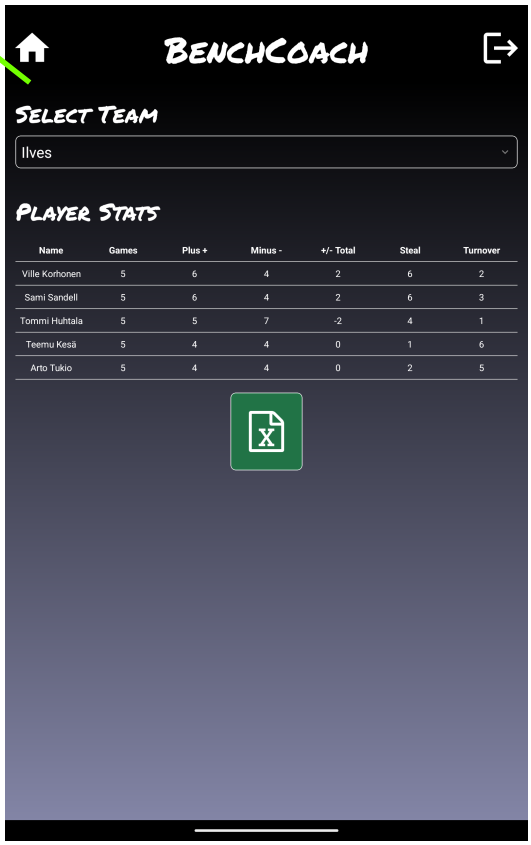
Tämä nappi luo excel-tiedoston ottelusta ja voit jakaa tiedoston.

Roskakori-nappi poistaa ottelun tietokannasta.

# Season stats

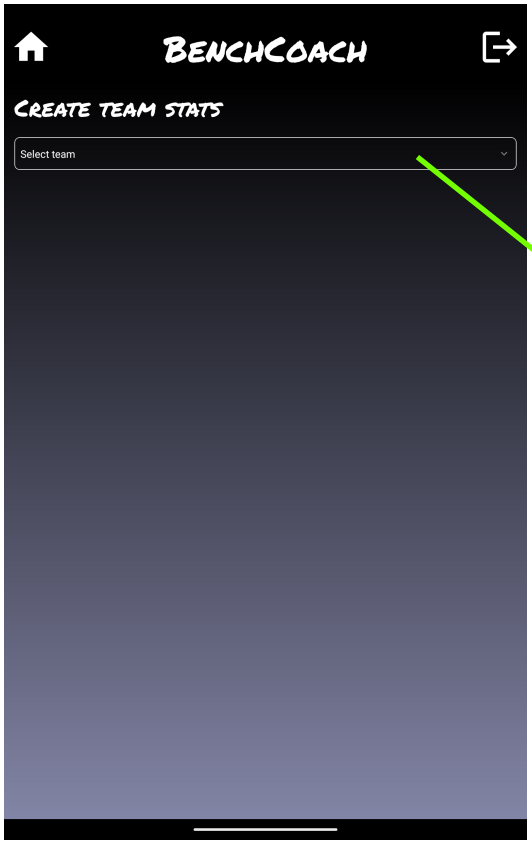



Valitse joukkue ja oman joukkueesi koko kauden pelattujen otteluiden henkilökohtaiset tilastot aukeavat esiin.

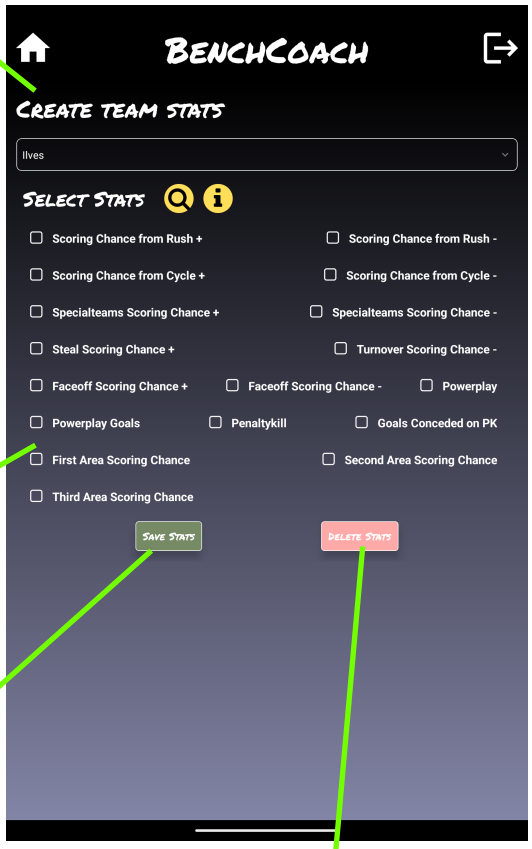



Tämä nappi luo excel-tiedoston koko kauden henkilökohtaisista tilastoista ja voit jakaa tiedoston.

# Create team stats




Valitse joukkue, jolle haluat asettaa joukkueen tilastoja tallennettavaksi. Muista luoda joukkue ensin ”Create new team”-osiossa.

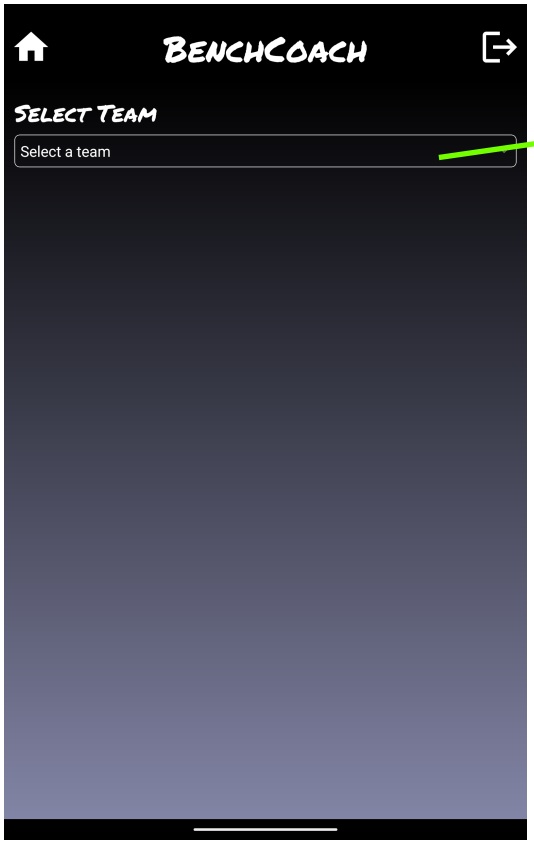



Valitse tallennettavat tilastot klikkaamalla tilastoa.

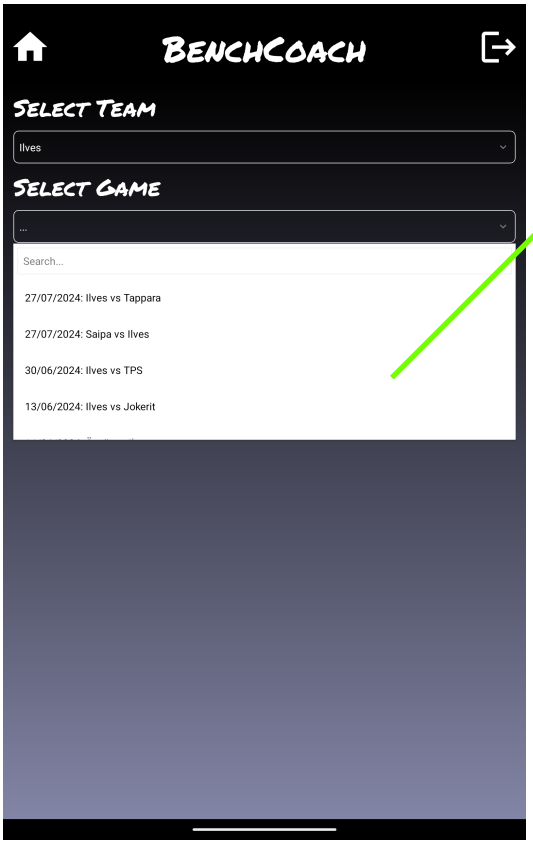
”Save stats”-nappia painettaessa tallennetaan tilastot joukkueen kohdalle tietokantaan ja niitä voidaan alkaa tallentaa toisessa osiossa sovellusta.

Jos käy niin, että tilastoja ei haluta tallentaa tai halutaan muuttaa jo valittuja tilastoja, tällä napilla joukkuetilastot voidaan poistaa.

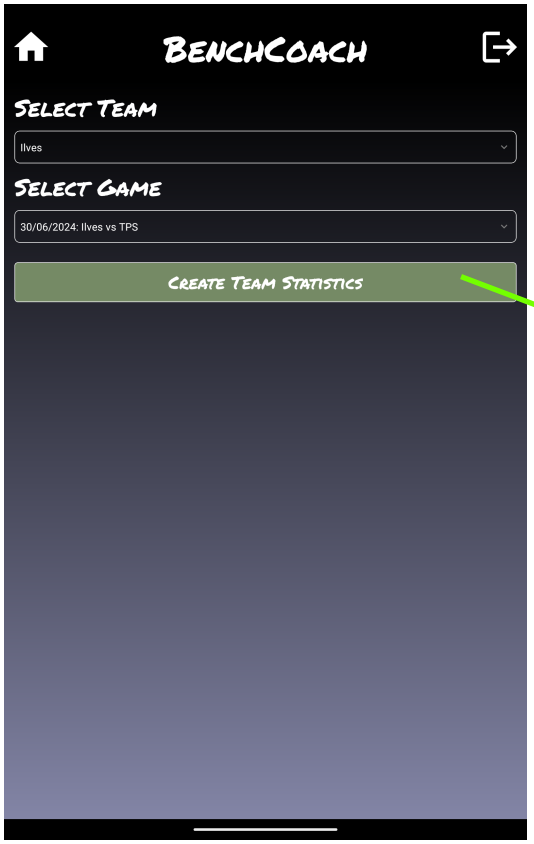
# Save team stats



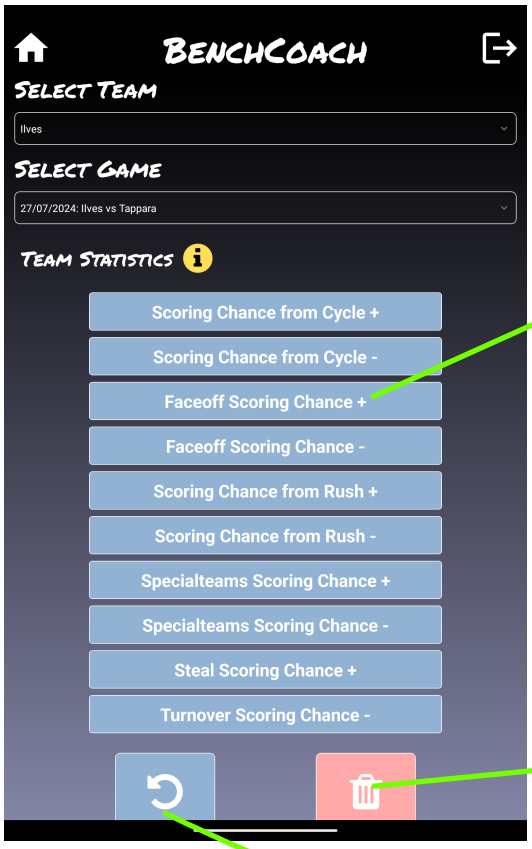

Valitse joukkueesi alasvetovalikosta.




Valitse ottelu, jonka tilastoja joukkueen osalta haluat tallentaa.




Lopuksi paina ”Create team statistics”-nappia, joka luo haluttuun otteluun joukkueen tilastot.



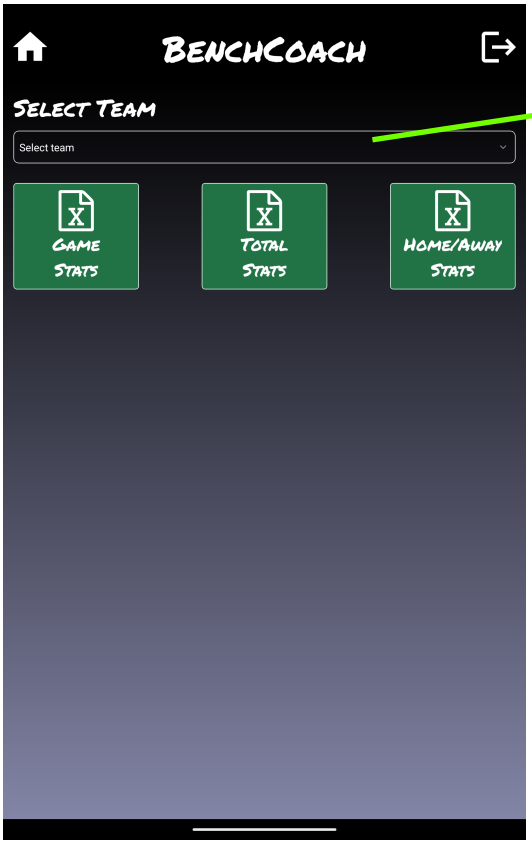

Kun ”Create team statistics”-nappia on painettu, aukeaa ruudulle aikaisemmin joukkueelle valitut tilastot. Painamalla nappia valittu tilasto tallentuu tietokantaan.

Virheellisen painalluksen pystyy perumaan tätä nappia painamalla.

Tällä roskakori-napilla, voi poistaa ottelusta tehdyt joukkueen tilastot.

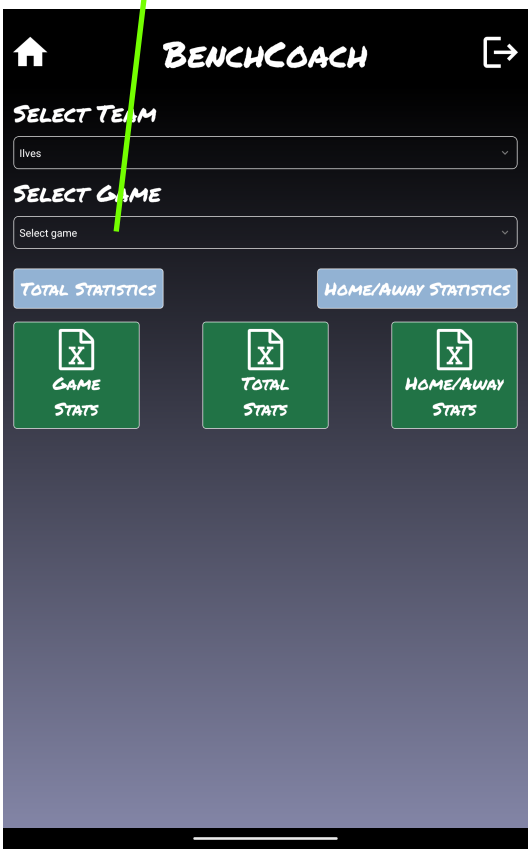
Huom!
Näitä joukkueen tilastoja pääsee täydentämään uudestaan, vaikka poistuisit tästä osioista tai koko sovelluksesta.

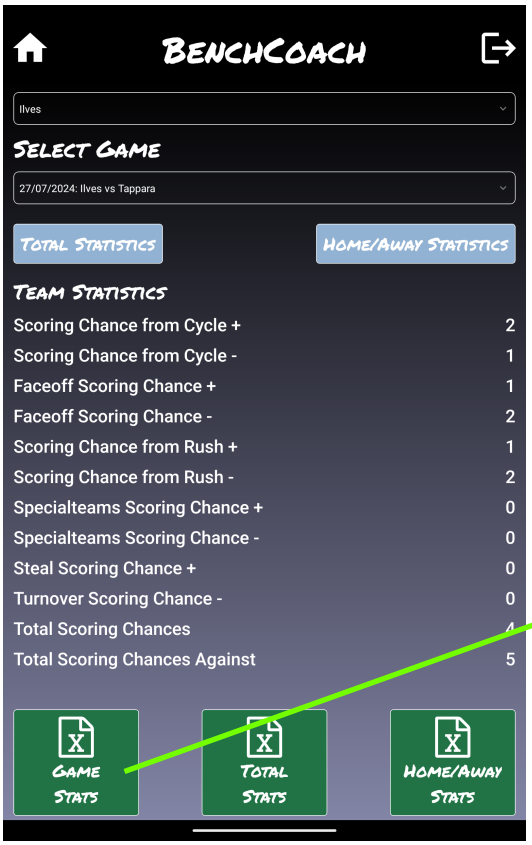
# Team Stats




Valitse joukkueesi alasvetovalikosta.

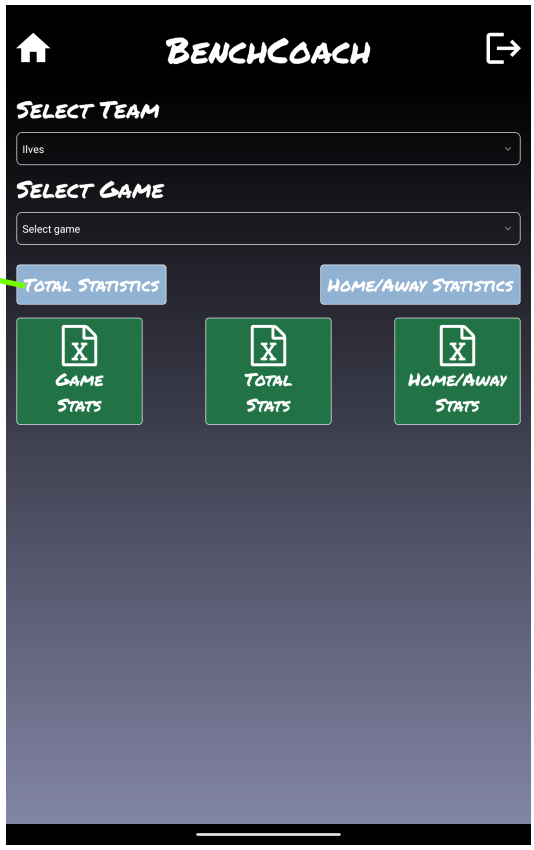
Valitse ottelu, jonka tilastoja haluat tarkastella.

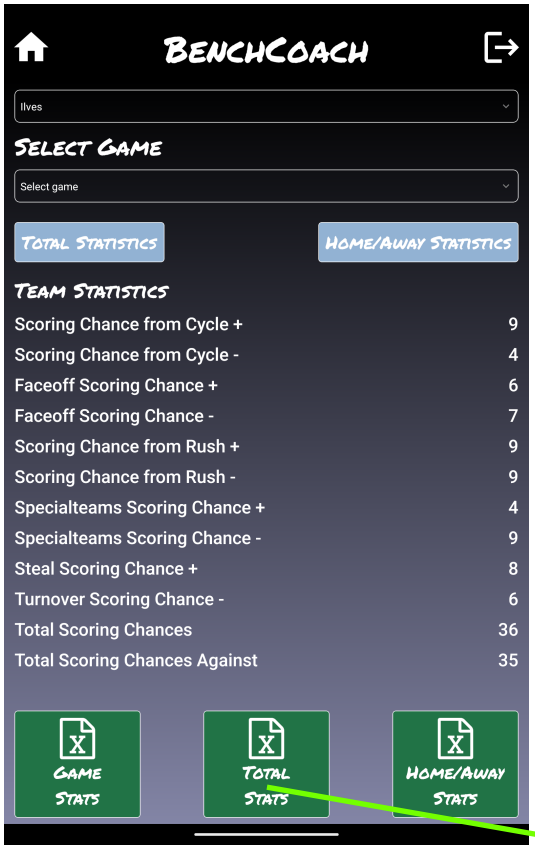


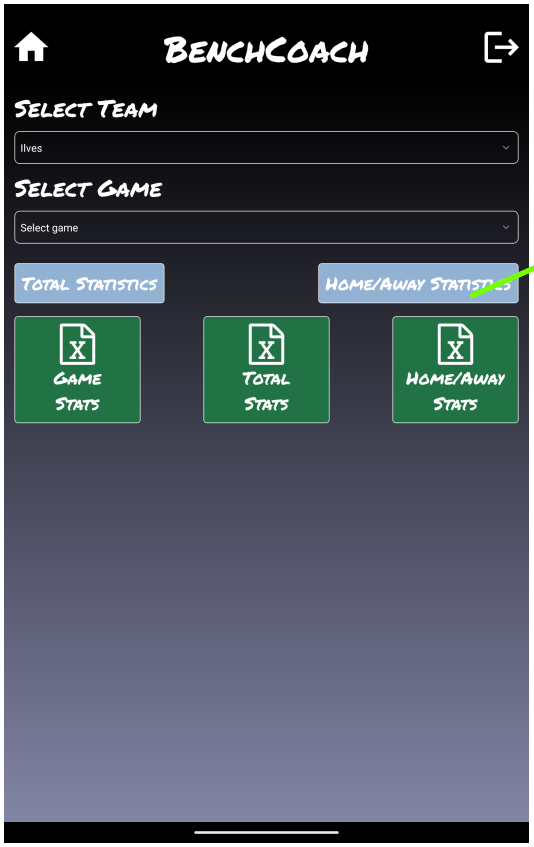

Kun olet painanut halutun ottelun tilastot näkyviin, voit jakaa ottelun tilastot excel-tiedostona painamalla ”Game Stats”-nappia.

”Total Statistics”-nappia painamalla sovellus näyttää kaikkien otteluiden tilastot, jotka käyttäjä on määritellyt omalle joukkueelleen.

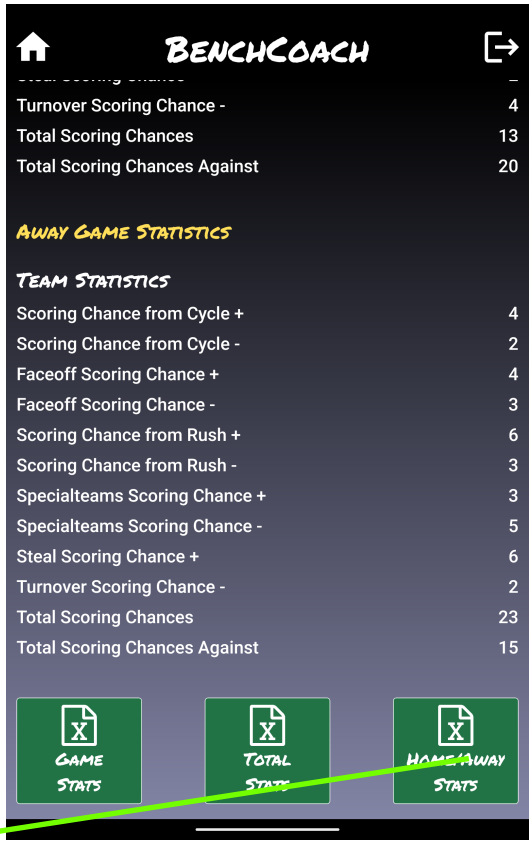



Kun olet painanut koko kauden tilastot näkyviin, voit ”Total Stats”-nappia painamalla jakaa koko kauden joukkueen tilastot excel-tiedostona.




”Home/Away Statistics”-nappia painamalla sovellus näyttää joukkueen tilastot koti-ja vierasotteluiden osalta.




Kun olet painanut koti- sekä vierasotteluiden tilastot näkyviin, voit jakaa ne excel-tiedostona painamalla ”Home/Away Stats”-nappia.