

Jere Hagman

TIETOVISAPELI ANDROIDILLE

TIETOVISAPELI ANDROIDILLE

Jere Hagman
Opinnäytetyö
Syksy 2020
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

Tekijä: Jere Hagman
Opinnäytetyön nimi suomeksi: Tietovisapeli Androidille
Opinnäytetyön nimi englanniksi: Quiz Game for Android
Työn ohjaaja: Eino Niemi
Työn valmistumislukukausi ja -vuosi: Syksy 2020
Sivumäärä: 36

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa tietovisapeli Android-käyttöjärjestelmälle Java-ohjelmointikielellä. Työn tavoitteena oli perehtyä syvemmin Full Stack -mobiilikehitykseen sekä Amazonin AWS-pilvipalvelu-alustan tarjontaan ja soveltaa näitä toteutuksessa.

Työ aloitettiin perehtymällä mahdollisiin toteutuksen osassa käytettäviin työkaluihin ja teknologioihin. Työn ensimmäisessä osassa esitellään valitut työkalut ja teknologiat sekä kerrotaan, kuinka niitä käytettiin tämän opinnäytetyön toteutuksessa. Toisessa osassa esitellään työn toteutus ja käydään läpi sovelluksen rakennetta ja kooditason ratkaisuja.

Opinnäytetyön tuloksena saatiin toimiva prototyypisovellus, josta olisi kohtuullisella jatkokehityksellä mahdollista saada Google Play -kauppaan julkaistava peli. Työn avulla tekijä sai myös paremman ymmärryksen Full Stack -mobiilikehityksestä ja pilvipalveluista.

Asiasanat: Android, AWS, mobiilipeli, mobiilikehitys, ohjelmointi, pilvipalvelu

ABSTRACT

Oulu University of Applied Sciences
Bachelor of Engineering, programming

Author: Jere Hagman

Title of thesis: Quiz Game for Android

Supervisor: Eino Niemi

Term and year when the thesis was submitted: Autumn 2020

Pages: 36

The purpose of this thesis was to design and implement a quiz game for the Android operating system using the Java programming language. The goal of this thesis was to develop a deeper understanding of Full Stack mobile development and explore what Amazon's AWS cloud computing platform has to offer and make use of that in the technical implementation.

The thesis work began with getting familiar with the possible tools and technologies to be used. The chosen tools and technologies are introduced and explored in the first part of this written thesis. The technical implementation details are explained in the second part.

As a result of this thesis, a functioning demo application was successfully developed. It would be possible to publish the game on Google Play with further minor to moderate development and time investment. Consequently, the author gained a deeper understanding of Full Stack mobile development and cloud computing.

Keywords: Android, AWS, mobile game, mobile development, programming, cloud computing

SISÄLLYS

TIIVISTELMÄ	3
ABSTRACT	4
SISÄLLYS	5
SANASTO	6
1 JOHDANTO	8
2 TYÖKALUT JA TEKNOLOGIAT	9
2.1 Android Studio	9
2.2 Lottie-kirjasto	10
2.3 Jira Software	10
2.3.1 Scrum ja Kanban	10
2.3.2 Muut ominaisuudet	13
2.4 Amazon Web Services	14
2.4.1 Amazon DynamoDB	15
2.4.2 AWS Amplify	16
2.5 Git ja GitHub	16
3 TOTEUTUS	19
3.1 Käyttöliittymäkulku	19
3.2 Graafinen ilme	20
3.3 Käyttäjän autentikointi	23
3.4 Kysymykset ja vastaukset	25
3.5 Pelin kulku	27
3.6 Tietokanta	29
4 POHDINTA	32
LÄHTEET	34

SANASTO

API	Ohjelmointirajapinta (engl. application programming interface). Ohjelmien välillä toimiva ”välittäjä”, joka mahdollistaa ohjelmien välisen keskustelun ja tiedonsiirron.
AWS	Amazon Web Services, Amazonin tarjoama kattava pilvipalvelualue, joka sisältää monia alapalveluita.
Backend	Termi ohjelmistokehitykselle, joka keskittyy datan käsittelemiseen ja bisneslogiikkaan.
CSV	Comma-separated values, yksinkertainen ja kevyt tiedostomuoto.
Frontend	Termi ohjelmistokehitykselle, joka keskittyy siihen, mitä loppukäyttäjä näkee ruudullaan.
Full Stack	Yleisesti käytetty nimitys ohjelmistokehitykselle, joka kattaa sekä frontend- että backend-kehityksen.
Google Play	Googlen digitaalinen sisältöpalvelu. Palvelusta löytyy mm. Android-sovelluksia, elokuvia, musiikkia ja kirjoja.
GUI	Graphical user interface, graafinen käyttöliittymä.
JSON	JavaScript Object Notation, yksinkertainen ja kevyt tiedostomuoto.

Pilvipalvelu	Paikasta ja fyysisistä rajoitteista riippumaton tietotekninen palvelu. Ohjelmat tai tiedot ovat kolmannen osapuolen hostaamana pilvessä, eivätkä esimerkiksi omalla tietokoneella.
Pull request	Versionhallinnan käytäntö, jolla kehittäjä voi pyytää koodimuutoksiansa viemistä esimerkiksi repositorion pääharaan.
Sovelluskehys	Ohjelmistokehitystä helpottamaan ja nopeuttamaan luotu ohjelmisto, joka yleensä toimii runkona jollekin järjestelmälle.

1 JOHDANTO

Tämän opinnäytetyön lähtökohtana oli oma kiinnostukseni mobiiliohjelmointiin ja haluni tutustua ohjelmistokehityksen alalla suosittuun Amazonin AWS-pilvipalveluun. Työllä ei ollut erillistä tilaajaa, vaan se tehtiin oman ideani pohjalta. Ohjaajana toimi tietotekniikan lehtori Eino Niemi.

Työn aiheena oli tehdä tietovisapeli Android-käyttöjärjestelmälle. Pelin ominaisuuksiin kuuluu neljä eri kysymyskategoriaa (Suomi, tiede, biologia, kulttuuri), peliastin, vaikeustason valitseminen, pisteytys sekä AWS-pilvessä oleva tietokanta, johon pelaajat voivat tallentaa pisteitään.

Pelin ideana on vastata mahdollisimman nopeasti oikein esitettyihin monivalintakysymyksiin. Pelaajalla on kolme ”elämää”, eli väärin voi vastata enintään kolme kertaa. Peli päättyy, kun kaikkiin annettuihin kysymyksiin on vastattu, pelaajan elämät loppuvat tai aika loppuu. Pisteitä saa oikeista vastauksista valitun vaikeustason kertoimen mukaan. Vaikeustason nostaminen lyhentää pelin ajastimen aikaa.

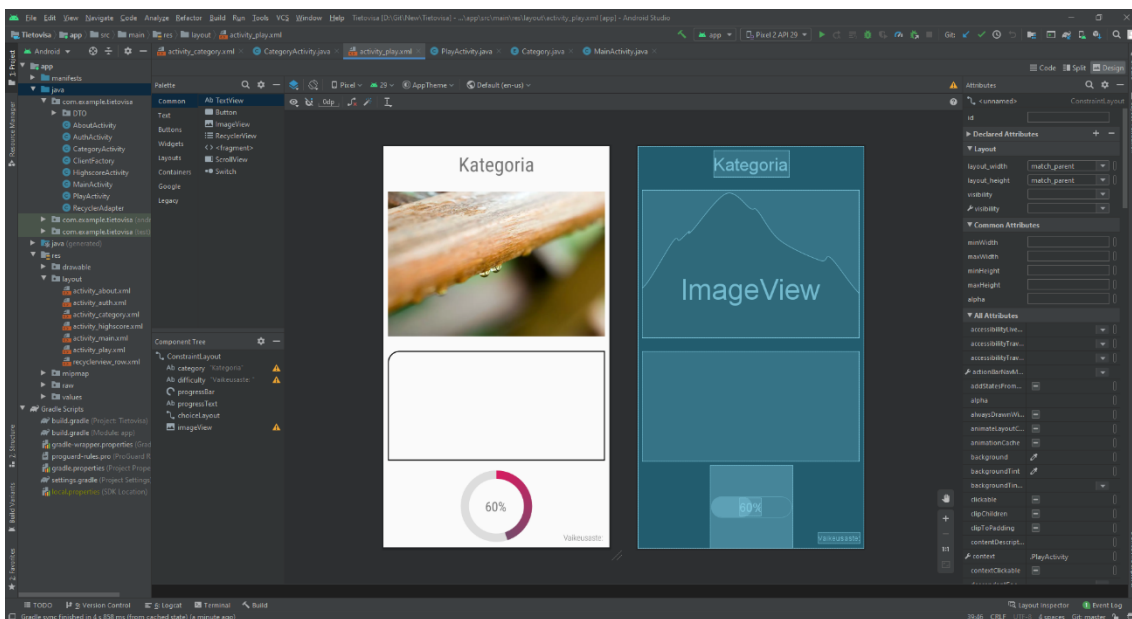
Tietovisatyyllisiä sovelluksia on tilastojen mukaan saatavilla sovelluskaupoissa suuri määrä ja osa on varmasti luokittelematta, jolloin niitä ei näy tilastoissa. AppBrain-sivuston mukaan Google Play -sovelluskaupassa on hakuhetkellä ladattavissa noin päälle 3 miljoonaa Android-sovellusta (1). Näistä sovelluksista on hakuhetkellä Trivia-kategorian alla 18 414 sovellusta (2).

2 TYÖKALUT JA TEKNOLOGIAT

Tämä luku kattaa teoriaosuuden opinnäytetyössä käytetyistä työkaluista ja teknologioista. Luvussa esitellään valitut työkalut ja teknologiat sekä käydään läpi, kuinka niitä käytettiin opinnäytetyön toteutuksessa.

2.1 Android Studio

Projektin tekoon käytettiin Android Studio -ohjelmaa ja Java-ohjelmointikieltä. Android Studio on Androidin virallinen, Googlen kehittämä ohjelmointiympäristö. Android Studio (kuva 1) toimii alustana kaikelle Android-pohjaiselle kehitykselle ja virheenkorjaukselle. Se kehitettiin JetBrainsin IntelliJ IDEA -ohjelmointiympäristön pohjalta (3).



KUVA 1. Android Studion Design-näkymä

Android Studio käyttää Gradle-pohjaista kääntäjää, joka mahdollistaa laajasti kustomoitavan käännösaunomaation sovelluksille. Android Studiossa on mahdollista testata sovellusta kehityksen aikana emulaattorilla, joka simuloi erilaisia Android-laitteita. Emulaattorin käyttö nopeuttaa kehitystä, helpottaa ongelmien löytämistä ja sen avulla voidaan testata sovellusta eri valmistajien Android-lait-

teilla. Android Studioon on myös integroitu mahdollisuus käyttää eri versionhallintajärjestelmiä, kuten Git, GitHub, CVS, Mercurial, Subversion ja Google Cloud Repository. (4.)

Toinen saatavilla oleva Android-ohjelmointiympäristö on Eclipse Foundationin Eclipse-ohjelmointiympäristö. Android Studio on nykyisin syrjäyttänyt Eclipsen pääasiallisena Android-kehitysympäristönä.

2.2 Lottie-kirjasto

Lottie on Airbnb:n luoma avoimen lähdekoodin kirjasto, jonka avulla voidaan vauhtomasti lisätä Adobe After Effects -animaatioita sovelluksiin. Se on saatavilla Androidille, iOS:lle ja React Nativelle. Tässä opinnäytetyössä Lottie-kirjastoa käytettiin muutamien animaatioiden näyttämiseen.

Hernan Torrisin kehittämä Bodymovin-niminen avoimen lähdekoodin After Effects -laajennuksen avulla animaatioita voidaan muuntaa JSON-muotoon. Lottie parseroi JSON-tiedostoiksi viedyt After Effects -animaatiot ja renderöi ne ruudulle. (5.)

2.3 Jira Software

Projektin hallinnassa käytettiin Atlassianin kehittämää Jira Software -tuotetta. Jira on alalla hyvin tunnettu ja se onkin yksi suosituimmista ohjelmistoprojektien tehtävienhallintatyökaluista (6).

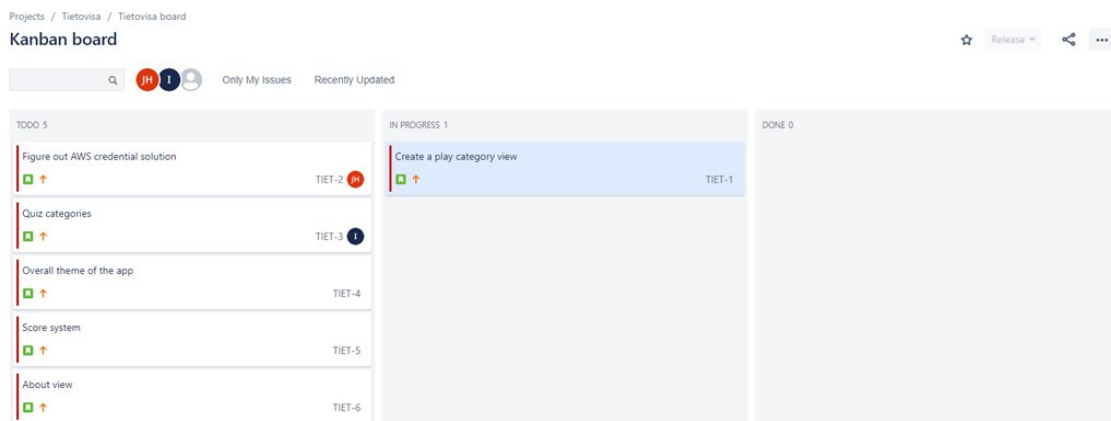
Jiran tärkeimpiin ominaisuuksiin kuuluvat Scrum- ja Kanban-taulut, roadmap-työkalut ja muut erilaiset ketterän kehityksen edistämiseen tarkoitetut työkalut.

2.3.1 Scrum ja Kanban

Scrum- ja Kanban-taulut ovat melko samanlaisia ominaisuuksiltaan, mutta niitä käytetään eri tavalla. Molemmat on tarkoitettu kehitystehtävien edistymisen seurantaan. Scrum-tyyliä käytetään yleisesti, jos ohjelmistokehitystä tehdään sykleissä, niin kutsutuissa sprinteissä. Sprintti on ajanjakso, jonka aikana siihen valikoidut kehitystehtävät suoritetaan sekä testataan toimiviksi. Sprintin kesto voi

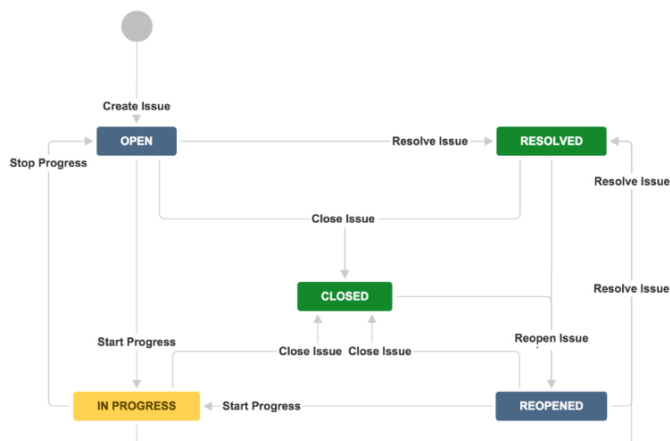
vaihdella viikosta jopa kahteen kuukauteen. Ohjelmistoprojekteissa, joissa opinäytetyön tekijä on itse ollut mukana, on totuttu kahden viikon sprintteihin, jotka tekijä on todennut sopivan pituisiksi. (7.)

Kanban-tyyliä (kuva 2) puolestaan käytetään, kun kehitystä tehdään niin sanotusti jatkuvalla toimituksella. Tällöin ei yleensä käytetä sprinttiä tai muuta kehitystä kohdistavaa ajanmäärettä. Molempiin tyyliin kuuluvat kehitysjonot (engl. backlog), joihin kerätään kehitystä vaativia asioita. Näistä kehitysjonosta nostetaan tehtäviä ja käyttäjätarinoita taululle, ja ne pyritään tyylistä riippumatta viemään työnkulun (engl. workflow) läpi ”Done”-kolumniin asti. (7.)



KUVA 2. Tietovisapelin Kanban-näkymä Jirassa

Jirassa edellä mainituille tauluille luodaan eri tyyppin issueita, joita viedään kehityksen mukaan työnkulun läpi. Issue on Jirassa yleinen termi vielä tyypittämättömille kirjauksille (8). Työnkulun tyyppin voi valita taululleen monesta Jiran oletus-työnkulusta (kuva 3) ja lisää on saatavissa muun muassa Atlassianin Marketplace-palvelusta (9). Tässä opinäytetyössä käytettiin mukautettua työnkulua, jossa oli vain kolme kolumnia: *TODO*, *IN PROGRESS* ja *DONE*.



KUVA 3. Jiran oletustyönkulku (9)

Issueille on valittavissa viisi issue-tyyppiä. Ensimmäinen näistä on story-tyyppi, joka vastaa ohjelmistokehityksessä käytettävää termiä ”käyttäjätarina” (kuva 4). Käyttäjätarina on ytimekkäästi ilmaistu kuvaus kehitettävästä ominaisuudesta loppukäyttäjän näkökulmasta. Task-tyyppi on tarkoitettu pienille työtehtäville. Sub-task on nimensä mukaan edellä mainitusta task-tyypistä vielä pienemmäksi pilkottu osa. Bug-tyyppi on myös suhteellisen itsestään selvä; se on tarkoitettu ohjelmointivirheiden korjauksiin. Nämä kaikki edellä mainitut yhteen nippuun laitettava tyyppi on epic, jonka alle tiettyyn asiaan liittyvät bugit, storyt ja taskit listataan. (10.)

TIET-12 Give feedback 1

Leaderboard for scores

Attach Create subtask Link issue

Description
As a player, I want to be able to save my high scores so that I can compete with my friends.

Activity
Show: Comments History Work log

Assignee
Unassigned

Reporter
Jere Hagman

Labels
None

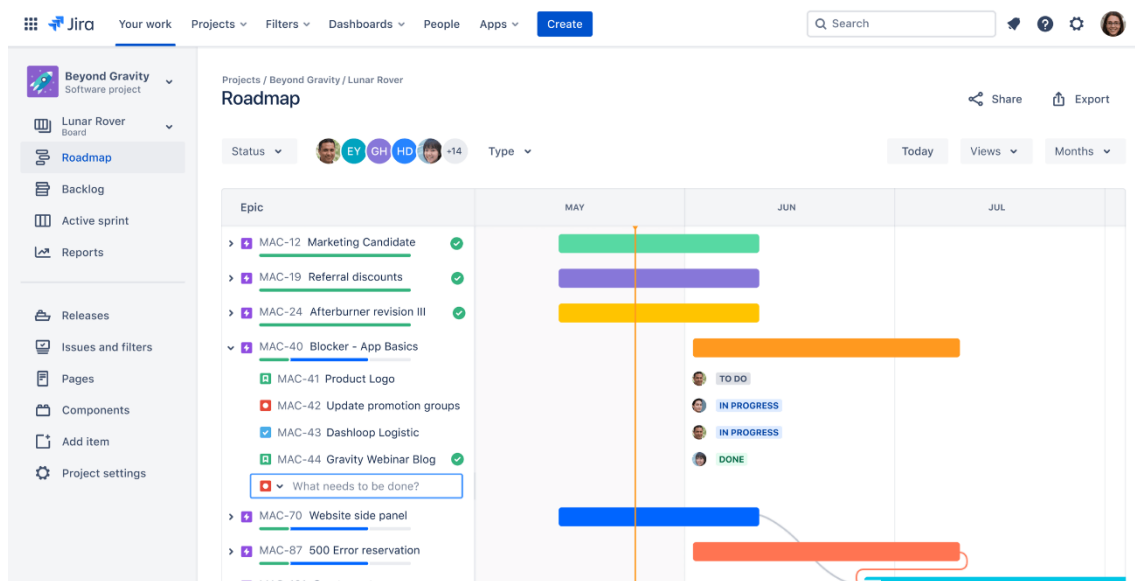
Priority
Medium

Pro tip: press **M** to comment

KUVA 4. Esimerkki käyttäjätarinasta Jirassa

2.3.2 Muut ominaisuudet

Jira tarjoaa myös työkaluja roadmapien eli etenemissuunnitelmien luomiseen ja hallintaan. Roadmapit antavat laajemman kuvan projektien aikataulusta ja edistyksestä. Jira tarjoaa kahta erilaista roadmap-tyyppiä: advanced ja basic (kuva 5). Basic-tyypin roadmap on käytävissä Jiran ilmaisversiossa, kun taas advanced on vain Jira Premiumin käyttäjille (11). Tässä opinnäytetyössä ei käytetty roadmap-ominaisuutta ollenkaan, sillä sen ei nähty tuovan arvoa yhden henkilön projektiin.



KUVA 5. Basic-tyypin roadmap Jirassa (11)

Monet muut Atlassianin tuotteet integroituvat Jiraan vaivatta ja parantavat käyttökokemusta ja ohjelmistokehityksen ketteryyttä. Suosittuja Jiraan liitettäviä tuotteita ovat muun muassa Confluence ja Bitbucket.

Confluence on organisaatiolle ja tiimeille tarkoitettu dokumentointialusta, johon on helppo kerätä kaikki tarvittava dokumentaatio. Näiden dokumenttien linkittäminen esimerkiksi käyttäjätarinoille Jirassa onnistuu helposti. (12.)

Bitbucket on Atlassianin tarjoama versionhallinnan järjestelmä, jossa ylläpidetään projektien lähdekoodit. Myös Bitbucket toimii suoraan Jiran kanssa mahdollistaen esimerkiksi automaattisesti päivittyvät branch- ja pull request -näkyvät käyttäjätarinoille. (13.)

Tässä opinnäytetyössä Jirasta oli erityisesti hyötyä kehitettävien ominaisuuksien seurannassa sekä projektin kokonaiskuvan ja tilanteen hahmottamisessa. Jiraa käytettiin pelkästään seuraamaan teknisen toteutuksen osuutta opinnäytetyöstä.

2.4 Amazon Web Services

Amazon Web Services (AWS) on mahdollisesti maailman suosituin pilvipalvelu-alusta. Se tarjoaa yli 175 yksilöille sekä yrityksille tarkoitettua palvelua ja tuotetta (14). Lähes kaikissa AWS:n tarjoamissa pilvipalveluissa avainasemassa on niiden helppo skaalautuvuus.

AWS:n palvelut on hostattu ympäri maailmaa hyvällä kattavuudella. Saatavuudesta puhuttaessa käytetään termejä Region ja AZ (Availability Zone). Regioneita on kirjoitushetkellä yhteensä 25 ja ne sijoittuvat Pohjois-Amerikkaan, Etelä-Amerikkaan, Aasian ja Tyynenmeren alueelle, Eurooppaan, Lähi-Itään sekä Afrikkaan. Palvelinkeskukset sijoittuvat Availability Zoneille, joita on kirjoitushetkellä Regioneissa yhteensä 77. Availability Zonet mahdollistavat esimerkiksi yritysten applikaatioiden jakauttamisen monelle AZ:lle, mikä turvaa applikaation toimivuutta esimerkiksi sähkökatkosten tai luonnonkatastrofien sattuessa. (15.)

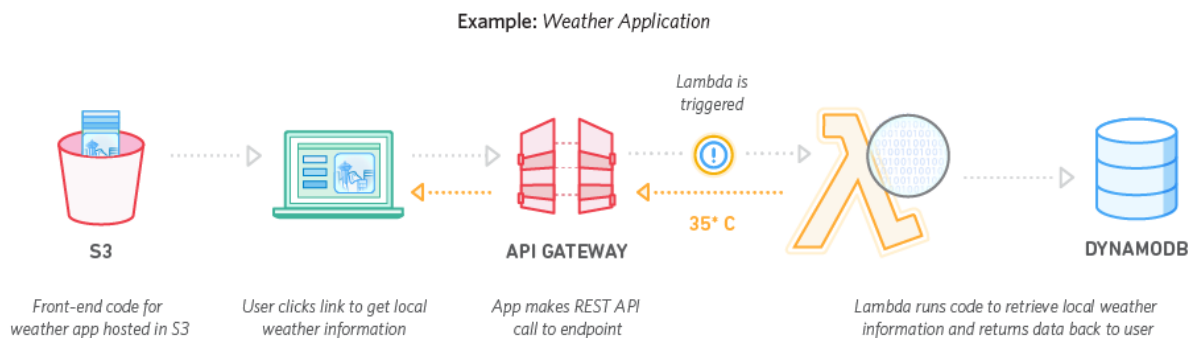
Muutamia esimerkkejä suosituimmista palveluista ovat Amazon Simple Storage Service (S3), AWS Lambda, Amazon EC2 ja Amazon DynamoDB. Amazon Simple Storage Service eli S3 on skaalautuva oliopohjainen datavarasto. S3:een voi tallentaa lähes minkä tahansa tyyppin olioita. Sen käyttötapauksia ovat esimerkiksi suuren datamäärän varmuuskopiointi ja palautus, datan arkistointi ja datan määräaikainen säilytys. Ennen kuin dataa pystyy tallentamaan, täytyy ensin luoda niin kutsuttu bucket eli säiliö, johon dataa varastoidaan. Bucketit ovat samankaltaisia kuin tietokoneen hakemistot. (16.)

AWS Lambda on eventteihin ja palvelimettomuuteen perustuva koodinajoalusta, jossa maksetaan vain siitä ajasta, kun koodia ajetaan. Lambdalla voi helposti käytännössä tehdä minkälaisen applikaation vain ja Amazon hoitaa automaattisesti resurssien ja kulujen laskemisen. Lambdalla on myös kattava yhteensopi- vuus AWS:n muiden tuotteiden kanssa. (17.)

Amazon EC2 eli Amazon Elastic Compute Cloud on skaalautuva pilvilaskennan palvelu. EC2:n avulla voidaan nopeasti luoda halutunlainen virtuaalinen palvelin pilveen. Fyysistä palvelinta ei tarvita esimerkiksi nettisivujen hostaamiseen. Näitä virtuaalisia ympäristöjä kutsutaan instansseiksi. (18.)

2.4.1 Amazon DynamoDB

DynamoDB on Amazonin tarjoama pilvessä oleva NoSQL-tietokanta (kuva 6). Se on nopeasti käyttöön otettava, käytön mukaan skaalautuva ja joustava tietokanta. Tunnettuja DynamoDB:tä käyttäviä yrityksiä ovat muun muassa Samsung, Netflix ja Airbnb. (19.)



KUVA 6. Esimerkki käytöstä DynamoDB:lle (19)

NoSQL-tyylisiä tietokantoja kehitettiin vaihtoehdoksi normaaleille SQL-tietokannoille jo 90-luvulla, mutta itse termi tuli käyttöön vasta 2000-luvulla. Niiden suosion kasvua edesauttoi tietojen säilyttämisen kustannusten raju lasku sekä tarve ketterämpään kehitykseen. NoSQL-tietokantojen hyvä puoli on niiden joustavuus, sillä ne toimivat periaatteella, jossa niihin tallennettavan datan tyyppiä ja kokoa ei välttämättä tarvitse ennalta määrittää. Niiden joustavuuden ansiosta saadaan nopeampi kehitystahti, sillä tietokannan rakenteiden muokkaaminen muuttuvan datan perusteella on paljon helpompaa. NoSQL-tietokantojen päätyypit ovat document, key-value, wide-column ja graph. DynamoDB:ssä voidaan käyttää key-value- ja document-mallisia ratkaisuja. (20.)

Tässä opinnäytetyössä käytettiin DynamoDB:tä pelaajien pisteiden tallentamiseen ja lukemiseen, koska tähän käyttötapaukseen haluttiin helposti ymmärrettävä ja muokattava tietokanta.

2.4.2 AWS Amplify

AWS Amplify on Amazonin tarjoama sovelluskehys (engl. framework). Amplifyn avulla voidaan muun muassa rakentaa web- ja mobiilisovelluksien serverless backend -ratkaisuja. Amplify-sovelluskehys koostuu kolmesta pääkomponentista: CLI-työkalusta (Command-line interface), kirjastoista ja UI-komponenteista. (21.)

CLI-työkalu on nimensä mukaan komentorivipohjainen työkalu. Sen avulla pystyy hallinnoimaan sovelluksen backend-konfiguraatiota lisäämällä siihen haluttuja komponentteja, kuten ohjelmointirajapintoja tai autentikaatoratkaisuja. Amplifyhin voi liittää Amazonin itse kehittämiä avoimen lähdekoodin kirjastoja, kuten ohjelmointirajapinta-, tietovarasto- ja analytiikkakirjastoja. Saatavilla on myös valmiita UI-komponentteja käytettäväksi Amplify-sovelluksissa. (22.)

2.5 Git ja GitHub

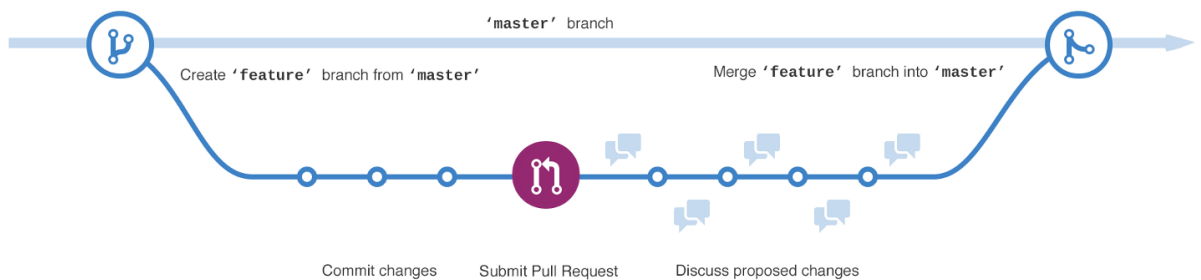
Työssä käytettiin alalla vakiintunutta versionhallinnan työkalua, Gitiä. Kunnollisen versionhallinnan tärkeyttä ei pidä koskaan unohtaa ohjelmistoprojekteissa, sillä sen avulla voidaan välttyä monilta katastrofeilta.

Git on ilmainen, avoimen lähdekoodin versionhallintajärjestelmä, jonka on alun perin luonut suomalainen Linus Torvalds. Git kehitettiin vuonna 2005 korvaamaan Linux-ytimen versionhallintaan käytettyä BitKeeperiä. Syynä korvaamiselle oli Linux-yhteisön riitaantuminen BitKeeperiä tarjoavan yhtiön kanssa sen jälkeen, kun yhtiö päätti muuttaa BitKeeperin maksulliseksi. Git on julkaistu GNU-lisenssin alla. (23.)

Gitissä käytetään haaroja (engl. branch), joita yhdistetään (engl. merge) usein toisiin haaroihin ja yleensä lopuksi niin kutsuttuun master-haaraan. Haarat ovat yleensä aluksi vain paikallisia, eli ne ovat vain kehittäjän koneella. Yleisin käyttö-

tapaus haaroille on käyttää niitä uusien ominaisuuksien tai bugikorjauksien tekemiseen. Hyvänä nyrkkisääntönä on, että jokaiselle yksittäiselle ominaisuudelle tai virheenkorjaukselle olisi oma haaransa, joka kehityksen jälkeen yhdistetään master-haaraan.

Gitille voi omaksua monia erilaisia työkulkutyöskentelytapoja. Yksi suosittu työkulku on tunnetun Git repositorio -hostaajan GitHubin oma yksinkertainen työkulku nimeltään GitHub flow (kuva 7).



KUVA 7. GitHub flow (24)

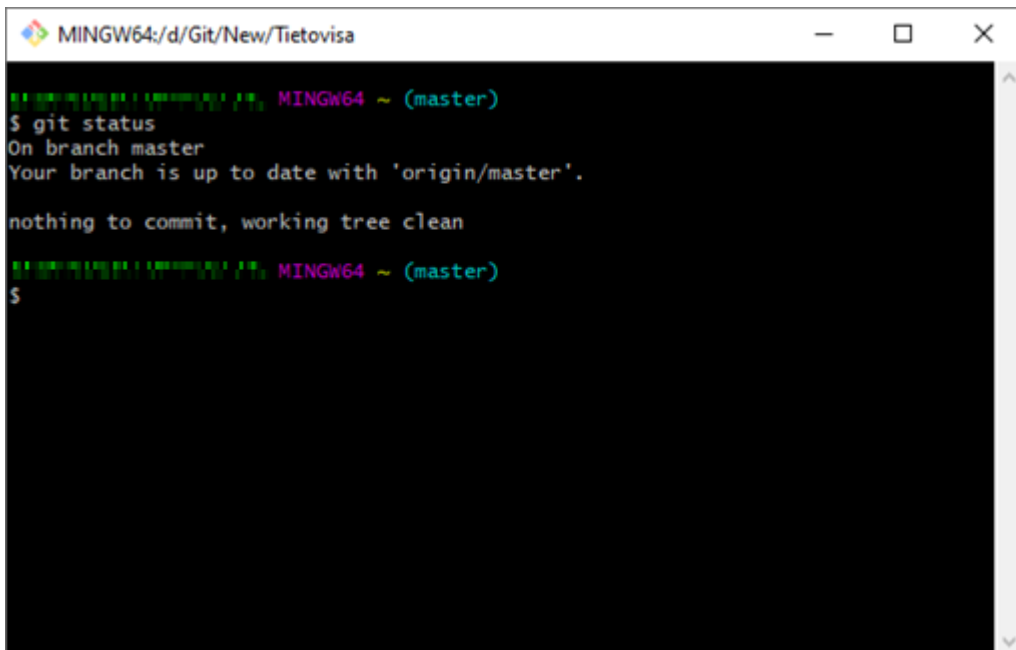
GitHub flow toimii seuraavasti:

1. Master-haarasta luodaan uusi feature-haara (esimerkiksi jollekin kehitettävälle ominaisuudelle).
 2. Feature-haaraan lisätään tiheästi koodia sitä mukaa, kun sitä syntyy.
 3. Luodaan pull request.
 4. Muut kehittäjät kommentoivat tarvittaessa koodimuutoksia.
 5. Koodiin tehdään tarvittaessa korjaavia muutoksia kommenttien perusteella.
 6. Pull request hyväksytään ja koodimuutokset yhdistetään master-haaraan.
- (25.)

Tämän opinnäytetyön koodi hostattiin yksityiseen GitHub-repositorioon. Työssä käytettiin GitHub flow'ta ilman muiden kehittäjien kommentointia pull requesteihin.

Gitille on olemassa monia erilaisia GUI-pohjaisia sovelluksia. Tässä työssä käytettiin kuitenkin komentorivipohjaista Git BASHiä (kuva 8), joka sisältyy Git for

Windowsiin. Git BASH jäljittelee Linuxin ja macOS:n Unix-tyylistä komentoriviä Windows-käyttöjärjestelmälle (26).



```
MINGW64:/d/Git/New/Tietovisa
MINGW64 ~ (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
MINGW64 ~ (master)
$
```

KUVA 8. Git BASH

3 TOTEUTUS

Tässä luvussa esitellään opinnäytetyön toteutuksen osaa. Luvussa esitellään työn toteutus ja käydään läpi sovelluksen rakennetta ja kooditason ratkaisuja.

3.1 Käyttöliittymäkulku

Tyypillinen graafinen Android-sovellus rakentuu aktiviteeteista. Sovellusta kehitettäessä aktiviteetit voidaan mieltää näyttöruutuina, joille sovellus piirtää käyttöliittymän. Yleensä jokainen ruutu on oma aktiviteettinsa. Sovelluksen käyttöliittymäkulku (kuva 9) pyrittiin tekemään melko yksinkertaiseksi, jotta vältettäisiin projektin monimutkaisuuden kasvu odotettua suuremmaksi.



KUVA 9. Sovelluksen kulkukaavio

MainActivity toimii Tietovisa-sovelluksen alkupisteenä ja päävalikkona. MainActivity:n Rekisteröidy- ja Kirjaudu-painikkeet tuovat MainActivity:n päälle dialog-ikkunan, jossa suoritetaan edellä mainitut toiminnot. Päävalikosta voi navigoida lukemaan tietoja sovelluksesta AboutActivityyn, katsomaan huippupisteitä HighscoreActivityyn sekä pelaamaan itse peliä CategoryActivityyn kautta.

Pelaa-painiketta painettaessa jatketaan CategoryActivityyn, jossa aktiviteetin nimen mukaisesti valitaan haluttu kysymysten kategoria sekä pelin vaikeustaso. CategoryActivityyn pelaa-napilla siirrytään PlayActivityyn ja aloitetaan peli.

PlayActivityyn yläosassa näkyy valitun kategorian nimi, kategoriaan liittyvä pelin alussa vaihtuva pikkukuva, peliajastin, kysymyksen numero ja pelaajan elämät. Peliaktiviteetin alaosassa on vaihtuva monivalintakysymys ja sen vastaukset. Oikeista vastauksista saa pisteitä valitun vaikeustason määräämällä kertoimella. Seuraavaan aktiviteettiin siirrytään, kun kysymykset loppuvat kesken tai pelaajan elämät tai aika loppuvat.

Viimeinen aktiviteetti pelin käyttöliittymäkulussa on HighscoreActivity, johon pääsee myös suoraan päävalikosta. Jos tähän aktiviteettiin tullaan pelin kautta, näytetään pelistä kertynyt pistemäärä ja lyhyt konfettianimaatio. Saadut pisteet voidaan haluttaessa tallentaa, jos pelaaja on rekisteröitynyt ja kirjautunut sovellukseen. Alhaalla näkyvät kymmenen korkeinta tietokannassa olevaa pistemäärää. Jos aktiviteettiin tullaan päävalikosta ja pelaaja on kirjautunut, näytetään pelaajan henkilökohtainen huippupistemäärä. Muutoin näytetään vain tietokannasta kymmenen korkeinta pistemäärää.

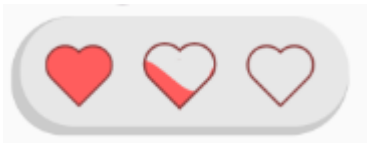
3.2 Graafinen ilme

Sovellukseen pyrittiin tekemään yksinkertainen ja siisti graafinen ilme. Valikot tehtiin yhtenäisen näköisiksi luomalla niille tausta, josta on kolme eri versiota eri sävyillä (kuva 10). Näin pyrittiin saamaan kolmiulotteinen vaikutelma, joka tuo valikoille paremmin ilmettä. Kirkkaimman sävyistä taustaa käytettiin aina pohjana ja sen päälle tuleville elementeille käytettiin tummempia taustoja.



KUVA 10. Valikoiden ja elementtien eri tasot

Sovelluksessa käytettiin icons8.com-sivustolta ladattuja ilmaisia ikoneita muun muassa Back-painikkeeseen ja eri kysymyskategorioiden hahmottamiseen. Peliin lisättiin myös muutamia animaatioita Lottie-kirjastoa käyttäen. Näitä animaatioita käytettiin hahmottamaan pelaajan elämiä (kuva 11) ja pelin päättymistä (kuva 12).



KUVA 11. Ruudunkaappaus animaatiosta pelaajan menettäessä elämän



KUVA 12. Ruudunkaappaus konfettianimaatiosta pelin päätyttyä

Sovellukseen haluttiin yhtenäinen tausta, joka ei ole vain pelkkä kuva tai väri. Taustakuva ei myöskään saanut olla liian päällekkäyvä eikä vaikeuttaa sovelluksen tekstien lukemista. Päädyttiin tekemään raidallinen, kolmesta eri sävystä koostuva animoitu taustakuva, jossa raidat ja värit liukuvat hitaasti näytön läpi (kuva 13).



KUVA 13. Sovelluksen taustakuva

Värit määriteltiin yleisen tason `colors.xml`-tiedostoon, jotta ne ovat helposti käytettävissä muualla koodissa. Taustakuvalle tehtiin neljä eri `xml`-tiedostoa: `start-`, `middle-`, `end-` ja `stripeGradient`. Näissä tiedostoissa määritellään animaatioissa käytettävät vaiheet. Viimeisiksi resurssitiedostoiksi luotiin vielä kaksi `xml`-tiedostoa, joissa edellä mainitut taustat asetetaan itemeiksi `animation-list`-elementin sisälle ja säädetään jokaiselle animaation sisällä olevalle taustalle kesto millisekunneissa.

Animaatioiden käynnistämistä varten luotiin yksinkertainen apuluokka (kuva 14), jonka `setUpAnimatedBackground`-metodia kutsutaan aktiviteeteissä, joihin animoitu taustakuva halutaan. Metodissa asetetaan kutsuva aktiviteetti full screen -tilaan, jotta yläpalkki ei olisi näkyvässä. `AnimationDrawable`ksi asetetaan kutsuvan aktiviteetin juurilayoutin tausta sekä asetetaan `enter-` ja `exit-`häivytysajat millisekuntien tarkkuudella. Lopuksi animaatiot aloitetaan.

```

public class BackgroundHelper {
    public static void setUpAnimatedBackground(Window window, Context context, ConstraintLayout rootLayout, LinearLayout stripeLayout) {
        window.getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
        AnimationDrawable animationDrawable = (AnimationDrawable)rootLayout.getBackground();
        Animation anim = AnimationUtils.loadAnimation(context, R.anim.stripe_animation);
        animationDrawable.setEnterFadeDuration(10);
        animationDrawable.setExitFadeDuration(3000);
        animationDrawable.start();
        stripeLayout.startAnimation(anim);
    }
}

```

KUVA 14. BackgroundHelper-apuluokka

3.3 Käyttäjän autentikointi

Sovellukseen tarvittiin tapa autentikoida käyttäjä, jotta käyttäjäkohtaiset pisteet voitaisiin tallentaa tietokantaan. Tietokantana tässä työssä käytettiin AWS DynamoDB:tä. Sovellukseen lisättiin autentikointi AWS:n Amplify-sovelluskehiksen avulla.

Kun Amplify backend -runko on lisätty projektiin, voidaan siihen lisätä erinäisiä kategorioita, kuten tässä tapauksessa Auth-kategoria. Tähän käytettiin Amplify CLI -työkalua. Prosessi aloitetaan "amplify add auth" -komennolla, jonka jälkeen ohjelma kysyy autentikaation konfigurointiin liittyviä kysymyksiä, kuten millä tavalla käyttäjien halutaan kirjautuvan (käyttäjänimellä, sähköpostilla tai puhelinnumerolla). Lopuksi muutokset tallennetaan AWS:ään "amplify push" -komennolla. Amplify luo sovelluksen backend-runkoon linkitetyn Cognito Identity- ja User Poolin, joista jälkimmäisessä voi verkossa hallinnoida sovelluksen käyttäjiä (kuva 15). Prosessi luo myös sovellukseen linkitetty IAM-roolit (Identity and Access Management), authRole ja unauthRole, joiden sisäisiä oikeuksia voidaan säätää. Tässä tapauksessa authRoleen asetettiin kirjoitus- ja lukuoikeudet pelaajien pisteitä varten luotuun DynamoDB-tietokannan tauluun, jotta autentikoidut pelaajat voivat tallentaa pisteitään. UnauthRoleen määriteltiin vain lukuoikeudet, jotta pelaajat, jotka eivät ole kirjautuneet, voivat kuitenkin nähdä top 10 -pisteet.

Users > JereTest

Add to group Reset password Enable SMS MFA Disable user

Groups -

Account Status Enabled / CONFIRMED

SMS MFA Status Disabled

Last Modified Oct 10, 2020 3:05:41 PM

Created Oct 10, 2020 3:04:08 PM

sub [REDACTED]

email_verified true

email [REDACTED]

KUVA 15. Käyttäjän hallinnointi Cognito User Poolissa

Sovellukseen tehtiin rekisteröitymistä ja kirjautumista varten dynaamiset dialogikkunat MainActivityyn. Kun käyttäjä on syöttänyt tarvittavat tiedot ja painaa Rekisteröidy-painiketta, kutsutaan koodissa Amplifyn Auth -kategorian signUp-metodia, joka hoitaa käyttäjän rekisteröinnin ja lähettää käyttäjän antamaan sähköpostiin vahvistuskoodin (kuva 16). Seuraavaksi dialogi laajenee ja pyytää käyttäjää syöttämään vahvistuskoodin rekisteröinnin loppuun suorittamiseksi (kuva 16).



KUVA 16. RegisterDialog-näkymä

Kun rekisteröinti on suoritettu onnistuneesti, voi käyttäjä kirjautua sisään käyttäjätunnuksella ja salasanallaan (kuva 17).



KUVA 17. LoginDialog-näkymä

3.4 Kysymykset ja vastaukset

Tietovisan kysymykset ja vastaukset koottiin alun perin Google Sheets -sovellukseen. Tiedot lisättiin CSV-tyyliä ajatellen sarakkeisiin niin, että jokaisella sarakkeella on oma ylätunnus. Tässä tapauksessa sarakkeita oli kuusi: question, a,

b, c, d ja rightAnswer. Sovelluksen koodia varten päätettiin tietojen käsittelyn helpottamiseksi käyttää JSON-formaattia, joten kysymykset ja vastaukset kopioitiin sellaisenaan Sheets-sovelluksesta csvjson.com-verkkotyökaluun (27), joka hoiti datan muuntamisen JSON-formaattiin. Tuloksena saatiin alla oleva tietorakenne.

```
{
  "questions": [
    {
      "question": "Mikä on Suomen kansalliseläin?",
      "a": "Karhu",
      "b": "Joutsen",
      "c": "Ilves",
      "d": "Susi",
      "rightAnswer": "Karhu"
    }
  ]
}
...
```

Eri kategorioiden JSON-tiedostot tallennettiin projektin sisäiseen resurssikansioon. Tiedostojen parsimista varten luotiin Questions-luokka, joka vastaa JSON-datan rakennetta. Avuksi luotiin myös JsonHelper-luokka (kuva 18), joka käyttää Googlen avoimen lähdekoodin Gson-kirjastoa muuntamaan annettu JSON-merkijono Questions-objektiksi.

```
public class JsonHelper extends AppCompatActivity {

    public static Questions convertJsonToModel(Context context, int resourceId) {
        String json = inputStreamToString(context.getResources().openRawResource(resourceId));
        return new Gson().fromJson(json, Questions.class);
    }

    static String inputStreamToString(InputStream inputStream) {
        try {
            byte[] bytes = new byte[inputStream.available()];
            inputStream.read(bytes, 0, bytes.length);
            return new String(bytes);
        } catch (IOException e) {
            return null;
        }
    }
}
```

KUVA 18. JsonHelper-luokka

Pelin alustuksessa arvotaan pelaajan valitsemasta kysymyskategoriasta viisi-toista satunnaista kysymystä sekä niiden vastaukset erillisen metodin avulla (kuva 19).

```

private ArrayList<Questions.QuestionAndAnswer> draftFifteenRandomQuestions() {
    Questions questionPool = JsonHelper.convertJsonToModel(context, this, getCategoryId());
    ArrayList<Questions.QuestionAndAnswer> draftedQuestions = new ArrayList<>();
    for(int i = 0; i < 15; i++) {
        int randomIndex = ThreadLocalRandom.current().nextInt(questionPool.list.size());
        draftedQuestions.add(questionPool.list.get(randomIndex));

        if(questionPool.list.size() > 0) {
            questionPool.list.remove(randomIndex);
        }
    }
    return draftedQuestions;
}

```

KUVA 19. Metodi kysymysten arpomiseen

3.5 Pelin kulku

Pelin alkaessa onCreate-metodissa alustetaan aktiviteetin elementit ja käynnistetään säie, joka vastaa ajastimen päivittämisestä. Alustuksessa volatile int -tyyppiseen progress-muuttujaan asetetaan sekuntimäärä, joka saadaan convertDifficultyToSeconds-metodin avulla. Edellä mainittu metodi ottaa sisään CategoryActivityssä valitun vaikeustason int-arvon väliltä 1–3. Ajastimen säie vähentää saatua progress-muuttujan arvoa yhdellä kerran sekunnissa. ProgressBar-elementti ja ajastimen teksti päivitetään progress-muuttujan perusteella. Kun progress-muuttuja päättyy 0 arvoon eli aika loppuu, kutsutaan BuildAndShowDialog-metodia (kuva 20) ja säie pysäytetään. BuildAndShowDialog-metodia kutsutaan myös kahdessa muussa pelin loppumisen tilanteissa: elämien loppuessa ja kun kaikkiin kysymyksiin on keretty vastata ennen ajastimen pysähtymistä. Metodissa rakennetaan AlertDialog-elementti ja asetetaan dialogin napit ja niiden toiminnot. Jatkettaessa eteenpäin HighscoreActivityyn lähetetään Intent, johon lisätään int-tyypin score-muuttuja, jossa on pelaajan nykyiset pisteet ja boolean-arvo true, joka määrittää, näytetäänkö seuraavaan aktiviteettiin tultaessa konfettianimaatio.

```

private void buildAndShowDialog(String messageText) {
    keepRunningThread = false;
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setMessage(messageText)
        .setTitle("Tietovisa")
        .setNegativeButton( text: "Palaa alkuun", (dialog, which) -> {
            startActivity(new Intent( packageContext: PlayActivity.this, MainActivity.class));
            finish();
        })
        .setPositiveButton( text: "Jatka", (dialog, which) -> {
            startActivity(new Intent( packageContext: PlayActivity.this, HighscoreActivity.class)
                .putExtra( name: "score", score)
                .putExtra( name: "confetti", value: true));
            finish();
        })
        .setCancelable(false);
    AlertDialog dialog = builder.create();

    dialog.show();
}

```

KUVA 20. BuildAndShowDialog-metodi

Aina kun pelaaja vastaa kysymykseen, tarkistetaan, oliko vastaus oikea. Jos vastaus oli oikea, lisätään pelaajan kokonaispistemäärään sata pistettä kerrottuna valitulla vaikeustasolla. Jos vastaus oli väärä, kutsutaan loseLife-metodia, joka vähentää pelaajalta elämän ja käynnistää siihen liittyvän animaation. LoseLife-metodi hoitaa myös pelin päättämisen, jos elämät loppuvat. Näiden lisäksi pelaajan vastatessa kysymykseen kutsutaan updateCurrentQuestion-metodia (kuva 21), joka päivittää ruudulle uuden satunnaisen kysymyksen aiemmin arvotuista viidestätoista kysymyksestä. Jos edellinen kysymys oli viimeinen, peli päättyy.

```

private void updateCurrentQuestion() {
    // End game if current question was the last
    if(currentQuestion == 15) {
        buildAndShowDialog( messageText: "Vastasit kaikkiin kysymyksiin. Peli ohi!");
        return;
    }
    currentQuestion++;
    questionCountText.setText(currentQuestion + " / 15");

    int randomIndex = ThreadLocalRandom.current().nextInt(currentQuestionSet.size());
    Questions.QuestionAndAnswer questionAndAnswer = currentQuestionSet.get(randomIndex);

    questionText.setText(questionAndAnswer.question);
    choiceA.setText(questionAndAnswer.a);
    choiceB.setText(questionAndAnswer.b);
    choiceC.setText(questionAndAnswer.c);
    choiceD.setText(questionAndAnswer.d);
    currentRightAnswer = questionAndAnswer.rightAnswer;

    // Remove current question from the list
    currentQuestionSet.remove(randomIndex);
}

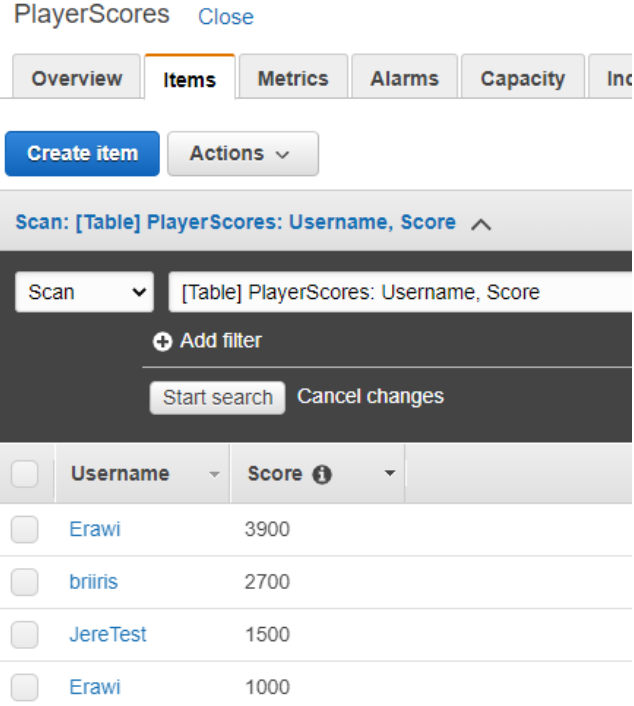
```

KUVA 21. UpdateCurrentQuestion-metodi

3.6 Tietokanta

Kuten aiemmin mainittiin, sovelluksen tietokantana käytettiin Amazonin NoSQL-tietokantaa DynamoDB:tä. Tietokannan tarkoituksena oli tässä sovelluksessa vain säilyttää pelaajien pisteet.

Tietokantaan luotiin PLayerScores-niminen tietokantataulu (kuva 22) AWS Management Consolen kautta selaimella. Taulun perusavaimeksi (engl. primary key) luotiin yhdistelmäavain (engl. composite key), joka koostuu ositusavaimesta (engl. partition key) ja järjestysavaimesta (engl. sort key). Ositusavaimeksi valittiin String-tyyppin "Username" ja järjestysavaimeksi number-tyyppin "Score". Tällainen yhdistelmäavain mahdollistaa sen, että tallentaessa ositusavain (käyttäjänimi) voi olla jo tietokantataulussa, mutta järjestysavain (pistemäärä) täytyy olla vastaavalle käyttäjälle uniikki. Näin tietokantaan ei synny kaksoiskappaleita.



PlayerScores [Close](#)

Overview **Items** Metrics Alarms Capacity Inc

Create item Actions

Scan: [Table] PlayerScores: Username, Score

Scan [Table] PlayerScores: Username, Score

+ Add filter

Start search Cancel changes

<input type="checkbox"/>	Username	Score
<input type="checkbox"/>	Erawi	3900
<input type="checkbox"/>	briiris	2700
<input type="checkbox"/>	JereTest	1500
<input type="checkbox"/>	Erawi	1000

KUVA 22. PLayerScores-taulun tietoalkioita järjestettynä Score-avaimen mukaan

Koodiin luotiin DatabaseAccess-luokka (kuva 23), jonka tehtävänä on alustaa DynamoDB-yhteyttä varten tarvittavat muuttujat. Luokan konstruktorissa luodaan

AmazonDynamoDBClient-instanssi ja ”ladataan” nimetty taulu tietokannasta. Luokassa on myös metodit ladatun taulun käsittelyä varten.

```
public class DatabaseAccess {  
  
    private static volatile DatabaseAccess instance;  
    private final String COGNITO_IDENTITY_POOL_ID = "arn:aws:iam::123456789012:role/cognito-identity-provider";  
    private final Regions COGNITO_IDENTITY_POOL_REGION = Regions.US_EAST_1;  
    private final String DYNAMODB_TABLE = "PlayerScores";  
    private CognitoCachingCredentialsProvider credentialsProvider;  
    private AmazonDynamoDBClient dbClient;  
    private Table dbTable;  
  
    private DatabaseAccess(Context context) {  
        credentialsProvider = new CognitoCachingCredentialsProvider(context, COGNITO_IDENTITY_POOL_ID, COGNITO_IDENTITY_POOL_REGION);  
        dbClient = new AmazonDynamoDBClient(credentialsProvider);  
        dbClient.setRegion(Region.getRegion(COGNITO_IDENTITY_POOL_REGION));  
        dbTable = Table.loadTable(dbClient, DYNAMODB_TABLE);  
    }  
  
    public static synchronized DatabaseAccess getInstance(Context context) {  
        if (instance == null) {  
            instance = new DatabaseAccess(context);  
        }  
        return instance;  
    }  
  
    public void createItem(Document doc) {  
        dbTable.putItem(doc);  
    }  
  
    public List<Document> getAllItems() {  
        return dbTable.scan(new Expression()).getAllResults();  
    }  
}
```

KUVA 23. DatabaseAccess-luokka

HighscoreActivityyn tultaessa luodaan GetAllItemsAsyncTask-luokasta asynkroninen tehtävä (engl. async task), joka käynnistetään execute-metodilla. Tämä tehtävä hakee taustasäikeessä kaikki tietoalkiot (engl. items) tietokanta-taulusta DatabaseAccess-luokan avulla. Kun säie on hakenut tiedot, ylikirjoitettussa onPostExecute-metodissa kutsutaan populateScoreList- ja tryPopulateUserTopScore-metodeja. PopulateScoreList-metodi käy for-silmukassa läpi haetut tietoalkiot ja lisää ne väliaikaiseen listaan. Tämä lista järjestetään tietoalkioiden Score-kentän mukaan käyttäen mukautettua Comparator-rajapinnan toteuttavaa ScoreSorter-vertailijaa (kuva 24). Järjestystä listasta valitaan 10 ensimmäistä indeksiä ja näiden perusteella alustetaan RecyclerView, joka renderöi näytölle huippupistetaulukoon nämä kymmenen suurinta pistemäärää ja nämä pistemäärät saaneet käyttäjät.

TryPopulateUserTopScore-metodi toteutuu vain, jos kirjautunut käyttäjä on tullut päävalikosta HighscoreActivityyn. Se suodattaa haetuista tietoalkioista vain nykyisen käyttäjänimen perusteella löytyvät alkiot, järjestää ne ScoreSorter-vertailijan avulla sekä valitsee järjestämisen jälkeen 0-indeksissä

olevat pisteet eli pelaajan suurimman pistemäärän. Tämä pistemäärä näytetään ruudun yläosassa.

```
public class ScoreSorter implements Comparator<Score> {  
    @Override  
    public int compare(Score s1, Score s2) {  
        return s2.getScore().compareTo(s1.getScore());  
    }  
}
```

KUVA 24. ScoreSorter-vertailija.

Jos pelaaja on tullut HighscoreActivityyn itse pelin kautta, luodaan Tallenna-painiketta painettaessa scoreEntry-niminen Document-tyypin muuttuja, johon lisätään tietokannan rakenteen mukaisesti nykyisen käyttäjän nimi ja pelistä saadut pisteet. CreateItemAsyncTask-luokasta luodaan asynkroninen tehtävä ja se käynnistetään execute-metodilla, joka ottaa parametrinä sisään scoreEntry-muuttujan. Tämä tehtävä tallentaa DatabaseAccess-luokan avulla taustasäikeessä uuden tietoalkion tietokantatauluun ja hakee tallennuksen jälkeen kaikki tietoalkiot uudelleen, jotta aktiviteetissa näkyvä huippupistetaulukko päivittyisi.

4 POHDINTA

Alkuperäisessä lähtötietomuistiossa työn kuvaus oli yksinkertainen:

Työssä tehdään tietovisasovellus Android-alustalle. Sovellus esittää kysymyksiä ja väittämiä käyttäjälle. Oikeista vastauksista saa pisteitä ja pelin lopussa on mahdollista tallentaa pisteet. Sovelluksessa on myös mahdollista nähdä lista kaikkien käyttäjien huippupisteistä. (28.)

Kuvaus vastasi siis mielestäni lopullista toteutusta melko hyvin. Projektin näkyvyysalue (engl. scope) kasvoi väistämättä projektia tehdessä ja uusia ominaisuuksia lisättiin sen mukaan. Näitä oli muun muassa mahdollisuus rekisteröityä, kysymyskategoriat, vaikeustason valinta ja animaatiot.

Lähtötietomuistiossa opinnäytetyön tavoitteiksi päätettiin: ”Päätavoitteena on tehdä toimiva sovellus Kotlin-kielellä ja julkaista se Play-kaupassa. Tarkoituksena on myös perehtyä käytettyihin teknologioihin ja kertoa niistä” (28). Alkuperäiset tavoitteet eivät vastanneet lopullista todellisuutta. Ohjelmointikieli vaihdettiin Javaksi projektin alkupuolella, sillä se oli minulle jo entuudestaan tuttu kieli. Peli oli tarkoitus alun perin julkaista Google Play -kaupassa, mutta projektin jatkuessa tuli selväksi, että en halua julkaista peliä, ennen kuin se on hiottu kunnolla valmiiksi. Käytettyihin teknologioihin perehdyttiin ja niistä luonnollisesti kerrottiin osana opinnäytetyötä.

Muutamia ongelmakohtia ja jatkokehitysmahdollisuuksia huomattiin opinnäytetyön toteutusosaa tehdessä. Vaikeustaso tuntuu keinotekoiselta, sillä sen nostaminen lyhentää vain peliajastimen aikaa. Parempi toteutus vaikeustasosta voisi olla esimerkiksi kysymysten kategorisointi niiden vaikeuden mukaan. Peliajastin voisi tässä tapauksessa myös olla sellainen, että se käynnistyy uudelleen jokaisen kysymyksen jälkeen, eikä niin kuin se on nykyisellään toteutettu, eli peliajastin on koko pelin ajan sama. Nykyisen toteutuksen takia suurin pistemäärä, jonka voi saada, on 4 500. Pisteytyksen toteutus pitäisi suunnitella uudelleen, jotta pisteiden vaihteluväli olisi isompi. Kysymyksiä on tällä hetkellä liian vähän, mutta jos niitä laitettaisiin sovellukseen lisää, voitaisiin esimerkiksi pisteongelman lievittä-

miseksi poistaa pelin 15 kysymyksen yläraja ja näin ollen lisätä pisteiden vaihteluväliä. Alustavien pelaajatestausten mukaan käyttöliittymän selkeyttä voitaisiin parantaa esimerkiksi tekemällä paneelien värimuutoksista selkeämpiä. Ulkoasua pitäisi testata kattavammin erikokoisilla laitteilla ja tehdä parannuksia ulkoasuun tämän perusteella, jotta siitä saataisiin yhtenäinen kaikille laitteille.

Kaiken kaikkiaan opinnäytetyö oli mielestäni onnistunut. Sitä tehdessä opin paljon uutta ja uraa varten hyödyllistä tietoa ja taitoa.

LÄHTEET

1. Android and Google Play statistics. 2020. AppBrain. Saatavissa: <https://www.appbrain.com/stats>. Hakupäivä: 29.8.2020.
2. Top categories, Category statistics table. 2020. AppBrain. Saatavissa: <https://www.appbrain.com/stats/android-market-app-categories> Hakupäivä 29.8.2020.
3. Meet Android Studio. 2020. User guide. Google. Saatavissa: <https://developer.android.com/studio/intro>. Hakupäivä 30.8.2020.
4. Version control basics. 2020. User guide. Google. Saatavissa: https://developer.android.com/studio/intro#version_control_basics. Hakupäivä 1.9.2020.
5. Abdul-Karim, Salih. Lottie – Behind the scenes of our new open-source animation tool. Saatavissa: <https://airbnb.design/introducing-lottie/>. Hakupäivä: 5.10.2020.
6. Jira Software Alternatives. 2020. Atlassian. Saatavissa: <https://www.atlassian.com/software/jira/comparison>. Hakupäivä 30.8.2020.
7. Scrum Board vs Kanban: Choosing the Right Agile Tool. 2020. Lucidchart Content Team. Saatavissa: <https://www.lucidchart.com/blog/kanban-vs-scrum>. Hakupäivä 1.9.2020.
8. What is an issue? 2020. Jira Software Support. Saatavissa: <https://support.atlassian.com/jira-software-cloud/docs/what-is-an-issue/>. Hakupäivä: 2.9.2020.
9. Working with workflows. 2018. Administering JIRA Server 7.9 applications. Atlassian support. Saatavissa: <https://confluence.atlassian.com/adminjiraserver079/working-with-workflows-950288650.html> Hakupäivä 4.9.2020.
10. What are issue types? 2020. Atlassian Support. Saatavissa: <https://support.atlassian.com/jira-cloud-administration/docs/what-are-issue-types/>. Hakupäivä: 2.9.2020.

11. Roadmaps in Jira Software. 2020. Atlassian. Saatavissa: <https://www.atlassian.com/software/jira/features/roadmaps?tab=basic>. Hakupäivä 5.9.2020.
12. Confluence's features create a single source of truth. 2020. Atlassian. Saatavissa: <https://www.atlassian.com/software/confluence/features> Hakupäivä 5.9.2020.
13. The code collaboration platform for modern software teams. 2020. Atlassian. Saatavissa: <https://bitbucket.org/product/features>. Hakupäivä 5.9.2020.
14. Brandon, John 2020. Complete list of Amazon Web Services. TechRadar. Saatavissa: <https://www.techradar.com/news/aws#complete-list-of-amazon-web-services>. Hakupäivä 6.9.2020.
15. Regions and Availability Zones. 2020. Amazon Web Services. Saatavissa: https://aws.amazon.com/about-aws/global-infrastructure/regions_az/?p=ngi&loc=2. Hakupäivä 8.9.2020.
16. Amazon S3. 2020. Amazon Web Services. Saatavissa: <https://aws.amazon.com/s3/>. Hakupäivä 9.9.2020.
17. AWS Lambda. 2020. Amazon Web Services. Saatavissa: <https://aws.amazon.com/lambda/>. Hakupäivä 9.9.2020.
18. What is Amazon EC2. 2020. User Guide for Linux Instances. Amazon Web Services. Saatavissa: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>. Hakupäivä 10.9.2020.
19. Amazon DynamoDB overview. 2020. Amazon Web Services. Saatavissa: <https://aws.amazon.com/dynamodb/>. Hakupäivä 23.9.2020.
20. Schaefer, Lauren 2020. What is NoSQL. MongoDB. Saatavissa: <https://www.mongodb.com/nosql-explained>. Hakupäivä 23.9.2020.
21. AWS Amplify. 2020. Amazon Web Services. Saatavissa: <https://aws.amazon.com/amplify/>. Hakupäivä: 20.10.2020.

22. AWS Amplify features. 2020. Amazon Web Services. Saatavissa: <https://aws.amazon.com/amplify/features/>. Hakupäivä 20.10.2020.
23. A Short History of Git. Git-scm. Saatavissa <https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>. Hakupäivä 27.9.2020.
24. Esquivel, Gabo 2018. 15 Tips to Enhance your Github Flow. Hackernoon. Saatavissa: <https://hackernoon.com/15-tips-to-enhance-your-github-flow-6af7ceb0d8a3>. Hakupäivä: 27.9.2020.
25. Understanding the GitHub flow. 2020. GitHub. Saatavissa: <https://guides.github.com/introduction/flow/>. Hakupäivä 27.9.2020.
26. Git Bash. Atlassian. Saatavissa: <https://www.atlassian.com/git/tutorials/git-bash>. Hakupäivä: 15.11.2020.
27. Drapeau, Martin 2014–2019. CSVJSON Online tool. Saatavissa: <https://csvjson.com/csv2json>. Hakupäivä: 29.9.2020.
28. Lähtötietomuistio 22.1.2020. Oulun ammattikorkeakoulu. Tietotekniikan tutkinto-ohjelma. Tekijän hallussa.
29. Robert, Pierre. Heart 5. 2020. LottieFiles. Saatavissa: <https://lottiefiles.com/16868-heart-5>. Hakupäivä: 1.10.2020.
30. Geyer, Mark. Confetti cannons. 2019. LottieFiles. Saatavissa: <https://lottiefiles.com/7893-confetti-cannons>. Hakupäivä: 3.10.2020.