Leo Partanen

Microsoft Officen käyttö Android-sovellusympäristössä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

24.5.2021

Tiivistelmä

Tekijä:	Leo Partanen
Otsikko:	Microsoft Officen käyttö Android-sovellusympäristössä
Sivumäärä:	31 sivua
Aika:	24.5.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mobile Solutions
Ohjaaja:	Yliopettaja Kari Aaltonen

Insinöörityön tarkoituksena oli tutustua Microsoft Officen käyttöön Android-sovellusympäristössä, minkä, lisäksi perehdyttiin myös Kotlin-ohjelmointikielen käyttöön ja sen etuihin Android-kehityksessä. Työssä perehdyttiin myös Microsoft Officeen ja yleisiin Android-kehityksen teknologioihin.

Työtä varten kerättiin tietoa verkosta englanninkielisistä lähteistä, koska sopivaa sisältöä ei juuri löydy suomeksi, ja tietotekniikan monet osa-alueet kehittyvät jatkuvasti, siksi ajan tasalla olevaa kirjallisuuttakin on vain rajatusti.

Insinöörityössä luotiin esimerkkisovellus Apache POI -rajapintaa käyttämällä, ja käyttöliittymän luomiseen käytettiin Android Studiota. Esimerkkisovelluksen ohjelmointiosuus toteutettiin kokonaan Kotlin-ohjelmointikieltä käyttäen. Sovellus keskittyi Excel-tiedostojen käsittelyyn.

Työ tehtiin Helsingin kaupungin työntekijän toiveesta yhdistää työajanseuranta ja laskutusta varten tarvittava työtehtävien kirjaaminen. Työntekijä selvitti myös, että kaupungilla on todella tarvetta uudistaa työajan kirjaaminen. Tällä hetkellä työajanseuranta on toteutettu mobiilisovelluksella, mutta työntekijät joutuvat erikseen kirjaamaan tekemänsä työt käsin. Ongelmana on eri tarkoitusperiin tarvittavien ohjelmistojen ja sovellusten yhteensopivuus, esim. laskutusohjelma.

Avainsanat: Microsoft Office, Android, ohjelmointi, mobiilisovellus

Abstract

Author:	Leo Partanen
Title:	Usage of Microsoft Office in Android application environment
Number of Pages:	31 pages
Date:	24 May 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technologies
Professional Major:	Mobile Solutions
Supervisor:	Kari Aaltonen, Principal Lecturer

The purpose of this thesis was to become acquainted with the usage of Microsoft Office in Android application environment. Another aim was to introduce the usage of Kotlin programming language and its advantages in Android development.

In this study, Microsoft Office and common Android development technologies were introduced as well. Information was retrieved from the internet using sources written in English, because suitable content was difficult to find in Finnish; many fields of information technology are constantly evolving, therefore up-to-date literature is mostly available in English.

In this study, the example application was created by using Apache POI interface, and the user interface was created with Android Studio. The programming of the example application was done entirely by using Kotlin programming language. The application focused on manipulating Excel files.

This study was done at the request of an employee of the City of Helsinki to combine time-tracking and the recording of work tasks required for invoicing. The employee also found out the city really needs to update this system. Currently, time-tracking is done with a mobile application, but employees have to record their work tasks manually. The problem is the compatibility of software and applications needed for different purposes, for example, invoicing software.

Keywords: Microsoft, Office, Android, programming, mobile application

Sisällys

Lyhenteet

MS: Microsoft. Yhdysvaltalainen monikansallinen teknologiayritys.

BIFF: Binary Interchange File Format. Excel-binääritiedostomuoto.

XML: Extensible Markup Language. Merkintäkieli, joka auttaa jäsentämään laajoja tietomassoja selkeämmin ja on sekä ihmisen että koneen luettavissa.

OLE: Object Linking & Embedding. Microsoftin kehittämä patentoitu tekniikka, joka mahdollistaa sulauttamisen ja linkittämisen asiakirjoihin ja muihin objekteihin.

DSL: Domain-specific language. Tietylle sovellusalueelle erikoistunut ohjelmotikieli eli ns. täsmäkieli.

ART: Android Runtime. Android-alustalla käytettävä ajoympäristö.

HAL: Hardware Abstraction Layer. Ohjelmistoalijärjestelmä UNIX-tyyppisille laitteiston omaaville käyttöjärjestelmille.

DEX: Dalvik Executable. Androidille suunniteltu tavukoodimuoto.

NDK: Native Development Kit. Ohjelmistokehityspaketti, joka on kokoelma ohjelmistokehityksen työkaluja.

JVM: Java virtual machine. Abstrakti kone, joka suorittaa sille käännettyjä Java-ohjelmia.

ADB: Android Debug Bridge. Monipuolinen komentorivityökalu, jonka avulla voi kommunikoida laitteen kanssa.

1 Johdanto

Opinnäytetyön aiheeksi valikoitui Microsoft Officen käyttö Android-sovellusympäristössä, sillä itse olen erityisen kiinnostunut Android-sovellusten suunnittelusta ja kehittämisestä ja sain tuttavaltani ehdotuksen hänen työajankirjaamistansa ja työtuntien raportointia helpottavasta sovelluksesta, johon liittyi Excel-tiedoston luominen mobiilisovelluksella.

Esimerkkisovelluksena lähdin toteuttamaan automatisoitua työajankirjausjärjestelmää, jonka tehtävänä on laskea tehdyt työtunnit ja raportoida ne suoraan Excel-tiedostoksi. Lähtökohtana on, ettei käyttäjän tarvitsisi tehdä muuta kuin kirjata työnsä nappia painamalla. Käyttäjän tulee kuitenkin rekisteröidä manuaalisesti työnumero ja kirjoittaa haluamansa työnkuvaus tehdylle työlle. Käyttäjän ei tarvitse miettiä, kuinka kauan aikaa on kulunut kutakin työtehtävää kohden, eikä myöskään muistella, mitä töitä on tullut tehtyä minäkin päivänä, ja tehdyt työtunnit myös täsmäävät enempää miettimättä. Esimerkkisovelluksen tarkoituksena ei ole ainoastaan helpottaa työntekijän tuntikirjausta, vaan myös vähentää tuntien kirjaukseen käytettävää työaikaa, mikä työnantajan näkökulmasta on sille eduksi.

2 Toimintaympäristön teknologiat

2.1 Microsoft Office

MS Office on Microsoft Corporationin oma tuote, joka julkaistiin ensimmäisen kerran vuonna 1990. Se on työpöydän tuottavuussovellusten sarja, joka on suunniteltu erityisesti toimisto- tai yrityskäyttöön. Se on maailmanlaajuisesti erittäin tunnettu ja yrityselämässä sekä kouluissa yleisesti käytetty sovellussarja. Se on saatavana jopa 35 eri kielellä, ja sitä tukevat MS:n oman Windows-käyttöjärjestelmän lisäksi myös Mac ja useimmat Linux-variantit, kuten Android, jota tässä opinnäytetyössä käsitellään tarkemmin. MS Office tarjoaa monia erilaisia työskentelyyn tarkoitettuja sovelluksia, kuten Word, Excel, PowerPoint, Outlook, OneNote, Access ja Publisher, joista käytetyimpiä ovat kolme ensimmäisenä lueteltua sovellusta. (1; 2.)

MS Office on esiasennettu kaikkiin Windows-käyttöjärjestelmän omaaviin kannettaviin ja pöytätietokoneisiin. MS Officen käytön laajuudesta kertoo myös se, että suuret tietokone- ja älylaitevalmistajat, kuten Sony, LG, Haier, Dell ja Samsung, ovat useiden valmistajien joukossa, jotka ovat allekirjoittaneet strategisen sopimuksen MS:n johtavina yhteistyökumppaneina. Sen seurauksena moniin Android-käyttöjärjestelmää käyttäviin älypuhelimiin ja tabletteihin on esiasennettu MS Office -sovelluksia. MS Office -sovellukset julkaistiin ensimmäisen kerran ilmaisena latauksena iPhone- ja Android-käyttäjille vuonna 2014. (2; 3.)

MS Excel on MS:n julkaisema laskentataulukkosovellus, joka on osa MS Office -tuoteohjelmistoa. Se on yksi toimistojen tuottavuuden sovelluksista, jonka ensimmäinen versio julkaistiin vuonna 1985. Excel on työkalu, jota käytetään datan laskemiseen ja hallintaan. Se voi analysoida tietoja, laskea tilastotietoja, luoda kääntötaulukoita ja esittää tietoja kaaviona tai kuvaajana. MS Excel on saatavana Windows-, macOS-, Android- ja iOS-käyttöjärjestelmiin. Se voi myös toimia Linux-käyttöjärjestelmässä tietokoneohjelmisto WINE:n avulla, joka mahdollistaa monien MS Windows -sovelluksien käytön Linux-käyttöjärjestelmissä. Excel järjestää tiedot sarakkeissa ja riveissä. Niiden leikkaamaa aluetta kutsutaan soluksi, ja

jokainen solu voi sisältää yhden tyyppisen tiedon, kuten tekstin, numeerisen arvon tai laskukaavan. (4.)

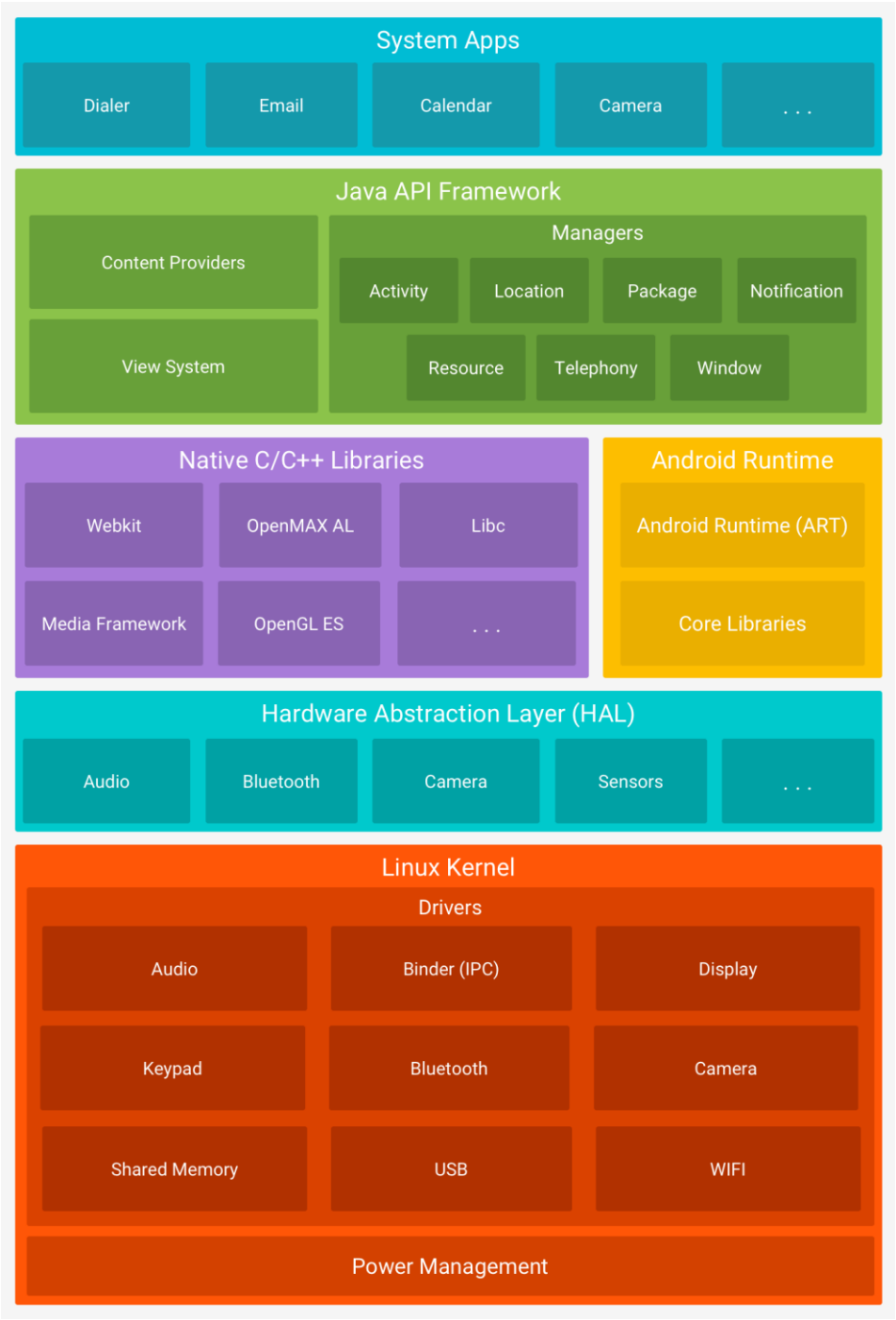
XLS ja XLSX ovat MS:n kehittämiä ja patentoimia tiedostomuotoja, jotka on tarkoitettu käytettäväksi Excelin kanssa. Excel-tiedostot käyttävät näitä tiedostomuotoja Excel-asiakirjojen tallentamiseen, ja tiedostopäätteinä käytetään päätteitä .xls ja .xlsx. XLSX on Excelin uusin tiedostomuoto Excel-versioille vuodesta 2007 eteenpäin, kun taas XLS on tiedostomuoto vuoden 2003 ja sitä vanhemmille Excel-versiolle. Uudemmissa Excel-versioissa, kuten vuosien 2007 ja 2012 versioissa, on kuitenkin integroitu tuki XLS-tiedostojen avaamiseen, muokkaamiseen ja luomiseen. Vaikka ne ovat saman yhtiön ja täysin samaan käyttötarkoitukseen kehitettyjä menetelmiä, käyttävät ne tietojen tallentamiseen kuitenkin eri tekniikoita. Vanhempi perustuu BIFF-tiedostoon, jossa tiedot tallennetaan suoraan binäärimuotoon. Uudempi taas puolestaan perustuu Office Open XML -muotoon, joka johdettiin nimensä mukaisesti XML:stä. XLSX-tiedoston tiedot tallennetaan tekstipohjaiseen tiedostoon, joka määrittelee kaikki parametrit XML:n avulla. (5; 6.)

2.2 Android-käyttöjärjestelmä

Android on Googlen kehittämä Linux-pohjainen käyttöjärjestelmä, eli Linux-käyttöjärjestelmäydin on Android-käyttöjärjestelmän alin osa. Android on lähtökohtaisesti kosketusnäytöllisille mobiililaitteille suunniteltu käyttöjärjestelmä, joka on jaettu useisiin versionumeroihin merkittävien ominaisuuksien ja toimintojen perusteella, mm. Android 5 (Lollipop), 6 (Marshmallow), 7 (Nougat), 8 (Oreo), 9 (Pie) ja 10. Käyttöliittymä perustuu pääosin suoriin manipulaatioihin, joissa käytetään kosketuseleitä, jotka vastaavat löysästi reaalimaailman toimintoja, kuten pyyhkäisemistä, napauttamista ja puristamista, sekä tekstinsyöttöön käytettävää virtuaalista näppäimistöä. Android on kehittynyt vuosien varrella, ja siitä on nopeasti tullut hallitseva mobiililaite ympäri maailmaa. Sillä oli hallussaan vuoden 2019 loppuun mennessä yli 74 % maailman mobiilisovellusten markkinaosuudesta. Google julkaisee sen lähdekoodin avoimen lähdekoodin lisensseillä, vaikka suurin osa laitteista toimitetaan viime kädessä avoimen lähdekoodin ja patentoitujen

ohjelmistojen yhdistelmällä, mukaan lukien patentoidut ohjelmistot, joita tarvitaan Googlen palveluiden käyttämiseen. Avoimuus onkin Androidin yksi suurimmista eduista, mutta sen lisäksi käyttöjärjestelmiin on saatavilla jatkuvasti uusia päivityksiä vanhemmille laitteille sekä uusia sisäänrakennettuja ominaisuuksia edistyneille käyttäjille. (7; 8.)

Rakenteellisesti Android on avoimen lähdekoodin joustava ohjelmistopaketti, joka on luotu monentyyppisille ja monen eri kokoluokan laitteille, ja se koostuu useista erilaisista komponenteista. (Kuva 1.) Alustan perustana on Linux-käyttöjärjestelmäydin, jonka päälle on koottu mukailtuja komponentteja, jotka käyttävät Linuxille tyypillisiä ominaisuuksia. Esimerkiksi ART (Android Runtime) luottaa Linuxin taustalla oleviin toimintoihin, kuten suoritussäikeisiin ja matalan tason muistinhallintaan. Linuxin käyttäminen pohjaratkaisuna antaa alustalle myös erinomaisen lähtökohdan hyödyntää sen tärkeimpiä suojausominaisuuksia. (9.)



Kuva 1. Android-ohjelmistopino ja sen tärkeimmät komponentit (9).

Linuxin päälle koottu HAL (Hardware Abstraction Layer) tarjoaa standardisoidun käyttöliittymän, joka liittää laitteen ominaisuudet ylemmän tason Java-ohjelmisto-rajapintaan. HAL koostuu useista kirjastomoduuleista, joista kukin toteuttaa liitän-

nän tietyntyyppiselle laitteistokomponentille, kuten kameralle tai Bluetooth-moduulille. Kun kehysrajapinta lähettää pyynnön laitteen laitteistoon pääsemiseksi, Android-järjestelmä lataa kyseisen laitekomponentin kirjastomoduulin. Android 5:sta (Lollipop) lähtien jokainen sovellus toimii omassa prosessissaan ja omalla ART-ilmentymällä. Tätä ennen alustassa käytettiin Dalvik-virtuaalikonetta, jota varten kehitetty DEX-tiedostomuoto on edelleen käytössä. ART pystyy ajamaan useita virtuaalikoneita vähän muistia käyttävissä laitteissa suorittamalla DEX-tiedostoja, jotka on optimoitu minimaaliseen muistipaikkaan. Android 9:stä (Pie) lähtien alusta sisältää sovelluspaketin DEX-tiedostojen muuntamisen vieläkin pienemmäksi konekoodiksi. ART:n etuihin kuuluu myös parempi virheenkorjaustuki, yksityiskohtaiset virheraportit sekä mahdollisuus asettaa tarkkailupisteitä tiettyjen kenttien seuraamiseksi. (9.)

Useat alustan komponenteista, kuten edellä mainitut HAL ja ART, on kirjoitettu C- tai C++-ohjelmointikielellä, ja näin ollen ne vaativat C- ja C++-kielellä kirjoitettuja natiivikirjastoja. C- ja C++-kielellä kirjoitettu Android NDK (Native Development Kit) tarjoaakin kehittäjille mahdollisuuden käyttää joitain näitä natiivikirjaston ominaisuuksia tarvittaessa. Tämä mahdollistaa esimerkiksi tuen 2D- ja 3D-grafiikan piirtämiselle ja käsittelemiselle. Kaikki käyttöjärjestelmän ominaisuudet ovat käytettävissä Java-kielellä kirjoitettujen ohjelmointirajapintojen kautta. Nämä rajapinnat muodostavat rakennuspalikoita, joita tarvitaan luomaan sovelluksia. Ne helpottavat kehitystä yksinkertaistamalla ytimen, modulaaristen järjestelmäkomponenttien ja palveluiden uudelleenkäyttöä. Seuraavassa on lueteltuna ko. rajapintoja ja niiden käyttötarkoituksia Android-kehityksessä:

- View Systemiä tarvitaan sovellusten käyttöliittymän rakentamiseen. Se sisältää yleishyödylliset luettelot, ruudukot, tekstikentät, painikkeet, yms.
- Resource Manager tarjoaa pääsyn ei-koodikielisille resursseille, kuten paikallisille merkkijonoille, grafiikoille ja käyttöliittymän asetteluun.
- Notification Managerin avulla kaikki sovellukset voivat näyttää mukautettuja hälytyksiä statuspalkissa.
- Activity Manager hallitsee sovellusten elinkaarta ja tarjoaa yleisen navigointipinopaketin.

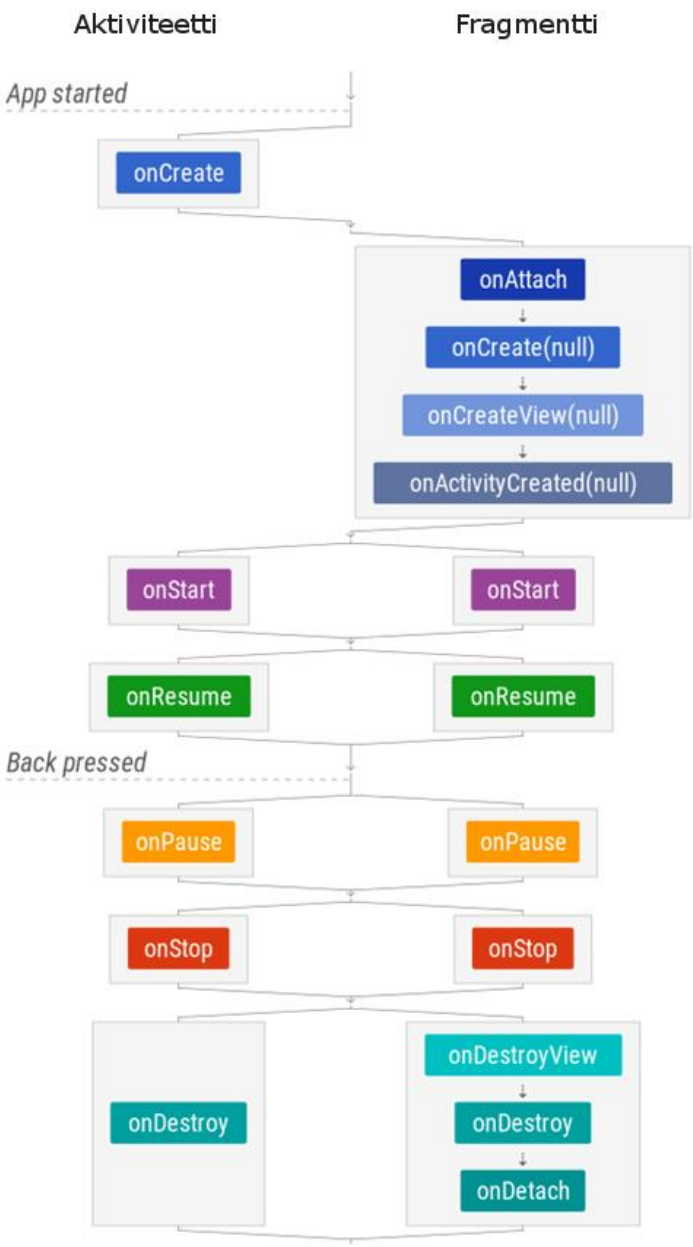
- Content Providers pyytää sovelluksille lupaa käyttää tietoja muista sovelluksista tai jakaa omia tietojaan. (9.)

Androidin näkyvin osa on sen käyttäjillekin tuttu, se koostuu järjestelmäsovelluksista. Kehittäjillä on täysi käyttöoikeus samoihin rajapintakehyksiin, joita järjestelmäsovellukset käyttävät. Androidin mukana tulee joukko erilaisia järjestelmäsovelluksia, joita ovat esimerkiksi sähköposti, tekstiviestit, kalenteri, yhteystiedot ja Internet-selain. Näitä järjestelmäsovelluksia ei voi poistaa, mutta alustan mukana tulleilla sovelluksilla ei ole erityistä asemaa verrattuna niihin sovelluksiin, jotka käyttäjä itse voi asentaa, joten käyttäjä voi asettaa kolmannen osapuolen sovelluksen esimerkiksi oletusselaimeksi, tekstiviestisovellukseksi tai vaikka oletusnäppäimistöksi. Joitain poikkeuksiakin on, kuten käyttöjärjestelmän Asetukset-sovellus, jota ei voi korvata muilla sovelluksilla. Järjestelmäsovellukset toimivat sekä käyttäjien sovelluksina että tarjoavat avaintoimintoja, joita kehittäjät voivat käyttää kehittämässään sovelluksessa. Kehittäjien ei tarvitse itse kehittää toimintoja esimerkiksi tekstiviestien lähettämiseen. (9.)

Ohjelmoinnin kannalta Activity Manager, View System ja Resource Manager ovat tärkeimmät ominaisuudet, joita tarvitaan jo yksinkertaisimmassakin Android-sovelluksessa. Activity Managerin tarjoamat aktiviteetit ovat Android-sovellusten perustavanlaatuisia osia, jotka toimivat lähtökohtaisesti vuorovaikutuksessa käyttäjän ja sovelluksen kanssa. Käytännössä ne toimivat pohjarakenteena kaikille toiminnoille niin taustalla tapahtuville, kuin käyttäjälle näkyvissä oleville toiminnoille. Ne ovat myös keskeisessä roolissa sovelluksen sisäisen navigoinnin osalta. Aktiviteetteja tarvitaan vähintään yksi, mutta mitä laajempi kokonaisuus ja mitä enemmän erilaisia toimintoja sovellus kattaa, sitä tarpeellisemmaksi useamman aktiviteetin käyttäminen osoittautuu.

Pelkän näkymän ja toimintojen vaihtamiseen käyttäjänäkymässä, ei vaadita uutta aktiviteettia. Näkymä ja toiminnot voidaan toteuttaa fragmentteina, jotka ovat aktiviteetteja kevyempiä rakenteita ja vuorovaikutuksessa aktiviteettien kanssa. Aktiviteetissa toimivalla fragmentilla voi olla oma näkymä sekä toimintoja, ja se voi myös käyttää taustalla olevan aktiviteetin toimintoja, mikä puolestaan helpottaa toistuvien rakenteiden käyttöä. Aktiviteetit voivat sisältää useampia fragmentteja,

ja niiden käyttäminen on suositeltavaa. Aktiviteetin elinkaari ohjaa myös fragmenttien elinkaarta: kun aktiviteetti pysähtyy, niin tekee myös fragmentti. (Kuva 2.) Aktiviteetit ovat yksi Android-alustan sovellusten perustekijöistä. Ne toimivat lähtökohtana käyttäjän vuorovaikutuksessa sovelluksen kanssa ja ovat myös keskeisiä sille, miten käyttäjä liikkuu sovelluksen sisällä (kuten Takaisin-painikkeen kanssa) tai sovellusten välillä (kuten Viimeaikaiset-painikkeen kanssa).



Kuva 2. Aktiviteetin ja fragmentin elinkaaret rinta rinnan (10).

Materiaalisuunnittelu (Material Design) on Googlen kehittämä Androidille suuntautunut suunnittelukieli, joka tukee kosketusnäyttökokemuksia luoden illuusion kolmiulotteisuudesta rikkaiden ominaisuuksien ja luonnollisten liikkeiden avulla, jotka matkivat reaalimaailman esineitä. Materiaalisuunnittelun tavoitteena on tuottaa korkealaatuista tuotantoa johdonmukaisesti eri alustoilla, jotta käyttäjät

voivat hallita selkeitä ja miellyttävän näköisiä komponentteja, jotka käyttäytyvät kuin todelliset kohteet. Materiaalisuunnittelun komponentit soveltavat fyysisen maailman luonnollisia peruslakeja, jotka koskevat pääasiassa valaistusta ja liikettä. Fyysistä maailmaa matkimalla on tarkoitus vähentää käyttäjien kognitiivisia kuormia, ja tämä saavutetaan kiinnittämällä huolellista huomiota asetteluun, visuaaliseen kieleen ja mallikirjastoon, maksimoimalla ennustettavuus ja poistamalla epäselvyydet. Esimerkiksi painikkeiden tulisi näyttää mahdollisimman selvästi olevan painettavia ja painettaessa käyttäjän tulisi huomata myös fyysistä painallusliikettä matkiva animaatio, jotta käyttäjä myös ymmärtää painavansa ko. painiketta. (11.)

2.3 Java-virtuaalikone

Java-ohjelmointikieli on alun perin Sun Microsystemsin kehittämä, ja se julkaistiin vuonna 1995 Sun Microsystemsin Java-alustan ydinkomponenttina. Java suunniteltiin mahdollisimman yksinkertaiseksi ja helpoksi oppia. Javan kehittyessä ja sen laajan suosion myötä useita kokoonpanoja rakennettiin sopiviksi erityyppisille alustoille. Java on alustariippumaton olio-ohjelmointikieli ja tästä syystä helposti laajennettavissa. (12.)

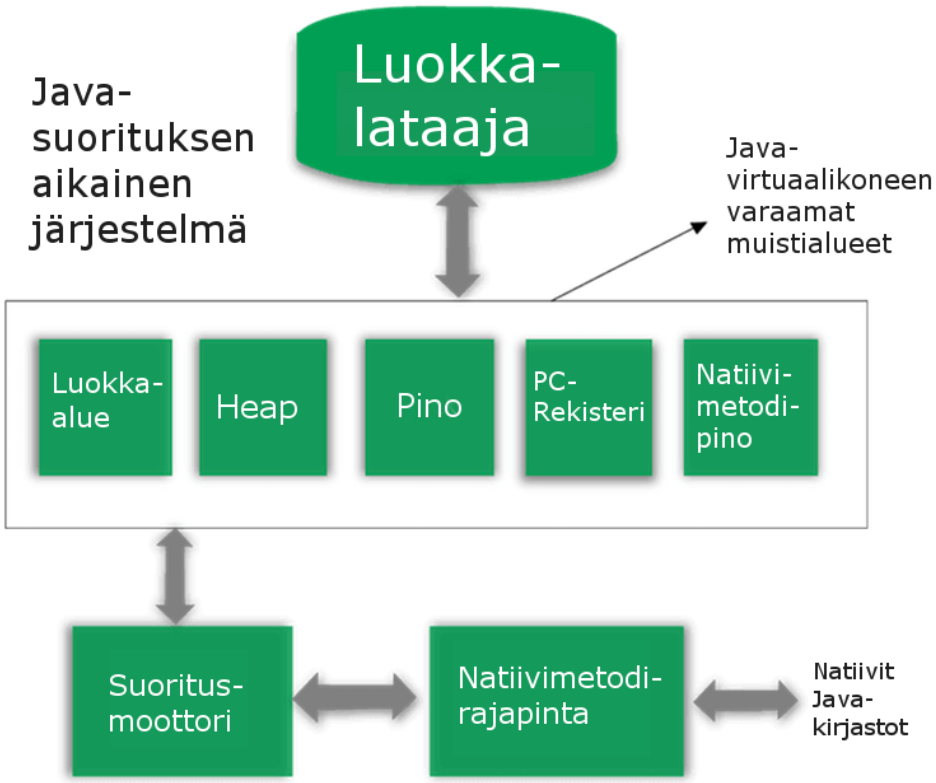
Suurin osa ohjelmointikielistä kääntää lähdekoodin suoraan konekoodiksi, joka soveltuu suoritettavaksi tietyllä mikroprosessoriarkkitehtuurilla. Toisin kuin monia muita ohjelmointikieliä käännettäessä, Javaa ei käännetä alustakohtaiseksi konekoodiksi, vaan alustasta riippumattomaksi tavukoodiksi. Tavukoodi jaetaan verkossa, ja virtuaalikone (JVM) tulkitsee sen, millä alustalla sitä ajetaan. Tavukoodi käännetään lennossa alkuperäisiksi koneen toimintaohjeiksi, eikä sitä tallenneta mihinkään. Kehitysprosessi on nopea ja analyyttinen, koska linkittäminen on asteittainen ja kevyt suorittaa. (13.)

Java-kääntäjä luo arkkitehtuurineutraalin oliotiedostomuodon, joka Javan ajonaikaisen järjestelmän ohella mahdollistaa käännetyn koodin suoritettavuuden useille erityyppisille prosessoreille. Javan arkkitehtuurineutraalius ja alustariippumattomuus tekevät siitä helposti siirrettävän eri alustoille. Eri käyttöjärjestelmät

käsittelevät suoritinkantoja eri tavoin, ja JVM:n tehtävänä on käsitellä näitä käännöksiä, joten käyttöjärjestelmä ja suorittimen arkkitehtuuri ovat kehittäjälle merkityksettömiä. (12; 13.)

Javan katsotaan olevan dynaamisempi kuin C tai C++, koska se on suunniteltu mukautumaan muuttuvaan ympäristöön. Javan monisäikeisen ominaisuuden avulla on mahdollista kirjoittaa ohjelmia, jotka voivat suorittaa useita tehtäviä samanaikaisesti. Monisäikeisyys mahdollistaa mutkattomasti toimivien interaktiivisien sovelluksien luomisen. Javan Just-In-Time-kääntäjät takaavat myös korkean suorituskyvyn. Java-ohjelmat voivat sisältää suuren määrän ajonaikaista informaatiota, jota voidaan käyttää olioiden ajonaikaiseen tunnistukseen ja selvittämiseen. Java pyrkii poistamaan virheiden aiheuttamat ongelmatilanteet kääntöajan virheiden ja ajonaikaisen tarkistuksen avulla. Javan suojattujen ominaisuuksien avulla voi kehittää viruksettomia ja muutenkin peukaloimattomia järjestelmiä, sillä sen todentamistekniikat perustuvat julkisen avaimen salaukseen. (12.)

JVM on abstrakti kone, joka tarjoaa ajonaikaisen ympäristön, jossa Java-tavukoodi voidaan suorittaa. JVM on alustasta riippuvainen, mutta se on saatavana useille eri laitteisto- ja ohjelmistoalustoille. JVM koostuu luokkalatausohjelmasta (ClassLoader), muistialueesta (Memory Area), suoritusmoottorista (Execution Engine) ja natiivimetodirajapinnasta (Native Method Interface). (14.) (Kuva 3.)



Kuva 3. Java-virtuaalikoneen (JVM) arkkitehtuuri (14).

Luokkalatausohjelma on JVM:n alijärjestelmä, jota käytetään luokkatiedostojen lataamiseen. Java-ohjelmaa suoritettaessa, luokkalatausohjelma lataa sen ensin. Javassa on kolme sisäänrakennettua luokkalatausohjelmaa, joista ensimmäinen on esilatausohjelma (Bootstrap ClassLoader), joka on laajennuslatausohjelman (Extension Classloader) superluokka. Esilatausohjelma lataa rt.jar-tiedoston, joka sisältää kaikki Java SE:n luokkatiedostot, kuten java.lang-, java.net-, java.util-, java.io- ja java.sql-luokkapaketit. Laajennuslatausohjelma on esilatausohjelman aliluokka ja järjestelmälatausohjelman (System ClassLoader) yliluokka. Laajennuslatausohjelma lataa .jar-tiedostot $JAVA_HOME/jre/lib/ext-hakemistosta. Järjestelmälatausohjelma lataa luokkatiedostot luokan polulta, joka on oletusarvoisesti asetettu nykyiseen hakemistoon. (14.)

Muistialueeseen kuuluu luokkametodialue (Class Area), keko (Heap), pino (Stack), ohjelmalaskurirekisteri (PC Register) ja natiivimetodipino (Native Method Stack). (Kuva 3.) Luokkametodialue tallentaa luokkakohtaiset rakenteet, kuten

ajonaikaisen vakiovarannon, alue- ja metodidatan sekä metodikoodin. Keko on olioille varattu ajonaikainen data-alue. Pino tallentaa kehykset ja pitää yllä paikallisia muuttujia sekä osittaisia tuloksia, ja näin se on osa metodin kutsua ja paluuta. Uusi kehys luodaan aina, kun metodia kutsutaan, ja se tuhoutuu, kun sen metodin kutsu on valmis. Jokaisella säikeellä on oma JVM-pino, joka on luotu samanaikaisesti säikeen kanssa. Ohjelmalaskurirekisteri sisältää senhetkisen suoritettavan Java-virtuaalikoneen osoitteen. Natiivimetodipino sisältää kaikki sovelluksessa käytetyt natiivimetodit. (14.)

Suoritusmoottori koostuu virtuaalisesta prosessorista, ohjelmatulkista ja ajonaikaisesta kääntäjästä, joiden tehtävänä on yksinkertaisesti lukea tavukoodi ja suorittaa sen antamat käskyt. Natiivimetodirajapinta on kehys, joka tarjoaa käyttöliittymän kommunikoimaan toisen sovelluksen kanssa, joka on kirjoitettu jollain muulla kielellä, esimerkiksi C-kielellä. Java käyttää tätä kehystä tulosteen lähettämiseen konsolille ja on vuorovaikutuksessa käyttöjärjestelmän kirjastojen kanssa. (14.)

2.4 Kotlin-ohjelmointikieli

Tärkeä piirre Android-sovellusympäristön kannalta on, että sovellukset on alun alkaen ohjelmoitu Java-ohjelmointikielellä. Nykyään on mahdollista ohjelmoida ne myös suhteellisen uudella Googlen kehittämällä Kotlin-ohjelmointikielellä. Se on JetBrainsin vuonna 2011 julkaisema ohjelmointikieli, joka on suunniteltu toimimaan yhdessä Javan kanssa. Tarkoituksena oli lisätä moderneja ominaisuuksia Java-mobiilikehitykseen. Kotlin on avoimen lähdekoodin staattisesti kirjoitettu ohjelmointikieli, joka perustuu Java-virtuaalikoneeseen (JVM). JVM:n avulla tietokone voi suorittaa Java-ohjelmia sekä muilla ohjelmointikielillä kirjoitettuja ohjelmia, jotka voidaan kääntää Java-tavukoodiksi. Google päätti vuonna 2017 tukevansa Kotlinia Android-kehitysympäristössään, ja Android Studio 3.0:n julkaisun jälkeen se onkin sisällytetty vaihtoehtona tavalliselle Java-kääntäjälle. Se on integroitu saumattomasti Android Studioon, ja sitä voi käyttää myös projekteissa yhdessä Java-luokkien kanssa. (16; 17; 18.)

Perustamisestaan lähtien Kotlin on kehittynyt jatkuvasti paitsi kielenä myös kokonaisena ohjelmistoekosysteeminä. Vuonna 2019 Google ilmoitti, että Kotlin on suositeltava kieli Android-sovellusten kehittämiseen. Sillä on suuri tuki ja paljon yhteisöä, joka kasvaa kaikkialla maailmassa, ja sitä käytetään aktiivisesti monissa yrityksissä sovellusten kehittämiseen. Googlen mukaan yli 60 % Play Kaupan 1 000 suosituimmasta sovelluksesta käyttää Kotlinia. Tämä puolestaan osoittaa sen, että ohjelmointikieli on erittäin käytännöllinen ja toimiva ratkaisu Android-kehityksessä. (18.)

Kotlinin käyttämisestä Javan sijaan kehityksessä on useita hyötyjä. Ensinnäkin se on selkeämmin luettavaa kuin Java ja vaatii myös vähemmän koodia, mikä käytännössä tarkoittaa, että sen tuottamiseen kuluu vähemmän aikaa. Vähemmällä koodilla ja selkeämmällä luettavuudella on myös se etu, että ne vähentävät mahdollisia virheitä. Kotlin tukee nykyaikaisia ohjelmointikonsepteja, kuten laajennettavia funktioita, korkeamman tason funktioita ja delegaatteja. Sitä voi kuitenkin käyttää Android-kehityksen lisäksi myös palvelinpään ratkaisuissa sekä iOS- ja verkkosovellusten kehittämiseen. (17; 18.)

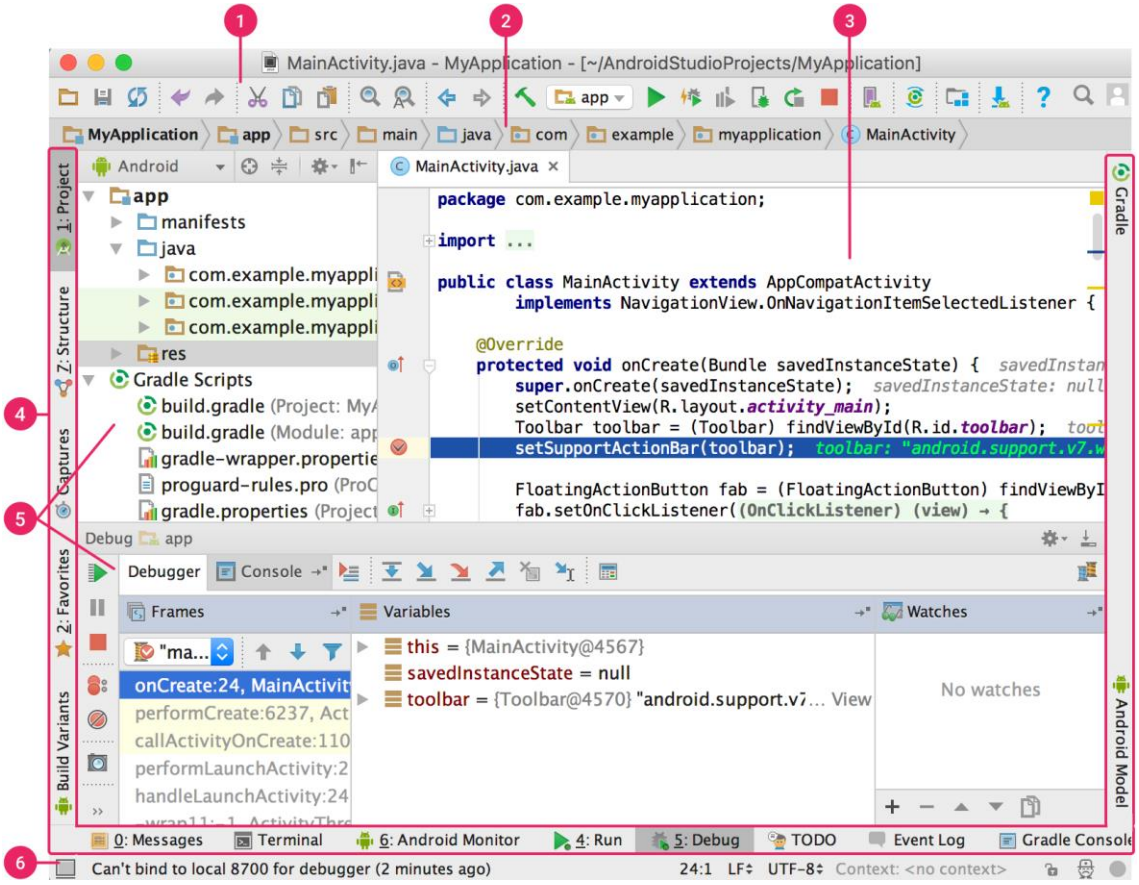
3 Ohjelmointiympäristö ja -rajapinta

3.1 Android Studio -ohjelmointiympäristö

Ennen kuin Android Studiosta tuli Android-kehittäjien pääasiallinen ohjelmointiympäristö, useimmat kehittäjät käyttivät Eclipseä tai muita vastaavanlaisia alustoja Android-sovellustensa luomiseen. Vuosien ajan nämä ohjelmistoympäristöt aiheuttivat paljon ongelmia kehittäjille jatkuvien virheilmoitusten takia, jotka viivästyttivät sovellusten rakentamisprosessia. Android Studio on integroitu kehitysympäristö (IDE) Android-sovelluskehitykselle ja se perustuu IntelliJ IDEA -ohjelmaan. IntelliJ:n tehokkaan koodieditorin ja kehittäjätyökalujen lisäksi Android Studio tarjoaa ominaisuuksia, jotka parantavat sovelluksen rakentamisen tuottavuutta, kuten erittäin suorituskykyisen ja joustavan Gradle-pohjaisen rakennusjärjestelmän sekä yhtenäisen ympäristön, joka mahdollistaa kehittämisen kaikille Android-laitteille. Android Studio mahdollistaa myös nopean ja monipuolisen emulaattorin käytön. Sieltä löytyvät laajat testaustyökalut ja kehykset sekä Linttyökalut, joita se käyttää suorituskyvyn ja käytettävyyden parantamiseen sekä version yhteensopivuuden ja muiden ongelmien selvittämiseen. (19; 20.)

Jokainen Android Studiolla tehty projekti sisältää vähintään yhden, mutta usein useamman moduulin, josta löytyvät projektin lähdekooditiedostot ja resurssitiedostot. Moduulityyppejä ovat sovellusmoduulit, kirjastomoduulit ja Google App Enginen moduulit. Oletusarvoisesti Android Studio näyttää projektitiedosto projektinäkymässä, joka on järjestetty moduuleilla, jotta projektin tärkeimmät lähdetiedostot ovat helposti nähtävillä ja nopeasti käytettävissä. Jokainen sovellusmoduuli sisältää manifests-, java- ja res-kansiot, joista ensimmäisenä mainittu sisältää Manifest.xml-tiedoston, java-kansio sisältää Java-lähdekooditiedostot, mukaan lukien JUnit-testikoodin, ja res-kansio sisältää puolestaan kaikki ei-koodiresurssit, kuten XML-asettelut, käyttöliittymämerkkijonot ja bittikarttakuvat. (20.)

Projektitiedostojen näkymää on mahdollista muokata sovelluskehityksen tiettyihin näkökohtiin, jotta niiden käyttäminen työskennellessä olisi sujuvampaa. Esimerkiksi projektin Ongelmat-näkymän valitseminen näyttää linkit lähdetiedostoihin, jotka sisältävät tunnistetut koodaus- ja syntaksivirheet, kuten asettelutiedostosta puuttuvan XML-elementin sulkutunnisteen. Android Studion päänäkymä koostuu useista eri palkeista ja ikkunoista. (Kuva 4.) Päänäkymän voi haluttaessa järjestää antamaan enemmän näyttötilaa piilottamalla tai siirtämällä työkalupalkkeja ja -ikkunoita. Useimpia ohjelmointiympäristöominaisuuksia voi käyttää myös pikanäppäimillä. Valmiiden perspektiivien käyttämisen sijaan Android Studio seuraa kontekstia ja tuo automaattisesti esiin asiaankuuluvat työkaluikkunat. Oletusarvoisesti yleisimmin käytetyt työkaluikkunat kiinnitetään sovellusikkunan reunojen työkalupalkkiin. (20.)



Kuva 4. Android Studion päänäkymä. 1. Työkalupalkki 2. Navigointipalkki 3. Editori-ikkuna 4. Työkaluikkunapalkki 5. Työkaluikkunat 6. Statuspalkki. (20.)

- Työkalupalkin avulla voi suorittaa monenlaisia toimintoja, mukaan lukien sovelluksen ajaminen ja työkalujen käynnistäminen.
- Navigointipalkin avulla voi navigoida projektissa ja avata tiedostoja muokkausta varten. Se esimerkiksi tarjoaa tiiviimmän näkymän projektin rakenteesta Projektinäkymässä.
- Editori-ikkunassa luodaan ja muokataan koodia. Editori vaihtuu editoitavan tiedostotyypin mukaan. Esimerkiksi XML-asettelutiedostoa tarkasteltaessa editori-ikkuna näyttää asettelueditorin.
- Työkalu-ikkunapalkki on sijoitettu ohjelmistoympäristön ikkunan ulkopuolelle, ja se sisältää painikkeet, joiden avulla voi laajentaa tai tiivistää yksittäisiä työkaluikkunoita.
- Työkaluikkunoista pääsee tiettyihin tehtäviin, kuten projektinhallintaan, hakukoneisiin, versionhallintaan. Niitä voi haluttaessa laajentaa ja tiivistää pois näkymästä käytettävyyden helpottamiseksi.
- Statuspalkki näyttää projektin ja itse ohjelmointiympäristön tilan sekä kaikki varoitukset ja muut ilmoitukset. (20.)

Android Studio tukee useita versionhallintajärjestelmiä, mukaan lukien Git, GitHub, CVS, Mercurial, Subversion ja Google Cloud Source Repositories. Käyttäjä voi valita mieleisensä versionhallintajärjestelmätuen halutulle versionhallintajärjestelmälle, luoda arkiston, tuoda uudet tiedostot versionhallintaan ja tehdä muita versionhallintatoimenpiteitä. Versionhallintajärjestelmävalikko näyttää useita versionhallintavaihtoehtoja valitun järjestelmän perusteella. Android Studio käyttää Gradlea rakennusjärjestelmän perustana, ja laajennukseen Gradle tarjoaa enemmän kohdennettuja ominaisuuksia. (20.)

Gradle toimii integroituna työkaluna Android Studio -valikossa ja riippumatta komentorivistä. Rakennusjärjestelmän avulla voi mukauttaa, konfiguroida ja laajentaa rakentamisprosessia. Sillä voi myös luoda sovellukselle useita sovellusohjelmien pakettitiedostoja eri ominaisuuksilla käyttämällä samaa projektia sekä samoja moduuleja. Järjestelmän ominaisuuksiin kuuluu myös koodin ja resurssien uudelleenkäyttö lähdekoodeissa. Käyttämällä Gradlen joustavuutta voi saavuttaa kaikki nämä asiat muuttamatta sovelluksen ydintiedostoja. (20.)

Android Studion rakennustiedostot ovat nimeltään build.gradle ja niissä määritetään projektin riippuvuudet. Moduuliriippuvuudet, etäbinääririippuvuudet ja paikalliset binääririippuvuudet voidaan ilmoittaa build.gradle-tiedostossa. Gradle

huolehtii riippuvuuksien löytämisestä ja niiden saatavuudesta. Jokaisessa projektissa on yksi ylätason rakennustiedosto koko projektille ja erilliset moduulitason rakennustiedostot kullekin moduulille. Kun olemassa oleva projekti tuodaan, Android Studio luo tarvittavat rakennustiedostot automaattisesti. Rakennusjärjestelmä voi auttaa luomaan saman sovelluksen eri versioita yhdestä projektista. Tämä on hyödyllistä esimerkiksi silloin, kun halutaan tehdä sekä ilmainen että maksullinen versio. (20.)

Android Studion sisäiset virheenkorjaukset ja suorituskyvyn analysointityökalut auttavat kehittäjää virheenkorjauksessa ja koodin suorituskyvyn parantamisessa. Sisäisen virheenkorjauksen avulla voi parantaa koodin läpivientejä virheenkorjausnäkymässä viitteiden, lausekkeiden ja muuttujien arvojen sisäisellä tarkistuksella. Sisäisiin virheenkorjaustietoihin kuuluvat muuttujien arvot, olioiden viitteet, metodien palautusarvot, lambdalausekkeet ja työkaluvinkkien luomat arvot. Android Studiossa on suorituskykyprofiileja, joiden avulla on mahdollista seurata sovelluksen muistia ja suorittimen käyttöä helpommin, löytää vapautettuja objekteja, paikantaa muistivuotoja, optimoida grafiikan suorituskykyä ja analysoida verkkopyyntöjä. Ohjelmistokehityspaketin työkalut, kuten Systrace ja logcat, tuottavat suorituskykyä ja virheenkorjaustietoja yksityiskohtaista sovellusanalyysia varten. (20.)

Ohjelmaa kääntäessä ohjelmointiympäristö suorittaa automaattisesti määritetyt Lint- sekä muut ohjelmistoympäristötarkastukset, jotka auttavat tunnistamaan ja korjaamaan koodin rakenteelliseen laatuun liittyvät ongelmat. Lint-työkalu tarkistaa projektin lähdetiedostoista mahdolliset virheet ja optimointiparannukset ja käy läpi niiden oikeellisuuden, turvallisuuden, suorituskyvyn, käytettävyyden, saavutettavuuden ja kansainvälisyyden. Lint-tarkastusten lisäksi Android Studio suorittaa myös IntelliJ-kooditarkastuksia ja validoi huomautusilmoitukset koodauksen työnkulun virtaviivaistamiseksi. Se tukee muuttujien, parametrien ja palautusarvojen huomautusilmoituksia virheiden, kuten null-osoittimen poikkeuksien ja resurssityyppiristiriitojen, havaitsemiseksi. Android Studiolla sovellusta rakentaessa ja ajaessa kehittäjä voi tarkastella ADB-tulostetta ja laitelokiin kirjoitettuja ilmoituksia Logcat-ikkunassa. (20.)

3.2 Apache POI -rajapinta

Apache POI on Apache Software Foundationin kehittämä ohjelmointirajapinta, jonka avulla ohjelmoijat voivat luoda ja muokata MS Office -tiedostoja Java-ohjelmien avulla. Apache POI:n avoimen lähdekoodin kirjasto sisältää valmiita luokkia ja metodeja, jotka helpottavat ohjelmistojen välistä vuorovaikutusta dekoodaamalla käyttäjän syötetietoja MS Office -asiakirjoihin. Sen vanhemmat versiot tukevat vain binääritiedostomuotoja, ja versiosta 3.5 eteenpäin tuettuina ovat myös MS Officen Office Open XML -tiedostomuodot. Uusimmat versiot sisältävät kaikkien MS Office -sovellusten OLE 2.0 Compound document -tiedostomuotojen käsittelemiseen tarvittavat luokat ja metodit. (21.) Seuraavassa on luettelo Apache POI -ohjelmistorajapinnan komponenteista:

- POIFS (Poor Obfuscation Implementation File System) - Tämä komponentti on kaikkien muiden POI-elementtien perustekijä. Sitä käytetään eri tiedostojen lukemiseen.
- HSSF (Horrible Spreadsheet Format) - Tätä käytetään MS Excel -tiedostojen XLS-tiedostomuodon lukemiseen ja kirjoittamiseen.
- XSSF (XML Spreadsheet Format) - Tätä käytetään MS Excel -tiedostojen XLSX-tiedostomuodossa.
- HPSF (Horrible Property Set Format) - Tätä käytetään MS Office -tiedostojen ominaisuusjoukkojen dekoodaamiseen.
- HWPF (Horrible Word Processor Format) - Tätä käytetään MS Word -tiedostojen DOC-muotojen lukemiseen ja kirjoittamiseen.
- XWPF (XML Word Processor Format) - Tätä käytetään MS Word -tiedostojen DOCX-tiedostomuodossa.
- HSLF (Horrible Slide Layout Format) - Tätä käytetään PowerPoint-esitysten lukemiseen, luomiseen ja muokkaamiseen.
- HDGF (Horrible DiaGram Format) - Tämä sisältää luokkia ja metodeja MS Visio -binaaritiedostoille.
- HPBF (Horrible PuBlisher Format) - Tätä käytetään MS Publisher -tiedostojen lukemiseen ja kirjoittamiseen.
- HSMF (Horrible Stupid Mail Format) - Tätä käytetään MS Outlook MSG -tiedostojen lukemiseen ja kirjoittamiseen.
- DDF (Dreadful Drawing Format) – Tätä käytetään MS Drawing -tiedostomuodon dekoodaamiseksi. (21.)

3.3 Järjestelmätyökalut Gradle ja Maven

Lähdekoodin käyttöä helpottamaan on luotu käännösautomaatiojärjestelmätyökaluja, joiden tehtävänä on lähdekoodin kääntäminen ohjelmiin ja kirjastoihin, binääritiedostojen pakkaaminen paketinhallintajärjestelmään sekä testaamisen automatisoiminen. Niitä käytetään ohjelmistoissa luomaan riippuvuuksia kolmannen osapuolen moduuleihin ja käännösjärjestykseen sekä kuvaamaan tarvittavia laajennuksia. Metodeille, kuten pakkaamiselle ja kääntämiselle, on ennalta määritetyt tavoitteet. Java-pohjaisissa ohjelmissa yleisessä käytössä ovat Apache Maven, tai yleisimmin lyhennettynä pelkkä Maven, sekä Gradle, joka perustuu Apache Antin ja Mavenin käsitteisiin ja käyttää Apache Groovy -pohjaista täsmäkieltä (DSL) Mavenin käyttämän XML-tiedostomuodon sijaan. (22.)

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.9</version>
</dependency>
```

```
dependencies {
    implementation group: 'org.apache.poi', name: 'poi', version: '3.9'
}
```

Esimerkkikoodi 1. Ylempänä Apache POI -riippuvuuden käyttäminen Mavenin käyttämällä XML-kielellä ja sen alapuolella sama esitettynä Gradlen käyttämällä täsmäkielellä.

Sekä Gradle että Maven käyttävät jonkinlaista rinnakkaista projektin rakentamiskäytäntöä ja rinnakkaista riippuvuusresoluutiota. Gradlen suurimmat edut ovat sen mekanismit, jotka välttävät ylimääräistä rakentamistyötä ja helpottavat projektin laajentamista. Gradlea tulisi käyttää, kun joustavuus, helppokäyttöisyys, nopeus ja projektin kasvavat rakenteet ovat tärkeitä. Mavenia sen sijaan tulisi käyttää, kun riippuvuus, hallinta, modulaarisuus, johdonmukaisuus ovat tärkeämpiä. Seuraavassa on lueteltuna kolme Gradlen parasta suorituskykyominaisuutta, jotka tekevät siitä monissa tapauksissa jopa moninkertaisesti nopeamman kuin Maven. (22; 23.)

- Gradlen asteittainen rakennustuki välttää työtä seuraamalla tehtävien syöttöä ja tuloa, ajamalla vain tarvittavan ja käsittelemällä vain tiedostoja, jotka ovat muuttuneet.
- Gradlen rakentamisvälimuisti säästää aikaa käyttämällä uudelleen muiden rakentamiskertojen tuottamia tuotoksia ja mahdollistaa muuttumattomien syöttöjen hakemisen välimuistista.
- Gradle Daemon parantaa rakentamiskertojen nopeutta käyttämällä aiempien rakennusten laskelmia uudelleen. (23.)

4 Työajankirjaussovellus

Tein esimerkkisovelluksena työajankirjaussovelluksen, jonka tarkoitus on laskea automaattisesti työtunnit ja rekisteröidä se suoraan Excel-tiedostoiksi. Sovellus ei tule suoraan minkään yrityksen käyttöön, mutta tarkoituksena oli kuitenkin luoda käyttökelpoinen ratkaisu, jota on mahdollista soveltaa myös yritystarpeisiin. Tärkeimpänä lähtökohtana oli selvittää, kuinka hyvin Office-tiedostojen luominen ja editoiminen ohjelmoinnin kannalta soveltuvat mobiilisovelluskäyttöön. Esimerkkisovellus on Android Studiolla kehitetty Apache POI -rajapintaa käyttävä automatisoitu työajankirjausjärjestelmä. Työ on toteutettu kokonaan Kotlin-ohjelmointikielellä.

4.1 Sovelluksen rakenne

Esimerkkisovellus koostuu yhdestä aktiviteetista ja neljästä fragmentista. Aktiviteetti toimii sovelluksen pohjana ja sen tehtävänä on vastata sovelluksen sisäisestä navigoinnista ja toimia yhteydessä rajapinnan kanssa.

Navigointi toteutettiin valikon avulla ja navigointi tapahtuu käytettävien fragmenttien välillä. Ensimmäisen fragmentin tehtävänä on näyttää käyttäjälle kaikki luodut Excel-tiedostot ja mahdollistaa näiden tiedostojen avaaminen, lähettäminen ja poistaminen. Tiedosto lähetetään kolmannen osapuolen sovelluksella, esimerkiksi pilvipalvelun, pikaviestintäsovelluksen tai sähköpostin välityksellä.

Toisen fragmentin tehtävänä on laskea työtunnit ja välittää muut tarvittavat tiedot Aktiviteetille. Kolmannen fragmentin tehtävänä on toimia vapaamuotoisena työajan kirjaamisen välikappaleena, jossa käyttäjä ilmoittaa itse päivämäärän sekä alkamis- ja päättymisajat. Neljännen fragmentin tehtävänä on toimia poissaoloajan kirjaamisen helpottajana, jolla mahdollistetaan useamman peräkkäisen arkipäivän kirjaaminen samalle työnumerolle yhdellä kertaa.

Rajapinnan kanssa kommunikointi tapahtuu aktiviteetista käsin rajapinnan olioluokkia käyttäen. Apache POI:n ydinluokkia ovat työkirja, taulukko, rivi, solu, solumuotoilu, väri, fontti, hyperlinkki, luomisavustaja ja tulostusasetus. Käytin tässä esimerkkisovelluksessa näistä olioluokista seitsemää ensimmäisenä lueteltua. Työkirja eli HSSFWorkbook-tyypitetty olio on luotava, jotta muut oliot saadaan linkitettyä tähän tiettyyn Excel-tiedostoon. Työkirjaoliolla voidaan luoda muiden luokkien olioita. (Esimerkkikoodi 2.)

```
private fun createTaskRow(wb: Workbook, sheet: Sheet, row: Row) {
    cellStyleTask = wb.createCellStyle()
    cellStyleTask.setupCellStyleTask(wb)

    sheet.setColumnWidth(0, 15 * 200)
    sheet.setColumnWidth(1, 15 * 200)
    sheet.setColumnWidth(3, 15 * 500)

    listOfTasks.forEach {
        val c = row.createCell(listOfTasks.indexOf(it))
        c.setCellValue(it)
        c.cellStyle = cellStyleTask
    }
}
```

Esimerkkikoodi 2. Esimerkkikoodin funktiossa luodaan rivi solumuotoilulla ja mm. solumuotoiluolio, jolle annetaan määritteet laajennettavalla funktiolla.

Olioluokat määrittelevät attribuutit sekä yhteydet muiden olioiden välillä, esimerkiksi rivin ja solun väliset suhteet. Apache POI:n olioluokille annettaan attribuutteina esimerkiksi lukuarvoja, kuten taulukko-olion rivin korkeus ja sarakkeen leveys. Soluoliolle voidaan antaa attribuutiksi soluarvo, niin kuin tavallisestikin Excel-taulukkoa käytettäessä, ja se voi olla esimerkiksi tekstiä, numero, päivämäärä tai laskukaava. Attribuutit voivat olla myös olioita, kuten solumuotoiluoliolle annettava fonttiattribuutti on fonttiolio. (Esimerkkikoodi 3.)

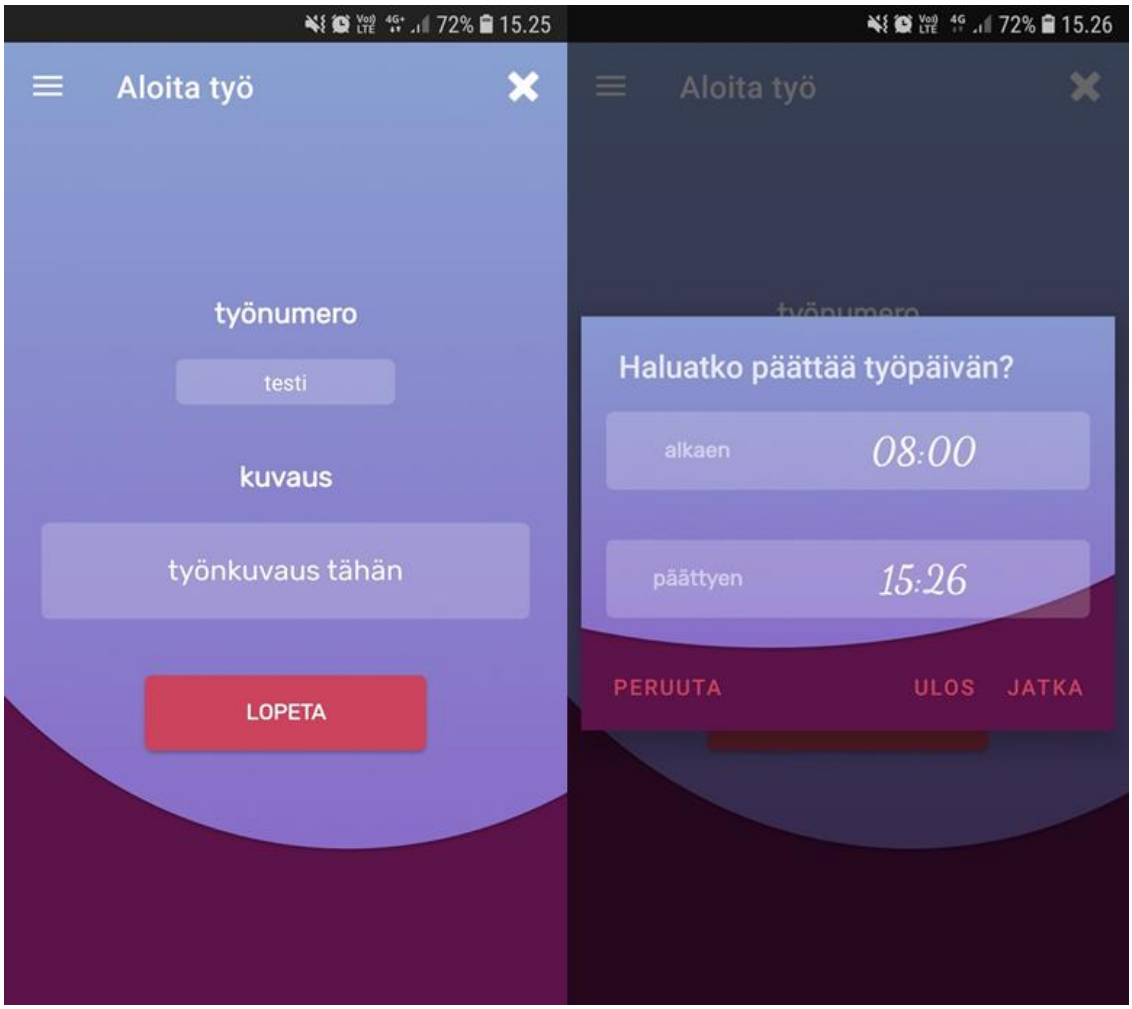
```
private fun CellStyle.setupCellStyleTask(wb: Workbook) {
    fillForegroundColor = HSSFColor.GREEN.index
    fillPattern = HSSFCellStyle.SOLID_FOREGROUND
    alignment = ALIGN_CENTER
    verticalAlignment = VERTICAL_CENTER
    val font = wb.createFont()
    font.color = HSSFColor.WHITE.index
    font.boldweight = HSSFFont.BOLDWEIGHT_BOLD
    setFont(font)
```

```
    wrapText = true
}
```

Esimerkkikoodi 3. Esimerkki laajennettavasta funktiosta, jossa HSSFCellStyle-luokkatyypitetty olio kutsuu useampaa attribuuttiaan ja niille annetaan ennalta määritetyt arvot.

4.2 Sovelluksen toiminnallisuus

Työajankirjaussovelluksen käyttäjä aloittaa työn painamalla painiketta ”Aloita”, jolloin näytölle ilmestyy kirjoituskentät työnumerolle ja kuvaukselle. Käyttäjän ei tarvitse tässä vaiheessa tietää työnumeroa tai työtehtäviään. Työnumero ja työnkuvaus ovat kuitenkin pakolliset tiedot, vasta silloin, kun käyttäjä haluaa lopettaa kyseiselle työnumerolle tehdyn työn painamalla painiketta ”Lopeta”. Tällöin sovellus laskee työn aloitus- ja lopetusajan perusteella työhön käytetyn ajan minuutin tarkkuudella ja rekisteröi työhön käytetyn ajan lisäksi aloitus- ja lopetusajan suoraan Excel-tiedostoon. Näiden lisäksi sovellus rekisteröi automaattisesti myös viikonpäivän ja päivämäärän työtehtävälle, eikä käyttäjän tarvitse näitä tietoja missään vaiheessa ilmoittaa. Sovellus laskee myös tehtyjen työtuntien kokonaismäärän ja myös työnumerokohtaisesti eriteltynä. Rekisteröinnissä otetaan huomioon kirjaamiseen liittyvät mahdolliset inhimilliset unohdukset, niinpä vielä ennen rekisteröintiä käyttäjän on mahdollista muuttaa työtehtävän alkamis- ja päättymisaikaa. (Kuva 5.)

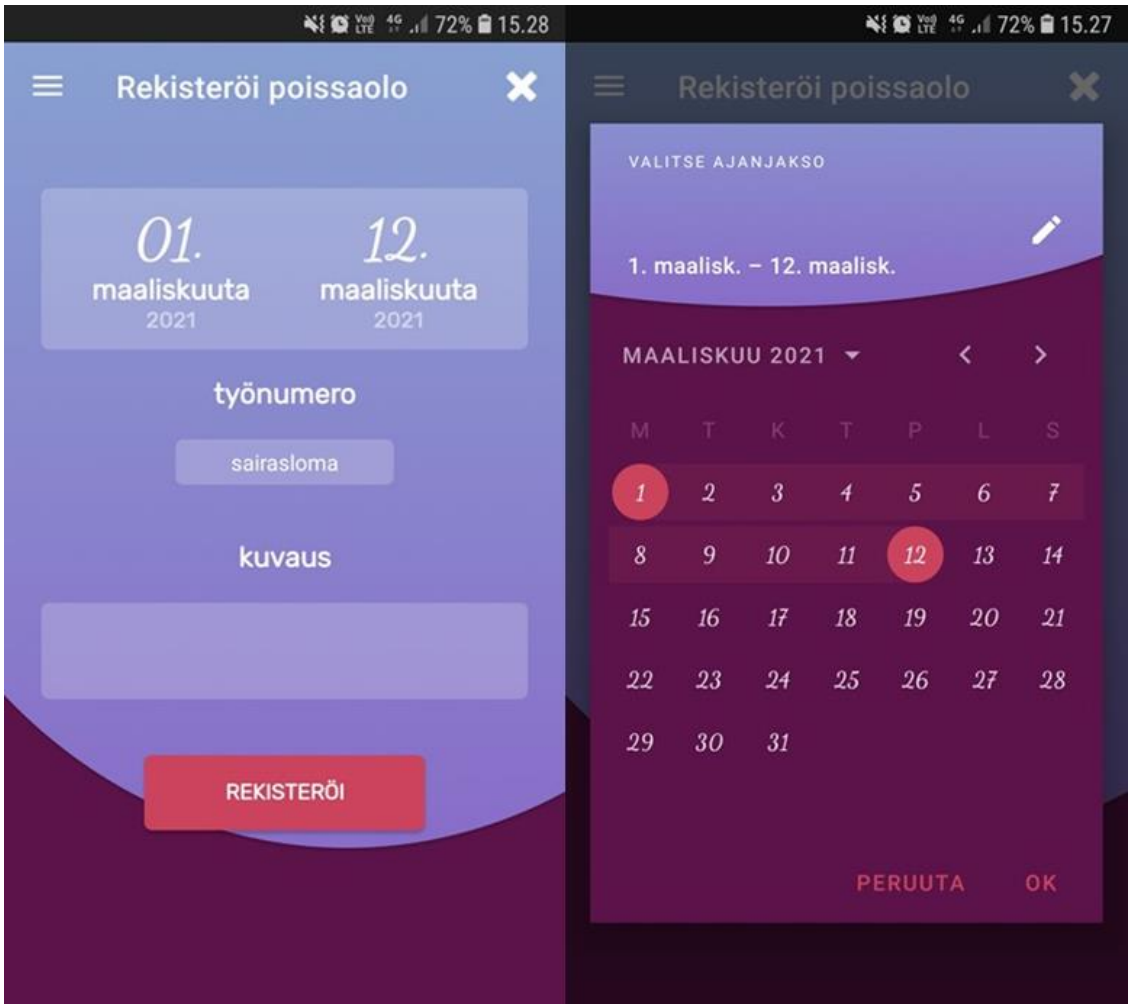


Kuva 5. Vasemmalla näkymä, jossa on tekstikentät työnumerolle ja työnkuvaukselle sekä ”Lopeta”-painike. Oikealla dialogi, jossa kysytään, haluaako käyttäjä jatkaa työpäiväänsä vai lopettaa sen, sekä mahdollisuus muokata vielä kellonaikoja.

Sovellus lähettää ilmoituksen ja kysyy 15 minuuttia ennen työpäivän päättymistä: haluaako käyttäjä jatkaa työpäiväänsä. Jos käyttäjä ei vastaa tähän ilmoitukseen tai valitsee haluavansa jatkaa työpäivää, niin työpäivä rekisteröityy automaattisesti päättyneeksi ajallaan seuraavan kerran, kun käyttäjä avaa sovelluksen. Sovellus kysyy tätä vain kerran, niinpä käyttäjän pitää muistaa lopettaa työpäivä työpäivän pidentyessä. Tämäkin on mahdollista korjata jälkikäteen. Aloittamisaikaa on myös mahdollista muokata jälkikäteen, jos käyttäjä unohtaa rekisteröidä työn aloittamisajankohdan.

Työnkuvausta voi kirjoittaa ennen työtä tai tehdyn työn jälkeen, tai aina, kun työt joutuvat syystä tai toisesta odottamaan. Tehdyt työtehtävät kannattaa kirjoittaa työnkuvaukseen tarkoitettuun kenttään aina, kun on saanut ne valmiiksi tai kun on keskeyttänyt työnteon esimerkiksi lounastaukoa varten. tällöin aiemmin tehdyt työt ovat yhä tuoreessa muistissa eikä käyttäjän tarvitse enää myöhemmin muistella, mitä kaikkia työtehtäviä on tullut tehtyä.

Käyttäjän on mahdollista merkitä tehdyt työtunnit jälkikäteen. Se tapahtuu niin, että käyttäjä asettaa työnumeron ja työnkuvauksen lisäksi päivämäärän sekä aloitus- ja lopetusajan. Käyttäjällä on myös mahdollisuus merkitä oma-ajat valitsemalla alkamis- ja päättymispäivä. Tällöin sovellus kirjaa automaattisesti työpäivän mittaiset poissaolot ko. työnumerolle, esimerkiksi sairaspoissaolo. (Kuva 6.)



Kuva 6. Vasemmalla näkymä, jossa on ajanjakson valitsin sekä tekstikentät työnumerolle ja työnkuvaukselle. Oikealla kalenteri ajanjakson valitsemista varten.

Sovellus luo automaattisesti jokaiselle kuukaudelle oman tiedoston. Sovelluksessa on myös toimintoja, jotka vaativat kolmannen osapuolen sovelluksen. Esimerkiksi Excel-tiedostojen avaamiseen tarvitaan Excel-sovellus, jotta työtuntitaulukoiden tarkastelu ja jälkikäteen muokkaaminen Excel-tiedostosta käsin on mahdollista. MS Officen sovellukset ovat esiasennettuja kaikkiin Android-puhelimiin. Työtuntitaulukot voidaan myös lähettää kolmannen osapuolen sovelluksella, esimerkiksi sähköpostilla, pilvipalvelulla tai pikaviestimellä.

5 Yhteenveto

Apache POI on avoimen lähdekoodin Java-kirjasto, jonka avulla voidaan luoda ja käsitellä erilaisia Microsoft Officen tiedostomuotoja. Se tarjoaa tiedostojen käsittelyyn ohjelmointiparadigman, joka mahdollistaa raskaan laskennan ja kehittäjälle helppokäyttöisyyden. Se soveltuu suurille tiedostoille ja vaatii myös vähän muistia. Se tarjoaa lähes jokaista Excelin ominaisuutta vastaavat selkeästi jaotellut luokat, joille on helppo antaa halutut parametrit. Ainoastaan sarakkeille ei ole luotu omaa luokkaa, sillä solun paikka määritellään rivin eli HSSFRow-luokan funktioilla, jolla luodaan solua vastaava Cell-olio tämän rivin sarakkeeseen. Vaikka yksinkertaiseenkin Excel-tiedostoon on luotava useita olioita ja annettava useitakin eri attribuutteja, on tiedoston muokkaus tehty ohjelmoijalle todella selkeäksi.

Excel-tiedostoon kirjoitettaessa voidaan samalla luoda dynaamisesti Apache POI:n luokkien olioita, kuten Sheet, Row ja Cell. Myös solumuotoiluun tarkoitetun HSSFCellStyle-luokan solumuotoiluolio on mahdollista luoda dynaamisesti jokaiselle solulle erikseen Excel-tiedostoon kirjoittamisen aikana, mutta tällöin annetaan mahdollisuus solumuotoilun epäonnistumiseen, varsinkin silloin, kun luodaan useampi solu kerralla. Siirrettäessä tätä dataa Excel-tiedostoon solumuotoilun voi huomata puuttuvan jokaisella uudella solulla. Kun halutaan varmistaa solumuotoilun onnistuminen, on solumuotoiluoliot luotava kerran ennen niiden asettamista usean solun attribuutiksi. Tästä huolimatta Apache POI tarjoaa erinomaisen tuen Excelin lisäominaisuuksille, kuten solumuotoiluun ja laskukaavojen käsittelyyn. Soluille annettavat laskukaava-attribuutit ovat Excelin omiin laskukaavoihin pohjautuvia eivätkä juurikaan niistä eroa.

Kotlin soveltuu erinomaisesti Android-kehitykseen. Laajennettavien funktioiden ansiosta olioille annettavien attribuuttien asettaminen vähentää koodia merkittävästi. Muita hyviä ominaisuuksia ovat null safety ja smart cast, jotka sujuvoittavat osaltaan koodin kirjoittamista.

Insinöörityössä tehty esimerkkisovellus oli yhdellä työntekijällä useamman viikon testikäytössä, ja testikäytöstä saadun palautteen, kehitysehdotusten ja ilmenneiden ongelmien perusteella tein muutamia parannuksia sovellukseen. Suurin osa palautteesta koski sovelluksen käytettävyyttä. Sovellus osoittautui toimivaksi, jopa käyttökelpoiseksi työajankirjausjärjestelmäksi.

Lähteet

1 Microsoft Office. 2019. Verkkoaineisto. Techopedia. <https://www.techopedia.com/definition/20737/microsoft-office>. 30.8.2019. Luettu 2.4.2020.

2 Microsoft Office Application. Verkkoaineisto. Educba. <https://www.educba.com/microsoft-office-application/>. Luettu 2.4.2020.

3 Kedmey, Dan. 2015. Why Your Next Tablet Might Come With Word and Excel. Verkkoaineisto. < https://time.com/3896695/microsoft-office-android/>. 26.5.2015. Luettu 3.4.2020.

4 Excel. 2020. Verkkoaineisto. Computer Hope. <https://www.computerhope.com/jargon/e/excel.htm>. 30.4.2020. Luettu 4.5.2020.

5 Understanding XLS Vs. XLSX: 7 Major Differences (With Comparison Chart). Verkkoaineisto. Viva Differences. <https://vivadifferences.com/understanding-xls-vs-xlsx/>. Luettu 10.4.2020.

6 .XLS Tiedostopääte. Verkkoaineisto. ReviverSoft. <https://www.reviversoft.com/fi/file-extensions/xls>. Luettu 10.4.2020.

7 Cervantes, Edgar. 2020. What is Android? Here’s everything you need to know. Verkkoaineisto. <https://www.androidauthority.com/what-is-android-328076/>. 10.5.2020. Luettu 13.5.2020.

8 Presentation On Android. 2017. Verkkoaineisto. TeachMission. <https://www.slideshare.net/TeachMission/presentation-on-android-71192538>. 19.1.2017. Luettu 1.7.2020.

9 Platform Architecture. 2020. Verkkoaineisto. Google. <https://developer.android.com/guide/platform>. 7.5.2020. Luettu 6.7.2020.

10 Alcérreca, Jose. 2017. The Android Lifecycle cheat sheet - part III: Fragments. Verkkoaineisto. <https://medium.com/androiddevelopers/the-android-lifecycle-cheat-sheet-part-iii-fragments-afc87d4f37fd>. 5.12.2017. Luettu 3.1.2021.

11 Material Design. Verkkoaineisto. Interaction Design Foundation. <https://www.interaction-design.org/literature/topics/material-design>. Luettu 29.1.2021.

12 Java – Overview. Verkkoaineisto. Tutorialspoint. <https://www.tutorialspoint.com/java/java_overview.htm>. Luettu 18.11.2020.

13 Reilly, David. 2006. Inside Java: The Java Virtual Machine. <http://www.javacoffeebreak.com/articles/inside_java/insidejava-jan99.html>. 5.6.2006. Luettu 25.1.2021.

14 JVM (Java Virtual Machine) Architecture. Verkkoaineisto. Javatpoint. <https://www.javatpoint.com/jvm-java-virtual-machine>. Luettu 18.11.2020.

15 Tyson, Matthew. 2020. What is the JVM? Introducing the Java Virtual Machine. Verkkoaineisto. <https://www.infoworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html>. 17.1.2020. Luettu 20.11.2020.

16 Ableson, Frank. 2018. Introduction to Android development. Verkkoaineisto. <https://developer.ibm.com/technologies/mobile/articles/os-android-devel/>. 30.5.2018. Luettu 4.7.2020.

17 Jackowski, Krzysztof. 2019. Kotlin vs. Java: What To Choose for Your Next Android App. Verkkoaineisto. <https://www.netguru.com/blog/kotlin-java-which-one-you-should-choose-for-your-next-android-app>. 2.8.2019. Luettu 4.11.2020.

18 Using Kotlin for Android Development. Verkkoaineisto. Kotlinlang. <https://kotlinlang.org/docs/reference/android-overview.html>. Luettu 2.11.2020.

19 The Android development evolution across time. 2017. Verkkoaineisto. Galileo University. <http://androiddeveloper.galileo.edu/2017/09/13/android-development-evolution-across-time/>. 13.9.2017. Luettu 17.5.2020.

20 Meet Android Studio. Verkkoaineisto. Google. <https://developer.android.com/studio/intro>. Luettu 15.11.2020.

21 Apache POI - Overview. Verkkoaineisto. Tutorialspoint. <https://www.tutorialspoint.com/apache_poi/apache_poi_overview.htm>. Luettu 20.5.2020.

22 Stringfellow, Angela. 2017. Gradle vs. Maven. Verkkoaineisto. <https://dzone.com/articles/gradle-vs-maven>. 30.6.2017. Luettu 21.5.2020.

23 Gradle vs. Maven. Verkkoaineisto. Educative. <https://www.educative.io/edpresso/gradle-vs-maven>. Luettu 23.6.2020.