Tampereen ammattikorkeakoulu

Kotlin verkkosovelluksen kehityksessä

Ville Kautto

OPINNÄYTETYÖ
Marraskuu 2021

Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Ohjemistotuotanto

KAUTTO, VILLE:
Kotlin verkkosovelluksen kehityksessä

Opinnäytetyö 30 sivua
Marraskuu 2021

Tämä opinnäytetyö tutki modernin verkkokehitykseen käytettyjä työkaluja ja menetelmiä. Työssä esitellään Kotlinia ohjelmointikielenä ja tutkitaan sen soveltumista verkkokehitykseen. Teksti on suunnattu ohjelmoinnin ja verkkokehityksen perusteet tuntevalle henkilölle. Työssä käsitellään verkkokehityksen eri konsepteja ja ne esitellään lukijalle.

Tutkimuksessa tavoitteena oli selvittää, kuinka hyvin Kotlin soveltuu verkkosovellusten kehitykseen ja mitkä ovat sille ominaisia etuja verrattuna muihin verkkokehityksessä käytettyihin ohjelmointikieliin. Tekstin tarkoituksena oli keskittyä tutkimaan Kotlinin ominaisuuksia ja kerätä tietoa sen soveltumisesta verkkosovellusten kehitykseen. Työn teoriaosuudessa Kotlinin ominaisuuksia verrattiin sovelluskehityksessä yleisemmän Javan ominaisuuksiin.

Työssä otettiin huomioon verkkosovellusten kehitykseen liittyviä yleisiä ominaisuuksia ja tarpeita. Työ toteutetaan käyttämällä Java sekä Kotlin -ohjelmointikieliä. Sovellusten koodin tuottamiseen käytetään IntelliJ Idea -sovelluskehitysympäristöä ja Spring boot -verkkokehityskehystä. Sovellusten koodin muutokset tallennettiin käyttämällä Git -versionhallintaohjelmistoa. Työn aikana tuotettu koodi julkaistiin vapaan lähdekoodin sovelluksina GitLab -säilytytysalustalle, jossa sovellusten koodit ja niiden kehityshistoria on vapaasti tarkasteltavissa.

Asiasanat: verkkokehitys, verkkokehys, verkkosovellus

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Software Production

KAUTTO, VILLE:
Kotlin in web application development

Bachelor's thesis 30 pages
November 2021

This thesis inspects the tools and methods used in modern web application development. The text focuses on Kotlin and common problems encountered during web application development. The work's nature is practical and it's text is directed to a person, who is already capable of the basics of programming and web development.

The purpose of the work was to examine how well Kotlin serves as a programming language in the development of a web application and which features are useful in web development. The goal of the study was to examine Kotlin's features and collect information about Kotlin's viability in web application development. In the research Kotlin's features are compared with the more common Java -programming language.

The modern needs for common language features and features specific to web application development were taken account during the research. The practical part of the study was done with Kotlin and Java programming languages. The applications made during the study were made by using IntelliJ IDEA programming enviroment and Spring framework. The code made during the study was also published to GitLab as open-source software where the applications' code and development history can be examined.

Key words: web development, web application framework, web application

SISÄLLYS

LYHENTEET JA TERMIT

HTML	HyperText Markup Language, Verkkosivujen esittämiseen käytetty dokumenttityyppi.
JVM	Java Virtual Machine, Virtuaalikone, joka ajaa Java -ohjelmointikielellä tuotettua koodia
Verkkosovelluskehys	Verkkosovelluksen kehityksessä käytetty työkalu, joka suoraviivaistaa sovellusten tuotantoa
IDE	Integrated Development Environment, Ohjelmien kehitykseen käytetty ohjelmistokokonaisuus
JetBrains	Ohjelmistojen tuotantoon keskittyvä yritys, joka on tuottanut monia IDE -kehitysympäristöjä ja Kotlin -ohjelmointikielen
CLBG	Computer Language Benchmarks Game, ohjelmointikielien suorituskyvyn mittaukseen käytetty vapaan ohjelmiston testiohjelma
Vapaa ohjelmisto	(eng. Free and Open-Source Software) Ilmainen ja avoimen lähdekoodin ohjalmisto, jota käytetään ohjelmointikielien suorituskyvyn mittaamiseen
Null	Tarkoituksellisesti tai oletusarvoisesti tyhjäksi asetettu arvo ohjelmointikielen muuttujassa
Null-safety	Tapa, jolla ”null” -arvojen aiheuttamia ongelmia voidaan välttää
Backend	Sovelluksen osio, joka vastaa käyttöliittymän toimintojen suorittamisesta
Frontend	Sovelluksen osio, joka vastaa sovelluksen käyttöliittymästä
CSS	Cascading Style Sheet, HTML koodin ulkoasun muokkaukseen käytetty kieli
JavaScript	Verkkosivuilla käytetty ohjelmointikieli, joka hallitsee sivuilla näkyvää sisältöä
PoC	Proof of Concept, soveltuvuusselvitys, jossa pyritään selvittämään idean toimintaa käytännössä
REST	Representational State Transfer

Refaktorointi	(eng. Refactor) Koodin muuttaminen ilman alkuperäisen toiminnallisuuden muuttamista

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on analysoida Kotlin -ohjelmointikielen soveltuvuutta verkkosovelluksen kehitykseen käytettävänä ohjelmointikielenä. Kotlinin toiminnallisuutta verrataan verkkokehityksessä yleisemmin käytettyyn Java -ohjelmointikieleen, joka on yksi suosituimmista kielistä verkkosovellusten backend -kehitykseen käytetyistä ohjelmointikielistä.

Opinnäytetyöllä ei ole toimeksiantajaa ja aihe perustuu kirjoittajan omaan mielenkiintoon tutkia vaihtoehtoisia menetelmiä verkkosovellusten kehitykseen.

Opinnäytetyössä perehdyttiin molempiin ohjelmointikieliin, modernia verkkosovelluksen kehitystä ja keskityttiin selvittämään vaihtoehtoisten menetelmien soveltumista ja kannattavuutta verkkosovelluksen kehityksessä. Opinnäytetyö keskittyi tutkimaan Kotlinin käyttöä erityisesti backend -kehityksessä, jossa on yleisemmin on käytössä Java -ohjelmointikieli.

Työn aikana suunniteltiin ja toteutettiin kaksi pienikokoista verkkosovellusta, jotka toimivat vertailukohteina toisillensa ja ovat toiminnallisuudelta identtisiä. Verkkosovellusten backendin toteutuksessa käytettiin kummankin ohjelmointikielen osalta Spring -verkkokehityskehystä sekä frontendin toteutuksessa Angular -kehityskehystä ja TypeScript -ohjelmointikieltä. Verkkosovellusten avulla pyritään havainnollistamaan eroja backendissä käytettävien ohjelmointikielien välillä. Back- ja frontendin teknologioiden valinta perustuu omaan kokemukseen kummankin ohjelmiston osan kohdalla.

2 KOTLIN

2.1 Lyhyt katsaus Kotlinin historiaan

Kotlin on JetBrainsin kehittämä yleiskäyttöön tarkoitettu ohjelmointikieli, joka on suunniteltu toimimaan monilla eri alustoilla. JetBrains on tunnettu ohjelmistokehitykseen keskittyvä yritys ja sen merkittävimmät tuotteet ovat muun muassa eri ohjelmien kehityksessä käytetyt IDE (Integrated Development Environment) -ohjelmistot, kuten PyCharm (Python), IntelliJ IDEA (Java) ja Rider (C#).

JetBrains julkaisi ensimmäisen kerran tietoa Kotlinista vuoden 2011 heinäkuussa nimellä Project Kotlin. JetBrainsin kehitysjohtaja, Dmitry Jemerow, kertoi Kotlinin kehityksen alkaneen yrityksen tarpeisiin sopivan ohjelmointikielen puutteesta. Hänen mukaansa ainoa vastaava vaihtoehto olemassa olevista JVM -virtuaalikonetta käyttävistä ohjelmointikielistä oli Scala, mutta sen käytöstä luovuttiin ohjelmien pitkien kääntöaikojen vuoksi. (Krill, 2011.)

John Rymer on kuvaillut Kotlinin yksinkertaistavan monimutkaisten sovellusten tuottoa ja tuovan lisää työntövoimaa jokaiseen koodiriviin. Hänen mukaansa Kotlin ei kuitenkaan tule tavoittamaan samanlaista suosiota kuin Java, koska moni kokee Javan jo ennaltaan tutuksi ja ei ole sen vuoksi valmis vaihtamaan. Lisäksi hän toteaa oman yrityksensä asiakkaiden standardien olevan sellaiset, jotka estävät Kotlinin käyttöönoton usean vuoden ajaksi. (Krill, 2011.)

Vuoden 2016 helmikuun 15. päivä Andrey Breslav julkaisi uuden JetBrainsin verkkosivuille uuden blogikirjoituksen, jossa kerrottiin Kotlinin ensimmäisen version julkaisusta. Andrew kuvaa blogissa Kotlinin kehitykseen liittyviä keskeisiä arvoja, joita ovat käytännönläheisyys, ongelmien ilmenemistä ehkäisevä tyyppijärjestelmä ja koodin uudelleenkäyttöä hyödyntävä toimintoja. (Breslav, 2016.)

Nykypäivänä kotlinin käyttö on yleisintä Android -käyttöjärjestelmää käyttävissä puhelimien sovelluksissa, koska Google on tukenut kotlinin käyttöä vuodesta

2017 alkaen ja suositellut sen käyttöä Android -sovelluksissa vuodesta 2019 lähtien (Lardinois, F. 2019). Kotlinia on käytetty myös verkkosovellusten backend -kehityksessä, mutta sen käyttö on varsin harvinaista. Kotlinille on myös kehitetty oma JavaScript kirjastonsa, joka mahdollistaa Kotlinin käytön myös verkkosovelluksen frontendin kehityksessä (Kotlin/JS, n.b.a).

2.1.1 Kotlinin suunnitteluperiaatteet

Kotlinin suunnittelussa on koko kehityksen aikana otettu huomioon sen käyttö yrityksen olosuhteissa, koska JetBrainsin alkuperäinen tarve Kotlinin kaltaiselle kielelle oli tarkoitus käyttää yrityskäytössä. Kotlinin alkuperäinen tarkoitus on myös toimia saumattomasti Java -koodin kanssa, joka tekee siihen tutustumisen helpommaksi Javan tuntevien kehittäjien keskuudessa. Kotlin pystyy myös tämän takia hyödyntämään Javalla toteutettuja kirjastoja. (Breslav, A. 2016.)

Saumaton yhteensopivuus Javan kanssa tarkoittaa, että Kotlin ei menetä mitään Javan tarjoamien ominaisuuksien puolesta. Kotlinilla on myös mahdollista käyttää ohjelmiston koodissa Javaa ja Kotlinia samanaikaisesti (Kotlin-Java interop. n.d.). Yhteensopivuus jatkuu myös Javalle tuotetuille kirjastoille, joiden käyttö on mahdollista myös Kotlinissa.

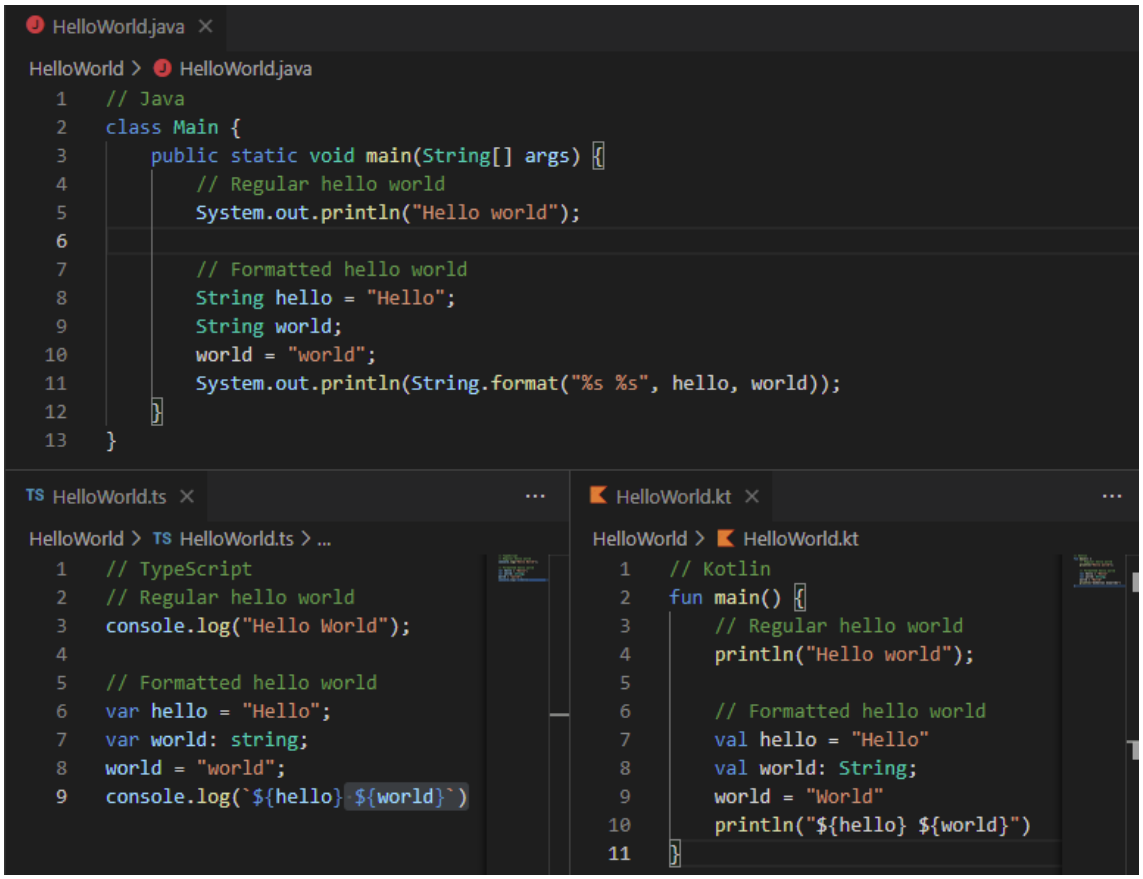
Kotlinin keskeinen idea on helpottaa toiminnallisen koodin tuotantoa, joka saadaan lisättyä automatisoimalla vähäistä toiminnallisuutta sisältäviä toimintoja. Kotlin lisää toiminnallisen koodin määrää, jonka avulla ohjelmiston koodista voidaan saada helpommin hahmotettava kokonaiskuva.

2.2 Kotlinin ominaisuudet

Kotlinin yrittää saavuttaa keskeisen ideansa yhdistämällä objektikohtaisen ohjelmoinnin (eng. Object-Oriented Programming) ja toiminnallisen ohjelmoinnin (eng. Functional programming). Kotlin sai yhdistettyä nämä ideat ja siitä on

kehitetty ohjelmointikieli, jolla voidaan tuottaa erittäin tiivistä ja toiminnallista koodia.

Tämä näkyy erityisen hyvin Kotlinin tiedon tulostamiseen käytetyssä metodissa, joka toimii sekä tavallisessa tulostuksessa että muotoillussa tulostuksessa. Javan tapauksessa muotoilu toteutetaan käyttämällä String kirjaston metodia *format*, joka muotoilee tekstin käyttäjän syöttämien määrittelyiden mukaisesti. Kotlinin ja TypeScriptin tapauksissa muotoilu voidaan suorittaa itse tulostuksen yhteydessä, jolloin erillisen metodin kutsuminen ei ole tarpeellista. (KUVA 1)



KUVA 1: Yläosassa Javalla, vasemmalla TypeScriptillä ja oikealla Kotlinilla toteutettu ”Hello world” -ohjelma

Tyyliltään Kotlin ei muistuta merkittävästi Javaa, vaan se voi muistuttaa enemmän esimerkiksi TypeScriptiä (KUVA 1). Kotlinin toiminta muistuttaa myös huomattavasti TypeScriptin tavasta suosia muuttujien tyyppien asettamista. TypeScriptin tapauksessa tyyppiä ei vaadita, mutta sen käyttö on siitä huolimatta suositeltavaa.

Kotlinin kirjastoihin on myös toteutettu monia tapoja ajaa koodia asynkronisesti, joka mahdollistaa useamman koodin ajamisen samanaikaisesti. Ohjelmoinnissa monisäikeistäminen on yleinen termi, kun puhutaan asynkronisesta koodista. Asynkronisuus on hyödyllistä erityisesti verkkosovellusten tuotannossa, jossa palvelinohjelmisto toimii pääasiassa kuuntelijana ja vastaa tarvittaessa käyttäjän pyyntöihin. Kotlinin asynkronisiin funktioihin kuuluvat esimerkiksi coroutines, callbackit ja futuurit. (Kotlin Coroutines. n.d.)

Kotlinin suunnittelussa on otettu myös huomioon seikkoja, jotka tekevät tavanomaisesti monisäikeistämisestä hankalampaa. Monisäikeistetty toiminta voi jäädä tuhlaamaan resursseja, mikäli toiminnassa ilmenee ongelmia tai virheitä. Kotlinin coroutineScope voi auttaa monisäikeistetyn koodin suorittamisessa, jossa jokainen operaatio pitää suorittaa onnistuneesti loppuun saakka. CoroutineScopen avulla voidaan esimerkiksi välttää pullonkaulojen syntymistä suoritettavassa koodissa. (Elizarov, 2018.)

Myös korkeamman tason funktiot ovat mahdollisia Kotlinissa. Korkean tason funktioiden avulla voidaan luoda ja suorittaa funktioita toistensa sisällä. Korkean tason funktioilla voidaan luoda monikäyttöisiä ja yksinkertaisia operaatioita, joilla voidaan saada kompakteja ratkaisuja monenlaisiin ongelmiin. Esimerkissä oleva korkeamman tason funktio (KUVA 2). (Hariri, 2018 .)

```
val repeatFun: String.(Int) -> String = { times -> this.repeat(times) }
val twoParameters: (String, Int) -> String = repeatFun // OK

fun runTransformation(f: (String, Int) -> String): String {
    return f("hello", 3)
}
val result = runTransformation(repeatFun) // OK
```

KUVA 2: Korkean tason metodi. (https://kotlinlang.org/docs/lambdas.html)

2.3 Kotlinin käytön hyödyt ja haitat

2.3.1 Kotlinin käytön hyödyt

Edellisessä alaluvussa käytiin läpi kotlinin yksittäisiä ominaisuuksia ja tässä alaluvussa keskitytään enemmän Kotlinin hyötyihin modernina ohjelmointikielenä. Kotlin on modernina ohjelmointikielenä suunniteltu vastaamaan suoraan moderneihin tarpeisiin, joihin vanhemmat ovat joutuneet kehittämään aikanansa omanlaiset ratkaisunsa.

Kotlinin kehityksessä yksi sen suurimmista eduista on kielen nuori ikä, koska sen suunnittelussa on otettu huomioon viimeisen parin vuosikymmenten aikana ilmenneitä ongelmia. Kotlinin tarkoitus on jatkaa Javan kaltaista toimintamallia, jossa ohjelmointikieli soveltuisi mahdollisimman monipuolisten ohjelmistojen tuotantoon.

Kotlin suosii datan pitämistä muuttumattomana. Muutettavissa olevien tieto on milloin tahansa muutettavissa ja se voi aiheuttaa koodissa ongelmia, kuten väärän arvon välittämistä pyydettyyn paikkaan. Tiedon pitäminen muuttamattomana helpottaa liikkuvan tiedon käsittelyä ja vähentää koodissa esiintyvien virheiden määrää. (McGregor & Pryce, 2021; Macdonald, 2021.)

Kotlin suosii tyyppien muuttamista täsmentävästi. Kotlinissa tyyppien muuntaminen tyypistä toiseen pitää tapahtua selkeiden tyyppien kääntämiseen tarkoitettujen metodien avulla. Javassa esimerkiksi integer -tyyppisiä arvoja voidaan asettaa suoraan long tyyppisiin arvoihin ilman erillistä muuntamista. Tyyppien muuttaminen suoraan toisen tyyppisiksi arvoiksi muuttaa ohjelman valvonnan kulkua (eng. control flow), joka voi haitata tyyppien hallintaa. (McGregor, D. & Pryce, N. 2021.)

Kotlinin yhteensopivuus Javan kanssa tekee siitä myös mielekkäämmän kielen käyttää sitä vanhan Java -koodin refaktorointiin, koska yleensä kielten välillä refaktorointi edellyttää koodin kääntämiseen käytettyjen työkalujen käyttöä. Ulkoisten työkalujen käyttö ei yleensä saa koko ohjelmaa ja sen toiminnallisuutta käännettyä suoraa, vaan sen saaminen toimivaksi kokonaisuudeksi edellyttää manuaalisia toimia. (McGregor, D. & Pryce, N. 2021.)

2.3.2 Kotlinin käytön haitat

Kotlin ei kuitenkaan ole suora päivitys verrattuna Javaan. Monet Javan tavoista tehdä ovat suorituskyvyn puolesta tehokkaampia kuin Kotlinissa. Kotlinin käyttämä tapa ajaa konekoodia JVM -virtuaalikoneen avulla on alkuperäisesti suunniteltu Javan käytettäväksi, joka on optimoitu Javan käyttöä varten.

Kotlin ominaisuus automatisoi monia operaatioita, mutta operaatioiden automatisointi ei tule ilman omia kustannuksia. Automatisointi piilottaa koodista yksityiskohtia, joka voi haitata sen toiminnallisuuden tulkittavuutta. Tämä tekee koodista helpommin luettavaa ja toiminnallisempaa, joka voi puolestaan tehdä koodin toiminnan tulkinnasta hankalampaa.

Vähäinen käyttöönotto tuo myös mukana omat ongelmansa. Monet tuntevat Javan ja ovat tutustuneet sen ominaisuuksiin ja tottuneet sen ongelmiin. Pienen käyttäjäkunnan takia Kotlinin käyttöön liittyvä kirjallisuus ja ohjeet ovat vähäisiä, joka mahdollisesti vähentää kiinnostusta uuden kielen opiskeluun ja käyttöönottoon.

2.3.3 Kotlinin suorituskyky

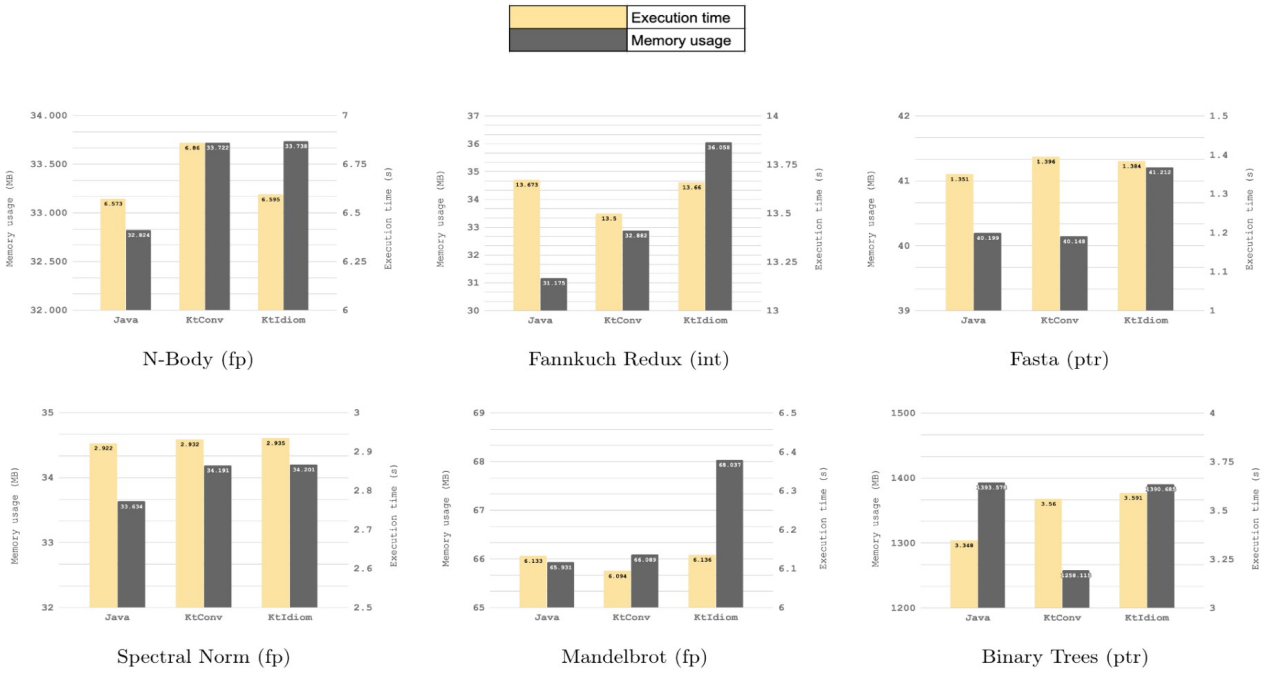
Ohjelmointikielien vertailussa yksi olennaisista vertailukohteista on suorituskyky. Suorituskyvyn mittaamiseen on suunniteltu menetelmiä, joista yksi on CLBG (Computer Language Benchmarks Game). CLBG on vapaan ohjelmiston projekti, jonka tarkoituksena on antaa karkea kuva ohjelmointikielien suorituskyvystä yksinkertaisten algoritmien avulla. CLBG koostuu 10:stä algoritmista, joista jokaisen tarkoitus on mitata ohjelmointikielien suorituskykyä eri osa-alueilla. (CLBG. n.d.)

Jakub Anioła (2019) on perehtynyt kaksiosaisessa artikkelissaan Kotlinin ja Javan välisiin eroihin ja hänen tavoitteenaan on selvittää, onko näiden kahden kielen väliset erot suorituskyvyssä merkittäviä. Artikkelissaan Anioła käy läpi 6

CLBG:n kuuluvista algoritmeistä, jotka on valittu hänen asettamien kriteerien mukaisesti. Testeissä piti olla kirjoittajan asettamien kriteerien mukaan paljon käsiteltävää dataa ja testien piti olla käännettävissä automaattisesti Kotlinille. Tekstissään Anioła vertaa Javan suoriutumista automaattisesti luotuun Kotlin versioon ja Kotlinin dokumentaatiota vastaavaan versioon.

Artikkelin ensimmäisessä osa keskittyy ohjelmointikielien suorituskyvyn ja resurssien käytön mittaamiseen ja artikkelin toinen osa keskittyy kääntöprosessissa tuotetun bittikoodin tutkimiseen. Tässä osiossa keskitytään pääasiassa ensimmäisessä osiossa löydettyihin havaintoihin.

Artikkelissa tutkittujen testien tulokset antavat vaikutelman, että yleisesti ottaen Kotlin pärjää monessa eri operaatiossa lähes yhtä hyvin kuin Java. Testien tuloksissa. Kotlinin dokumentaation mukaisesti käytettynä ei kuitenkaan saavuttanut yhdessäkään testissä suorityskyvyllisesti parhaita tuloksia. Javasta automaattisesti luotuna Kotlin suoriutui kuitenkin mediaani parhaiten kahdessa testissä kummassakin mitatussa kategoriassa.



Kuva 3: Ohjelmointikielien suoriutuminen testeissä. Kaavion vasemmalla puolella Java, keskellä automaattisesti luotu Kotlin, ja oikealla Kotlinin

dokumentaatiota vastaava versio. (https://medium.com/rsq-technologies/comparative-evaluation-of-selected-constructs-in-java-and-kotlin-part-1-dynamic-metrics-2592820ce80)

Aniołan julkaisema artikkeli on tyyliltään läpinäkyvä, työhön liittyvät parametrit ja testauksessa käytetyt metodit on esitelty selkeästi. Myös benchmarkkaukseen käytetty ohjelma on vapaasti ladattavissa ja tarkasteltavissa GitHubissa. Artikkelin kirjoittaja on kuitenkin saanut artikkelin kommenteissa kritiikkiä dokumentaatiota vastaavasta versiosta, koska dokumentaatiota vastaava version koodin laatu vaihtelee merkittävästi eri benchmarkkien kohdalla. Vastaavasti kirjoittaja on saanut myös positiivista palautetta tekemästään työstä.

Automaattisesti luotuna Kotlin vastaa monessa testissä Javalle kehitetyn ohjelman tuloksia. Karkeasti katsottuna Java käytää ohjelman ajon aikana vähemmän muistia, mutta suoritusajan puolesta erot ohjelmointikielien välillä ovat pieniä. Poikkeuksina on kuitenkin ”N-Body” ja ”Binary Trees” -testit. ”N-Body” -testissä Java suoriutuu pienemmässä ajassa sekä pienemmällä resurssien käytöllä kuin kumpikaan Kotlin -versioista. ”Binary Trees” -testissä puolestaan automaattisesti luotu kotlin -ohjelma suoriutuu hitaammin ajassa, mutta käyttää vähemmän resursseja kuin Java -ohjelma.

2.4 Javan suunnitteluperiaate ja historia lyhyesti

Javan oli alunperin suunniteltu käytettäväksi kodinkoneissa ja interaktiivisissa televisioissa. Javan käyttö nousi merkittävästi, kun Netscape -internetselaimeen lisättiin tuki Java sovelluksille. Myös Microsoft lisäsi omaan selaimeen Internet Exploreriin tuen Java sovelluksille, jolloin sen ajan kaksi suurinta selainta sisälsi sisäänrakennetun tuen Java -sovelluksille. Tämä toi Javalle uuden käyttötarkoituksen, joka johti ohjelmointikielen kasvavaan suosioon. (McGregor & Pryce, 2021, 4.)

Yksi javan merkittävistä ominaisuuksista kielen suunnittelussa alkuaikoina oli virallinen nimeämiskäytäntö ohjelmiston eri osille. Virallinen nimeämiskäytäntö

teki ulkoisten kirjastojen lisäämisestä verrattain saumattomaa, koska muilla sen ajan kielillä ei ollut virallista nimeämiskäytäntöä, joka vaihteli jokaisen kirjaston kohdalla. (McGregor & Pryce, 2021.)

Myöhemmissä Javan versioissa lisättiin virallinen tapa lisätä ohjelmiin käyttöliittymiä ja JavaBeans API, joka lisäsi mahdollisuuden rakentaa itsenäisiä osia ohjelmaan. Ajan myötä Java yleistyi myös yrityskäytössä, jolloin JavaBeans API:n tuomia ominaisuuksia varten kehitettiin kehityskehyksiä ohjelmistojen resurssienhallintaa varten, joista yksi tunnetuimmista on Javalle kehitetty Spring -kehityskehys. (McGregor & Pryce, 2021.)

Viimeisimpien Javan versioiden aikana on lisätty useita uusia moderniin ohjemistokehitykseen kuuluvia ominaisuuksia, kuten null-safety, lambda -funktiot ja streams -ohjelmointirajapinta. Nämä ominaisuudet ovat toimineet merkittävinä ominaisuuksia Javan aikaisemmin kohtaamien ongelmien ratkaisussa.

2.5 Yleiset ongelmat Javassa

Java on nyt yli 25 vuotta vanha ohjelmointikieli ja vastaavasti sen yksi suurimmista ongelmista on sen iän mukana tulleet ongelmat. Java on toiminut edelläkävijänä joidenkin asioiden suhteen, joka on tarkoittanut uusien ominaisuuksien kehittämistä ennen toimiviksi todettujen ratkaisujen löytymistä. Javan kehityshistorian aikana sille on lisätty monia ominaisuuksia, joista osa ei ole saanut positiivista vastaanottoa.

Yksi eniten kritiikkiä saaneista ominaisuuksista on JavaBeans API:n set ja get metodit, joilla vaikutetaan objektien ominaisuuksien hallintaan. Set ja get -metodit lisäsivät huomattavasti ohjelmissa olevan toimettoman koodin määrää, joka haittaa koodin muokkausta ja tekee luokkien koosta huomattavasti kookkaamman. JavaBeansin aiheuttamaa ongelmaa on pyritty ratkaisemaan ulkoisten kirjastojen avulla (esim. Javan Lombok -kirjasto), jolloin ohjelmassa olevaa toimetonta koodia on saatu vähennettyä. (McGregor & Pryce, 2021, 5-7.)

Ohjelmointikielen käyttöikä tuo mukanaan omia ongelmia, jotka voivat edellyttää muutoksia ohjelmointikielen vanhemmissa ominaisuuksissa. Myös vuosikymmenien jälkeen vanhojen perusominaisuuksien päivitys muuttuu hankalaksi, koska kielen perusominaisuudet ovat mahdollisesti toimineet uusien ominaisuuksien perustana. Tämä voi muutosten yhteydessä johtaa joidenkin toiminnallisuuksien rikkomiseen, joka ei yleensä ole haluttu lopputulos.

3 VERKKOKEHITYS

3.1.1 Verkkokehitys käsitteenä

Verkkokehitys itsessään on vielä varsin uusi ohjelmistokehityksen muoto, joten käsite ”moderni verkkokehitys” itsessään saattaa kuulostaa jossain määrin kummalliselta. Verkkokehityksessä on tästä huolimatta tapahtunut huomattavia muutoksia verrattuna ensimmäisten verkkosivujen tuotantotapoihin. Ensimmäiset verkkosivut olivat staattisia verkkosivuja, joiden sisältö on valtaosin yksittäisiä verkkosivuja, joiden toiminnallisuudet olivat rajoitettu aktiiviselle sivulle.

3.1.2 Mitä on moderni verkkokehitys

Yleensä modernissa verkkokehityksessä kuitenkin ohjelmistokokonaisuutta, joka koostuu monesta eri teknologiasta ja on yleensä jaettu tietokantaan, backendiin ja frontendiin. Yleisesti modeniin verkkokehitykseen liittyvää kokonaisuutta kutsutaan myös Stackiksi, joka viittaa sovelluksen käyttämään pinoon erilaisia teknologioita.

Moderniin verkkokehitykseen on kehitetty myös monenlaisia erilaisia ohjelmistokokonaisuuksia. Yleisin tällaisista kokonaisuuksista on CMS (Content Management System) -järjestelmä, jolla pystytään tuottamaan monenlaisiin tarkoitukseen soveltuvia verkkosovelluksia. Esimerkkejä yleisitä CMS -järjestelmistä ovat esimerkiksi Wordpress, Joomla ja Durpal. Myös Kotlinille on kehitetty omia CMS, järjestelmiä, joista yksi on Strapi.

Kotlinilla on mahdollisuus toimia hyvänä kielenä modernissa verkkokehityksessä, koska sen moderni tyyli sopii hyvin modernien verkkokehitykseen käytettyjen teknologioiden kanssa. Kotlinin kehitys on alkanut modernien verkkosovellusten kehityksen yleistymisen ajalla, joten sen kehityksessä on pystytty ottamaan huomioita erityisesti verkkokehityksen suunnalta tulleiden tarpeiden mukaan.

4 Projekti

4.1 Projektin tarkoitus ja sen lopputuotteet

Kotlinin ja Javan käyttö verkkokehityksessä keskittyy pääasiassa sovelluksen backend -osioon. Yleensä frontendin kehitykseen käytetään eri ohjelmointikieliä kuin backendissä. Frontendin kehitykseen käytetään yleensä ohjelmointikieliä, jotka tuottavat lopputuotteeksi HTML, JavaScript ja CSS -tiedostoja. Backendin osa verkkosovelluksessa vastaa tietojen välittämisestä tietokannan ja frontendin välillä.

Osana opinnäytetyötä toteutettiin kaksi toiminnallisesti samanlaista verkkosovellusta, joista toinen on totetutettu Javalla ja toinen Kotlinilla. Verrattavia ohjelmointikieltä käytetään vain sovelluksen backendin kehittämiseen ja sovellukset käyttävät toiminnassaan identtistä frontendiä, jotta ohjelmien toiminta olisi mahdollisimman samanlainen.

Ohjelmointkielillä toteutettu sovellus on verkkopalvelu, johon lisätyt julkaisut poistuvat määritetyn ajanjakson kuluttua niiden julkaisusta. Kuka tahansa voi julkaista palvelussa omia julkaisuja ja jokainen pystyy näkemään sovellukseen lisätyt julkaisut.

4.1.1 Backendin kehitykseen käytetyt teknologiat

Ensimmäiseksi projektissa kehitetään Javalla ja Kotlinilla verkkosovellus, joista ensimmäiseksi toteutetaan Javalla toteutettu versio. Toisen verkkosovelluksen työstö aloitetaan vasta ensimmäisen sovelluksen valmistuttua. Projektin lopputuotteiden toteutuksen tärkein osuus on sovellusten backendin toteutus, jossa nähdään ohjelmointikielien erot verkkosovelluksen toteutuksessa.

4.1.2 Frontendin kehitykseen käytetyt teknologiat

Verkkosovelluksen frontend vastaa verkkosovelluksen käyttöliittymästä, jonka asvulla sovelluksen toimintoja ohjataan. Frontendin toteutukseen käytetään TypeScriptille kehitettyä verkkokehityskehystä, Angularia.

4.1.3 Tietokannan tenknologiat

Projektissa tarvitaan tietokanta sovelluksessa kehitetyn tiedon tallentamiselle. Ylesiä verkkosovellusten vaihtoehtoja tietokantoja ovat muunmuassa MariaDB, MySQL, postgreSQL ja h2 Database.

Sovelluksessa päädyttiin käyttämään h2 tietokantaa, koska se soveltuu hyvin PoC (Proof of Concept) -tyyppisille sovelluksille sen ohjelmistosta riippumattoman toimintatavan takia. Suuremman mittakaavan projektissa muiden tietokantaohjelmistojen käyttäminen on suotavaa palvelun suorituskyvyn ylläpitämiseksi.

4.2 Projektissa käytetyt työkalut

Projektin koodin luomiseen käytetään verkkokehitykseen yleisesti käytettyjä työkaluja. Työkaluihin kuuluvat integroitu kehitysympäristö (IntelliJ IDEA), versionhallintajärjestelmä (Git), Java projektin hallintaan käytettävä työkalu (Maven) ja frontendin kirjastojen hallintaan käytetty paketinhallintaohjelma (NPM).

4.2.1 IntelliJ IDEA

IntelliJ IDEAn tarkotuksena on tuoda kaikki ohjelmistokehitykseen liittyvät työkalut saman käyttöliittymä alle. IntelliJ -kehitysohjelmisto soveltuu hyvin projektin kehitysympäristöksi, koska se tukee kaikkia projektissa käytettäviä työkaluja ja Kotlinin kielituki on valmiiksi asennettuna kehitysympäristöön.

IntelliJ IDEAn tarkoitus tässä projektissa on toimia keskeisenä kehitysympäristönä, jossa projektin koodi tuotetaan ja sen avulla hallitaan kaikkia muita projektiin liittyviä työkaluja. Projektin tuotannossa on ollut käytössä IntelliJ IDEAn Community -versio, joka on ilmainen ja avoimen lähdekoodin versio IntelliJ Ultimate ohjelmistosta.

4.2.2 Git ja GitLab

Git on ilmainen ja avoimen lähdekoodin ohjelmisto, jota käytetään ohjelmistojen versionhallintaan. Gitin tarkoitus projektissa on luoda selkeä historia ja luoda mahdollisuus palauttaa sovellus aikaisempaan versioon tarvittaessa. Git toimii myös työkaluna, jolla ohjelmiston koodi lisätään GitLab -säilytyspalveluun.

GitLab on koodin katselmointiin ja säilytykseen käytetty verkkopalvelu, jonka tarkoitus on toimia projektissa koodin säilytyspalveluna. Projektin koodi julkaistaan julkisesti nähtäväksi projektin valmistuttua. GitLab sisältää monia tilastoja projektin sisältöön liittyen, josta voidaan nähdä esimerkiksi projektissa käytettyjen koodin määrän katselmointiin.

4.2.3 Maven

Maven on ohjemistoprojekteille suunniteltu työkalu, joka luo koontitiedostot automaattisesti projektin koodista ja sen käyttämistä lisäosista ja kirjastoista. Mavenin valmistelee projektin tuotantokäyttöä varten ja sen avulla projektiin voidaan lisätä uusia lisäosia, jotka tuovat projektin backendille uusia ominaisuuksia. Ominaisuuksiin voi kuulua esimerkiksi projektin ulkopuolisia kirjastoja, ohjelmointirajapintoja ja projektin toimintamallin muuttamiseen liittyvä ominaisuuksia.

4.2.4 NPM

NPM on JavaScript -ohjelmointikielen ja Node.js koodin ajoympäristön hallintaan käytettävä paketinhallintaohjelmisto, jolla projektiin saadaan lisättyä. NPM tarjoaa kattavan määrän JavaScriptille ja TypeScriptille kehitettyjä kirjastoja, jotka on vapaasti käytettävissä erilaisissa projekteissa. Tässä projektissa NPM –ohjelmistoa käytetään ainoastaan projektin frontendissä käytettävien kirjastojen hallintaan.

4.3 Java -projektin kulku

Ensimmäisen projektin työstäminen alkoi projektissa käytettävien työkalujen käyttöönotolla. Valtaosa projektiin liittyvistä työkaluista ja teknologioista saatiin valmiiksi kootussa paketissa start.spring.io -verkkosivulta, jossa voi luoda ja ladata tyhjän Spring -kehityskehystä käyttävän verkkosovelluksen. Sivustolla pystyy lisäämään projektiin myös monia riippuvuuksia, joilla pohjaan saa lisättyä lisää toiminnallisuutta. Pohjan ladattua se puretaan omaan kansiooonsa, jolloin tyhjä projekti on valmis kehitystyötä varten.

Projektin työstä aloitettiin työn backend osiosta, jossa ensimmäinen tehtävä oli REST (Representational State Transfer) -ohjelmointirajapintojen asettaminen projektin frontendiä varten. Rajapinnoilla määritetään, mitä tietokannan tietoja välitetään frontendille. Rajapintojen asettamisen jälkeen projektille luodaan palvelu nimeltä BlogService, joka vastaa REST -rajapinnan välittämiin pyyntöihin. BlogServicen tarkoituksena on välittää rajapinnan välittämät kutsut tietokannalle ja palauttaa tiedot tietokannasta takaisin rajapinnalle.

Rajapintojen asettamisen jälkeen projektin backendiin luotiin objekti julkaisuille, jota käytetään frontendistä saatujen tietojen välittämiseen tietokannalle. Julkaisut sisältävät tietoa mm. Julkaisun otsikon, julkaisun sisällön, julkaisun kirjoittajan ja julkaisun aikaan liittyvää tietoa. Objektissa pitää määrittää myös get ja set -metodit jokaiselle määritetylle muuttujalle, jotta objektin arvojen asettaminen ja noutaminen olisi mahdollista (KUVA 4).

```
25
26      public String getTitle() { return title; }
29
30      public void setTitle(String title) { this.title = title; }
33
34      public String getContent() { return content; }
37
38      public void setContent(String content) { this.content = content; }
41
42      public void setAuthor(String author) { this.author = author; }
45
46      public String getAuthor() { return author; }
49
50      public LocalDateTime getCreatedAt() { return createdAt; }
53
54      public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }
57
58      public long getId() { return id; }
61
62      public void setId(long id) { this.id = id; }
65
66      public LocalDateTime getExpiresAt() { return expiresAt; }
69
70      public void setExpiresAt(LocalDateTime expiresAt) { this.expiresAt = expiresAt; }
73
74      public Long getTimeActive() { return timeActive; }
77
78      public void setTimeActive(Long timeActive) { this.timeActive = timeActive; }
81  }
```

KUVA 4: Java -olioiden muuttujien hallinta ilman ulkoisten kirjastojen käyttöä.

Tämän jälkeen projektin backend on valmis frontendin toimintaa varten. Projektiin lisättiin ennen frontendin kehittämistä Maven -lisäosa, joka kokoaa ja välittää frontendin tiedostot backendille aina sovelluksen käynnistämisen yhteydessä. Tämän lisäosan ainoa tarkoitus on nopeuttaa projektin kehitystä.

Frontendin tärkeimmät osuudet ovat lomake julkaisujen lisäämiselle ja alue, johon tietokannasta haetut julkaisut lisätään. Angularissa kummallekkin osalle voidaan luoda omat komponentit, jolloin ne ovat helposti yksitellen hallittavissa. Näiden osien lisäksi angularissa luodaan malli (model) julkaisuille, johon lisätään kaikki julkaisun luomiseen liittyvät tiedot ennen sen lähettämistä backendille. Malli välitetään luonnin jälkeen frontendin BlogServicelle, joka vastaa yheyksien välityksestä backendille.

Tämän jälkeen sivu on perusominaisuuksiltaan täysin kunnossa ja projektissa päästiin keskittymään sivujen muihin ominaisuuksiin, kuten sivujen frontendin ulkoasun asetteluun. Sivuille lisättiin toiminnallisuuden puolesta vielä aikaan

perustuva poistotoiminto, joka poistaa sivuille lisätyt julkaisut frontendissä määritetyn ajan jälkeen.

4.4 Kotlin -projektin kulku

Kotlin -projektin aloitus on toimii käytännössä samalla tavalla kuin Java -projektin aloituksessa, mutta tyhjää projektia ladatessa start.spring.io -sivulta valitaan ohjelmointikieleksi Kotlin Javan sijasta. Frontendin lisättiin projektiin heti kehityksen alkuvaiheilla, jossa projektiin kopioitiin toisen projektin frontendin tiedostot. Frontendin kokoamiseen ja tiedostojen siirtoon backendille käytetään samaa Mavenin lisäosaa kuin Java -projektissa.

Kotlin projekti työstämiseen on tässä vaiheessa kaksi vaihtoehtoa: Projektin työstö voidaan aloittaa tyhjältä pohjalta tai projektiin voidaan ottaa toisen sisältämä koodi käyttöön. Tämän projektin laajuus on kuitenkin pieni, joten vanhan koodin käyttöön ottamisen hyöty jäisi erittäin pieneksi. Tässä projektissa projekti aloitetaan puhtaalta pohjalta, jotta Kotlinin ominaisuuksia tulisi käytyä läpi mahdollisimman paljon perusteista lähtien.

Kehitysprosessissa käytiin samoja asioita kuin Java -sovelluksen kehityksessä: Ensin aloitettiin backendin rajapintojen lisäämisestä, jonka jälkeen lisättiin palvelu backendin tietojen välitykselle tietokantaan ja päinvastoin. Tämän jälkeen lisättiin olio frontendistä tulevien tietojen hallitsemiseen. Suurimmat erot javaan verrattuna olivat uuden olion luonnissa, jossa oliolle ei tarvinnut luoda get ja set -metodeita (KUVA 5). Toiminnallisuus pysyi kuitenkin identtisenä Kotlinin sisäänrakennettujen toiminnallisuuksien takia.

```
@Entity
data class BlogDto (
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(nullable = false, updatable = false)
    val id: Long,

    @Column(nullable = false)
    val title: String,

    @Column(nullable = false)
    val content: String,

    @Column(nullable = false)
    val author: String,
    val timeActive: Long,

    @Column(name = "created_at")
    var createdAt: LocalDateTime?,

    @Column(name = "expires_at")
    var expiresAt: LocalDateTime?
)
```

KUVA 5: Kotlinin dataluokan olioiden muuttujien hallinta ilman ulkoisten kirjastojen käyttöä.

4.5 Merkittävät erot sovellusten tuotannossa

Javassa ja Kotlinissa käytetyt syntaksit ovat toisiinsa verrattuna merkittävästi erilaisia. Ohjelmiston toimintaa ohjataan kuitenkin käyttämällä samoja kehitykseen käytettyjä kirjastoja, joten toiminnallisuuden kehitys on backendin kehityksen osalta samankaltaista. Frontendin puolella eroja ei ole, mikäli sovellusten väliset rajapinnat ovat identtisiä.

Liikkuminen backendin koodin ja frontendin koodin välillä tuntuu huomattavasti luontevammalta Kotlinin ja TypeScriptin välillä verrattuna Javan ja TypeScriptin välillä. Merkittävä osa tästä johtuu Kotlinin ja TypeScritpin käyttämästä

syntaksista. TypeScriptin ja Javan välillä erot ovat huomattavasti suuremmat, joka voi tehdä kielien syntaksien välillä liikkumisesta hitaampaa.

Javassa tiedon säilömiseen käytetyissä luokissa on huomattavasti enemmän koodia, vaikka Kotliniin verrattuna koodin toiminnallisuus on identtistä. Javalla jokainen koodissa käytetty vaatii toteutukset set ja get -metodit, jotta muuttujien tilaa saadaan muutettua. Kotlinissa get ja set -metoidien asettaminen on toteutettu dataluokille automaattisesti, joka vähentää koodin määrää huomattavasti (KUVA 4 & KUVA 5).

Erot pienessä projekteissa eivät tietenkään ole kovinkaan isoja, mutta projektien koon kasvaessa projekteissa saattaa olla kyse sadoista ja tuhansista koodiriveistä, joka voi aiheittaa ongelmia koodin luettavuudessa ja tulkinnassa.

5 POHDINTA

Työn aikana ei ollut tarkoitus ottaa kantaa siinä arvioitujen ohjelmointikielien toimintaan tai korostaa kumpaakaan työssä käsiteltyä ohjelmointikieltä. Tämä on työn kontekstissa hankalaa, koska työssä vertaillaan kahta ohjelmointikielen ominaisuuksia toisiinsa. Tämän takia työssä on etsitty ja perusteltu siinä esiintyviä väitteitä ulkopuolisilla lähteillä, jotka pyrkivät tarkastelemaan asiaa eri näkökulmista. Opinnäytetyön läpinäkyvyyden takaamiseksi siinä tuotetut lopputuotteet julkaistaan avoimen lähdekoodin sovelluksina, jolloin sovellusten käyttämä koodi on vapaasti tutkittavissa.

Yleisesti ottaen Kotlin on kuitenkin rakennettu tehokkaaksi ja yleispäteväksi ohjelmointikieleksi, joka on tällä hetkellä tuettu jo ohjelmistokehityksen monilla eri saroilla. Suorituskyvyltään Kotlin ja Java ovat verrattain tasaväkisiä, mutta Java on suorituskyvyltään lievästi tehokkaampi.

Kotlin on saanut hyvän alun kielen kehitykselle ja sen yleistyminen Android -laitteissa voi toimia vakaana kivijalkana sen suosion kasvattamisessa. Kotlin voi hyvinkin saada lisää suosiota tilanteissa, jossa vanhaa Java -koodia pitäisi päivittää modernimmaksi. Lähitulevaisuudessa Kotlin ei välttämättä tule yleistymään näistä seikoista huolimatta. Tällä hetkellä Kotliniin siirtyminen voidaan nähdä riskinä, joka pätee erityisesti yritysten olosuhteissa. Myös tottumukset Javan käyttämiseen ja Javan saamat uudet ominaisuudet voivat hidastaa kielen yleistymistä.

Kielen yleistyminen edellyttäisi suuremman ominaisuuden kehitystä yksinomaisesti Kotlinille. Javalla tämä tapahtui, kun kaksi yleisintä verkkoselainta lisäsivät sisäänrakennetun tuen Java sovelluksille. Kielen yleistyminen voi tapahtua myös muista syistä, kuten kielen ominaisuuksien poiston takia, mutta sen tapahtuminen on erittäin epätodennäköistä.

Itse näen Kotlinin suurimman hyödyn tulevan yhteensopivuudesta Javan koodin kanssa. Tämä mahdollistaa Kotlinin ja Javan käytön saman projektin alla, jonka avulla voidaan tarvittaessa hyödyntää tiettyjä ominaisuuksia kummastakin kielestä ilman uusien ongelmien ilmenemistä. Ominaisuus helpottaa myös

ohjelmistojen kääntämistä kielestä toiseen ja kielestä toiseen siirtyminen voi tapahtua samassa koodissa pidemmällä aikavälillä vaikka ominaisuus kerrallaan.

Tämä opinnäytetyö keskittyi Kotlinin yleiskuvan tutkimiseen verkkokehityksen kontekstissa. Jatkokehityksen mahdollisuuksia voivat olla esimerkiksi Kotlin/JS -frontend -kirjastoon käyttöönottoon tai Kotlinin monisäikeistyksen (Concurrency) eri ominaisuuksiin liittyen.

LÄHTEET

Krill, P. 2011. JetBrains readies JVM-based language. Infoworld. Uutisartikkeli. Viitattu 24.10.2021. https://www.infoworld.com/article/2622405/jetbrains-readies-jvm-based-language.html

Breslav, A. 2016. Kotlin 1.0 Released: Pragmatic Language for the JVM and Android. JetBrains -blogi. Viitattu 24.10.2021. https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/

Hariri, H. 2018. Functional Programming with Kotlin. YouTube. Julkaistu 27.6.2018. Viitattu 26.10.2021. https://www.youtube.com/watch?v=eNe5Nokrjdg

Elizarov, R. 2019. Server-side Kotlin with Coroutines - GOTO 2019. YouTube. Julkaistu 14.6.2019. Viitattu 26.10.2021. https://www.youtube.com/watch?v=hQrFfwT1IMo

Anioła, J. 2019. Medium. Medium -blogi. Java vs. Kotlin — Part 1: Performance. Viitattu 30.10.2021 https://medium.com/rsq-technologies/comparative-evaluation-of-selected-constructs-in-java-and-kotlin-part-1-dynamic-metrics-2592820ce80

McGregor, D. & Pryce, N. 2021. Java to Kotlin: a Refactoring Guidebook. 1st edition. O'Reilly Media.

Lardinois, F. 2019. TechCrunch. Kotlin is now Google's preferred language for Android app development. Uutisartikkeli. Julkaistu 07.05.2019. Viitattu 03.11.2021. https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

Kotlin/JS. n.d. Kotlinlang.org. Ohjelmointikielen dokumentaatio. https://www.infragistics.com/community/blogs/b/mobileman/posts/building-a-better-web-a-brief-history-of-web-development

Kotlin couroutines. n.d. Kotlinlang.org. Ohjelmointikielen dokumentaatio. https://kotlinlang.org/docs/coroutines-overview.html

Kotlin-Java interop. n.d. Kotlinlang.org. Ohjelmointikielen dokumentaatio. https://kotlinlang.org/docs/java-interop.html

Macdonald, M. 2021. Tiny.cloud. Tiny Programming Principles: Immutability Blogiartikkeli. Viitattu 10.11.2021. https://www.tiny.cloud/blog/mutable-vs-immutable-javascript/