A Comparative Study of Java and Kotlin for Android Mobile Application Development

Bambang Purnomosidi Dwi Putranto
Master of Information Technology
STMIK Akakom
Yogyakarta, Indonesia
bpdp@akakom.ac.id

Robertus Saptoto
Master of Information Technology
STMIK Akakom
Yogyakarta, Indonesia
robertus.saptoto@gmail.com

Ovandry Chandra Jakaria
Master of Information Technology
STMIK Akakom
Yogyakarta, Indonesia
ovandrychandrajakaria@gmail.com

Widyastuti Andriyani
Master of Information Technology
STMIKAkakom
Yogyakarta, Indonesia
widy.ugm@gmail.com

2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) | 978-1-7281-8406-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/ISRITI51436.2020.9315483

Abstract— **There are several programming languages that can be used to develop Android application, such as C++, Java, Kotlin, JavaScript, and many more. Each programming language certainly has some advantages and disadvantages in the development of Android application. Therefore, a comparison to see the values of those programming languages is needed. This research compared two popular programming languages for Android application development, namely Java and Kotlin. The testing was done by building two applications using Java and Kotlin that access data in a remote server. Our comparison includes source code evaluation, testing on the performance of the app performed on two devices, and testing on the data usage. From the test results, it was proved that Kotlin is superior in terms of more concise lines of code and less data usage which will reduce bugs substantially therefore will lead to faster development time. On the other hand, Java is superior in its compiling time (for first time build but on par with Kotlin for incremental build) and APK size albeit not significant. We also compare ecosystem and programming language constructs for both programming languages. Overall, for Android mobile application development, Java should be used if mobile application development priority is the APK size and compilation / build time while Kotlin should be used if mobile application development priority is lesser bugs, concise code, and faster development time.**

Keywords— Programming Language, Java, Kotlin, Android, Comparison.

I. Introduction

Programming language plays an important role in developing an application and allows programmers to provide computer-understandable instructions and allow computers to process large and complex information quickly and efficiently [1], [2], [3]. Programming language is considered as a means of expressing computations in a comprehensible form for both people and machines. Much like human language, there are many computer programming languages which can be used by programmers to communicate with computers [4], [5] . The syntax of a language specifies how various sorts of phrases (expressions, commands, declarations, etc.) are combined to form programs [6], [7].

Some programming languages that can be used to develop Android applications are Java and Kotlin. Java is an object-oriented programming language (and currently also includes functional programming) that lifts objects that exist in the real world and has good flexibility as one of the programming languages because of its multiplatform; which means that it can be run on various platforms, such as: Linux, Windows, Solaris, and various mobile devices without having to change the existing code, which is commonly referred to the term "write once, run anywhere" [8], [9]. By compiling source code into intermediate Java byte code, and then executing it on a virtual machine on the target system, Java provides portability and platform independency which few other languages can offer [10]. However, Java is a verbose language, thus one of the main drawbacks of the language is that even simple tasks often need a significant amount of code. To allow programmers to write concise code, JetBrains created a new language named Kotlin [11]. Kotlin has more modern language features than Java. Kotlin is a pragmatic programming language that combines object-oriented (OO) and functional programming [12], [13]. Kotlin has attracted many developers because of its simple syntax and its main focus on mobile development in the beginning. Yet, it is mainly because of its compatibility with Java [14], [15].

In this study, the authors used an application to display international actors and artists that are at the peak of their career, the application will feature photos of 20 actors and the best international artist according to The Movie Database (TMDb).

II. Background

A. Java

Java is a language and a platform originated by Sun Microsystems [16], [17], [18]. Java language is an object-oriented programming language created by James Gosling and several other engineers at Sun Microsystems. Java was first developed in 1991 as part of the Green Project. Initially, Java was designed to replace the C++ language and it was known as Oak.

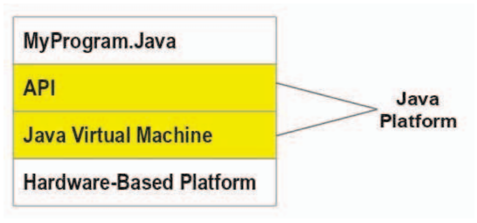


Fig. 1. Java Platform

Java platform is different from most other platforms. Within the Java platform, the software platform runs on top of the hardware-based platform. Most other platforms are a combination of hardware and operating systems. Java Platform has two components [19]:

1. Java Virtual Machine (JVM)
2. Java Application Programming Interface (Java API)

Java API is a collection of ready-made software components that provides a variety of facilities, such as: many GUI widgets, collections data structures, and many more. The Java API is grouped in the packages of related components [20], [21], [22], [23].

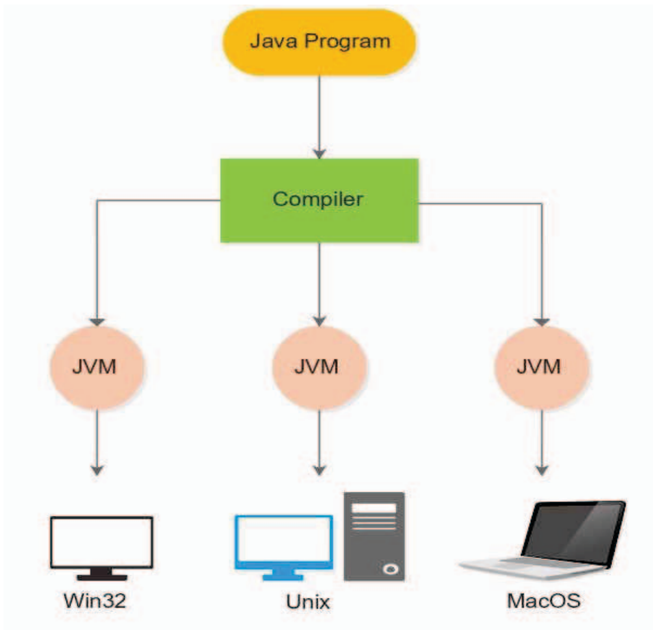


Fig. 2. Java Multiplatform

As can be seen in Fig. 2 above, Java program runs on top of the Java platform. Java Platform isolates Java programs with hardware, so Java programs do not rely on hardware independently. Java has several advantages when compared to other programming languages, such as: object-oriented language, multithreading, garbage collector support, statically typed, multiplatform.

B. *Kotlin*

Kotlin is a statically typed programming language that targets Java Virtual Machine (JVM), Android, JavaScript, and Native. At first, Kotlin was developed by JetBrains and Kotlin project was started in 2010. Then it became an open-source programming language. Kotlin version 1.0 was officially released in February 2016. Kotlin is a great fit for developing Android applications, bringing all of the advantages of a modern language to the Android platform without introducing any new restrictions like compatibility, performance, interoperability, footprint, compilation time and learning curve [24]. Kotlin is compiled to Java byte-code, which means that an application written in Kotlin can be executed on the Java virtual machine, and Kotlin is fully interoperable with Java [25], [26], [27]. As a result, Kotlin becomes very easy to get into for developers. As with the full interoperability with Java, it becomes very easy to get into the language by migrating small parts of the code into Kotlin [28].

C. *The Movie Database (TMDb)*

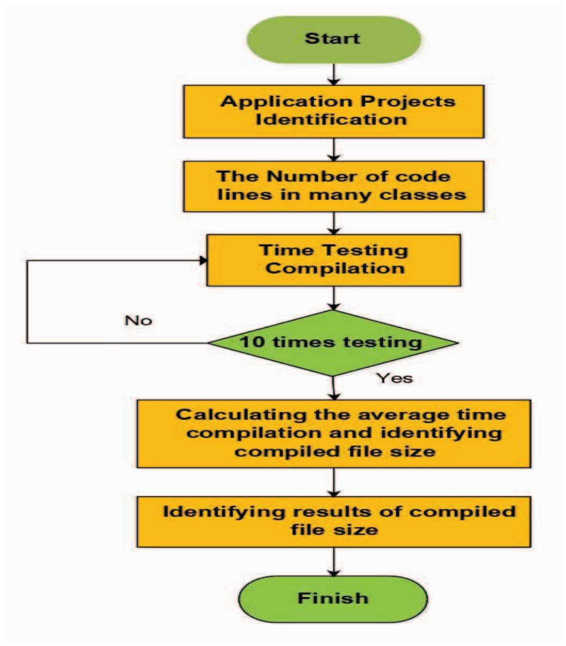
The Movie Database (TMDb) is an online database that provides information related to movies and TV programs [29]. The information provided among other popular movies and TV listings, detailed information, such as: release date, director, movie star, actor, synopsis, photos, video trailers, etc. TMDb provides API services so that the application developers can add information about movies and TV programs on their applications. Since 2008, TMDb has been used by more than 200.000 developers and companies.

D. *Android Profiler*

Android Studio has a great tool for application profiling. In addition, it provides the closest metal-to-metal interface with the mobile Android device. However, its usage is not simple and therefore, it becomes the third profiling tools compared to other tools. Android profiler provides real-time data applications, such as: CPU usage, memory, network, and battery. Android Profiler helps developers in measuring application performance [30].

III. Research Methodology

This research will develop two applications in which each represents the programming examples languages of Java and Kotlin. Both applications have the same requirements and end results so they can be analyzed and compared for their performance. Research flows in this study can be seen in Fig. 3.



Start
Application Projects Identification
The Number of code lines in many classes
Time Testing Compilation
No
10 times testing
Yes
Calculating the average time compilation and identifying compiled file size
Identifying results of compiled file size
Finish

Fig. 3. Research Flows

A. *Literature Review*

The purpose of a literature review is to gain an understanding of the existing research and debates relevant to a particular topic or area of study and to present that knowledge in a written report form. In this study, many research papers, on-line articles, and previous projects on different topics are reviewed to come up with an appropriate solution of how programming languages affect application performance.

B. *Applied Methodology*

This stage aims to analyze the application needs, design of the system as a whole, application implementation, and runtime testing to ensure that the application used is running based on the particular requirements. The application is a popular person application, a useful application for showcasing popular people in the world. Applications are built using Java and Kotlin programming language. Next, both applications will be measured by their performance measurement and efficiency performance to determine which application and efficiency performance are better. A Use Case diagram in Fig. 4 is created to describe interactions between users and applications. In general, the application takes personal data obtained from a server and users can only see the person's data.

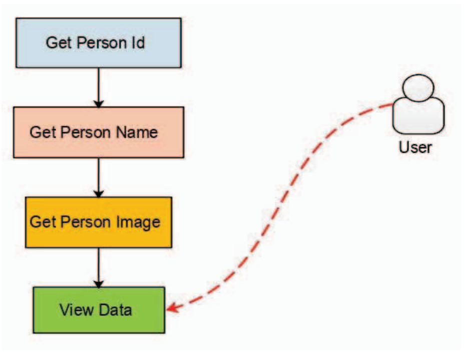


Fig. 4. Use Case Diagram

C. *Experiment*

Experiments are performed by performing performance measurements on individual Java and Kotlin applications in Android studio which uses Gradle for build tools and an emulator to run the application. The measurement scenario is to do application coding and building for 10 sessions. In each coding and building session, we change some source code for feature addition and/or bugs fixes (done on purpose to record incremental build). For those 10 sessions, the results for compilation and build time are averaged. Application with fewer number of lines of code, fewer number of classes and lower compilation and build time is considered as an application with better performance.

Experiments are performed using Oracle JDK 1.8.0 update 271, targeting Android SDK version 27. For Kotlin, we use the latest Kotlin version (1.4.10) as of november 2020 on Oracle JDK 1.8.0 update 271. We use Gradle 6.7 for our build tools. For hardware, our experiments are performed on two devices with specification:

Table 1. Device Specifications

Device A	Device B
Processor Intel®Core™i5-5200U CPU @2.20GHz 2.19GHz	Intel® Core™ i3-6006U CPU @ 2.00GHz
Random Access Memory (RAM) 8,00 GB	Random Access Memory (RAM) 4,00 GB
SSD 256 GB	Hard Disk 1 Terabyte
Hard Disk 1 Terabyte	

IV. Result and Analysis

Data processing and presentation are obtained from the experiment results for compilation time and lines of code while ecosystem and programming language constructs are based on current available resources. Based on the resulting data, the performance of both applications can be seen. Our analysis will show which programming language is more efficient to be used on Android-based application development. Our measurements are done by building applications on each programming language; Java and Kotlin. Next, we measure the application's efficiency by running the application. The metrics itself is obtained by using lines of code, class, and compilation time in Android Studio.

A. *Compilation Time and APK Size Results*

Comparison for compilation time on the application project applied to both programming languages at two computers showed that Java is superior in 10 times testing compared to Kotlin as shown in Fig. 5. First compilation always has the longest compilation time since Gradle will fetch any libs (.jar) needed for compilation. Subsequent compilation time is reduced significantly since Gradle caches all unchanged results and only builds the changed parts.

Table 2. Compilation Time (in seconds)

Testing	Device A		Device B	
	Java	*Kotlin*	*Java*	*Kotlin*
Test 1	46.592	76.592	16.212	26.212
Test 2	1.775	3.775	1.358	2.358
Test 3	1.744	2.744	1.270	2.270
Test 4	1.582	2.582	1.187	2.187
Test 5	1.555	2.555	1.133	2.133
Test 6	1.552	2.552	1.166	2.166
Test 7	1.241	2.241	1.480	2.480
Test 8	1.438	2.438	1.182	2.182
Test 9	1.126	2.126	1.250	2.250
Test 10	1.120	2.120	1.840	2.840
Average	5.973	9.973	2.808	4.708

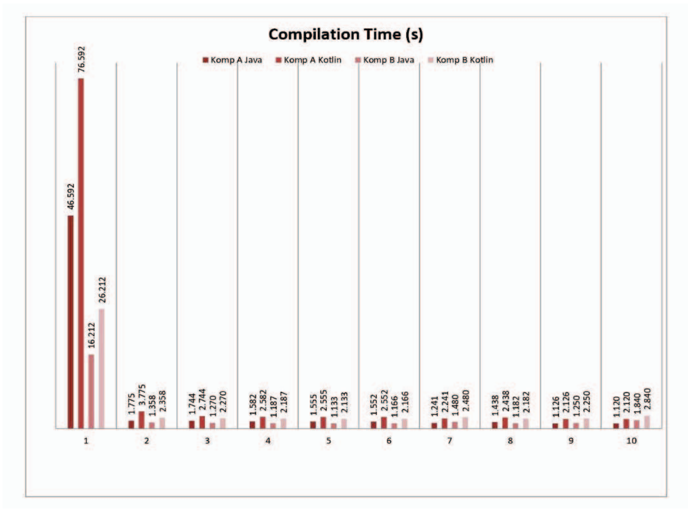


Fig. 5. Compilation Time Comparison

The final results of the APK size between the Java and Kotlin versions differ by about 77%. The Java version is 1,809,987 KB in size and the Kotlin version is 2,349,921 KB.

B. *Lines of Code*

For lines of code comparison in application projects for both programming languages, Java's lines of code is 8624 lines compared to Kotlin with 8055 lines of code. Therefore, in this case Kotlin is superior to Java as it is shown in Fig. 6.

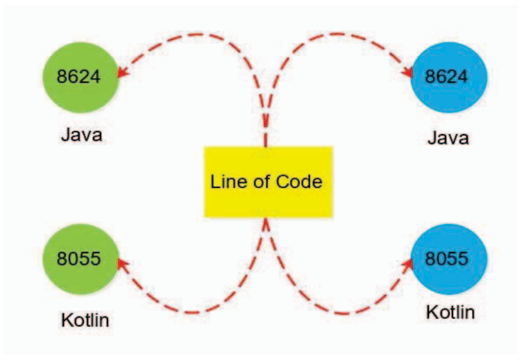


Fig. 6. Lines of Code Comparison

C. *Number of Class*

A comparison chart for the number of classes on application projects in both languages showed that Java has more classes since it uses 30 classes, while Kotlin only uses 22 classes. Therefore, it can be concluded that Kotlin is superior to Java as it can be seen in Fig. 7.

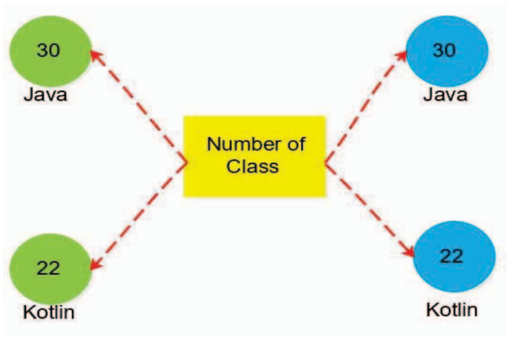


Fig. 7. Number of Class Comparison

D. *Ecosystem*

Both programming languages are JVM programming languages and both are officially supported by Google for Android application development. It implies that both are interoperable, means Kotlin can use Java libraries and vice versa. In this regard, all maven repositories can be used by both programming languages.

For IDE, currently Android Studio supports both programming languages, therefore, Java and Kotlin are supported from initial projects, during iterative development, debugging, profiling, and packaging for deployment. If developers prefer development without Android Studio, many available development tools exist to support this kind of development - for example SpaceVim enables syntax highlighting, completion, and many more using LSP (Language Server Protocol) for both Java and Kotlin.

For application development, even though we do not discuss multi platform development in this paper, note that Kotlin has more mature framework which is free and open source (KMM - Kotlin Multi Mobile)[1] while Java has similar free - proprietary commercial offering with Gluon Mobile[2]. This has to be considered upfront before application development is started.

E. *Programming Language Constructs*

Effective development and less development time are usually attributed to programming language constructs. We discuss comparison between Java and Kotlin in Table 3. Note that we use JDK 8 and Kotlin 1.4.10 for this comparison. We compare standard programming language constructs without third party libraries and add-ons. We use Kotlin documentation[3] and Java documentation[4] for this purpose.

Table 3. Comparison of Programming Language Constructs

Parameter	**Java**	**Kotlin**
Compilation target	Bytecodes for JVM. Native target available using GraalVM	Bytecodes for JVM, native, JavaScript
Null safety	No	Yes
Lambda expressions	No	Yes, also inline functions
Extension functions	No	Yes
Smart casts	No	Yes
Invariant array	No	Yes
Non-private field	Yes	No, forces setter and getter for properties.
Static member	Yes	No, bu tcan be replaced with companion objects, top-level

[1] https://kotlinlang.org/lp/mobile/

[2] https://gluonhq.com/products/mobile/

[3] https://kotlinlang.org/docs/reference/comparison-to-java.html

[4] https://docs.oracle.com/javase/8/

		functions, extension functions, or @JvmStatic
Wildcard types	Yes	No, but can be replaced with declaration-site variance and type projections
Singletons objects	Yes	Yes
Checked exceptions	Yes	No
Primitive types that are not classes	Yes	No
Ternary operator	Yes	No, but can be replaced with **if**.
Raw types	Yes	No
Properties	No	Yes
Primary constructors	No	Yes
String template	No	Yes
First-class delegation	No	Yes
Type inference for variable and property types	No	Yes
Declaration-sit e variance & type projections	No	Yes
Range expressions	No	Yes
Operator overloading	No	Yes
Companion objects	No	Yes
Data classes	No	Yes
Separate interfaces for read-only and mutable collections	No	Yes
Coroutines	No	Yes
Verbosity	Very verbose	Concise

V. Conclusion

Based on the research that has been done, it can be concluded as follows:

1. Testing on an application project showed that Kotlin is superior in terms of lines of code and the number of classes while Java is superior in terms of compilation time and APK size although they are not too significant.
2. Ecosystem for both programming languages is complete and interoperability between Kotlin and Java is easy and seamless, therefore both can use available development tools, libraries, and frameworks. However, for multi platform development, Kotlin is better in framework support as KMM is freely available and fully supported.
3. From a programming language constructs point of view, Kotlin has more advantages and less disadvantages compared with Java. Also, source code for Java is far more verbose than Kotlin which can lead to more bugs and / or code smell.
4. Based on the overall measurement results, it can be concluded that Java should be used if mobile application development priority is the APK size and compilation / build time while Kotlin should be used if mobile application development priority is lesser bugs, concise code, and faster development time.

References

[1] A. Stefik, S. Siebert, K. Slattery, and M. Stefik, "Toward intuitive programming languages," *IEEE Int. Conf. Progr. Compr.*, pp. 213–214, 2011.

[2] B. Frey, J. Doddridge, and C. Seaman, "Chasing the AHA! moment: Exploring initial learnability of programming languages," *Proc. IEEE Symp. Vis. Lang. Human-Centric Comput. VL/HCC*, vol. 2017-Octob, pp. 329–330, 2017.

[3] M. Guzdial, W. M. McCracken, and A. Elliott, "Task specific programming languages as a first programming language," *Proc. - Front. Educ. Conf.*, vol. 3, pp. 1359–1360, 1997.

[4] L. N. Thin and M. H. Husin, "Smart flyers mobile application," *Proceeding - 2017 3rd Int. Conf. Sci. Inf. Technol. Theory Appl. IT Educ. Ind. Soc. Big Data Era, ICSITech 2017*, vol. 2018-Janua, pp. 195–199, 2017.

[5] T. T. Cheng, E. D. Lock, and N. S. Prywes, "Use of Very High Level Languages and Program Generation by Management Professionals," *IEEE Trans. Softw. Eng.*, vol. SE-10, no. 5, pp. 552–563, 1984.

[6] R. Harper, *Practical foundations for programming languages, second edition*. Cambridge University Press, 2016.

[7] S. Tigrek and M. Obadat, "Teaching smartphones programming using (Android Java): Pedagogy and innovation," *2012 Int. Conf. Inf. Technol. Based High. Educ. Training, ITHET 2012*, 2012.

[8] D. Gracanin, M. Matijasevic, and K. P. Valavanis, "Virtual environment testbed for underwater robotics applications," *Int. Conf. Adv. Robot. Proceedings, ICAR*, pp. 793–797, 1997.

[9] C. Cota, L. Aguilar, and G. Licea, "A java compatible virtual machine as an embedded middleware for wireless sensor networks," *Proc. - 2010 IEEE Electron. Robot. Automot. Mech. Conf. CERMA 2010*, pp. 265–270, 2010.

[10] P. Schwermer, "Performance Evaluation of Kotlin and Java on Android Runtime," *Degree Proj. Comput. Sci. Eng.*, 2018.

[11] M. Flauzino, J. Veríssimo, R. Terra, E. Cirilo, V. H. S. Durelli, and R. S. Durelli, "Are you still smelling it?," pp. 23–32, 2018.

[12] D. Gotseva, Y. Tomov, and P. Danov, "Comparative study Java vs Kotlin," *27th Natl. Conf. with Int. Particip. Ways to Connect Futur. TELECOM 2019 - Proc.*, pp. 86–89, 2019.

[13] D. Stepanov, M. Akhin, and M. Belyaev, "ReduKtor: How we stopped worrying about bugs in kotlin compiler," *Proc. - 2019 34th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2019*, pp. 317–326, 2019.

[14] H. ZAYAT, "Kotlin and Android applications : diffusion and adoption of characteristic constructs," POLITECNICO DI TORINO, 2020.

[15] V. Oliveira, L. Teixeira, and F. Ebert, "On the Adoption of Kotlin on Android Development: A Triangulation Study," *SANER 2020 - Proc. 2020 IEEE 27th Int. Conf. Softw. Anal. Evol. Reengineering*, pp. 206–216, 2020.

[16] J. Friesen, *Learn Java for Android Development*. 2013.

[17] Z. Mu, Y. Peng, and Y. Liu, "E-reading system based on android," *Proc. - 2019 12th Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2019*, pp. 487–491, 2019.

[18] A. Pramono, "First Aid Instructional Media using Android Platform," *2017 4th Int. Conf. Comput. Appl. Inf. Process. Technol.*, pp. 3–7, 2018.

[19] L. Rapanotti, J. G. Hall, and Z. Li, "Problem Reduction: a systematic technique for deriving Specifications from Requirements," *IEE Proceedings–Software*, vol. 153, no. 5, pp. 183–198, 2006.

[20] K. Jezek, J. Dietrich, and P. Brada, "How Java APIs break - An empirical study," *Inf. Softw. Technol.*, vol. 65, pp. 129–146, 2015.

[21] D. Larsson and W. Mostowski, "Specifying Java Card API in OCL,"

Electron. Notes Theor. Comput. Sci., vol. 102, pp. 3–19, 2004.

[22] L. Ardito, R. Coppola, G. Malnati, and M. Torchiano, “Effectiveness of Kotlin vs. Java in android app development tasks,” *Inf. Softw. Technol.*, vol. 127, no. July, 2020.

[23] Á. Gamaza, G. Ortiz, J. Boubeta-Puig, and A. Garcia-de-Prado, “REST4CEP: RESTful APIs for complex event processing,” *Sci. Comput. Program.*, vol. 198, p. 102515, 2020.

[24] JetBrains, “Kotlin Language Documentation,” 2016.

[25] M. Martinez and B. G. Mateus, “How and Why did developers migrate Android Applications from Java to Kotlin? A study based on code analysis and interviews with developers,” pp. 1–29, 2020.

[26] S. Samuel and S. Bocutiu, *Programming Kotlin*. 2017.

[27] P. Späth, *Learn Kotlin for Android Development*. 2019.

[28] N. Everlönn, S. Gakis, and A. Nilsson, “Java and Kotlin, a performance comparison,” 2020.

[29] M. Burch *et al.*, “IMDb Explorer: Visual exploration of a movie database,” *ACM Int. Conf. Proceeding Ser.*, pp. 88–91, 2018.

[30] M. Lanham, *Learn ARCore - Fundamentals of Google ARCore : Learn to build augmented reality apps for Android, Unity, and the web with Google ARCore 1.0*. 2018.