RANCANG BANGUN PLUGIN KONVERSI KOTLIN DATA CLASS KE PROTOCOL BUFFERS MESSAGE BERBASIS METODE EXTREME PROGRAMMING

TUGAS AKHIR

DISUSUN OLEH:

FATKHI NUR AKHSAN

NIM. 20106050026

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2024

LEMBAR PENGESAHAN

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-1463/Un.02/DST/PP.00.9/08/2024

Tugas Akhir dengan judul : Rancang Bangun Plugin Konversi Kotlin Data Class ke Protocol Buffers Message Berbasis Metode Extreme Programming

yang dipersiapkan dan disusun oleh:

Nama	: FATKHI NUR AKHSAN
Nomor Induk Mahasiswa	: 20106050026
Telah diujikan pada	: Senin, 12 Agustus 2024
Nilai ujian Tugas Akhir	: A-

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Dr. Agung Fatwanto, S.Si., M.Kom.
SIGNED

Valid ID: 66bf1afaad284

Penguji I

Dr. Ir. Aulia Faqih Rifa'i, M.Kom.
SIGNED

Valid ID: 66bf12eee6f07

Penguji II

Dwi Otik Kurniawati, M.Eng.
SIGNED

Valid ID: 66bece1eb2225

Yogyakarta, 12 Agustus 2024
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 66c2d9cd012e1

LEMBAR PERNYATAAN

SURAT PERNYATAAN KEASLIAN/BEBAS PLAGIASI

Yang bertanda tangan dibawah ini:

Nama	: Fatkhi Nur Akhsan
NIM	: 20106050026
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Dengan ini menyatakan bahwa isi tugas akhir saya yang berjudul "Rancang Bangun Plugin Konversi Kotlin Data Class ke Protocol Buffers Message Berbasis Metode *Extreme Programming*" merupakan hasil pekerjaan penulis sendiri, tidak terdapat karya yang pernah diajukan untuk memperoleh gelar sarjana di suatu Perguruan Tinggi, bukan duplikasi atau saduran dari karya orang lain, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain sepanjang pengetahuan penulis, kecuali yang secara tertulis diacu dalam tugas akhir ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 4 Agustus 2024

Fatkhi Nur Akhsan

NIM. 20106050026

LEMBAR PERSETUJUAN TUGAS AKHIR

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Tugas Akhir
Lamp :

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa tugas akhir Saudara:

Nama : Fatkhi Nur Akhsan
NIM : 20106050026
Judul Tugas Akhir : Rancang Bangun Plugin Konversi Kotlin Data Class ke Protocol Buffers Message Berbasis Metode *Extreme Programming*

Sudah dapat diajukan kembali kepada Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami mengharap agar tugas akhir Saudara tersebut di atas dapat segera di-*munaqasyah*-kan. Atas perhatiannya kami ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta, 5 Agustus 2024

Dr. Agung Fatwanto, S.Si., M.kom.
NIP. 197701032005011003

INTISARI

Perkembangan teknologi, terutama dalam perangkat komputasi mobile dan smartphone, telah mendorong penggunaan aplikasi mobile secara masif. Dua sistem operasi yang mendominasi pasar adalah Android dan iOS. Dalam pengembangan aplikasi Android, Google telah mengadopsi Kotlin sebagai bahasa resmi sejak 2017. Salah satu fitur Kotlin adalah Data Class yang memudahkan penyimpanan data. Namun, penggunaan Jetpack DataStore untuk penyimpanan data di Android memerlukan definisi skema dengan Protocol Buffers, yang memiliki sintaks berbeda dengan Kotlin Data Class. Hal ini menimbulkan tantangan bagi *developer* dalam mengonversi struktur data Kotlin ke Protocol Buffers. Tugas akhir ini bertujuan untuk merancang dan membangun alat yang dapat mengubah Kotlin Data Class menjadi Protocol Buffers Message secara otomatis. Alat ini diimplementasikan sebagai *plugin* untuk Intellij IDEA dan Android Studio, yang diharapkan dapat menyederhanakan proses konversi, meminimalisir kesalahan, dan meningkatkan produktivitas *developer*.

Kata Kunci: Protocol Buffers, Protobuf, Kotlin, Data Class, Konversi, *Plugin*, Pengembangan Perangkat Lunak.

ABSTRACT

The rapid advancement of technology, particularly in mobile computing and smartphones, has led to massive adoption of mobile applications. Android and iOS are the two dominant operating systems. In Android application development, Google adopted Kotlin as the official language in 2017. One of Kotlin's features is Data Class, which facilitates data storage. However, using Jetpack DataStore for data storage in Android requires schema definitions with Protocol Buffers, which have different syntax from Kotlin Data Class. This presents a challenge for developers in converting Kotlin data structures to Protocol Buffers. This thesis aims to design and develop an automated tool to convert Kotlin Data Class to Protocol Buffers Message. The tool is implemented as a plugin for Intellij IDEA and Android Studio, which is expected to simplify the conversion process, minimize errors, and enhance developer productivity.

Keywords: Protocol Buffers, Protobuf, Kotlin, Data Class, Conversion, Plugin, Software Development.

MOTTO

فَإِنَّ مَعَ الْعُسْرِ يُسْرًا ۝٥

Maka, sesungguhnya beserta kesulitan ada kemudahan.

QS. Al-Insyirah 94: Ayat 5

HALAMAN PERSEMBAHAN

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Dengan penuh syukur atas karunia Allah yang Maha Pengasih dan Maha Penyayang, Tugas Akhir ini kupersembahkan dengan penuh rasa hormat dan cinta kepada:

1. Ibu tercinta, Rismawati, yang telah memberikan kasih sayang, doa, dan dukungan tanpa henti. Engkau adalah inspirasi utama dalam setiap langkah hidupku.
2. Bapak tercinta, Budiono, yang telah menjadi tiang penopang kehidupanku. Dengan kebijaksanaan dan kesabaranmu, Engkau telah membimbingku melalui setiap tantangan dan keberhasilan.

Doa dan pengorbanan Ibu dan Bapak adalah cahaya yang membimbing langkah-langkahku menuju kesuksesan. Terima kasih tak terhingga untuk kasih sayang dan pengorbanan tanpa batas kalian. Semoga Allah senantiasa melimpahkan rahmat-Nya.

KATA PENGANTAR

Alhamdulillahi Rabbil 'Alamin, penulis bersyukur kepada Allah *Subhanahu wata'ala* karena berkat Rahmat dan Karunia-Nya, penulis dapat menyelesaikan tugas akhir ini yang berjudul "Rancang Bangun Plugin Konversi Kotlin Data Class ke Protocol Buffers Message Berbasis Metode *Extreme Programming*." Sholawat dan salam selalu diberikan kepada junjungan kita Rasulullah SAW, yang telah menuntun kita ke zaman yang cerah, dan semoga kita semua mendapatkan syafaatnya di akhir zaman.

Penyelesaian tugas akhir ini membutuhkan usaha dan kerja keras dalam proses penyelesaiannya. Namun dalam proses penyelesaiannya tidak lepas dari bantuan dan dukungan berbagai pihak, secara langsung maupun tidak langsung. Dalam kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Prof. Dr.Phil. H. Al Makin, S.Ag., M.A. selaku Rektor UIN Sunan Kalijaga Yogyakarta.
2. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Ibu Ir. Maria Ulfah Siregar, S. Kom, MIT. Ph. D., selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
4. Bapak Muhammad Galih Wonoseto, M.T. selaku dosen penasihat akademik penulis selama kuliah di UIN Sunan Kalijaga.
5. Bapak Dr. Agung Fatwanto, S.Si., M.Kom. selaku pembimbing tugas akhir penulis yang sabar dan memberikan masukan selama penulisan dan penyusunan tugas akhir.
6. Seluruh Dosen dan Karyawan Program Studi Informatika UIN Sunan Kalijaga yang telah memberi ilmu dan bantuan selama perkuliahan penulis.

7. Kedua orang tua penulis Ibu Rismawati dan Bapak Budiono yang telah membesarkan dan merawat penulis hingga dapat menempuh pendidikan tinggi dan menjadi pribadi yang lebih baik.
8. Seluruh Angkatan Informatika UIN Sunan Kalijaga terutama teman-teman angkatan 2020 yang telah menjadi teman seperjuangan dan saling mendukung pada saat perkuliahan.

Tugas akhir ini pasti tidak terlepas dari kekurangan kualitas dan kuantitas materi. Untuk itu, mohon kritik dan saran yang konstruktif untuk membantu penulis memperbaiki kekurangannya. Semoga tugas akhir ini dapat memberikan manfaat dan dapat dikembangkan untuk ke depannya.

Yogyakarta, 2 Agustus 2024

Penulis

Fatkhi Nur Akhsan

NIM. 20106050026

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi dalam beberapa dekade terakhir sangatlah pesat, salah satu teknologi yang berkembang pesat adalah perangkat komputasi mobile dan smartphone dalam 3 dekade terakhir. Saat ini perangkat genggam sudah memiliki daya komputasi yang sama besarnya dengan ruangan yang penuh dengan komputer pada tahun 1970-an. Perkembangan daya komputasi tersebut tidak terlepas dari peningkatan tajam jumlah transistor yang dapat ditampung dalam sebuah *chip* akan berlipat ganda setiap dua tahun, sesuai dengan hukum Moore [1].

Meskipun memiliki kecepatan proses dan konektivitas yang masih lebih lambat dibandingkan dengan perangkat dekstop, penggunaan perangkat mobile dan aplikasi mobile lebih dipilih oleh pengguna dibandingkan dengan perangkat dekstop terutama untuk menyelesaikan pekerjaan yang mudah [2]. Salah satunya yaitu aplikasi *mobile-commerce* yang marak digunakan ketika masa pandemi COVID-19, penggunaan aplikasi *mobile-commerce* juga diprediksi masih berlanjut saat periode endemi [2]. Senada dengan hal itu, laporan dari Cisco menunjukkan bahwa hampir 300 miliar aplikasi seluler akan diunduh pada tahun 2023, dengan aplikasi berjenis media sosial, game, dan bisnis akan menjadi unduhan yang paling populer [3].

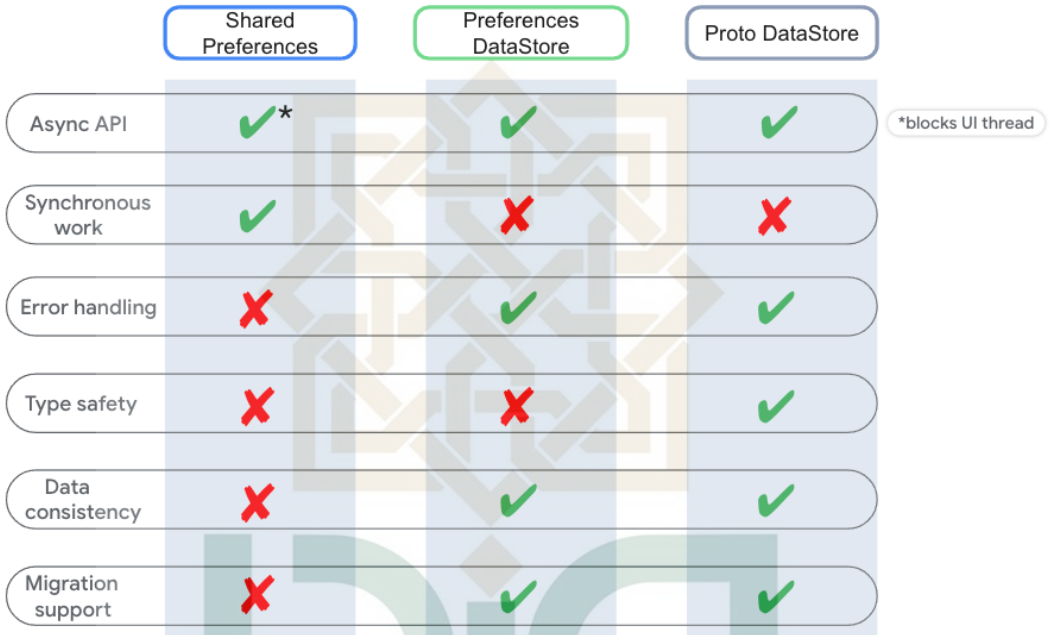
Diantara penggunaan masif perangkat mobile terdapat dua sistem operasi yang sudah mendominasi pasar dengan cakupan 90% dari market, yaitu Android (Google, Mountain View, CA) and iOS (Apple) sejak tahun 2008 [1]. Untuk mendukung produktifitas *developer*, Google mengumumkan dukungannya pada Google I/O 2017 untuk menjadikan bahasa pemrograman Kotlin sebagai bahasa resmi untuk pengembangan aplikasi Android [4]. Setelah dua tahun berselang, tepatnya saat Google I/O 2019, pengembangan aplikasi mobile Android akan menjadi *kotlin-first* setelah Google mengumumkannya, hal ini berarti Kotlin akan

menjadi pilihan pertama Google dalam mengembangkan alat dan konten pengembangan aplikasi Android [5].

Bahasa pemrograman Kotlin sendiri merupakan bahasa pemrograman open source berjenis statis yang mendukung pemrograman berorientasi objek dan fungsional [6]. Kotlin pada mulanya dirancang oleh JetBrains, kemudian terus dikembangkan oleh para kontributor dan dikelola oleh Kotlin Foundation yang didirikan oleh JetBrains dan Google [7]. Meskipun tergolong bahasa yang modern, Kotlin juga dapat dikatakan bahasa yang sudah matang. Bahasa ini ringkas, aman, dapat dioperasikan dengan Java dan bahasa lain, dan menyediakan banyak cara untuk menggunakan kembali kode di antara berbagai platform untuk pemrograman yang produktif [8]. Oleh karena itu, beberapa API pada Android seperti AndroidKTX yang dikhususkan untuk Kotlin tetap dapat kompatibel dengan Bahasa Java yang sudah lebih dulu menjadi bahasa resmi untuk pengembangan aplikasi Android. Dengan interoperabilitas yang baik dengan Java tersebutlah, Kotlin berhasil menghasilkan pengalaman pengembangan aplikasi Android yang lebih nyaman [6].

Salah satu library yang memiliki AndroidKTX sebagai ekstensi adalah library Jetpack DataStore. Jetpack DataStore adalah salah satu library yang menyediakan solusi penyimpanan data lokal yang terdapat pada platform Android, khususnya untuk kumpulan data yang kecil dan sederhana dan tidak memerlukan pembaruan parsial atau integritas referensial. Jetpack DataStore menggunakan Kotlin Coroutines dan Flow untuk menyimpan data secara asinkron, konsisten, dan transaksional menjadikannya lebih unggul untuk pengembangan aplikasi Android modern dibandingkan dengan solusi penyimpanan lain yang sejenis seperti Shared Preferences. Jetpack DataStore dapat menyimpan data berupa pasangan *key-value* atau objek yang ditulis menggunakan Protocol Buffers. Oleh karena itu, Jetpack DataStore dibagi menjadi dua jenis, yaitu Preferences DataStore, yang menyimpan dan mengakses data menggunakan kunci dan Proto DataStore, yang menyimpan data berupa instansiasi objek yang sebelumnya telah didefinisikan menggunakan *schema* yang ditulis menggunakan Protocol Buffers [9]. Sehingga apabila data yang disimpan berupa kelas-kelas yang kompleks, dianjurkan untuk menggunakan Proto

DataStore sehingga dapat memanfaatkan fitur *type safety* yang dimilikinya, sebaliknya, dianjurkan untuk menggunakan Preferences DataStore apabila yang disimpan merupakan data sederhana yang tidak terlalu membutuhkan fitur *type safety*, seperti data dengan tipe *basic* seperti integer, float, dan boolean. Hal-hal tersebut dapat digambarkan pada Gambar 1.1.1 dibawah [30].



Gambar 1.1.1 Perbandingan implementasi DataStore dengan SharedPreferences

Protocol Buffers atau yang juga dapat disingkat dengan protobuf, merupakan mekanisme yang dapat diekstensi untuk membuat serialisasi data terstruktur yang *language-neutral* dan *platform-neutral*. Protocol Buffers sendiri unggul dalam hal ukuran dan kecepatan dibandingkan dengan format data lain seperti JSON dan XML, dan dapat menghasilkan binding bahasa asli. Protocol Buffers adalah kombinasi bahasa definisi (dibuat dalam *file* .proto), kode yang dibuat oleh kompiler proto untuk berinteraksi dengan data, pustaka runtime yang *language-specific*, format serialisasi untuk data yang ditulis ke *file* (atau dikirim melalui koneksi jaringan), dan data yang terserialisasi [10].

Salah satu cara untuk menampung data pada bahasa pemrograman Kotlin adalah dengan menggunakan Data Class yang ditandai dengan keyword “data”. Untuk setiap Data Class, kompiler akan secara otomatis menghasilkan fungsi tambahan yang memungkinkan untuk mencetak *instance* ke *output* yang dapat

dibaca, membandingkan *instance*, menyalin *instance*, dan banyak lagi. Hal ini mengurangi kode *boilerplate* yang dihasilkan dibandingkan dengan mekanisme penampungan data yang terdapat pada bahasa pemrograman lain, seperti POJO pada bahasa Java [11]. Data Class sering digunakan untuk merepresentasikan bisnis model dari sistem atau aplikasi yang menggunakan bahasa pemrograman Kotlin sebagai bahasa pengembangan seperti pada pengembangan aplikasi Android modern [31]. Sama halnya dengan Data Class, Protocol Buffers juga erat berkaitan dengan data-data terutama untuk mendefinisikan protokol komunikasi (bersama dengan gRPC) dan untuk penyimpanan data. Struktur data pada Protocol Buffers didefinisikan di *file-file* dengan ekstensi .proto yang dibuat oleh engineer. Pada *file* tersebut service-service dan message-message didefinisikan. Message inilah yang dijadikan schema untuk memperoleh kode dan runtime library yang lebih *language-specific* sehingga dapat digunakan pada berbagai bahasa dan *framework* pemrograman, khususnya dalam konteks ini yaitu untuk menyimpan data menggunakan Jetpack Proto DataStore [10].

Meskipun keduanya, yaitu Kotlin Data Class dan Protocol Buffers memiliki beberapa persamaan. Namun, keduanya memiliki perbedaan sintaks, hal ini menjadi permasalahan dan tantangan tersendiri bagi *developer* Android yang ingin menggunakan Jetpack Proto DataStore sebagai solusi penyimpanan lokal pada aplikasi yang dikembangkan. Semakin kompleks struktur data yang dipakai tentunya akan menambahkan tingkat kesulitan yang dialami *developer* dalam merepresentasikannya kedalam bentuk skema yang dibuat dari Protocol Buffers Message. Bahkan, ketika struktur data tersebut tidaklah kompleks pengembang yang belum pernah memakai Jetpack Proto DataStore tetap perlu untuk mempelajari Protocol Buffers. Proses pembuatan skema inilah yang seringkali memakan waktu dan rentan terhadap kesalahan, terutama ketika berhadapan dengan struktur data yang rumit dikarenakan prosesnya yang masih manual [12]. Permasalahan tersebutlah yang menciptakan kebutuhan akan sebuah alat yang dapat mengotomatisasi proses perubahan struktur data yang berasal dari Kotlin Data Class menjadi struktur data berupa skema yang didefinisikan melalui Protocol Buffers Message dengan cepat dan efisien. Kebutuhan tersebut sangat penting bagi

pengembang yang perlu bekerja dengan data yang sudah ada dalam bentuk struktur data Kotlin Data Class kemudian ingin memperolehnya dalam bentuk Protocol Buffers Message dengan cepat dan efisien.

Kebutuhan akan alat konversi dari Kotlin ke Protocol Buffers ini meskipun kecil tetapi ada misalnya pada beberapa developer yang menyampaikan kebutuhannya pada *section* Issues nomor 34 di GitHub repository dari "kotlinx.serialization", seperti *developer* dengan *username* GitHub "elizarov" yang membutuhkan fungsionalitas tersebut sehingga dapat menjadikan Kotlin sebagai "master" *code* dari API-nya kemudian memanfaatkan hasil konversi berupa *file* ".proto" yang telah dihasilkan untuk dipakai pada bahasa pemrograman lainnya, selain itu terdapat juga *developer* dengan *username* GitHub "apatrida" yang menyampaikan pendapatnya terkait fungsionalitas tersebut akan memungkinkan penggunaan Kotlin bersama dengan GRPC yang membutuhkan *file* ".proto", demikian juga dengan dua *developer* dengan *username* GitHub "IgorKey" dan "RdeWilde" yang menyampaikan kebutuhan konversi dari Kotlin ke Protocol Buffers namun tidak menjelaskan alasannya [34]. Lalu, pada Issues nomor 477 di repository yang sama, *developer* dengan *username* GitHub "mikezliu" yang mempunyai *project* untuk Front-End (vue) dan iOS yang sebelumnya menggunakan Kotlin Data Class sebagai satu *source of truth* kemudian menginginkan untuk menggantinya dengan ".proto" *files* sehingga membutuhkan alat untuk konversi dari Kotlin Data Class ke Protocol Buffers, dan juga pada Issues yang sama terdapat *developer* dengan *username* GitHub "Marlinski" yang menyampaikan ketertarikannya akan *generator* untuk .proto files dari Kotlin Data Class sehingga dapat menyelesaikan masalahnya terkait proyek pada Back-End berbahasa pemrograman Golang yang membutuhkan Kotlin Data Class dengan cara melakukan konversi Kotlin Data Class ke .proto files terlebih dahulu sebelum memakainya pada Back-End yang berbahasa pemrograman Golang tersebut [35]. Kemudian juga terdapat *developer* dengan *username* GitHub "dzielins42" yang membutuhkan fungsionalitas tersebut untuk digunakan pada proyek IoT bernama "Nanopb" sehingga dapat membuat ".proto" *files* dari Kotlin [36].

Selain itu, kebutuhan akan alat yang dapat membuat *schema* Protocol Buffers dari sebuah *class* juga dapat ditemukan pada bahasa pemrograman lain. Seperti pada bahasa pemrograman Dart, dimana perusahaan SquareAlfa membuat sebuah *library* bernama "proto_generator" yang dapat mengubah kelas-kelas PODOs (plain-old-dart-objects) yang digunakan sebagai *business model* menjadi Protocol Buffer Messages, library ini juga telah mencapai 56% popularity yang menjadi ukuran seberapa banyak pengembang yang menggunakan paket tersebut selama 60 hari terakhir [37]. Lalu, juga terdapat perusahaan bernama "Infinispan" yang membuat *library* bernama "ProtoStream" untuk menserialisasikan Java berdasarkan format data pada Protocol Buffer [38]. API pada library tersebut sudah digunakan oleh perusahaan perangkat lunak RedHat untuk mengubah *objects* yang berasal dari *classes* pada bahasa pemrograman Java menjadi ".proto" *schema* pada salah satu produknya yang bernama "Red Hat Data Grid" yaitu sebuah penyimpanan data dalam memori yang terdistribusi dan berkinerja tinggi [39]. Alat yang dapat membuat *schema* Protocol Buffers dari sebuah *class* pada bahasa pemrograman lain tersebut menambah alasan penulis untuk membuat alat dengan fungsionalitas yang hampir sama yaitu membuat *schema* Protocol Buffers, namun kali ini dari Kotlin Data Class.

Untuk mengatasi permasalahan dan memenuhi kebutuhan dari pengembang tersebut, diperlukan sebuah alat yang dapat mengubah struktur atau model data dari Kotlin Data Class ke dalam bentuk Protocol Buffers Message secara otomatis. Alat tersebut diharapkan dapat mengatasi permasalahan dan memenuhi kebutuhan sekaligus menghadirkan beberapa manfaat bagi para pengembang, terutama pengembang aplikasi Android, seperti: menyederhanakan dan mempercepat proses pembuatan Protocol Buffers Message sebagai skema data yang sebelumnya sudah pernah dibuat dalam bentuk Kotlin Data Class, meminimalisir kesalahan yang mungkin terjadi selama proses perubahan dan pembuatan skema, memudahkan pengembang dalam memelihara kode program [13]. Penggunaan alat konversi otomatis juga dapat meningkatkan konsistensi dalam struktur kode sehingga praktik-praktik kode yang baik dapat dijalankan dengan lebih mudah, memfasilitasi pembaruan yang lebih mudah ketika skema data berubah, dan memungkinkan

pengembang untuk lebih berfokus pada logika bisnis inti daripada tugas-tugas transformasi data yang repetitif [14].

Terdapat dua *plugin* yang menjadi inspirasi dalam membuat perancangan dan pembangunan alat yang dapat mengubah Kotlin Data Class menjadi Protocol Buffers Message secara otomatis dalam bentuk *plugin* yang dapat dipasang dan berjalan pada Intellij IDEA dan Android Studio. Pertama, RoboPOJOGenerator, yaitu *plugin* yang dapat menghasilkan *file* Java, Java Records, dan Kotlin POJO dari JSON: GSON, FastJSON, AutoValue (GSON), Logan Square, Jackson, Lombok, Jakarta JSON Binding, dan *empty annotations template* [27]. Kedua, pojo to proto, yaitu *plugin* yang dapat menghasilkan google protobuf message dari Java POJO *class* [28]. Namun, keduanya tidak melakukan konversi dari Kotlin Data Class menjadi Protocol Buffers Message. Juga, belum ditemukan *plugin* untuk IDE Intellij IDEA maupun Android Studio dengan fungsionalitas yang sama sejauh pencarian penulis saat tugas akhir ini dibuat.

Hal-hal tersebutlah yang memotivasi penulis untuk merancang dan membangun alat yang dapat mengubah Kotlin Data Class menjadi Protocol Buffers Message secara otomatis dalam bentuk *plugin* yang dapat dipasang dan berjalan pada Intellij IDEA dan Android Studio, dimana Intellij IDEA sendiri merupakan Integrated Development Environment (IDE) terdepan dalam pengembangan yang menggunakan bahasa Kotlin [15], kemudian Android Studio sebagai IDE Resmi untuk mengembangkan aplikasi Android [16].

1.2 Rumusan Masalah

Dari latar belakang yang telah diuraikan, masalah dapat dirumuskan sebagai berikut: “Bagaimana cara merancang dan membangun *plugin* yang dapat mengubah struktur data pada Kotlin Data Class menjadi skema yang didefinisikan melalui Protocol Buffers Message berbasis metode *Extreme Programming*?”

1.3 Tujuan

Dari rumusan masalah yang telah disebutkan, tujuan yang ingin dicapai adalah untuk merancang dan membangun alat yang dapat mengubah struktur data pada Kotlin Data Class menjadi skema yang didefinisikan melalui Protocol Buffers Message berupa *plugin* berbasis metode *Extreme Programming* yang dapat dipasang dan dijalankan pada Intellij IDEA dan Android Studio.

1.4 Manfaat

Hasil dari rancang bangun ini diharapkan dapat memberikan beberapa manfaat sebagai berikut:

1. Membantu pengembang aplikasi Android yang menggunakan bahasa pemrograman Kotlin dalam membangun aplikasi Android yang menggunakan *library* Jetpack Proto DataStore menjadi lebih efisien, cepat, dan mudah dipelihara.
2. Meningkatan produktivitas *developer* dengan mempercepat proses pembuatan skema pada Protocol Buffers melalui proses perubahan yang diperoleh dari struktur data pada Kotlin Data Class, menghemat waktu *developer* sehingga dapat berfokus pada logika bisnis inti, dan meningkatkan efisiensi *workflow*.
3. Berkontribusi pada pengembangan bahasa pemrograman Kotlin, aplikasi Android, dan ekosistem keduanya.
4. Menyediakan studi kasus, membuka peluang penelitian optimasi konversi Protocol Buffers Message dan pengembangan *tools*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Plugin konversi struktur data pada Kotlin Data Class menjadi skema yang didefinisikan melalui Protocol Buffers Message yang dirancang dan dibangun mampu untuk melakukan konversi dengan cakupan minimal berupa seluruh tipe data skalar secara akurat. Selain itu, *plugin* yang dirancang dan dibangun menyediakan fungsionalitas untuk pengguna dapat menentukan nama *file output* dan konfigurasi tipe data *integer/numeric* sesuai yang diinginkan.

Plugin ini dapat dimanfaatkan oleh *developer* sehingga dapat menghemat waktu, meminimalisir kesalahan, meningkatkan konsistensi, dan memfokuskan diri pada logika bisnis aplikasi daripada tugas-tugas transformasi data yang repetitif. Hal-hal tersebut berpotensi untuk meningkatkan kualitas dan stabilitas serta memudahkan pemeliharaan sistem yang menggunakan Kotlin dan Protocol Buffers. Kemudahan dalam menggunakan *plugin* juga berpotensi mendorong adopsi teknologi yang menggunakan Protocol Buffers seperti *library* Jetpack Proto DataStore yang digunakan pada pengembangan aplikasi Android. Dengan demikian diharapkan perancangan dan pembangunan *plugin* ini dapat memberikan kontribusi positif bagi komunitas *developer* Kotlin dan Protocol Buffers beserta ekosistemnya sehingga dapat meningkatkan kualitas pengembangan sistem yang memakainya.

5.2 Saran

Beberapa saran yang dapat diterapkan untuk pengembangan *plugin* di masa depan antara lain:

1. Meningkatkan antarmuka pengguna pada *plugin* sehingga dapat lebih intuitif dan ramah pengguna.
2. Menambahkan dukungan konversi *field* bertipe well-known dan common.
3. Menambahkan dukungan untuk konversi dari Kotlin Array atau Collections dengan lebih dari satu dimensi.

4. Dukungan konversi dari bahasa pemrograman selain Kotlin.

DAFTAR PUSTAKA

[1] B. S. Rothman, R. K. Gupta, and M. D. McEvoy, "Mobile Technology in the Perioperative Arena: Rapid Evolution and Future Disruption," *Anesth. Analg.*, vol. 124, no. 3, pp. 807–818, 2017, doi: 10.1213/ANE.0000000000001858.

[2] M. Ashoer, M. H. Syahnur, J. S. Tjan, A. Junaid, A. Pramukti, and A. Halim, "The Future of Mobile Commerce Application in a Post Pandemic Period; An Integrative Model of UTAUT2," *E3S Web Conf.*, vol. 359, pp. 1–8, 2022, doi: 10.1051/e3sconf/202235905005.

[3] "Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper," *Cisco*, Jan. 2022. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed Jul. 04, 2024).

[4] Google for *Developers*. *Google I/O Keynote (Google I/O '17)*. (May 17, 2017). Accessed: Jul. 04, 2024. [Online video]. Available: https://www.youtube.com/live/Y2VF8tmLFHw?si=zgCrETDalg1xflXH&t=5230.

[5] "Android's Kotlin-first approach," Android *Developers*, 2019. https://*developer*.Android.com/Kotlin/first (accessed Jul. 04, 2024).

[6] "Ringkasan Kotlin," Android *Developers*, 2023. https://*developer*.Android.com/Kotlin/overview?hl=id (accessed Jul. 04, 2024).

[7] "Kotlin Foundation – official site," *Kotlin Foundation – official site*, 2023. https://Kotlinfoundation.org/ (accessed Jul. 04, 2024).

[8] "Kotlin Programming Language," *Kotlin*, 2024. https://Kotlinlang.org/ (accessed Jul. 04, 2024).

[9] "App Architecture: Data Layer - DataStore - Android *Developers*," Android *Developers*, 2024.

https://*developer*.Android.com/topic/libraries/architecture/datastore (accessed Jul. 04, 2024).

[10] Protocol Buffers, "Protocol Buffers," *Protobuf.dev*, 2023. https://protobuf.dev/ (accessed Jul. 04, 2024).

[11] Kotlin, "Data classes," Kotlin Documentation, 2024. https://Kotlinlang.org/docs/data-classes.html (accessed Jul. 04, 2024).

[12] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real Challenges in Mobile App Development," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, 2013, pp. 15–24, doi: 10.1109/ESEM.2013.9.

[13] M. Voelter, *Generic Tools, Specific Languages*. Delft University of Technology, 2014.

[14] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, vol. 1. 2012.

[15] JetBrains, "IntelliJ IDEA," *JetBrains*, 2024. https://www.jetbrains.com/idea/ (accessed Jul. 04, 2024).

[16] "Download Android Studio & App Tools - Android *Developer*s," Android *Developers*, 2024. https://*developer*.Android.com/studio (accessed Jul. 04, 2024).

[17] "Kotlin Help," *Kotlin Help*, 2016. https://Kotlinlang.org/docs/faq.html (accessed Jul. 06, 2024).

[18] "Kotlin *Developer* Stories," Android *Developers*, 2024. https://*developer*.Android.com/Kotlin/stories (accessed Jul. 06, 2024).

[19] H. Kolengsusu, "Rancang Bangun *Plugin* untuk Sistem Informasi Akademik dengan Ajax dan Web Services," *BIMAFIKA J. MIPA, Kependidikan dan Terap.*, vol. 4, no. 1, pp. 425–434, 2012, [Online]. Available: https://unidar.e-journal.id/bima/article/view/190.

[20] A. Shrivastava, I. Jaggi, N. Katoch, D. Gupta, and S. Gupta, "A Systematic Review on Extreme Programming," *J. Phys. Conf. Ser.*, vol. 1969, no. 1, 2021, doi: 10.1088/1742-6596/1969/1/012046.

[21] A. Restu Mukti, C. Mukmin, E. Randa Kasih, D. Palembang Jalan Jenderal Ahmad Yani No, S. I. Ulu, and S. Selatan, "Perancangan Smart Home Menggunakan Konsep Internet of Things (IOT) Berbasis Microcontroller," *J. JUPITER*, vol. 14, no. 2, pp. 516–522, 2022.

[22] "What is UML | Unified Modeling Language," *Uml.org*, 2024. https://www.uml.org/what-is-uml.htm (accessed Jul. 16, 2024).

[23] "What is Wireframing? The Complete Guide [Free Checklist]| Figma," *Figma*, 2022. https://www.figma.com/resource-library/what-is-wireframing/ (accessed Jul. 16, 2024).

[24] freeCodeCamp.org. *UML Diagrams Full Course (Unified Modeling Language)*. (April 22, 2021). Accessed: Jul. 16, 2024. [Online video]. Available: https://www.youtube.com/watch?v=WnMQ8HlmeXc&t=3869s

[25] A. Nordeen, *Learn Software Testing in 24 Hours: Definitive Guide to Learn Software Testing for Beginners*. Guru99, 2020.

[26] A. Verma, A. Khatana, and S. Chaudhary, "A Comparative Study of Black Box Testing and White Box Testing," *Int. J. Comput. Sci. Eng.*, vol. 5, no. 12, pp. 301–304, 2017, doi: 10.26438/ijcse/v5i12.301304.

[27] "RoboPOJOGenerator - IntelliJ IDEs *Plugin* | Marketplace," *JetBrains Marketplace*, 2024. https://*plugins*.jetbrains.com/*plugin*/8634-robopojogenerator (accessed Jul. 22, 2024).

[28] "pojo to proto - IntelliJ IDEs *Plugin* | Marketplace," *JetBrains Marketplace*, 2024. https://*plugins*.jetbrains.com/*plugin*/14691-pojo-to-proto (accessed Jul. 22, 2024).

[29] "Language Specification | The Protobuf Language," *Protobuf.com*, 2023. https://protobuf.com/docs/language-spec#identifiers-and-keywords (accessed Aug. 01, 2024).

[30] S. Milanović, "Introduction to Jetpack DataStore - Android Developers - Medium," *Medium*, Jan. 18, 2022. https://medium.com/androiddevelopers/introduction-to-jetpack-datastore-3dc8d74139e7 (accessed Aug. 13, 2024).

[31] "Data layer," *Android Developers*, 2024. https://developer.android.com/topic/architecture/data-layer (accessed Aug. 13, 2024).

[32] R. I. Borman, A. T. Priandika, and A. R. Edison, "Implementasi Metode Pengembangan Sistem Extreme Programming (XP) pada Aplikasi Investasi Peternakan," *J. Sist. dan Teknol. Inf.*, vol. 8, no. 3, p. 272, 2020, doi: 10.26418/justin.v8i3.40273.

[33] GeeksforGeeks, "Class-Responsibility-Collaboration Card," *GeeksforGeeks*, Apr. 10, 2024. https://www.geeksforgeeks.org/class-responsibility-collaboration-card/ (accessed Aug. 13, 2024).

[34] "Generator for .proto files based on serializable Kotlin classes · Issue #34 · Kotlin/kotlinx.serialization," *GitHub*, Nov. 24, 2017. https://github.com/Kotlin/kotlinx.serialization/issues/34 (accessed Aug. 16, 2024).

[35] "Is there any kotlin class generator based on .proto file available? · Issue #477 · Kotlin/kotlinx.serialization," *GitHub*, May 31, 2019. https://github.com/Kotlin/kotlinx.serialization/issues/477 (accessed Aug. 16, 2024).

[36] "How to generate .proto file from annotated class during build? · Issue #2498 · Kotlin/kotlinx.serialization," *GitHub*, Nov. 08, 2023. https://github.com/Kotlin/kotlinx.serialization/issues/2498 (accessed Aug. 16, 2024).

[37] "proto_generator," *Dart packages*, Mar. 21, 2021. https://pub.dev/packages/proto_generator (accessed Aug. 16, 2024).

[38] "infinispan/protostream: ProtoStream is a serialization library based on Protocol Buffers," *GitHub*, Aug. 13, 2024. https://github.com/infinispan/protostream (accessed Aug. 16, 2024).

[39] "Data Grid Developer Guide | Red Hat Product Documentation," *Redhat.com*, 2024. https://docs.redhat.com/en/documentation/red_hat_data_grid/8.0/html/data_grid_developer_guide/ (accessed Aug. 16, 2024).