Perancangan Back-end pada Startup SIABDes TAXion dengan Arsitektur Modular Monolitik

Ahmad Fathan Hanif
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
ahmadfathanhanif@student.telkomuniversity.ac.id

Arfive Gandhi
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
arfivegandhi@telkomuniversity.ac.id

Koenta Adji Koerniawan
Fakultas Ekonomi dan Bisnis
Universitas Telkom
Bandung, Indonesia
koentaadji@telkomuniversity.ac.id

***Abstrak* — Sistem Informasi Aplikasi Badan Usaha Milik Desa (SIABDes) sudah melakukan pengembangan Minimum Viable Product (MVP). MVP ini telah diuji pada Badan Usaha Milik Desa (BUMDes) sekitar kota Bandung. MVP ini mendapatkan umpan balik, berupa perubahan fitur pencatatan laporan keuangan dan perpajakan PPN dan PPh 21. Selain perubahan fitur, MVP ini memiliki isu keamanan, isu kesulitan pemeliharaan sumber kode dan tidak terdapat implementasi pengujian unit. Hal ini berpotensi menyebabkan kegagalan sistem, jika diadakan penambahan fitur atau pemeliharaan kode. Selain itu, SIABDes juga memiliki target pengguna 270 BUMDes. Berdasarkan tantangan tersebut, rekonstruksi menyeluruh dapat dilakukan agar memudahkan pemeliharaan dan penambahan fitur aplikasi di masa depan. Agar pemodelan rekonstruksi terstruktur dengan baik, digunakan arsitektur modular monolitik. Arsitektur ini berfokus pada kecepatan pengembangan dan modularitas kode, sehingga memudahkan pemeliharaan dan mudah di implementasikan pada tim kecil. Jenis pengujian pada implementasi arsitektur ini, dilakukan dengan pengujian performa. Pengujian ini dilakukan dengan menyimulasikan beban pengguna 270 BUMDes. Berdasarkan pengujian ini, dapat menerima beban pengguna tanpa peningkatan performa yang signifikan.**

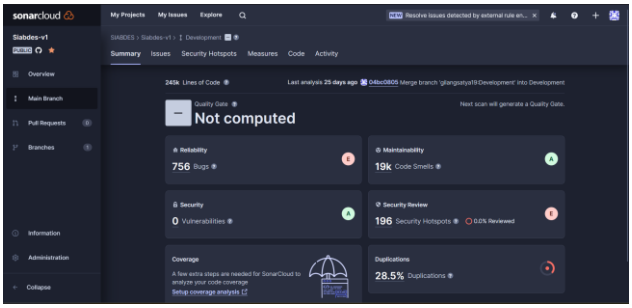
***Kata kunci*— modular monolitik, BUMDes, refactoring, pengujian performa**

I. PENDAHULUAN

Sistem Informasi Akuntansi Badan Usaha Milik Desa (SIABDes) adalah startup sekaligus aplikasi pencatatan laporan keuangan. Aplikasi ini dikembangkan oleh Fakultas Informatika dan Fakultas Ekonomi Bisnis Universitas Telkom. Aplikasi ini dicetuskan oleh founder SIABDes Dr. Koenta Adji Koerniawan. SIABDes menggunakan Standar Akuntansi Keuangan Entitas Mikro Kecil Menengah (SAK EMKM) yang kompatibel dengan kasus badan usaha milik desa (BUMDes). Startup SIABDes juga mengikuti program inkubasi WRAP Entrepreneurship yang diselenggarakan Bandung Techno Park. Pada program ini, terutama anggota startup, digilir setiap tahun.

Tim pengembang SIABDes mulai mengembangkan Minimum Viable Product (MVP) pada 4 April 2023. Pada bulan Mei 2023, tim pengembang SIABDes berhasil mengembangkan MVP sampai implementasi pencatatan laporan keuangan. MVP ini telah diujikan dan divalidasi pada BUMDes sekitar kota Bandung. Hasil pengujian MVP SIABDes ini mendapat beberapa umpan balik. Umpan balik yang pertama adalah User Experience (UX) yang kurang baik untuk pengguna awam akuntansi [1]. Kedua, terdapat BUMDes yang memiliki beberapa unit yang data per unit harus diisolasi, sehingga pencatatan keuangan setiap unit perlu dipisah [2]. Ketiga, tidak ada fitur perpajakan untuk memudahkan perhitungan pajak barang dan karyawan [1].

Selain dari sisi bisnis SIABDes, terdapat berbagai permasalahan pada tim pengembang SIABDes. Masalah pertama dalam tim pengembang SIABDes adalah tidak terdokumentasinya Spesifikasi Perangkat Lunak (SKPL). Hal ini menyebabkan tim pengembang sering melakukan revisi dan validasi kepada tim bisnis, sehingga memperlambat pengembangan. Selain tidak terdokumentasi SKPL, pengembangan MVP ini juga dalam kondisi tergesa-gesa dalam durasi satu minggu, sehingga kualitas kode tidak diimplementasikan dengan baik. Masalah kedua adalah Quality Assurance (QA) Engineer hanya bekerja sebagai pelengkap SKPL, sehingga proses pengujian tidak berjalan dengan baik. Ketiga proses deployment SIABDes ke production dilakukan secara manual. Hal ini juga mengurangi efisiensi dan dapat menyebabkan human error. Proses deployment secara manual berpotensi terjadi human error ketika terdapat pembaruan sistem atau fitur.



GAMBAR 1
LAPORAN HASIL ANALISIS KODE

Berdasarkan laporan analisis kode pada Gambar 1 menggunakan Sonar Cloud, ditemukan bahwa 642 code smells pada sumber kode MVP SIABDes yang ditinggalkan oleh tim pengembang MVP SIABDes. Isu code smells ini wajar dalam sumber kode yang bersifat MVP. Sumber kode MVP SIABDes menggunakan Laravel versi 9.19. Pada versi ini terdapat isu keamanan yang dilaporkan terkait otentikasi dan otorisasi [3]. Selain isu keamanan pada versi Laravel, terdapat isu halaman masih dapat mengakses halaman dashboard tanpa melakukan login. Hal ini menandakan bahwa tidak terdapat komponen pemeriksa bahwa pengguna sudah login atau belum. Pada sisi basis data, MVP SIABDes terdapat isu mengenai kualitas skalabilitas dalam pemodelan tabel jurnal umum. Tabel ini menyimpan hasil kalkulasi pada setiap akun yang digunakan dalam jurnal. Hal ini menjadi isu penting dalam konsistensi data. Jika terdapat kesalahan input dari sisi user, tidak dapat dilakukan rollback pada data, dikarenakan tidak terdapat riwayat transaksi pada jurnal. Pada sisi kebutuhan bisnis, founder SIABDes juga memperkirakan pada tahun 2024 akan ada 270 BUMDes kabupaten Bandung yang akan menggunakan SIABDes.

Selain faktor kurangnya dokumentasi dan technical debt pada aplikasi MVP, terdapat beberapa isu dalam arsitektur aplikasi MVP. Aplikasi MVP SIABDes menggunakan framework Laravel yang berbasis pola arsitektur Model View Controller (MVC). Dalam MVC terdapat komponen View sebagai representasi antarmuka pengguna, komponen Model sebagai representasi data dan Controller sebagai orkestrator logika bisnis. Dalam implementasi MVC pada aplikasi MVP SIABDes, komponen antarmuka dan logika bisnis tidak terisolasi dengan baik sesuai dengan tanggung jawab komponen yang semestinya. Hal ini menjadi penting karena artefak proyek tidak tersedia secara lengkap. Terdapat contoh sumber kode pada MVP SIABDes, yang melakukan kalkulasi kompleks pada komponen View untuk halaman Neraca Lajur dan Buku Besar. Tanpa adanya artefak tentang bagaimana Neraca Lajur dan Buku Besar diproses, dapat menyulitkan pemeliharaan atau bahkan pengembangan fitur baru. Hal ini juga menjadi kurang baik, karena pemrosesan Neraca Lajur dan Buku Besar pada komponen View tidak dapat digunakan kembali, sehingga mengakibatkan duplikasi kode dan tidak memiliki sumber perhitungan yang sama. Dalam sistem informasi, konsistensi data dari sisi pemrosesan dan penyimpanan harus dijaga, sehingga penggunaan prinsip penggunaan ulang sebuah logika bisnis harus diterapkan, agar menjaga konsistensi pemrosesan data untuk keseluruhan sistem.

Berdasarkan hal tersebut, peneliti menyarankan dilakukan rekonstruksi secara menyeluruh pada aplikasi SIABDes, dengan dipisahkan komponen antarmuka dan logika bisnis menjadi dua aplikasi berbeda. Aplikasi komponen antarmuka merupakan aplikasi front-end, sedangkan komponen yang berfokus pada orkestrasi logika bisnis adalah aplikasi back-end. Rekonstruksi aplikasi SIABDes disarankan, karena penambahan fitur dilakukan, sehingga kualitas kemudahan dalam pemeliharaan kode diperlukan. Selain itu, refactoring tanpa pengujian unit berpotensi untuk terjadinya fault yang tidak terdeteksi. Oleh karena itu, rekonstruksi secara menyeluruh masih dapat dilakukan, karena MVP SIABDes masih dalam tahap uji coba dan belum digunakan untuk pencatatan keuangan sehari-hari oleh BUMDes Dengan belum adanya penggunaan secara aktif, rekonstruksi secara menyeluruh dapat dilakukan dengan risiko yang minimal.

Untuk membedakan antara MVP SIABDes dengan hasil rekonstruksi MVP tersebut, penerus dari MVP SIABDes diberi nama SIABDes TAXion. SIABDes TAXion berfokus pada pengembangan fitur perpajakan dan perubahan sistem pencatatan laporan keuangan. Untuk mempermudah dan mempercepat proses pengembangan, SIABDes TAXion menggunakan arsitektur modular monolitik. Arsitektur ini diimplementasikan pada konteks aplikasi back-end dari SIABDes TAXion.

Arsitektur modular monolitik dipilih agar mudah diimplementasikan pada tim kecil. Selain itu, arsitektur ini dipilih agar tim pengembang di masa depan memiliki berbagai opsi dalam keputusan teknis. Hal ini dikarenakan modular monolitik mengelompokkan setiap domain ke dalam modul yang sesuai. Salah satu masalah yang sangat mungkin terjadi pada SIABDes adalah masalah skalabilitas. Salah satu solusi skalabilitas dari arsitektur monolitik adalah dengan migrasi ke microservice. Sehingga, modular monolitik dapat mempermudah migrasi ke arsitektur microservice yang sangat bergantung pada pembagian domain yang jelas.

Berdasarkan uraian sebelumnya, rekonstruksi SIABDes diajukan dengan mempertimbangkan faktor keamanan, perubahan fitur pencatatan laporan keuangan, penambahan fitur perpajakan. Selain peningkatan secara fungsionalitas dan keamanan, rekonstruksi SIABDes diharapkan dapat meningkatkan kualitas maintainability dengan konsep modular yang didukung oleh modular monolitik. Sehingga tim pengembang SIABDes saat ini, dapat membuka opsi keputusan teknis yang lebih luas kepada tim pengembang SIABDes di masa depan.

II. KAJIAN TEORI

A. Badan Usaha Milik Desa (BUMDes)

Badan Usaha Milik Desa adalah badan usaha yang bertujuan untuk mengelola perekonomian desa. Badan usaha ini dapat terdiri dari berbagai macam sektor, sehingga dapat memberdayakan sumber daya desa, sehingga kesejahteraan desa dapat berkembang *[4]*.

B. Standar Akuntansi Keuangan Entitias Mikro Kecil dan Menengah

SAK EMKM adalah standar keuangan yang digunakan untuk memisahkan kekayaan pribadi dan usaha. Standar ini mengatur transaksi umum secara historis. Standar keuangan ini dapat cocok digunakan oleh badan Usaha Mikro, Kecil dan Menengah (UMKM) [5].

C. Scrum

Scrum sebagai framework dalam mengembangkan aplikasi, bertujuan agar pengembangkan dilakukan secara inkremental dan cepat. Dalam Scrum, dinamika proyek dapat diatur berdasarkan prioritas pada setiap sprint. Hal ini membuat Scrum untuk memungkinkan mengembangkan

produk yang lebih berkualitas pada proyek yang kompleks [6].

D. Domain Driven Design (DDD)

Dalam memodelkan arsitektur microservice, salah satu pendekatan yang dapat dilakukan adalah dengan Domain Driven Design (DDD). DDD membantu dalam memetakan kebutuhan bisnis, sehingga kebutuhan tersebut dapat dikelompokkan kedalam sebuah domain [7].

E. Arsitektur Modular Monolitik

Modular monolitik adalah arsitektur yang memanfaatkan kesederhanaan monolitik pada umumnya, dengan tambahan modularitas pada struktur arsitektur [8]. Pendekatan DDD dapat diimplementasikan pada arsitektur modular monolitik, yang mengutamakan pengelompokan domain menjadi modul masing-masing [9]. Dengan demikian, secara sederhana arsitektur modular monolitik adalah arsitektur yang menggunakan konsep modularitas seperti microservice, namun berada dalam satu sumber kode [10]. Arsitektur modular monolitik juga memudahkan dalam melakukan migrasi ke arsitektur microservice. Hal ini karena kedua arsitektur menggunakan pendekatan yang sama, yaitu penggunaan DDD. Arsitektur microservice yang memecah komponen atau modul menjadi sebuah service tersendiri. Modularitas dari modular monolitik ini, yang mempermudah evolusi arsitektur dalam aplikasi menjadi microservice [9].

F. Model View Controller (MVC)

MVC adalah pola arsitektur yang membagi menjadi tiga komponen utama, yaitu komponen Model, View dan Controller. Komponen Model merepresentasikan data dan logika bisnis. Kedua, komponen View bertugas sebagai representasi dari antarmuka, agar pengguna dapat berinteraksi dengan aplikasi. Ketiga, komponen Controller sebagai komponen yang menerima input pengguna dan menentukan alur aplikasi [11].

G. Node.js

Node.js adalah lingkungan agar aplikasi berbasis JavaScript dapat berjalan secara asinkron, dengan eksekusi berbasis peristiwa. Node.js dapat menerima koneksi dan menangani secara konkuren, dikarenakan Node.js didesain untuk membangun jaringan yang dapat di skalakan [12].

H. Framework

Istilah framework adalah standar desain aplikasi yang berisi sekumpulan library dan tools. Framework memiliki tujuan agar desain aplikasi tetap konsisten, mudah dikembangkan dan dipelihara [13].

I. NestJs

NestJS adalah framework yang menggunakan TypeScript. NestJS merupakan framework yang opinionated, sehingga NestJS sudah memiliki standar tersendiri. Salah satu standar dalam NestJS adalah dengan diadakan pengelompokan domain bisnis kedalam sebuah modul. Dengan adanya standar ini, memudahkan kolaborasi para pengembang dalam mengembangkan dan memelihara sumber kode [14] [15].

J. PostgreSQL

PostgreSQL adalah sistem manajemen basis data objek-relasional. Postgres dikenal dengan reliabilitas, performa dan dukungannya terhadap transaksi Atomicity, Consistency, Isolation and Durability (ACID). Dengan menggunakan postgres sebagai basis data, dapat meningkatkan konsistensi data, pada saat melakukan mutasi data maupun membaca data [16].

K. Software Testing

Software Testing atau pengujian perangkat lunak adalah kegiatan untuk mengujikan sistem pada perangkat lunak, baik secara otomatis maupun manual. Software Testing bertujuan untuk membangun kepercayaan diri, bahwa kebutuhan fungsional maupun non-fungsional pada sistem terpenuhi [17].

L. Unit Testing

Unit Testing atau pengujian unit adalah kegiatan untuk mengujikan bagian terkecil dalam perangkat lunak. Pengujian ini dilakukan oleh pengembang karena berfokus dengan bagian terkecil, sehingga dapat dilakukan dengan mudah dan cepat [17]. Pengujian unit bertujuan untuk menguji logika, batasan dan penanganan error pada sebuah function. Selain pengecekan function, pengujian unit juga dapat digunakan sebagai media dokumentasi aturan bisnis pada sebuah function [18].

M. Load Testing

Load Testing adalah jenis pengujian performa pada kegiatan Software Testing, yang bertujuan untuk menilai bahwa sistem dapat menerima beban pengguna dalam jumlah tertentu. Jumlah pengguna yang diujikan dalam pengujian ini bergantung pada jumlah pengguna harian. Sehingga dapat didefinisikan, Load Testing adalah pengujian performa yang menyimulasikan beban pengguna harian, sehingga pengembang mendapat kepercayaan diri bahwa sistem dapat digunakan dengan beban harian [19].

N. Grafana k6

Grafana k6 atau k6 adalah alat untuk membantu Load Testing. Jenis pengujian yang dapat dilakukan oleh k6 adalah pengujian Load Testing dan pengujian perfroma browser. K6 memiliki kemampuan untuk menguji secara otomatis [20].

III. METODE

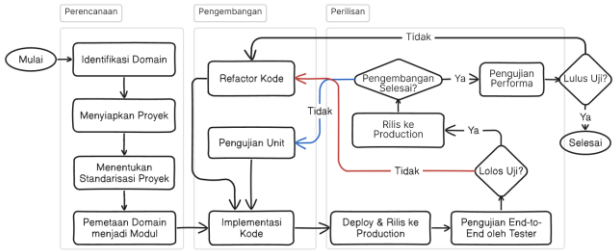
A. Metodologi

Dalam pengembangan aplikasi SIABDes TAXion, menggunakan tahapan yang ada pada Software Development Lifecycle (SDLC). Pada SDLC memiliki tahapan secara umum dengan berikut secara urut: Perencanaan, Analisis, Desain, Implementasi, Pengujian dan Perawatan. Untuk memudahkan manajemen dalam tim SIABDes TAXion, menggunakan proses perangkat lunak agile. Dalam proses perangkat lunak yang digunakan, tim SIABDes TAXion menyesuaikan proses dalam agile dengan keadaan dan kultur tim, sehingga terdapat perubahan pada penggunaan agile. Berikut proses yang sudah disesuaikan untuk tim SIABDes TAXion pada Tabel 1:

TABEL 1
PROSES PERANGKAT LUNAK SIABDES TAXION

Urutan	Agile Tradisional	Proses SIABDes TAXion
1	Daily standup dilakukan setiap hari dalam durasi yang singkat, untuk memastikan momentum dan status pekerjaan anggota tim.	Meeting dijadwalkan secara fleksibel, berdasarkan beban pekerjaan pada meeting sebelumnya.
2	Sprint review dilakukan untuk melihat progres produk, yang telah dikerjakan pada sprint tersebut.	Dilakukan demo progres pada pekerjaan yang sudah direncanakan pada meeting sebelumnya, sehingga dapat diketahui kekurangan dan penyesuaian pada hasil pekerjaan.
3	Sprint retrospective dilakukan dengan membahas kekurangan pada sprint sebelumnya, sehingga dapat diperbaiki pada sprint selanjutnya.	Kesulitan antar individu dalam tim dijelaskan pada meeting yang sedang berjalan, sehingga tim dapat memberikan umpan balik ke pada individu yang kesulitan dengan cepat.
4	Dilakukannya sprint planning untuk merencanakan daftar backlog, yang akan dikerjakan pada sprint selanjutnya secara detail.	Perencanaan pekerjaan apa yang akan dilakukan, beban pekerjaan ditentukan berdasarkan beban backlog dalam hari. Setelah ditentukan beban pekerjaan, ditetapkan deadline pekerjaan yang sekaligus sebagai jadwal meeting selanjutnya.

B. Alur Pengembangan



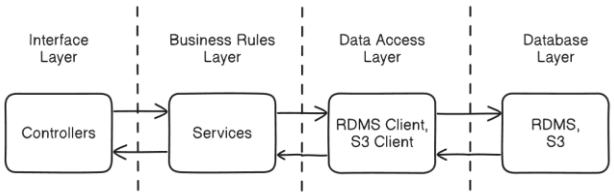
GAMBAR 2
ALUR PENGERJAAN BACK-END SIABDES TAXION

Alur pengembangan aplikasi back-end SIABDes TAXion mengikuti tahapan pada SDLC. Dalam pengembangan ini hanya berfokus pada tahap implementasi dan pengujian. Hal ini divisualisasikan melalui Gambar 2 yang merupakan alur pengembangan back-end SIABDes TAXion. Alur pengembangan ini memiliki tiga fase, yaitu perencanaan, pengembangan dan perilisan.

Secara garis besar, fase perencanaan bertujuan untuk memudahkan penyusunan kode. Pada tahap identifikasi domain, dilakukan pengelompokan kebutuhan fungsional menjadi sebuah domain. Setelah domain teridentifikasi, peneliti menyiapkan proyek dengan teknologi dan kerangka kerja yang dibutuhkan. Pada tahap ketiga, standarisasi sumber kode diperlukan agar menjaga konsistensi kode. Selain menjaga konsistensi kode, standarisasi ini juga memudahkan pemelihara kode yang lain untuk memahami sumber kode. Selanjutnya, domain tersebut dipetakan ke dalam sebuah modul. Modul ini berisi komponen-komponen arsitektur seperti Controller dan Service, yang bertujuan untuk menyelesaikan kebutuhan logika bisnis. Selanjutnya pada fase Pengembangan, pertama dilakukan implementasi kode yang sudah direncanakan pada fase sebelumnya. Pada tahapan ini, dilakukan implementasi logika bisnis sesuai dengan kebutuhan fungsional dan non-fungsional dari SKPL.

Pada alur pengerjaan back-end pada fase perilisan, tahap deploy ke lingkungan staging dilakukan secara otomatis dengan bantuan Continous Integration / Continous Deployment (CI/CD) oleh penyedia layanan awan. Selanjutnya, hasil deployment pada lingkungan staging diuji oleh tester. Setelah dilakukan pengujian oleh tester, pengembang melakukan merge dari branch yang sudah diuji, agar dapat dirilis pada lingkungan production. Pada tahapan ini jika pengembangan sudah memenuhi kebutuhan fungsional, dilakukan pengujian performa terhadap aplikasi back-end menggunakan Grafana k6. Namun, jika kebutuhan fungsional belum terpenuhi keseluruhan, iterasi diulang pada tahapan sebelumnya berdasarkan hasil pengujian. Jika hasil pengujian dilaporkan tidak ditemukan bug, maka pengembang melanjutkan pekerjaan pada tahap implementasi kode dengan fitur baru. Namun, jika pada pengujian dilaporkannya bug, dilakukan refactor atau perbaikan kode pada sumber kode yang terjadi bug.

C. Arsitektur Aplikasi

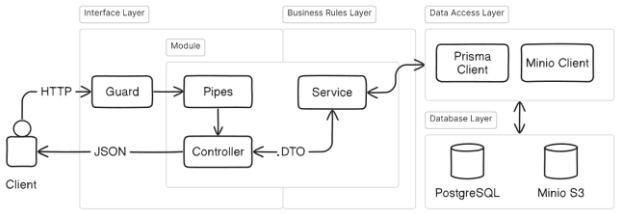


GAMBAR 3
ARSITEKTUR HIGH LEVEL SIABDES TAXION

Arsitektur aplikasi back-end SIABDes TAXion memiliki empat lapisan. Lapisan ini bertujuan untuk mengisolasi dan menerapkan prinsip satu tanggung jawab pada tiap lapisan. Relasi antar lapisan ini divisualisasikan melalui Gambar 3, yang menggambarkan setiap lapisan hanya dapat berkomunikasi dengan maksimal dua lapisan lainnya. Pada lapisan pertama yaitu interface layer, bertanggung jawab sebagai antarmuka agar aplikasi front-end dapat berkomunikasi dengan aplikasi back-end. Pada lapisan business rules, bertanggung jawab sebagai orkestrator logika bisnis dari aplikasi SIABDes TAXion. Lapisan ini bertujuan untuk mengolah data yang didapatkan dari lapisan Database melalui lapisan Data Access.

Lapisan Data Access berisi klien agar lapisan sebelumnya dapat berkomunikasi dengan lapisan Database. Pada lapisan Data Access ini memerlukan komponen klien yang kompatibel dengan lapisan Database. Pada SKPL SIABDes TAXion, kebutuhan fungsional pada aplikasi ini adalah basis data untuk penyimpanan file dan data relasional. Berdasarkan kebutuhan fungsional tersebut, penyimpanan dokumen dapat menggunakan Simple Storage Service (S3). Sedangkan data relasional dapat menggunakan Relational Database Management System (RDMS) berbasis

PostgreSQL. Hal ini memerlukan komponen klien pada lapisan Data Access kompatibel dengan basis data yang digunakan. Pada konteks aplikasi back-end SIABDes TAXion, menggunakan Object Relational Mapper (ORM) Prisma sebagai klien RDMS, dan Minio sebagai klien S3.



GAMBAR 4
DETAIL KOMPONEN PADA ARSITEKTUR BACK-END

Detail arsitektur pada Gambar 4, menunjukkan bagaimana Client atau aplikasi front-end berkomunikasi dengan aplikasi back-end. Lapisan interface berkomunikasi melalui endpoint yang disediakan oleh Controller. Endpoint ini diakses melalui request yang menggunakan Hypertext Transfer Protocol (HTTP) oleh aplikasi front-end. Dalam beberapa kasus yang memerlukan otorisasi, dapat digunakan komponen Guard yang bertanggung jawab untuk memastikan request yang diterima memiliki otoritas terhadap sumber daya yang diminta. Selanjutnya ada komponen Pipes yang bertanggung jawab untuk melakukan validasi data yang dikirimkan pada request. Data yang sudah divalidasi, diubah menjadi Data Transfer Object (DTO), dan disalurkan ke komponen Service yang digunakan sebagai argumen dalam mengolah data.

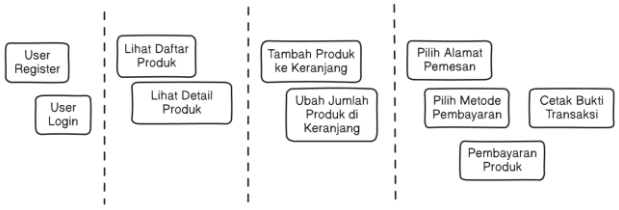
Berdasarkan uraian mengenai detail interaksi pada Arsitektur Modular Monolitik pada Gambar 2 dan Gambar 3, dapat memenuhi kebutuhan pada SKPL dan masalah keamanan pada proyek SIABDes TAXion. Hal ini didukung dengan adanya berbagai komponen pendukung. Komponen Guard yang bertugas sebagai mekanisme pengecekan otoritas dari pengguna. Komponen Controller, Pipes dan Service bertugas sebagai orkestrator yang bertujuan untuk memenuhi kebutuhan bisnis. Selanjutnya pada komponen pada lapisan Data Access dan Database bertugas sebagai layanan manajemen data.. Setiap komponen dalam arsitektur ini dapat meningkatkan modularitas pada setiap skenario bisnis, sehingga memudahkan dalam pemeliharaan, peningkatan skalabilitas aplikasi dalam konteks back-end dan memenuhi kebutuhan fungsional dari SKPL SIABDes TAXion.

D. Domain Driven Design (DDD)

DDD digunakan dalam memodelkan dan mengelompokkan berbagai kasus bisnis. DDD menjadi sebuah landasan, dalam membangun aplikasi dengan arsitektur modular monolitik. Arsitektur ini bergantung pada modularitas. Agar dapat mengidentifikasi domain dan mengelompokkan kasus-kasus tersebut, DDD menawarkan berbagai strategi untuk memperjelas domain. Pada konteks aplikasi back-end SIABDes TAXion, terdapat tiga langkah, yaitu Event Storming, Scenario Flows dan Bounded Context.

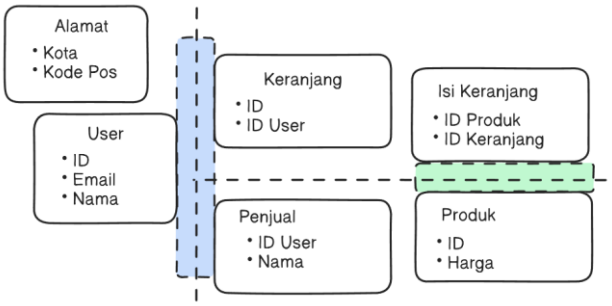
GAMBAR 5
KASUS BISNIS YANG BELUM DIKELOMPOKKAN

Event Storming bertujuan untuk mengidentifikasi kasus bisnis apa saja yang mungkin terjadi. Langkah ini dilakukan oleh tim pengembang, dengan cara menuliskan daftar kasus bisnis yang mungkin saja terjadi pada media tulis. Sebagai contoh kasus, diilustrasikan dengan kasus pembelian produk pada sebuah aplikasi e-commerce pada Gambar 5.



GAMBAR 6
KASUS-KASUS BISNIS MENJADI SKENARIO BISNIS

Pada langkah ini, dilakukan setelah diidentifikasi berbagai macam kasus bisnis yang mungkin terjadi. Langkah ini bertujuan untuk memudahkan pengelompokan tiap kasus bisnis, menjadi ke dalam sebuah domain. Langkah ini dapat dilakukan dengan melakukan urutan pada setiap kasus bisnis, sehingga menciptakan sebuah skenario bisnis. Berdasarkan skenario bisnis tersebut, dapat dengan lebih mudah untuk menemukan potensi kelompok kasus bisnis. Sebagai contoh langkah ini, pada Gambar 6 menggambarkan bagaimana kasus bisnis yang tidak terorganisir, menjadi sebuah skenario bisnis yang lebih mudah dipahami.



GAMBAR 7
CONTOH PENERAPAN BOUNDED CONTEXT

Setelah dilakukan penyusunan beberapa skenario bisnis, setiap kasus bisnis ini dapat ditentukan hubungan antar kasus bisnis lainnya. Dengan menentukan hubungan antar dapat lebih mudah untuk menentukan domain-domain yang berpotensi sesuai dengan skenario-skenario bisnis, sehingga memudahkan dalam pengembangan. Berikut contoh ilustrasi penerapan bounded context pada Gambar 7.

IV. HASIL DAN PEMBAHASAN

A. Identifikasi Domain

Dalam pengembangan aplikasi SIABDes TAXion, domain-domain bisnis diidentifikasi menggunakan pendekatan Domain-Driven Design (DDD). Proses ini dimulai dengan Event Storming, di mana kasus bisnis utama yang terjadi dalam aplikasi diidentifikasi dan dicatat. Contoh kasus bisnis termasuk pembuatan jurnal dan pencetakan laporan keuangan. Langkah ini membantu memetakan proses-proses penting yang terjadi dalam aplikasi.

Selanjutnya, proses tersebut diorganisasi menggunakan Scenario Flows, yang menyusun urutan logis dari setiap kasus bisnis. Setiap skenario menunjukkan aliran proses bisnis yang terjadi, seperti transaksi yang berhubungan dengan jurnal atau pencatatan pajak.

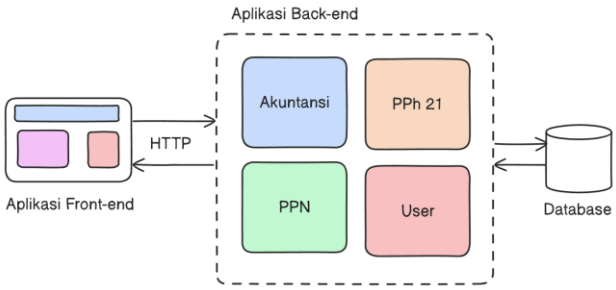
Tahap terakhir adalah Bounded Context, di mana domain-domain bisnis dipisahkan berdasarkan keterkaitannya. Setiap domain berfokus pada skenario bisnis yang spesifik, memastikan bahwa setiap komponen aplikasi hanya menangani tugas yang sesuai dengan batasan domainnya. Pada SIABDes TAXion, domain-domain yang terdefinisi meliputi Jurnal, PPN, PPh 21, dan Unit BUMDes. Pemisahan domain ini memastikan modularitas dan memudahkan pengembangan serta pemeliharaan aplikasi di masa mendatang.

B. Menyiapkan dan Menentukan Standar Proyek

Pengembangan back-end aplikasi SIABDes TAXion menggunakan framework NestJS, yang berbasis pada Node.js dan menggunakan PostgreSQL sebagai database relasional. Untuk memudahkan pengelolaan data, digunakan juga penyimpanan objek dengan MinIO. Infrastruktur aplikasi disiapkan menggunakan Docker, yang memungkinkan penggunaan container untuk menjalankan PostgreSQL dan MinIO secara terisolasi. Dengan Docker, proyek dapat diinisialisasi dengan perintah docker-compose up, yang akan menjalankan semua layanan yang dibutuhkan.

Setelah proyek disiapkan, langkah selanjutnya adalah menetapkan standarisasi proyek agar kode yang dikembangkan konsisten dan mudah dipahami oleh pengembang lain di masa mendatang. Penggunaan standar penamaan file dan pengorganisasian kode yang baik sangat penting untuk menjaga struktur yang rapi. Misalnya, file untuk controller, service, dan model dipisahkan sesuai tanggung jawab masing-masing. Setiap komponen seperti controller diberi akhiran .controller.ts, sedangkan untuk service menggunakan .service.ts. Standarisasi ini membantu menjaga keteraturan dan konsistensi dalam pengembangan kode, serta mempermudah pemeliharaan proyek ke depannya.

C. Pemetaan Domain menjadi Modul

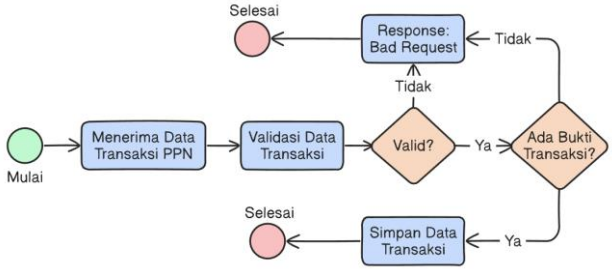


GAMBAR 8
ARSITEKTUR APLIKASI BACK-END SIABDES TAXION

Sistem SIABDes TAXion diorganisasikan ke dalam empat domain utama: User, Akuntansi, PPN, dan PPh 21, yang dipetakan dalam arsitektur modular monolitik. Masing-masing domain diimplementasikan sebagai modul terpisah, sebagaimana ditunjukkan pada Gambar 8. Aplikasi front-end berkomunikasi dengan back-end melalui protokol HTTP, di mana setiap domain memiliki logika bisnis dan skema data tersendiri, namun berbagi struktur data yang serupa.

D. Implementasi Kode

Dalam implementasi kode pada back-end SIABDes TAXion, mengikuti standarisasi yang sudah dibuat pada langkah sebelumnya. Komponen-komponen dalam implementasi, mengikuti penamaan file dan pola desain kode yang sesuai dengan standarisasi kode. Agar dapat memvisualisasikan hal tersebut, diagram alur ditunjukkan pada Gambar 10



GAMBAR 10
DIAGRAM ALUR MEMBUAT DATA PPN

Pada Gambar 10, menunjukkan alur penerimaan data dari front-end, lalu akan divalidasi data tersebut. Jika data tidak sesuai, back-end akan mengirimkan respons dengan kode status 400 bad request. Jika validasi dinyatakan lolos, data tersebut akan disimpan pada basis data. Implementasi kode tersebut dapat dilihat pada Gambar 11.

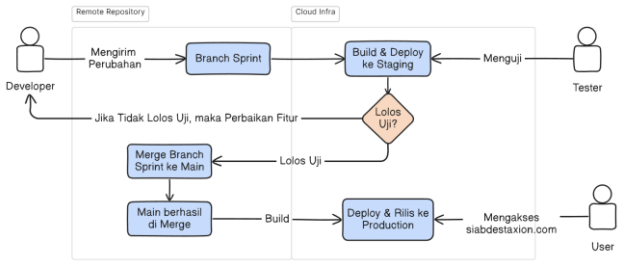
```
async addPpn(
  evidence: Express.Multer.File,
  ids: IdsDto,
  dto: AddPpnV2Dto,
): Promise<AddPpnV2Response> {
  const { key } = await this.fileService.upload(evidence, {
    ...ids,
    resource: FileResourceLocation.PPN,
  });

  const ppn = await this.ppnRepository.create({
    evidenceKey: key,
    dto,
    ids,
  });

  return {
    id: ppn.id,
    created_at: ppn.createdAt,
  };
}
```

GAMBAR 11
IMPLEMENTASI KODE MENAMBAHKAN DATA PPN

E. Deployment dan Perilisan



GAMBAR 12
STRATEGI DEPLOYMENT DAN RILIS BACK-END

Gambar 12 menggambarkan strategi deployment ke staging dan berakhir ke lingkungan production. Berdasarkan gambar tersebut, deployment ke lingkungan staging dilakukan setelah pengembang mengirim perubahan ke repository. Setelah perubahan terkirim, infrastruktur awan melakukan pull untuk melakukan build secara otomatis dan dilakukan deployment secara otomatis. Pada lingkungan staging, Tester melakukan pengujian end-to-end untuk menguji fungsionalitas secara menyeluruh. Jika pengujian tidak lolos, maka pengembang melakukan perbaikan. Tahap ini dilakukan sampai sistem lolos pengujian.

Setelah tahap pengujian oleh Tester dinyatakan lolos, tim pengembang melakukan merge branch untuk digabungkan ke branch utama. Setelah merge berhasil dilakukan, infrastruktur awan melakukan pull kembali pada branch utama secara otomatis. Strategi ini bertujuan untuk mengurangi kesalahan manusia yang melakukan deploy dan perilisan secara manual. Selain itu, dengan melakukan isolasi pada staging dan menguji pada lingkungan ini, dapat meminimalkan risiko bug dirilis ke lingkungan production.

F. Pengujian Unit

```
File                                | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
------------------------------------|---------|----------|---------|---------|-------------------
All files                           |      81 |    52.37 |   82.89 |   85.17 |
 common/helpers                     |   66.66 |    55.76 |   42.85 |    67.3 |
  file-path.helper.ts               |    61.9 |    56.09 |   33.33 |    62.5 | 13-22,37,40,46,56,69-83
  ptkp-mapper.helper.ts             |   83.33 |    54.54 |     100 |   83.33 | 26-27
 common/services                    |   61.53 |    29.03 |   57.14 |      65 |
  files.service.ts                  |   61.53 |    29.03 |   57.14 |      65 | 18,30-46
 common/utils                       |   82.75 |        0 |   88.88 |      84 |
  file-service-test.util.ts         |     100 |      100 |     100 |     100 |
  minio-test.util.ts                |     100 |      100 |     100 |     100 |
  prisma-test.util.ts               |     100 |      100 |     100 |     100 |
  retry.util.ts                     |   54.54 |        0 |   66.66 |      60 | 25-35
 modules/auth/helpers               |     100 |      100 |     100 |     100 |
  payload.helper.ts                 |     100 |      100 |     100 |     100 |
 modules/auth/services              |   87.83 |    43.93 |     100 |      95 |
  auth.service.ts                   |   89.36 |    48.71 |     100 |   94.87 | 27,85
```

GAMBAR 13
LAPORAN COVERAGE PENGUJIAN UNIT

Berdasarkan Gambar 13, pengujian unit aplikasi back-end SIABDes TAXion menunjukkan statement coverage berada diatas 80%. Pada laporan tersebut juga menunjukkan branch coverage pada diangka 52%. Berdasarkan persentase statement dan branch coverage, menunjukkan mayoritas function telah diujikan, namun function yang memiliki percabangan belum terujikan secara menyeluruh.

G. Pengujian Performa

Pengujian Load Testing pada aplikasi back-end SIABDes TAXion, menggunakan beberapa skenario bisnis yang sehari-hari dilakukan oleh pengguna. Skenario-skenario ini menjadi dasar penggunaan aplikasi SIABDes TAXion oleh pengguna. Sebelum pengujian dilakukan, ditambahkan data pengguna dummy yang digunakan sebagai representasi pengguna dalam pengujian. Pada pengujian ini, dilakukan iterasi sebanyak mungkin dengan 378 pengguna virtual dalam rentang waktu 13 menit. Berikut daftar skenario pada Tabel 2 yang digunakan dalam pengujian ini:

TABEL 2
DAFTAR SKENARIO PENGUJIAN PERFORMA

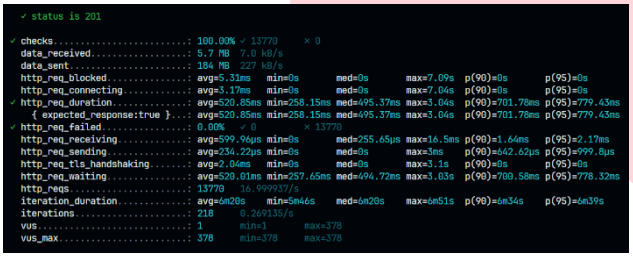
Urutan	Skenario Uji	Pemanggilan Tiap Iterasi
1	Login sebagai Unit yang sudah disediakan	1
2	Menambahkan Jurnal Umum untuk laporan laba rugi	8
3	Menambahkan Jurnal Umum untuk laporan posisi keuangan	8
4	Menambahkan sebuah data karyawan	1
5	Menambahkan pajak PPh 21 pada karyawan	12
6	Menambahkan pajak PPN	4

Selanjutnya setelah ditentukan skenario yang digunakan, diperlukan kriteria untuk mengukur apakah pengujian performa ini dapat digunakan oleh 270 BUMDes. Metrik kriteria yang menjadi indikator aplikasi back-end memiliki performa yang baik adalah: waktu response, kode status yang sesuai berdasarkan skenario yang dilakukan dan jumlah request yang gagal diproses oleh aplikasi. Berikut daftar kriteria beserta alasan kriteria tersebut diperlukan pada Tabel 3:

TABEL 3
DAFTAR SKENARIO PENGUJIAN PERFORMA

Kriteria	Alasan
90% request memiliki rata-rata waktu response di bawah dua detik	Berdasarkan kebutuhan non-fungsional NFR-01, menunjukkan sistem maksimal memberikan response dibawah 4 detik. Dalam konteks back-end, sebagian besar pemrosesan oleh back-end. Berdasarkan tersebut, back-end memiliki waktu pemrosesan maksimal setengah

	dari kebutuhan tersebut. Sehingga diperlukan mayoritas request memiliki waktu response dibawah dua detik.
Persentase request gagal di bawah 5%	Kriteria ini memastikan kegagalan mayoritas request tidak terjadi. Namun, dalam sebuah sistem masih terdapat celah kegagalan request, sehingga diperlukan persentase maksimum yang menggambarkan mayoritas request berhasil.
95% request memiliki kode status yang sesuai dengan skenario yang dilakukan. Dalam kasus operasi penambahan data, kode status dari response yang sesuai bernilai 201.	Pada kriteria ini hanya bertujuan sebagai pengecekan tambahan. Kriteria ini diperlukan untuk memastikan bahwa skenario dijalankan dengan baik oleh aplikasi back-end, sehingga memberikan response dengan kode 201.



GAMBAR 14
LAPORAN HASIL PENGUJIAN PERFORMA

Gambar 14 menunjukkan rangkuman laporan hasil pengujian performa pada back-end SIABDes TAXion. Dalam laporan ini, metrik yang diperhatikan adalah "http_req_duration", "http_req_failed" dan "checks". Ketiga metrik ini dapat menunjukkan performa aplikasi back-end SIABDes TAXion. Pada metrik pertama "http_req_duration", metrik ini menunjukkan durasi keseluruhan request yang dikirimkan dari klien hingga menerima response. Selanjutnya metrik "http_req_failed" menandakan terdapat request yang gagal, hal ini ditandai pada nilai status kode HTTP dengan kategori client error (4xx) atau server error (5xx). Metrik ketiga "checks" merupakan metrik tambahan untuk mengetahui apakah status kode response sudah sesuai dengan kasus uji, dalam pengujian ini dengan operasi penambahan data memiliki kode status 201.

Berdasarkan hasil laporan Gambar 14, menunjukkan metrik "http_req_duration" bernilai 701.78 milidetik pada persentil 90. Pada kriteria pertama terpenuhi karena 90% request, memiliki durasi di bawah 2 detik. Kriteria dua juga terpenuhi, karena tidak terdapat request yang mengalami error. Hal ini ditunjukkan melalui metrik "http_req_failed" yang bernilai 0%. Pada kriteria terakhir juga terpenuhi yang ditunjukkan melalui metrik "checks" dengan nilai 100%. Metrik "checks" menandakan bahwa setiap response memiliki status kode bernilai 201. Berdasarkan uraian ini, dapat disimpulkan bahwa pengujian performa berjenis Load Testing ini dianggap lolos.

V. KESIMPULAN

Berdasarkan uraian sebelumnya, implementasi aplikasi back-end menggunakan arsitektur modular monolitik dapat mengelola data laporan keuangan, PPN dan PPh 21. Fungsionalitas ini didukung dengan dilakukannya pengujian unit untuk memastikan kebenaran fungsionalitas kasus bisnis. Namun, pengujian unit tidak menjamin celah lain yang belum diujikan sudah sesuai dengan kebutuhan bisnis. Selain telah terpenuhi kebutuhan fungsional, juga tercapai tujuan agar dapat menahan beban pengguna 270 BUMDes, melalui pengujian performa dengan jenis Load Testing. Namun, tidak menutup kemungkinan bahwa hasil pengujian Load Testing, dapat mengetahui jika beban pengguna di atas 270 BUMDes.

REFERENSI

[1] SIABDes TAXion, "Angket Kepuasan Mitra (Peserta) Terhadap Kegiatan Penganbdian Kepada Masyarakat Universitas Telkom (Jawaban)," 9 November 2023. [Online]. Available: https://docs.google.com/spreadsheets/d/1vfVLbkyEUyYqotcOJLr8oId-QQML0rctP1E8pWS4sRY/edit#gid=87101240. [Diakses 2 December 2023].

[2] S. N. Trihapsari, "Rapat Koordinasi dengan Pengurus BUMDes," Bandung, 2023.

[3] National Institute of Standards and Technology, "CVE-2022-40482 Detail," [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-40482. [Diakses 25 November 2023].

[4] K. Nisaa dan N. Hidayati, "PERANAN BADAN USAHA MILIK DESA (BUMDes) DALAM PEMBERDAYAAN MASYARAKAT DI DESA LAMBANG SARI," *JURNAL SOSIAL DAN SAINS*, vol. 2, pp. 779-786, 2022.

[5] Ikatan Akuntansi Indonesia, "Tentang SAK EMKM," Ikatan Akuntansi Indonesia, [Online]. Available: https://web.iaiglobal.or.id/SAK-IAI/Tentang%20SAK%20EMKM#gsc.tab=0. [Diakses 5 August 2024].

[6] O. A. Dada dan I. T. Sanusi, "The adoption of Software Engineering practices in a Scrum environment," *African Journal of Science, Technology, Innovation and Development*, vol. 14, no. 6, 2022.

[7] H. Vural dan M. Koyuncu, "Does Domain-Driven Design Lead to Finding the Optimal Modularity of a Microservice?," *IEEE Access*, vol. 9, pp. 32721 - 32733, 22 February 2021.

[8] A. Shakir, D. Staegemann, M. Volk, N. Jamous dan K. Turowski, "Towards a Concept for Building a Big Data Architecture," *24th International Conference on Business Information Systems (BIS 2021)*, pp. 83-94, 2021.

[9] N. Goncalves, D. Faustino, A. R. Silva dan M. Portela, "Monolith Modularization towards Microservices: Refactoring and Performance Trade-offs," *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pp. 54-61, 2021.

[10] M. Tsechelidis, T. Maikantis, N. Nikolaidis dan A. Ampatzoglou, “Modular Monoliths the way to Standardization,” *Eclipse Security, AI, Architecture and Modelling Conference on Cloud to Edge Continuum,* pp. 49 - 52, 17 Oktober 2023.

[11] S. Khan dan A. T. Khanam, “Study on MVC Framework for Web Development in PHP,” *International Journal of Scientific Research in Computer Science, Engineering and,* vol. 9, no. 4, pp. 414-419, 2023.

[12] Node.js, “Node.js — About Node.js®,” Node.js, [Online]. Available: https://nodejs.org/en/about. [Diakses 5 August 2024].

[13] M. Laaziri, K. Benmoussa, S. Khoulji, K. M. Larbi dan A. E. Yamami, “A Comparative Study of Laravel and Symfony PHP Frameworks,” *International Journal of Electrical and Computer Engineering,* vol. 9, no. 1, pp. 704-712, 2019.

[14] Trilon, “NestJS Introduction,” Trilon, [Online]. Available: https://docs.nestjs.com/. [Diakses 2 December 2023].

[15] A. D. Pham, “Developing Back-end of a Web Application With NestJS Framework Case: Integrify Oy’s student management system,” 2021.

[16] L. Frank dan R. U. Pedersen, “Managing Consistency Anomalies in Distributed Integrated Databases With Relaxed ACID Properties,” *ICUIMC '14: Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication,* no. 30, pp. 1-7, 2014.

[17] A. Uddin dan A. Anand, “Importance of Software Testing in the Process of Software Development,” *IJSRD - International Journal for Scientific Research & Development,* vol. VI, no. 12, pp. 141-145, 2019.

[18] Amazon Web Service, “What is Unit Testing? - Unit Testing Explained - AWS,” Amazon Web Service, [Online]. Available: https://aws.amazon.com/what-is/unit-testing. [Diakses 5 August 2024].

[19] Grafana, “Average-load testing | Grafana k6 documentation,” Grafana, [Online]. Available: https://grafana.com/docs/k6/latest/testing-guides/test-types/load-testing/. [Diakses 5 August 2024].

[20] Grafana, “Grafana k6 | Grafana k6 documentation,” Grafana, [Online]. Available: https://grafana.com/docs/k6/latest/. [Diakses 5 August 2024].