Journal of Digital Business and Technology Innovation (DBESTI)

https://journal.nurulfikri.ac.id/index.php/DBESTI
P-ISSN: 3047-1028 E-ISSN: 3032-775X

REENGINEERING REST API MONOLIT KE *MICROSERVICE* MENGGUNAKAN *FRAMEWORK* LARAVEL

Asep Sahrudin[1], Ahmad Rio Adriansyah[2]

[1,2]Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Jakarta Selatan, DKI Jakarta, Indonesia 12640
asep20159ti@student.nurulfikri.ac.id, arasy@nurulfikri.ac.id

Abstract

The increasing population growth in Jakarta has also led to a surge in demand for housing, including boarding houses. This surge in demand has become a challenge in managing applications and websites that provide information and services related to boarding houses, particularly in terms of managing user activity within the system. Therefore, researchers are trying to restructure the existing boarding house application using a microservice model. This research discusses the restructuring of the backend application from a monolithic to a microservice model, with a case study on Kos Jenggot. The development method employed in this research is Extreme Programming (XP), which incorporates testing methods, including black-box and performance testing. The results of this research show that using the microservice model is superior to the monolithic model; with black box testing results of 100%, the feature can run well through performance for the most significant scenario, namely an access load of 200 users in 100 seconds which has a total error of 100% for the monolithic model and 40 % for microservice models. These results indicate that the microservice model is more effective than the monolithic model for restructuring boarding applications.

Keywords: *Backend, Microservice, Monolithic, Laravel, Locust*

Abstrak

Pertumbuhan penduduk di Jakarta yang semakin meningkat turut membuat permintaan akan kebutuhan tempat tinggal, termasuk kos-kosan menjadi melonjak. Lonjakan permintaan tersebut menjadi tantangan tersendiri dalam pengelolaan aplikasi dan situs web penyedia informasi dan layanan terkait kos-kosan dalam mengelola aktivitas *user* di dalam sistem. Oleh karenanya, peneliti mencoba untuk melakukan *reengineering* aplikasi kos yang telah ada dengan menggunakan model *microservice*. Penelitian ini membahas *reengineering* aplikasi kos *backend* dari model monolitik ke *microservice* dengan studi kasus pada Kos Jenggot. Metode pengembangan yang diterapkan dalam penelitian ini adalah *Extreme Programming* (XP) dengan metode pengujian yakni menggunakan *blackbox testing* dan *performance testing*. Hasil penelitian ini menunjukkan lebih unggul menggunakan model *microservice* dibandingkan model monolitik, dengan hasil *blackbox testing* sebesar 100% fitur dapat berjalan baik melalui performa untuk scenario paling besar yakni beban akses 200 user dalam waktu 100 detik yang memiliki total eror 100% untuk model monolitik dan 40% untuk model *microservice*. Dari hasil tersebut diperoleh, bahwa efektifitas penggunaan model *microservice* pada *reengineering* aplikasi kos ini lebih unggul dibandingkan model monolitik.

Kata kunci: *Backend*, *Microservice*, Monolitik, *Laravel, Locust*

1. PENDAHULUAN

Dengan adanya pertumbuhan penduduk yang signifikan dalam beberapa tahun terakhir di Jakarta, data menunjukkan peningkatan yang konsisten dari tahun ke tahun. Dengan angka pertumbuhan sebesar 0,04% pada tahun 2020, meningkat menjadi 0,45% pada tahun 2021, dan mencapai 0,66% pada tahun 2022 [1]. Pertumbuhan ini mengindikasikan adanya kebutuhan yang terus meningkat terhadap fasilitas tempat tinggal, termasuk kos-kosan sebagai salah satu pilihan utama bagi banyak penduduk perkotaan. Namun, dampak dari lonjakan permintaan tersebut dapat menimbulkan tantangan dalam pengelolaan aplikasi maupun situs web, yang menyediakan informasi dan layanan terkait kos-kosan. Oleh karena itu, penelitian ini bertujuan untuk melakukan *reengineering* aplikasi kos yang sudah ada dengan menggunakan arsitektur *microservice*. Pendekatan ini diharapkan dapat meningkatkan performa situs web serta mengurangi risiko

Diterima 27 Maret 2024, Direvisi 28 Mei 2025, Diterima untuk publikasi 30 Mei 2025

terhadap lonjakan permintaan yang mungkin terjadi di masa mendatang.

Berdasarkan hal tersebut, pengelola atau penyedia kos membutuhkan sebuah rancangan *website* yang mudah dipahami dan mudah diakses oleh semua orang. Pada dasarnya, rancangan desain web kos yang sudah ada sebelumnya masih menggunakan arsitektur monolitik. Arsitektur monolitik adalah pendekatan pengembangan perangkat lunak yang menggabungkan semua komponen sistem dalam satu entitas tunggal [2]. Meningkatnya penggunaan web Kos Jenggot turut berdampak pada proses pemeliharaan, performa, dan kompleksitas web yang dibangun menggunakan arsitektur monolit ini. Ketika melakukan peningkatan web atau memperbaiki sistem 2 yang *error* ketika menggunakan arsitektur monolitik, maka biasanya semua layanan atau *service* pada aplikasi akan ikut mati[3].

Dengan adanya kendala yang ditemui saat meningkatkan skalabilitas web kos tersebut, maka pengembangan perangkat lunak dengan menggunakan arsitektur *microservice* adalah salah satu cara untuk meningkatkan efisiensi dalam proses pengembangan dan perawatan aplikasi web kos ini. Arsitektur *microservice* adalah arsitektur pengembangan perangkat lunak yang memiliki tanggung jawab berdasarkan fungsionalitasnya atau memiliki fungsionalitas secara terpisah dan saling terhubung satu sama lain [3]. Dengan demikian, pengembang dapat melakukan pembaharuan yang berjalan sendiri dan saling berkaitan dengan mekanisme ringan seperti HTTP API. Untuk itulah penggunaan *microservice* adalah pilihan perangkat lunak yang tepat untuk membuat sebuah aplikasi web yang membutuhkan pembaharuan secara berkala seperti aplikasi Web Kos Jenggot.

Penelitian ini akan berfokus pada proses perancangan ulang atau *reengineering* pada arsitektur aplikasi yang sudah ada sebelumnya yakni pada proyek MSIB yang menggunakan arsitektur monolitik, untuk kemudian diubah menjadi *microservice* pada pengembangan aplikasi kos berbasis web di sisi *backend*-nya. Harapannya dengan adanya aplikasi kos yang dibangun menggunakan *microservice* nantinya dapat memberikan dampak positif yang sangat signifikan untuk beberapa pihak, seperti mahasiswa yang hendak mencari penginapan jangka panjang untuk kebutuhan kuliah lebih cepat dan terhindar dari *error* serta membantu pemilik Kos Jenggot dalam proses manajerial untuk meningkatkan efisiensi kos yang tersedia. Oleh karena itu peneliti mengambil penelitian yang berjudul "*Reengineering* REST API Monolit ke *Microservice* Menggunakan *Framework* Laravel".

Restrukturisasi

Sebuah konsep untuk "merestrukturisasi" atau "menyusun ulang" sesuatu agar mencapai tingkat kebaikan dan efisiensi dalam struktur atau tatanannya. "Penataan ulang" ini bisa terjadi dalam berbagai konteks, seperti bisnis, manajemen, teknologi, atau bahkan aspek-aspek sosial, dan sering kali diperlukan untuk menghadapi perubahan atau meningkatkan kinerja suatu sistem atau entitas [4].

Arsitektur

Arsitektur merupakan suatu manifestasi kompleks dari pengetahuan dan kreativitas yang bertujuan untuk menghasilkan struktur-struktur yang tidak hanya memenuhi fungsi praktis, tetapi juga menjadi lambang dan cerminan dari peradaban dan identitas manusia [4].

Indekos

Kos adalah suatu bentuk akomodasi sementara yang umumnya ditempati oleh individu atau kelompok orang yang membutuhkan tempat tinggal untuk periode tertentu, seperti mahasiswa, pekerja, atau individu lainnya yang butuh tempat tinggal sementara. Konsep kos sangat penting dalam mengakomodasi kebutuhan perumahan sementara di berbagai daerah, terutama di kota-kota besar [4].

Hypertext Preprocessor (PHP)

PHP sering digunakan untuk mengakses basis data, mengelola formulir, mengelola sesi pengguna, dan melakukan berbagai tugas lain yang mendukung fungsionalitas aplikasi web yang dinamis. Bahasa pemrograman ini telah menjadi salah satu alat yang sangat populer dalam pengembangan web [5].

Database

Database adalah suatu sistem perangkat lunak yang dirancang untuk menyimpan, mengelola, dan mengakses data dengan efisien. *Database Management System* (DBMS) adalah komponen utama dalam pengelolaan *database*, yang memungkinkan pengguna untuk melakukan berbagai operasi (CRUD). *Database* dan DBMS ini saling bekerja sama agar organisasi atau individu dapat menyimpan dan mengelola data mereka dengan cara yang efektif dan aman [6].

Laravel

Laravel adalah salah satu *framework open-source* yang didesain khusus untuk bahasa pemrograman PHP. *Framework* ini bertujuan untuk menyederhanakan proses pengembangan aplikasi web. Laravel menjadi pilihan yang sangat disukai oleh banyak pengembang karena mempercepat proses pengembangan dan membantu mereka menciptakan aplikasi web yang lebih efisien dan dapat diandalkan [7].

MySQL

Penggunaan SQL memungkinkan pengguna dapat merancang kueri yang kompleks untuk mengambil informasi dari *database* dengan efisien. DBMS yang menggunakan SQL umumnya digunakan dalam jenis aplikasi berbasis web yang memerlukan manajemen data yang efisien dan dinamis, seperti sistem manajemen konten atau aplikasi *e-commerce* [8].

Monolitik

Monolitik adalah pendekatan dalam pengembangan perangkat lunak di mana seluruh aplikasi atau sistem dibangun sebagai satu entitas tunggal yang utuh.

Microservice

Sebuah aplikasi dibangun sebagai sekumpulan layanan terpisah yang masing-masing bertanggung jawab atas fungsionalitas tertentu. Arsitektur *microservice* memiliki karakteristik layer yang terpisah dan saling terhubung [9].

Blackbox Testing

Blackbox testing merupakan salah satu teknik pengujian perangkat lunak yang dilakukan untuk mengamati *output* dari perangkat lunak berdasarkan fungsionalitasnya [10]. Metode *blackbox* ini mencakup beberapa pendekatan antara lain *Equivalence Partitioning, Boundary Value Analysis, Comparison Testing, Sample Testing, Robustness Tesing,* dan lain-lain [10].

2. METODE PENELITIAN

Pada bagian ini akan dibahas mengenai metode pengumpulan data, instrumen penelitian, metode pengujian, serta tahapan penelitian.

2.1. Metode Pengumpulan Data, Instrumen Penelitian, dan Metode Pengujian

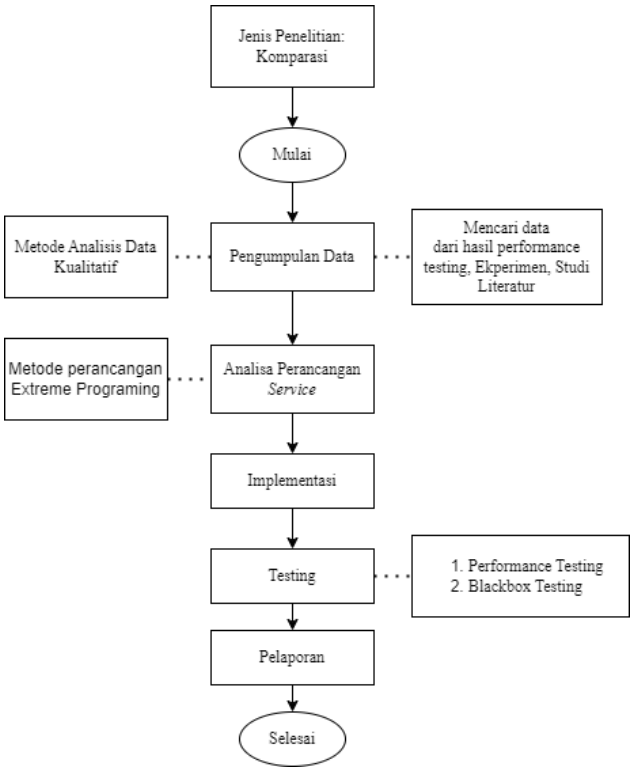
Penelitian ini mengadopsi pendekatan pengumpulan data yang bersifat komprehensif melalui penggunaan. Kombinasi metode ini dirancang untuk mendapatkan pemahaman yang mendalam mengenai fenomena yang menjadi fokus penelitian. Berikut untuk penjelasan secara detail maksud dari kedua metode pengumpulan data tersebut:

a. Studi Literatur
   Studi Literatur dalam penelitian ini merupakan bagian penting dari mencapai dan mendukung tujuan tugas akhir. Membaca, mencari, dan mengumpulkan data dari berbagai sumber termasuk dalam proses studi literatur. Sumber-sumber ini termasuk jurnal ilmiah, kertas, skripsi atau penelitian terkait, *website*, buku, dan alat elektronik lainnya. Selain itu, pengamatan atau observasi juga dilakukan untuk mendapatkan data atau informasi yang relevan tentang objek yang diamati.

b. Metode Eksperimen
   Dalam penelitian ini yang dimaksudkan dengan *experiment* adalah peneliti melakukan *reengineering* servis pada aplikasi yang sebelumnya sudah ada, kemudian diadopsi menggunakan servis yang berbeda. Lalu dilakukan dengan pengujian *performance testing* untuk melihat perbedaan di antara kedua servis tersebut.

2.2. Tahapan Penelitian

Dalam proses penelitian tentang "*Reengineering* REST API Monolit ke *Microservice* Menggunakan *Framework Laravel*". Penulis melaksanakan serangkaian alur dan tahapan, yang meliputi pengumpulan data, analisis perancangan *service*, implementasi, *testing*, dan pelaporan. Gambar 1 berikut ini adalah *helicopter view* yang dilakukan tentang tahapan penelitian yang pada penelitian ini.



Gambar 1. Tahapan Penelitian

2.3. Metode Pengembangan

Dalam penelitian ini, peneliti menggunakan metode pengembangan yang terbukti efektif, yaitu *Extreme Programming* (XP), untuk mencapai tujuan pengembangan perangkat lunak yang inovatif dan berkualitas. Dengan memanfaatkan pendekatan ini, peneliti berusaha untuk memahami bagaimana XP dapat menghasilkan perangkat lunak yang responsif, tangguh, dan memenuhi standar kualitas yang tinggi. *Extreme Programming* (XP) adalah salah satu metodologi dalam pengembangan *agile software development methodologies* yang memiliki fokus terhadap *(coding)* sebagain aktivitas utama dalam semua tahapan pada siklus pengembangan perangkat lunak [11].

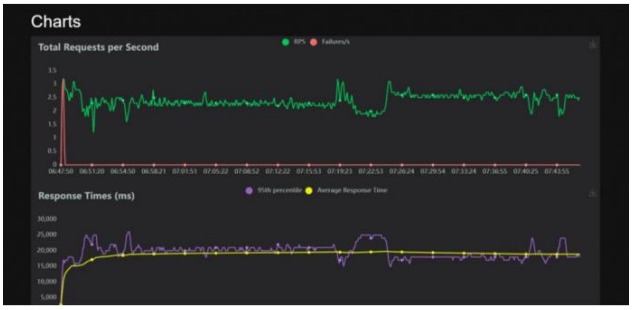
3. HASIL DAN PEMBAHASAN

Pada bagian ini akan dibahas mengenai analisis dan perancangan, analisis kebutuhan, dan perancangan sistem pada penulisan tugas akhir.

3.1. Analisis dan Perancangan

Analisis kebutuhan yang akan dibahas pada penulisan tugas akhir ini berdasarkan analisa pengujian sistem yang berjalan sebelumnya menggunakan arsitektur monolitik dengan melakukan pengetesan *performance testing* atau *load testing* untuk mendapatkan hasil perbedaan pengujian performa sistem berjalan.

Berikut adalah satu proses analisa yang dilakukan pada arsitektur sebelumnya yang menggunakan monolitik dengan melakukan proses pengujian atau *testing* dengan *performance testing* menggunakan *locust*.



Gambar 2. Proses Analisis Pertama untuk mengetahui hasil dari Performa menggunakan Arsitektur Monolitik

Gambar 2 tersebut merupakan salah satu hasil proses analisis yang dilakukan untuk mengetahui hasil performa yang dimiliki oleh arsitektur yang dibangun sebelumnya (monolitik) sebagai bahan pertama yang akan dilakukan proses komparasi hasilnya nanti (setelah perancangan dan pengembangan arsitektur *microservice* selesai) untuk melihat keunggulan performa kedua arsitektur yang akan dilakukan pada bagian hasil penelitian.

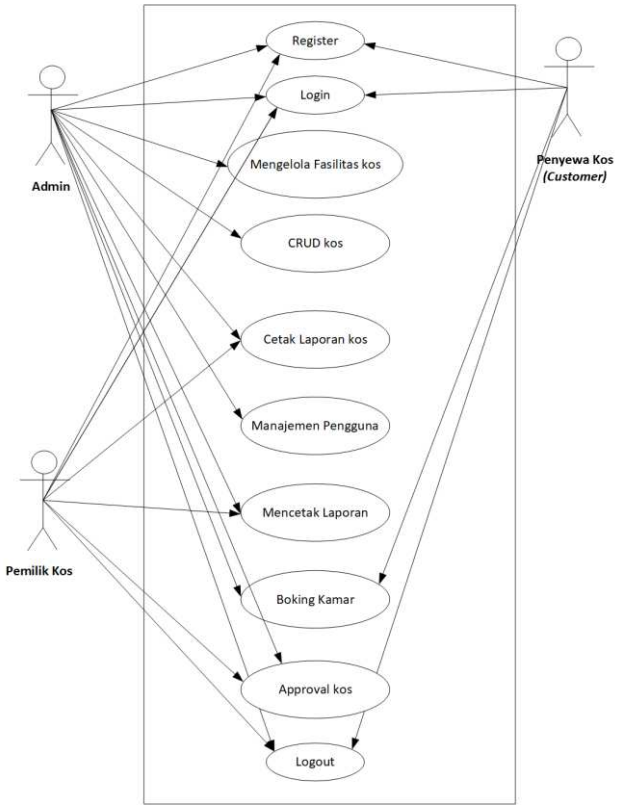
3.2. Analisis Kebutuhan Sistem

Pada bagian proses analisa kebutuhan sistem dilakukan dengan cara menganalisis sistem yang sudah ada yang dibangun menggunakan arsitektur monolitik melalui proses pengujian performa menggunakan *locust tools*.

3.3. Perancangan Sistem

3.3.1. *Use Case Diagram*

Untuk *use case diagram* yang digunakan dalam melakukan proses perancangan ini terdapat 3 *role user* yaitu admin, *customer* (penyewa) dan pemilik.



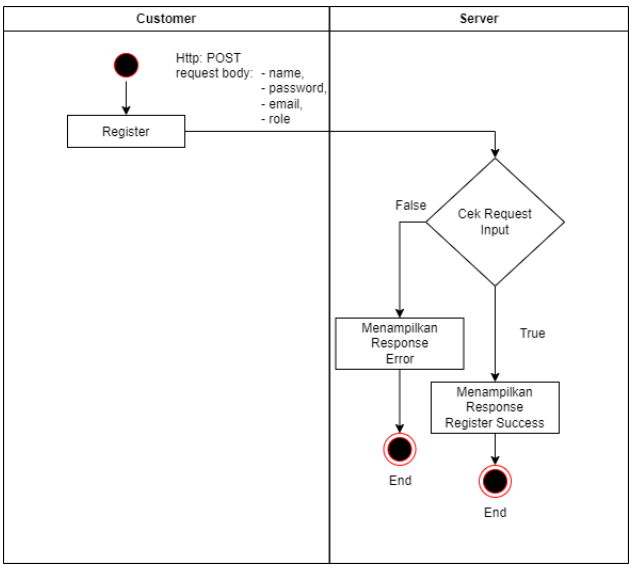
Gambar 3. *Use Case Diagram*

Dari gambar 3 tersebut dapat dijelaskan secara detail sebagai berikut:

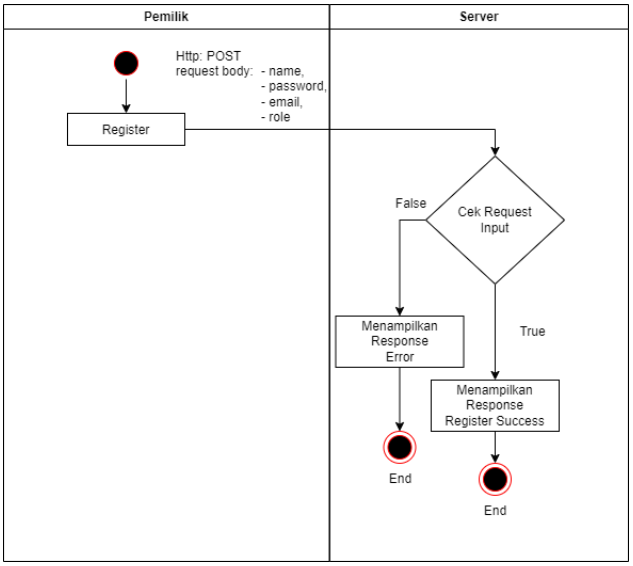
a. Admin (dapat mengelola seluruh akses pada sistem)
b. Penyewa dapat melakukan proses *booking* dan melihat informasi kos yang tersedia.
c. Pemilik dapat melakukan unggah kos yang dimiliki untuk supaya dapat diakses oleh *public*.

3.3.2. *Activity Diagram*

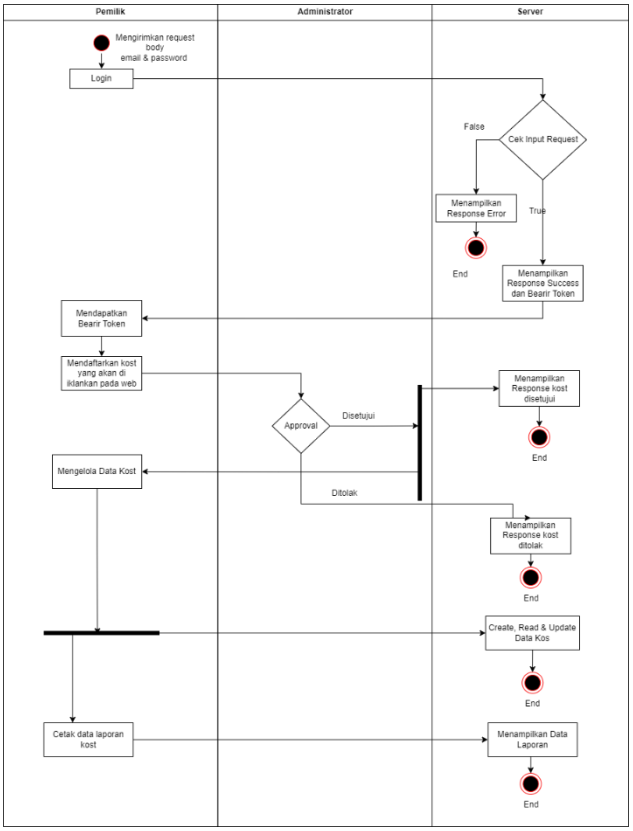
Activity Diagram dibuat untuk menggambarkan alur atau bisnis proses yang terjadi pada aplikasi kos ini.



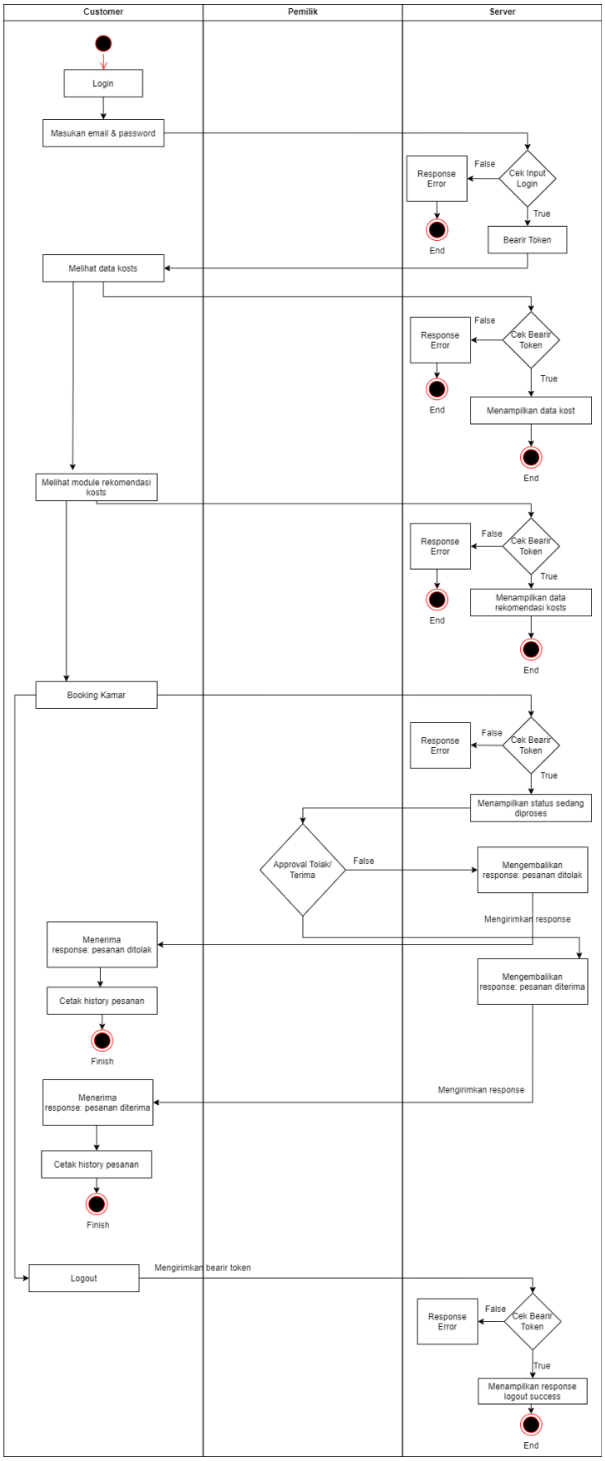
Gambar 4. *Activity Diagram Customer Login*



Gambar 5. *Activity Diagram Login Pemilik*



Gambar 6. *Activity Diagram* Akses Data Pemilik Kos

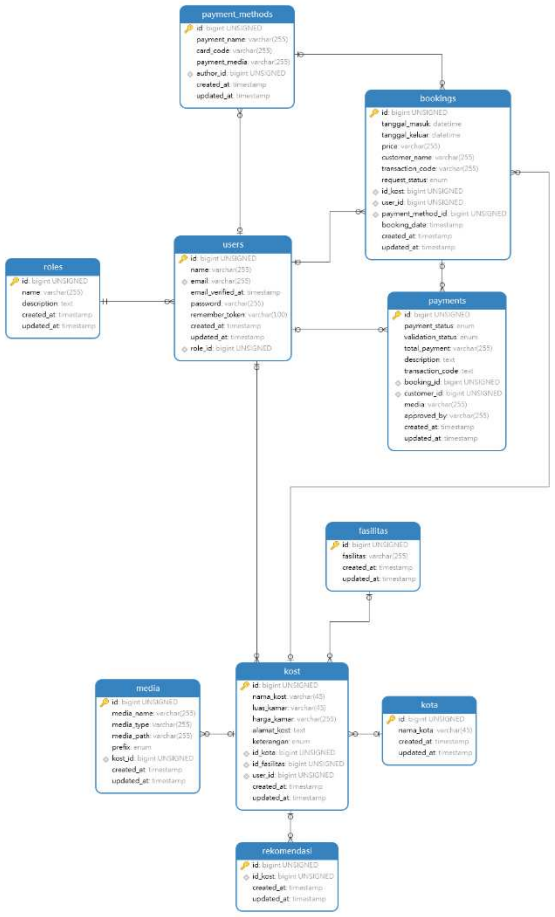


Gambar 7. *Activity Diagram* Akses Data *Resource Role Customer*

Gambar 4, gambar 5, gambar 6, dan gambar 7 di atas merupakan *activity diagram* yang menggambarkan bisnis proses pada aplikasi kos.

3.3.3. ERD Diagram

Perancangan *database* untuk kebutuhan aplikasi kos pada penelitian ini menggunakan *database* MySQL MariaDB. Untuk lebih jelasnya berikut gambar yang merupakan desain *database* ERD yang dirancang untuk kebutuhan aplikasi kos pada penelitian ini dapat dilihat pada gambar 8 berikut.



Gambar 8. ERD Diagram Project Kos

3.3.4. Daftar Fitur

Terkait daftar fitur yang tersedia pada arsitektur *microservice* ini di antaranya sebagai berikut:

a. Manajemen *users*
b. Manajemen kos
c. Manajemen fasilitas
d. Manajemen kota
e. Notifikasi email *approval*
f. Manajemen rekomendasi kos

3.4. Perancangan Pengujian

3.4.1. *Blackbox Testing*

Pengujian *blackbox* pada perancangan arsitektur ini dilakukan untuk mencari letak kesalahan dari sisi fungsionalitas sistem yang dirancang dalam penelitian ini, berikut merupakan daftar modul yang dilakukan proses pengujian:

a. Manajemen *users*.
b. Manajemen kos.
c. Manajemen fasilitas.
d. Manajemen kota
e. Notifikasi email *approval*
f. Manajemen rekomendasi kos.

Dari hasil pengujian *blackbox* tersebut secara fungsionalitas mendapatkan hasil yang sesuai dengan daftar fitur dokumen spesifikasi kebutuhan pada proses analisa perancangan.

3.4.2. *Locust Performance Testing*

Pada bagian ini merupakan proses pengujian beban pada arsitektur yang sudah tersedia (monolitik) dan pada arsitektur yang telah dilakukan proses *reengineering* untuk mendapatkan hasil tes performa yang lebih baik dari masing-masing arsitektur setelah proses komparasi. Untuk lebih jelasnya berikut tabel 1 merupakan daftar iterasi pengujian yang dilakukan selama tiga kali pengujian.

Tabel 1. Iterasi *Testing*

No	Jumlah *User*	*Rump Up (User Started/ Second)*
1	50	100 sec
2	100	100 sec
3	200	100 sec

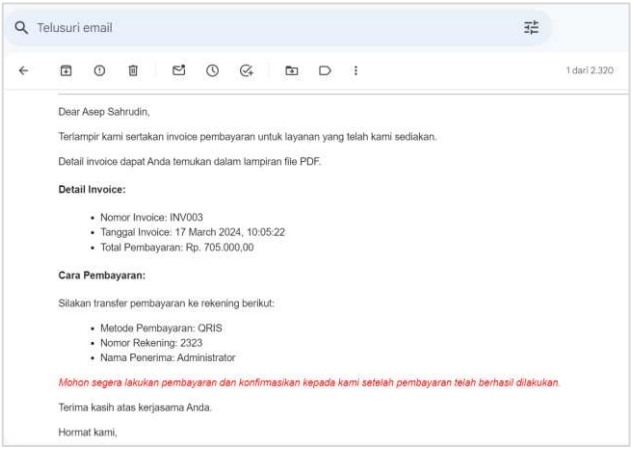
3.5. Implementasi Hasil

Berikut ini merupakan bagian dari proses implementasi hasil pengujian pada *project* kos penelitian ini. Ada 3 kriteria yang diuji pada penelitian ini diantaranya:

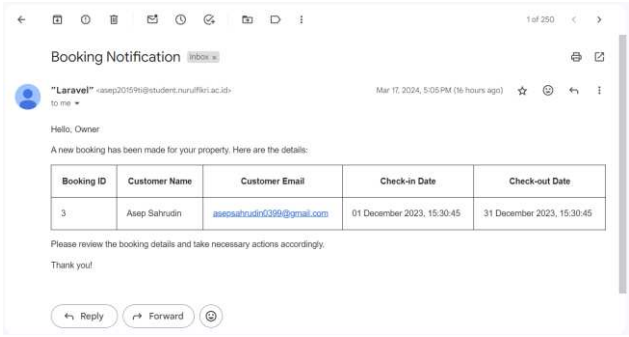
a. Bisnis proses pada *booking flow* pemesanan kos.
b. Monolitik vs *microservice* secara teknis dalam proses isolasi.
c. Hasil *testing* setelah dilakukan proses komparasi dua arsitektur (monolitik dan *microservice*).

3.5.1. Bisnis proses dan modul yang tersedia pada aplikasi kos

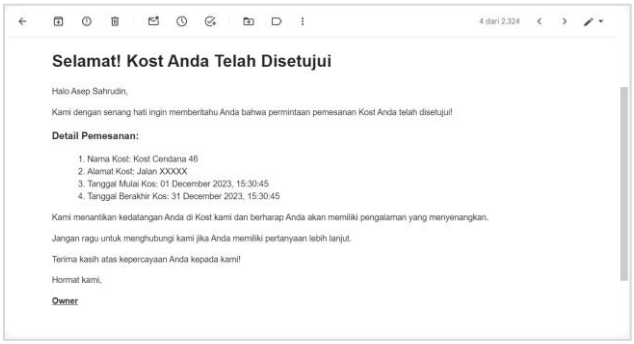
Berikut di bawah ini gambar 9, gambar 10, gambar 11, dan gambar 12 merupakan bisnis proses dan modul pada aplikasi kos.



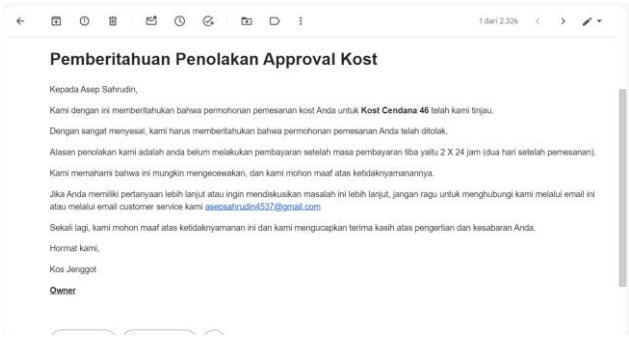
Gambar 9. Isi Pesan Email pada *Customer* Ketika Melakukan *Booking* Kos



Gambar 10. Isi Pesan Email yang Diterima Pemilik Ketika Ada Penyewa Melakukan *Booking* Kos



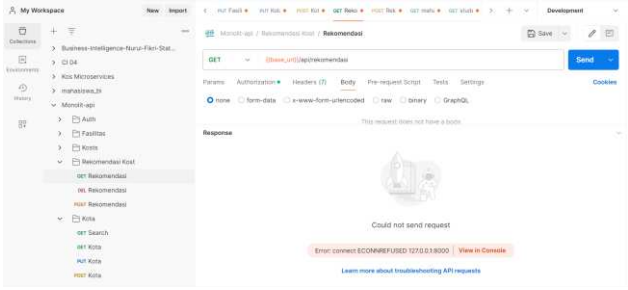
Gambar 11. Isi Pesan Email yang Diterima Penyewa Ketika Sudah Disetujui oleh Pemilik Kos



Gambar 12. Isi Email Penolakan Ketika Proses *Booking* Ditolak oleh Pemilik

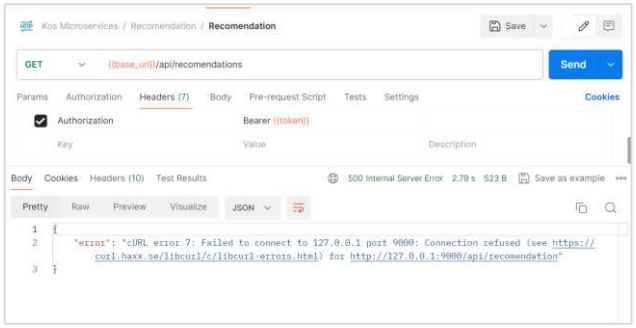
3.5.2. Monolitik vs *Microservice*

Arsitektur monolitik terdiri dari gabungan layer menjadi satu kesatuan yang utuh, sedangkan untuk arsitektur *microservice* memiliki karakteristik layer yang terpisah dan saling terhubung [8]. Pada bagian ini dijelaskan untuk membuktikan proses isolasi pada arsitektur *microservice* maupun monolitik.

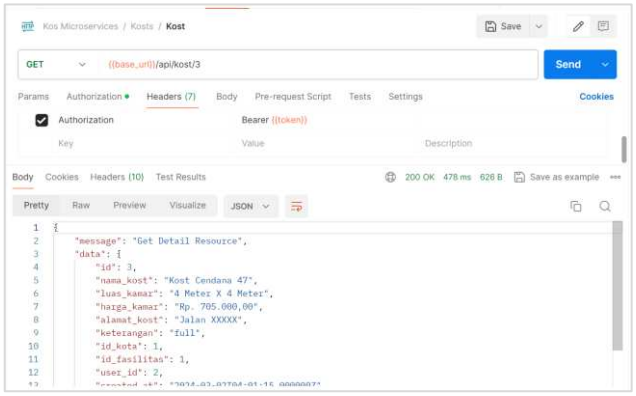


Gambar 13. Proses Isolasi pada Monolitik

Pada gambar 13 di atas merupakan salah satu bentuk hasil arsitektur monolitik ketika server *project* dimatikan maka seluruh servis pada *project* kos tersebut ikut mati karena seluruh modul maupun servis tersimpan dalam satu *project*.



Gambar 14. Hasil Tampilan *Microservice* Ketika Modul Rekomendasi Kos Dimatikan.



Gambar 15. Tampilan Modul Kos pada Arsitektur *Microservice*

Pada gambar 14 dan gambar 15 tersebut merupakan salah satu bentuk isolasi pada *microservice*, ketika salah satu modul dimatikan pada modul rekomendasi kos tapi pada saat kita akses modul kos masih berjalan dengan lancar.

3.6. Hasil Data Pengujian

Pada bagian ini setiap *service* diuji dengan metode pengujian *load testing*. Tujuan penggunaan *load testing* pada penelitian ini adalah untuk mengetahui hasil performa antara masing masing servis yang ada (monolitik dan *microservice*).

Berdasarkan dari ketiga hasil pengujian pada kedua arsitektur (monolitik dan *microservice*) tersebut peneliti coba jelaskan dan ringkas ke dalam suatu tabel 2 di bawah ini supaya lebih tergambar hasil komparasinya.

Tabel 2. Hasil Pengujian Ketiga Iterasi dari *Screenshot* di Atas

Arsitektur	***Users***	***Ramp up (Started Users / Second)***	**Laju Kinerja / *RPS* (*Request Persecond*)**	***Failures***
Skenario Pertama				
Monolitik	50	100 detik	2.6 Detik	38%

Arsitektur	*Users*	*Ramp up (Started Users / Second)*	Laju Kinerja / *RPS (Request Persecond)*	*Failures*
Microservice	50	100 detik	1.7 Detik	0%
Skenario Kedua				
Monolitik	100	100 detik	2.5 Detik	66%
Microservice	100	100 detik	3.3 Detik	30%
Skenario Ketiga				
Monolitik	200	100 detik	5.4 Detik	100%
Microservice	200	100 detik	1.6 Detik	40%

Berdasarkan dari ketiga hasil *scenario testing* di atas, pada tabel 2 menunjukkan bahwa beban akses *users* secara normalnya diangka 50/100 detik untuk mendapatkan total *error* sebesar 0% dan untuk *scenario* paling besar di waktu terakhir untuk beban akses 200 *user* dalam waktu 100 detik memiliki total *error* 100% untuk arsitektur monolitik dan masih tetap unggul untuk arsitektur *microservice* dengan mendapatkan total error dibawah 100% (40%) untuk arsitektur *microservice*. Dari hasil tersebut dapat disimpulkan, bahwa efektifitas penggunaan arsitektur *microservice* pada perancangan ulang *project* kos ini lebih disarankan menggunakan arsitektur *microservice* jika merujuk pada hasil *testing* performa tersebut.

3.7. Evaluasi Sistem

3.7.1. *Performance Testing*

Berdasarkan hasil penelitian dapat disimpulkan bahwa dengan melakukan *reengineering* arsitektur pada pengembangan aplikasi yang tadinya menggunakan arsitektur monolitik menjadi arsitektur *microservice* pada aplikasi kos jenggot menunjukkan hasil yang lebih bagus kualitasnya dengan mendapatkan hasil peningkatan yang lebih optimal dan lebih unggul pada arsitektur *microservice*. Dari hasil pengujian percobaan ke 1 untuk arsitektur monolitik diberikan beban akses sebesar 50 *user* mendapatkan hasil laju kerja 2.6/detik dan dengan memberikan beban akses yang sama 50 *user* pada arsitektur *microservice* mendapatkan hasil laju kerja 1.7/detik, pada percobaan yang kedua diberikan beban akses 100 *user* pada kedua arsitektur tersebut dengan mendapatkan hasil laju kinerja 2.5/detik pada arsitektur monolitik dan mendapatkan laju kinerja sebesar 3.3/detik pada arsitektur *microservice*, pada percobaan ketiga dilakukan peningkatan yang lebih tinggi untuk kedua arsitektur tersebut yaitu memberikan beban akses sebesar 200 *user* untuk melihat perbedaan yang lebih signifikan, dari hasil percobaan ke 3 tersebut mendapatkan hasil laju kerja sebesar 5.4/detik pada arsitektur monolitik dan 1.6/detik untuk arsitektur *microservice*. Dari seluruh percobaan pada *testing* yang telah dilakukan pada aplikasi kos jenggot dapat kita simpulkan bahwa dengan menggunakan arsitektur *microservice* lebih unggul dibandingkan menggunakan arsitektur monolitik.

3.7.2. *Blackbox Testing*

Pada bagian pengujian *blackbox* ini digunakan untuk mengetahui kesesuaian fungsi dari fitur yang dikembangkan pada *project* kos penelitian. Ada beberapa modul yang dilakukan pengujian menggunakan *blackbox* pada *project* kos ini di antaranya:

a. Modul *users*
b. Modul kos
c. Modul kota
d. Modul fasilitas
e. Modul kota
f. Modul rekomendasi kos
g. Fitur *booking*.

Berdasarkan dari yang telah dilakukan pengujian *blackbox* hasil yang didapatkan dari keseluruhan *scenario* dan butir uji mendapatkan hasil yang sesuai dengan spesifikasi kebutuhan fitur.

4. KESIMPULAN

Berdasarkan dari hasil penelitian yang telah dilakukan terdapat beberapa kesimpulan sebagai berikut:

1. Melakukan *reengineering backend* web dari monolit ke *microservice* menggunakan *Framework* Laravel merupakan salah satu solusi yang memberikan dampak positif khususnya pada kinerja aplikasi yang telah dirancang sehingga memudahkan *user* tanpa harus menunggu terlalu lama ketika aplikasi hendak diakses oleh banyak *user* dan dapat dilihat dari hasil setelah dilakukan pengujian dengan menggunakan *locust tools* pada *scenario* pertama mendapatkan 1.7 RPS (*Request per-second*) dengan stabil dengan tingkat *failures* 0%.
2. Untuk beberapa perbedaan antara penggunaan arsitektur monolitik dan *microservice* pada penelitian ini yang terletak pada tingkat kinerja aplikasi yang dirancang, dimana aplikasi yang menggunakan arsitektur *microservice* memiliki performa lebih baik dibandingkan menerapkan arsitektur monolitik. Hal tersebut dibuktikan pada saat pengujian kinerja aplikasi menggunakan *locust* untuk melihat kekuatan aplikasi pada saat menerima beban yang tinggi ketika diakses oleh banyak user. Untuk tingkat keberhasilan komparasi keunggulan performa *microservice* dan monolitik dapat dilihat pada poin 4.2.3 tentang hasil data pengujian di bagian tabel *summary* hasil *testing* 3 skenario yang telah dilakukan. Berdasarkan hasil pengujian pada tabel tersebut dapat disimpulkan bahwa hasil pengujian arsitektur *microservice* bisa dikatakan berhasil lebih unggul secara performa maupun dari tingkat *failures* yang rendah ketika diberikan peningkatan beban sampai mentok di 200 *users* dalam waktu 100 detik.

DAFTAR PUSTAKA

[1] Badan Pusat Statistik Provinsi DKI Jakarta, "Jumlah Penduduk Menurut Kabupaten/Kota di Provinsi DKI Jakarta (Jiwa)," *Jumlah Penduduk Menurut Kabupaten/Kota di Provinsi DKI Jakarta (Jiwa),* 2020-2022.

[2] S. Waruwu dan K. D. Nuryana, "Implementasi Arsitektur Monolitik Pada Rancang Bangun Sistem Informasi," *Journal of Informatics and Computer Science,* vol. 4, pp. 2686-2220, 2023.

[3] S. A. Karimah, H. H. Latif, dan S. Prabowo, "Analisis Performansi Layanan Web Menggunakan Arsitektur Microservice dan Monolitik," *KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen),* vol. 4, pp. 830-840, 2023.

[4] Badan Pengembangan dan Pembinaan Bahasa Kemdikbudristek RI, "Pengertian Restrukturisasi," 2016. [Online]. Available: https://kbbi.kemdikbud.go.id/entri/restrukturisasi. [Diakses 25 Maret 2024].

[5] M. Raharjo, M. Napiah, and R. S. Anwar, "Perancangan Sistem Informasi dengan PHP dan MySQL untuk Pendaftaran Sekolah di Masa Pandemi," *Computer Science (CO-SCIENCE),* vol. 2, no. 1, pp. 2808-9065, 2022.

[6] A. R. R. O'g'li, "The Difference Between The Concepts Of Database (DB) and Database Management System (DBMS)," *Humanity and Science Congress Hosted From Essen, Germany,* 2022.

[7] M. Toni dan A. Hadi, "Pengembangan Sistem Informasi Akademik Politeknik LP3I Kampus Padang Menggunakan Framework Laravel," *Jurnal Sains dan Teknologi Informatika,* vol. 1, no. 2, pp. 73-79, 2023.

[8] D. Y. Arimbi, D. Kartinah, dan W. A. N. Della, "Rancangan Sistem Informasi Kost Putri Malika Berbasis Website Menggunakan Framework Laravel dan MySQL," *Jurnal Ilmiah Multidisiplin (JUKIM), vol. 1, no. 3, pp. 93-103, 2022.*

[9] F. Arifien, Rozi, dan E. Sutomo, "Implementasi Arsitektur Microservices Pada Sistem Informasi Akademik Stmik Jakarta Sti&K Menggunakan Model Enterprise Javabeans (EJB) dan Polymer JS," *Seminar Nasional Teknologi Informasi dan Komunikasi STI&K (SeNTIK),* vol. 5, 2021.

[10] N. M. D. Febriyanti, A. A. K. O. Sudana, and I. N. Piarsa, "Implementasi Black Box Testing pada Sistem Informasi Manajemen Dosen," *JITTER-Jurnal Ilmiah Teknologi dan Komputer,* vol. 2, 2021.

[11] R. I. Borman, A. T. Priandika, and A. R. Edison, "Implementasi Metode Pengembangan Sistem Extreme Programming (XP) pada Aplikasi Investasi Peternakan," *Jurnal Sistem dan Teknologi Informasi (JUSTIN),* vol. 8, pp. 272-277, 2020.