RANCANG BANGUN SISTEM INFORMASI PENDAFTARAN DAN ABSENSI ASLAB BERBASIS *CONTAINER-AS-A-SERVICE* (CAAS) (STUDI KASUS: FASTAPI DAN GOOGLE CLOUD RUN)

Firman Ali[1], Hikmal Dwi Rifa'i[2], Putri Mentari Endaswari[3]

[123]Universitas Bangka Belitung, Gang IV No.1, Balun Ijuk, Kec. Merawang, Kabupaten Bangka, Kepulauan Bangka Belitung

Email : [1]firmanali2701@gmail.com, [2]hikmaldwirifai@gmail.com, [3]putrimentari@ubb.ac.id*

ABSTRAK

Pengelolaan asisten laboratorium (aslab) yang masih dilakukan secara manual sering kali menyebabkan masalah seperti ketidakefisienan dalam pengelolaan data, keterlambatan informasi, dan kurangnya transparansi dalam proses seleksi. Di sisi lain, pengembangan sistem informasi di lingkungan kampus sering terkendala masalah infrastruktur *server* tradisional (IaaS) yang rumit dalam pemeliharaan dan tidak efisien dalam hal biaya operasional terutama saat trafik rendah. Penelitian ini bertujuan untuk merancang bangun Sistem Informasi Pendaftaran, Administrasi, dan Absensi Aslab dengan menerapkan arsitektur modern berupa *Container-as-a-Service* (CaaS). Sistem dibangun menggunakan *framework* Python FastAPI, dikemas dengan teknologi *container* Docker, dan di-*deploy* pada layanan Google Cloud Run yang terintegrasi dengan Google Cloud SQL. Metode penelitian menggunakan model *Research and Development* (R&D) yang diadaptasi dari model Borg & Gall. Hasil dari pengujian *Black-Box* dan *User Acceptance Testing* (UAT) menunjukkan bahwa sistem berjalan dengan baik di lingkungan *cloud serverless* dan memenuhi kebutuhan fungsional pengguna. Penggunaan arsitektur CaaS memberikan keunggulan signifikan dalam hal skalabilitas otomatis (*auto-scaling*), efisiensi biaya melalui model *pay-per-request*, serta kemudahan proses *deployment* dibandingkan infrastruktur *server* konvensional.

Keywords: *Cloud Computing, Container-as-a-Service, Google Cloud Run, FastAPI, Docker.*

ABSTRACT

Manual management of laboratory assistants (aslab) often leads to problems such as inefficiency in data management, information delays, and a lack of transparency in the selection process. On the other hand, the development of information systems in campus environments is often hampered by traditional server infrastructure (IaaS) which is complicated to maintain and inefficient in terms of operational costs, especially during low traffic. This study aims to design and build an Aslab Registration, Administration, and Attendance Information System by implementing a modern architecture in the form of Container-as-a-Service (CaaS). The system is built using the Python FastAPI framework, packaged with Docker container technology, and deployed on Google Cloud Run services integrated with Google Cloud SQL. The research method uses a Research and Development (R&D) model adapted from the Borg & Gall model. The results of Black-Box testing and User Acceptance Testing (UAT) indicate that the system runs well in a serverless cloud environment and meets user functional needs. The use of a CaaS architecture provides significant advantages in terms of automatic scalability (auto-scaling), cost efficiency through a pay-per-request model, and ease of deployment compared to conventional server infrastructure.

Keywords: *Cloud Computing, Container-as-a-Service, Google Cloud Run, FastAPI, Docker.*

1. PENDAHULUAN

Perkembangan teknologi informasi di zaman digital telah mengubah secara drastis cara pengelolaan data dan administrasi, termasuk dalam dunia pendidikan tinggi. Institusi pendidikan tinggi harus terus menyesuaikan diri agar bisa memberikan layanan akademik yang efektif, efisien, dan dapat dipertanggungjawabkan. Salah satu isu yang sering timbul adalah proses rekrutmen dan pengelolaan asisten laboratorium (aslab) yang masih dilakukan dengan cara manual, mulai dari pendaftaran, pemeriksaan dokumen, hingga pelaporan aktivitas [1]. Metode manual ini berpotensi menyebabkan keterlambatan, kesalahan dalam pencatatan, serta ketidakefisienan dalam penyampaian informasi [2].

Sebagai jawabannya, sistem informasi dan absensi online berbasis web telah terbukti efektif dalam mengatasi banyak kekurangan dari metode manual [3]. Penelitian di bidang perusahaan dan pendidikan membuktikan bahwa sistem absensi digital memainkan peran penting dalam mengurangi beban administrasi, meminimalisir risiko kehilangan data, serta meningkatkan disiplin kerja [4]. Dalam konteks pendidikan tinggi, penerapan rekrutmen asisten praktikum yang berbasis digital memberikan kontribusi nyata terhadap peningkatan transparansi, akuntabilitas, dan efisiensi [2],[5].

Namun, hanya memiliki sistem informasi berbasis web yang fungsional saja tidak cukup untuk menghadapi tantangan modern, terutama tantangan terkait proses penyebaran (*deployment*) dan pemeliharaan (*maintenance*). Pendekatan tradisional umumnya menggunakan layanan *Infrastructure-as-a-Service* (IaaS), seperti menyewa *Virtual Private Server* (VPS), seringkali membebani pengembang dengan dituntut untuk mengelola semua hal secara manual, mulai dari penginstalan dan konfigurasi sistem operasi, penerapan *patch* keamanan, pengaturan web *server* seperti *Nginx*, hingga mengelola skalabilitas *server* saat trafik pengguna meningkat. Kompleksitas ini dapat mengalihkan fokus pengembang dari pengembangan fitur aplikasi itu sendiri [6]. Untuk mengatasi persoalan tersebut, arsitektur *cloud* modern seperti *Platform-as-a-Service* (PaaS) menawarkan solusi yang jauh lebih efisien. PaaS memungkinkan pengembang fokus sepenuhnya pada penulisan kode aplikasi tanpa mengelola infrastruktur dasarnya, sehingga meningkatkan efisiensi pengembangan [7]. Khususnya layanan *serverless container* seperti Google Cloud Run memungkinkan aplikasi berjalan dengan model sistem penagihan per permintaan (*pay-per-request*) dan skalabilitas otomatis termasuk *scale-to-zero*, sehingga jauh lebih efisien dari segi biaya dan pemeliharaan [8].

Dengan adanya kebutuhan akan infrastruktur yang efisien, memilih teknologi pengembangan aplikasi yang tepat sangat penting. Menerapkan *framework* modern yang cepat dan ringan sangat diperlukan agar bisa mendukung kinerja arsitektur *cloud* tersebut. Hal ini sejalan dengan temuan penelitian sebelumnya yang menunjukkan bahwa menggunakan virtual *server* berbasis *container* bisa menciptakan lingkungan pengembangan sistem yang lebih efisien, fleksibel, dan mudah diperluas dibandingkan metode konvensional [9]. Menggabungkan sistem informasi akademik yang dinamis dengan infrastruktur *serverless* menjadi salah satu topik yang menarik untuk dieksplorasi, khususnya dalam menangani pola akses kampus yang sering berubah. Oleh karena itu dibutuhkan sebuah penelitian implementatif yang menggabungkan keandalan sistem berbasis web modern dengan efisiensi arsitektur *deployment* masa kini agar dapat menyelesaikan permasalahan administrasi laboratorium secara menyeluruh.

Berdasarkan latar belakang tersebut, penelitian ini mengusulkan Rancang Bangun Sistem Informasi Pendaftaran, Administrasi, dan Absensi Aslab yang tidak hanya dibangun menggunakan *framework* FastAPI, namun juga diimplementasikan pada arsitektur *deployment* CaaS. Penelitian ini juga menyertakan integrasi notifikasi *real-time* melalui WhatsApp API. Tujuan dari penelitian ini adalah untuk (1) Merancang dan menerapkan arsitektur *deployment* CaaS untuk Sistem Informasi Pendaftaran, Administrasi, dan Absensi Aslab menggunakan Google Cloud Run dan Cloud SQL. (2) Menganalisis keunggulan kualitatif arsitektur PaaS/CaaS dibandingkan IaaS tradisional, terutama dari segi efisiensi *deployment*, skalabilitas, dan efisiensi biaya.

2. TINJAUAN PUSTAKA

2.1 Sistem Informasi Asisten Laboratorium

Sistem adalah sekumpulan elemen atau komponen yang saling berhubungan dan saling mempengaruhi antara satu dengan lainnya yang bekerja bersama-sama sesuai dengan aturan yang ditentukan untuk mencapai suatu tujuan [10][11]. Dalam sebuah sistem, jika salah satu komponennya tidak berfungsi atau rusak, maka sistem tidak akan bekerja seperti yang diharapkan. Dengan kata lain, sistem dapat diartikan sebagai sekumpulan variabel-variabel yang saling terhubung, berinteraksi, dan saling bergantung satu sama lain. Dalam konteks penelitian ini, sistem yang dimaksud secara spesifik merujuk pada Sistem Informasi Pendaftaran, Administrasi, dan Absensi Aslab. Sistem ini menggabungkan proses pendaftaran, seleksi, serta pelaporan absensi asisten laboratorium dalam satu alur kerja yang terstruktur dan terotomatis.

2.2 FastAPI

FastAPI adalah sebuah kerangka kerja (*framework*) *backend* yang cepat dan efisien untuk membangun aplikasi web dengan menggunakan Bahasa pemrograman Python. Keunggulan utamanya terletak pada kemampuan validasi data secara otomatis menggunakan Pydantic, serta pembuatan dokumentasi otomatis yang sangat baik, sehingga mempercepat proses pengembangan. FastAPI seringkali digunakan untuk membangun aplikas dengan kinerja tinggi dan validasi data yang ketat dalam lingkungan yang efisien [12].

2.3 Model Layanan *Cloud* (IaaS, PaaS, SaaS)

Dalam *cloud computing*, terdapat tiga model layanan utama []:

1. *Infrastructure-as-a-Service* (IaaS). Merupakan jenis layanan dalam komputasi awan di mana pengguna bisa menyewa infrastruktur teknologi informasi seperti *network*, *storage*, *memory* dan berbagai unit komputasi lainnya. Contoh penyedia dari layanan IaaS meliputi Amazon EC2, Rackspace Cloud, Windows Azure, dan lainnya [13]. Keuntungan dari layanan ini adalah pengguna tidak perlu membeli komputer fisik. pengguna juga dapat menambah atau mengurangi komponen dengan mudah sesuai kebutuhan tanpa menganggu proses yang sedang berjalan [14]
2. *Platform-as-a-Service* (PaaS). Merupakan jenis layanan dalam komputasi awan yang memungkinkan pengguna menyewa "rumah" beserta lingkungannya untuk menjalankan aplikasi yang telah dibuat. Pengguna tidak perlu mempersiapkan dan merawat "rumah" tersebut. Contoh penyedia layanan PaaS meliputi Amazon Web Service, Windows Azure, dan Google App Engine [13]. Layanan ini memudahkan pengguna dalam mengembangkan sebuah sistem, termasuk di dalamnya *environment*, sistem operasi, *database* serta keperluan lainnya yang menunjang pengembangan sistem [14].
3. *Software-as-a-Service* (SaaS). Merupakan jenis layanan dalam komputasi awan dimana pengguna bisa menggunakan perangkat lunak yang telah disediakan oleh penyedia layanan *cloud*. Contoh dari layanan SaaS di antaranya seperti Office 365, Gmail, Facebook, Yahoo Messenger dan lainnya [13]. Pada layanan ini pengguna biasanya dikenakan biaya tambahan jika ingin membuka fitur-fitur yang lebih lengkap [14].

2.4 *Containerization* (Docker)

Containerization (kontainerisasi) adalah teknologi yang memungkinkan aplikasi berjalan dalam lingkungan terisolasi, lengkap dengan semua dependensi dan konfigurasi yang diperlukan. Teknik ini mempermudah proses pengembangan, *deployment*, dan pengelolaan aplikasi, karena aplikasi bisa berjalan diberbagai lingkungan tanpa perlu penyesuaian tambahan. Docker adalah salah satu contoh platform kontainerisasi yang paling popular dan banyak digunakan, karena bisa memberikan fleksibilitas dan kemudahan dalam memindah aplikasi ke berbagai tempat [15].

2.5 *Container-as-a-Service* (CaaS)

Container-as-a-Service (CaaS) adalah model layanan komputing berbasis *cloud* yang memberikan kebebasan tinggi dalam mengatur lingkungan *server*. Berbeda dengan PaaS yang memiliki spesifikasi lingkungan yang lebih terbatas, CaaS memungkinkan pengembang untuk menyesuaikan lingkungan *server* sesuai dengan kebutuhan aplikasi yang akan di luncurkan [16]. CaaS dirancang khusus untuk mengelola *container*.

2.6 Google Cloud Run

Google Cloud Run adalah salah satu layanan komputasi yang mengimplementasi *Container-as-a-Service* (CaaS) yang *fully managed* dan bersifat *serverless* di mana menggabungkan kepraktisan dari *Platform-as-a-Service* (PaaS) dengan fleksibilitas dan portabilitas dalam mengatur tempat serta skalabilitas aplikasi berbentuk *container* [17]. *Serverless* sendiri berarti pengembang tidak perlu memikirkan *server* sama sekali. Model ini sangat efisien karena pengguna hanya perlu membayar berdasarkan *request* yang digunakan (*pay-per-request*).

3. METODE PENELITIAN

Metode penelitian yang digunakan dalam penelitian ini adalah *Research and Development* (R&D). *Research and Development* sendiri merupakan metode penelitian yang digunakan untuk menghasilkan suatu produk tertentu [18]. Tujuannya yaitu untuk membuat sistem berbasis web dengan arsitektur *deployment* yang modern. Model pengembangan yang digunakan mengadopsi model Borg & Gall yang disederhanakan menjadi beberapa tahapan utama sesuai dengan pengembangan perangkat lunak dan penerapan infrastruktur *cloud*. Adapun tahapan penelitian ini digambarkan secara detail dan dapat dilihat pada Gambar *1*.

Gambar 1. Tahapan Metode R&D

3.1 Identifikasi Masalah

Tahap awal dalam penelitian ini adalah untuk menemukan dan merumuskan permasalahan yang terjadi di lapangan. Melalui observasi terhadap proses rekrutmen dan pengelolaan asisten laboratorium yang masih dilakukan secara manual, ditemukan berbagai kendala mulai dari keterlambatan dalam penyampaian informasi dan komunikasi, kesalahan pencatatan data, dan kurang transparansi proses seleksi. Hasil identifikasi ini menjadi acuan dasar bagi peneliti untuk merancang solusi sistem yang saling terintegrasi.

3.2 Pengumpulan Data

Tahapan ini dilakukan untuk mengumpulkan data melalui wawancara langsung kepada kepala laboratorium terpadu, dosen pengampu matakuliah asisten laboratorium, dan mahasiswa calon asisten laboratorium. Selain itu, dilakukan studi terhadap dokumen administrasi yang dibutuhkan serta analisis studi literatur mendalam mengenai teknologi *cloud computing* (PaaS/CaaS) dan *Containerization* untuk menentukan solusi infrastruktur yang paling efisien.

3.3 Analisis Kebutuhan

Tahap ini bertujuan untuk mengidentifikasi kebutuhan baik dari fungsionalitas maupun non-fungsionalitas dari sistem. Adapun kebutuhan fungsional sistem meliputi fitur pendaftaran, seleksi, dan absensi serta kebutuhan non-fungsional terkait arsitektur *deployment*. Sistem memerlukan lingkungan yang *scalable*, minim perawatan (maintenance-free), serta efisien secara biaya, sehingga Google Cloud Run dipilih sebagai platform *deployment.*

3.4 Perancangan Sistem

Tahapan perancangan mencakup 2 aspek utama. Pertama, perancangan dilakukan dengan menggunakan pendekatan *Unified Modeling Language* (UML) yang mencakup *use case diagram, class diagram, activity diagram, Sequence diagram, dan entity relationship diagram* (ERD) untuk menggambarkan alur proses sistem. Kedua, perancangan arsitektur *cloud* yang mendefinisikan bagaimana aplikasi FastAPI dikemas dalam bentuk *container* Docker dan dihubungkan dengan layanan Google Cloud SQL.

3.5 Implementasi dan *Deployment* (*Build*)

Tahap ini adalah proses mengubah desain menjadi aplikasi nyata yang bisa digunakan. Kode program ditulis dengan menggunakan bahasa pemrograman Python dan *framework* FastAPI. Setelah itu, aplikasi melalui proses *containerization* menggunakan Docker. *Image* Docker yang terbentuk kemudian di-*deploy* ke layanan Google Cloud Run, sedangkan *database* dimigrasikan ke Google Cloud SQL.

3.6 Pengujian Sistem

Setelah sistem berhasil di-*deploy*, dilakukan pengujian dengan metode *Black-Box Testing* untuk memastikan setiap fitur dapat berjalan dengan baik sesuai dengan yang diharapkan. Selain itu, akan dilakukan pengujian mengenai aksesibilitas sistem agar dapat diakses melalui alamat URL publik yang diberikan oleh penyedia layanan *cloud*.

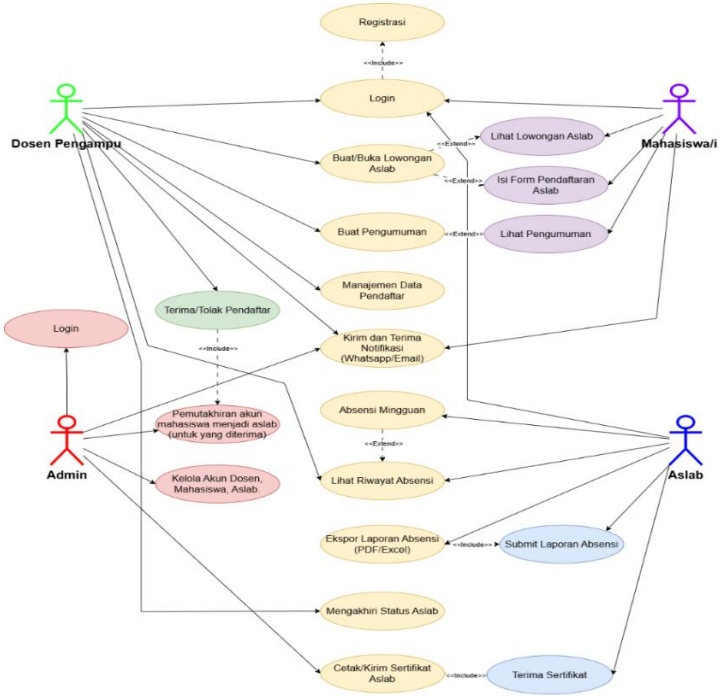
4. PEMBAHASAN

4.1 Perancangan Sistem

Sistem dirancang agar dapat menampilkan gambaran visual tentang alur kerja dan interaksi antar komponen dalam sistem. Pendekatan yang digunakan untuk memodelkan visual ini adalah *Unified Modeling Language* (UML). Dengan pendekatan UML, struktur dan cara kerja sistem dijelaskan secara standar agar lebih mudah dipahami bagaimana pengguna berinteraksi dengan sistem. Selain perancangan perangkat lunak menggunakan UML, tahap ini juga mencakup perancangan struktur *database* serta desain infrastruktur *cloud* yang akan digunakan sebagai lingkungan *deployment*.

A. *Use Case Diagram*

Use Case Diagram menggambarkan cara aktor saling berinteraksi dengan sistem secara menyeluruh. Terdapat empat aktor utama dalam sistem ini, yaitu Admin, Dosen, Mahasiswa, dan Aslab. Admin memiliki akses penuh untuk mengelola data pengguna. Dosen bertugas mengelola lowongan aslab dan menyeleksi Mahasiswa. Mahasiswa bisa mendaftar pada lowongan yang tersedia, sedangkan Aslab yang diterima bisa melakukan absensi mingguan. Visualisasi interaksi aktor dengan sistem tersebut dapat dilihat pada Gambar *2*.



Gambar 2. *Use Case Diagram*

B. Desain Arsitektur *Cloud* (CaaS)

Berbeda dengan cara penyebaran tradisional di VPS, sistem ini dirancang dengan menggunakan arsitektur *Container-as-a-Service* (CaaS). Aplikasi FastAPI dikemas dalam *container* Docker dan dijalankan di atas platform Google Cloud Run. Arsitektur ini memisahkan bagian aplikasi (*stateless container*) dengan bagian penyimpanan data (*manage database*). Permintaan dari pengguna (Mahasiswa/Dosen/Admin) diteruskan ke Google Cloud Run melalui protokol HTTPS, yang kemudian berkomunikasi dengan *database* di Google Cloud SQL. Rancangan arsitektur *deployment* tersebut diilustrasikan pada Gambar *3*.

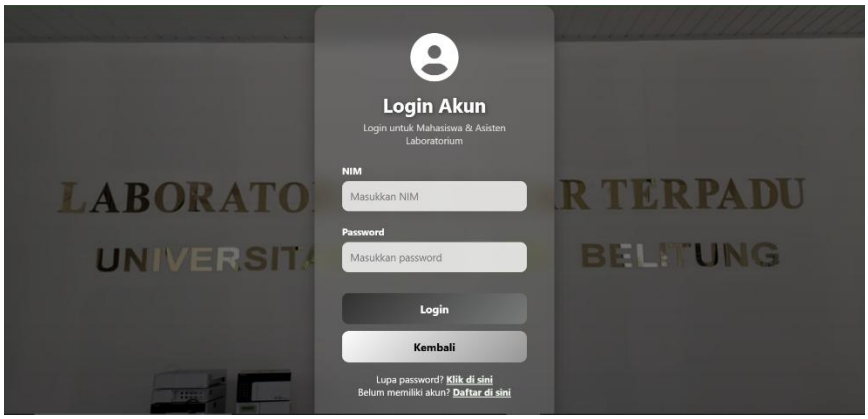
Gambar 3. Arsitektur *Deployment* pada Google Cloud Platform

4.2 Implementasi Antarmuka

Realisasi dari perancangan arsitektur dan logika sistem dilakukan pada tahap implementasi. Sistem Informasi Pendaftaran, Administrasi, dan Absensi Aslab ini diterjemahkan dari diagram UML dan skema database menjadi aplikasi web yang fungsional. Pada tahap perancangan, fokus utamanya adalah pada struktur logika dan aliran data, maka pada tahap ini fokus dialihkan ke tampilan antarmuka visual yang memudahkan pengguna dalam berinteraksi dengan sistem yang sudah diinstall di lingkungan cloud. Berikut adalah hasil implementasi antarmuka utama sistem ini.

A. Halaman *Login*

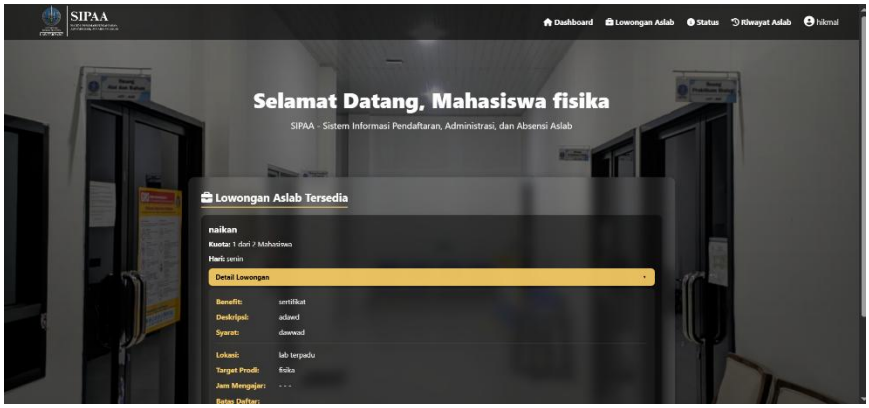
Halaman ini menjadi gerbang utama masuk ke dalam sistem. Sistem menggunakan mekanisme autentikasi untuk memverifikasi kredensial pengguna (NIM/NIP) dan kemudian diarahkan ke *dashboard* sesuai dengan perannya (Mahasiswa, Dosen, atau Admin). Tampilan halaman *login* akun dapat dilihat pada Gambar 4.



Gambar 4. Tampilan *Login*

B. Halaman Lowongan Aslab (Mahasiswa)

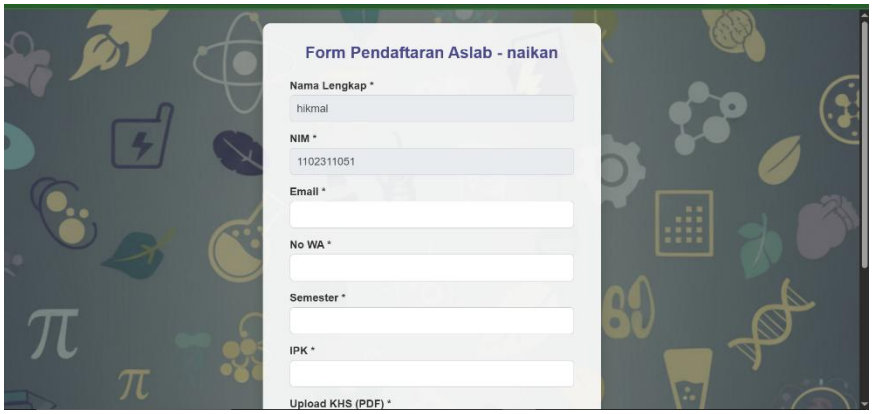
Pada halaman ini, Mahasiswa dapat melihat daftar lowongan asisten laboratorium yang telah dibuka oleh dosen. Informasi disajikan dalam bentuk kartu (*card*) yang memuat nama mata kuliah, kuota, deskripsi, syarat dan berbagai informasi lainnya. Terdapat juga tombol untuk melakukan pendaftaran pada lowongan. Tampilan halaman lowongan aslab dapat dilihat pada Gambar 5.



Gambar 5. Tampilan Detail Lowongan Aslab

C. Halaman Form Pendaftaran Aslab (Mahasiswa)

Halaman ini merupakan halaman untuk Mahasiswa mendaftar lowongan asisten laboratorium yang telah dibuka oleh dosen. Mahasiswa diminta untuk mengisi data seperti email, no.whatsapp, semester, IPK, dan berbagai dokumen administrasi lainnya. Pada bagian *field* nama merupakan *field absolute* berdasarkan akun yang mendaftar. Tampilan halaman form pendaftaran aslab dapat dilihat pada Gambar 6.



Gambar 6. Tampilan Form Pendaftaran Aslab

4.3 Analisis Keunggulan Arsitektur CaaS

Setelah sistem berhasil diimplementasikan dan antarmuka bisa diakses oleh pengguna, aspek penting berikutnya yang perlu dievaluasi adalah performa infrastruktur yang mendukung sistem tersebut. keberhasilan sistem ini tidak hanya bergantung pada kualitas kode programnya saja, tetapi juga pada efisiensi *server* yang menjadi tempat aplikas tersebut berjalan. Berdasarkan hasil *deployment* nyata di Google Cloud Run, penerapan arsitektur *Container-as-a-Service* (CaaS) terbukti memberikan sejumlah keunggulan kualitatif yang signifikan dibandingkan jika sistem dijalankan pada infrastruktur konvensional (VPS), yaitu:

1. Efisiensi *Deployment*: Proses *deployment* menggunakan Docker membuat proses distribusi aplikasi lebih mudah. Masalah ketidaksesuaian lingkungan (*environment dependency*) bisa dihindari karena aplikasi berjalan didalam wadah isolasi terpisah.
2. Skalabilitas Otomatis (*Auto-scaling*): Google Cloud Run bisa menyesuaikan jumlah *instance* secara otomatis. Saat tidak ada pengguna yang mengakses, *instance* aplikasi akan dimatikan agar tidak boros biaya. Namun ketika ada banyak pengguna yang mengakses sekaligus, sistem secara otomatis akan menambah *instance* untuk menangani beban trafik.
3. Efisiensi Biaya: Sistem pembayaran berdasarkan permintaan (*pay-per-request*) jauh lebih murah biaya operasionalnya dibandingkan menyewa *server* VPS yang harus selalu menyala 24 jam sehari, terutama untuk aplikasi kampus berskala menengah.

4.4 Pengujian Sistem (*Black Box*)

Analisis keunggulan infrastruktur sebelumnya menjamin efisiensi dari sisi *server*, namun keandalan fungsi fitur bagi pengguna akhir tetap harus divalidasi melalui pengujian langsung. Oleh karena itu, tahap akhir dari pembahasan ini adalah pengujian sistem menggunakan metode *Black-Box Testing* dan *User Acceptance* Testing (UAT). Pengujian ini dilakukan dengan tujuan untuk membuktikan bahwa sistem ini dapat memenuhi kebutuhan fungsional pengguna tanpa eror. Pengujian dilakukan pada tanggal 14 November 2025 di Laboratorium Terpadu Universitas Bangka Belitung. Berdasarkan dokumen UAT, pengujian mencakup 12 modul dengan total 96 skenario pengujian yang melibatkan skenario positif dan negatif. Hasil pengujian untuk scenario-skenario penting tersebut disajikan secara ringkas pada Tabel 1.

Tabel 1. Rekapitulasi Hasil Pengujian Sistem (Sampel UAT)

Kode Uji	Modul & Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian (Aktual)	Status
AUTH-15	Autentikasi (*Login* Multi-Peran) Mahasiswa yang juga Aslab melakukan *Login*.	Sistem mendeteksi peran ganda dan menampilkan *Pop-up* pilihan peran (Mahasiswa/Aslab).	Muncul *Pop-up* pilihan peran sesuai harapan.	OK
LOW-02	Manajemen Lowongan Dosen membuat lowongan baru dengan data lengkap.	Data tersimpan, notifikasi sukses muncul, dan *file* pamflet tersimpan di *server*.	Notifikasi sukses muncul, data dan *file* tersimpan.	OK
MHS-REG-05	Pendaftaran Aslab Mahasiswa mendaftar ulang di lowongan yang sama (Duplikasi).	Sistem menolak pendaftaran dengan pesan "Anda sudah mendaftar".	Sistem menolak dan pesan error muncul.	OK
SEL-04	Seleksi Aslab Dosen mengubah status pendaftar menjadi "Diterima".	Status berubah jadi "Diterima" (Hijau) dan mahasiswa menerima email notifikasi.	Status berubah dan email notifikasi terkirim.	OK
ACT-09	Aktivasi Aslab (Admin) Admin mengaktifkan akun mahasiswa yang diterima.	Data mahasiswa disalin ke tabel akun_aslab dan email instruksi *Login* terkirim.	Data tersalin dan email terkirim.	OK
ABS-02	Absensi Harian Aslab mengirim absensi dengan foto bukti.	Data absensi dan *file* foto tersimpan di database dan *server*.	Notifikasi sukses, data tersimpan.	OK
END-02	Pengakhiran Kontrak Dosen mengakhiri kontrak Aslab.	Status menjadi "Selesai" dan hak akses Aslab dicabut otomatis.	Pesan sukses muncul, akses dicabut.	OK
CERT-03	Sertifikat Digital Admin mengirim sertifikat ke Aslab yang selesai kontrak.	*File* PDF sertifikat ter-*generate* dan terkirim via email ke mahasiswa.	Notifikasi sukses, email sertifikat diterima.	OK

5. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Penelitian ini berhasil merancang dan mengembangkan Sistem Informasi Pendaftaran, Administrasi, dan Absensi Aslab menggunakan *framework* FastAPI. Sistem tersebut di-*deploy* dengan arsitektur *Container-as-a-Service* (CaaS) pada Google Cloud Run. Integrasi antara aplikasi (*stateless container*) dan *database* Google Cloud SQL (*manage service*) berjalan dengan baik dan bisa diakses melalui jaringan publik.
2. Sistem yang diimplementasi menggunakan teknologi Docker dan Google Cloud Run memberikan solusi infrastruktur yang lebih hemat dibandingkan menggunakan VPS biasa. Fitur *auto-scaling* dan *scale-to-zero* di Google Cloud Run memungkinkan penggunaan sumber daya secara optimal karena sistem akan secara otomatis mematikan instance saat tidak ada aktivitas, sehingga mengurangi biaya operasional.
3. Arsitektur yang digunakan (*container serverless*) dapat mengurangi beban administrasi *server* (instalasi OS dan *patching* keamanan) bagi pengembang. Dengan hal ini, tim pengembang dapat lebih berfokus pada pengembangan fitur-fitur fungsional dari sistem ini.

UCAPAN TERIMA KASIH

Terima kasih kepada LPMPP Universitas Bangka Belitung atas dukungannya terhadap program penelitian ini melalui Hibah Pendanaan skema Team Based Project (TBP). Terima kasih juga kepada Lab Terpadu UBB atas bantuannya yang luar biasa dalam memastikan kelancaran penelitian ini.

DAFTAR PUSTAKA

[1] A. Syam, A. Asniati, and W. O. D. S. Herman, “Rancang Bangun Sistem Pendaftaran dan Seleksi Asisten Laboratorium Berbasis Android,” *J. Inform.*, vol. 12, no. 2, pp. 121–129, 2023, doi: 10.55340/jiu.v12i2.1418.

[2] N. A. L. Siahaan, A. Junaidi, F. R. Lumbanraja, and B. Hermanto, “Pengembangan Sistem Informasi Administrasi Asisten Mata Kuliah Praktikum Dan Responsi Di Jurusan Ilmu Komputer Universitas Lampung,” *J. Pepadun*, vol. 3, no. 2, pp. 207–220, 2022, doi: 10.23960/pepadun.v3i2.116.

[3] M. Eric Iryanto Nur Efendi and B. Yulisa Geni, “Sistem Absensi Online Berbasis Web Dengan Fitur Radius Pembatasan Lokasi Absen Untuk Karyawan Outsourcing Departemen Contact Center Pt Pegadaian,” *JATI (Jurnal Mhs. Tek. Inform.*, vol. 8, no. 4, pp. 7428–7435, 2024, doi: 10.36040/jati.v8i4.10206.

[4] U. Aryanti and S. Karmila, “Sistem Informasi Absensi Pegawai Berbasis Web di Kantor Desa Nagreg,” *Intern. (Information Syst. Journal)*, vol. 5, no. 1, pp. 90–101, 2022, doi: 10.32627/internal.v5i1.532.

[5] D. Harmade, A. P. Aulia, I. Ash-shiddiqi, and Q. Adelia, “Information System Design For Open Recruitment of Laboratory Assistants for Information Systems Study Program at UIN Suska Riau Perancangan Sistem Informasi Open Recruitment Asisten Laboratorium Prodi Sistem Informasi UIN Suska Riau,” vol. 1, no. 1, pp. 42–50, 2024.

[6] D. Saxena, M. Ieee, A. K. Singh, and S. M. Ieee, “A Comprehensive Survey on Sustainable Resource Management in Cloud Computing Environments,” 2024.

[7] F. Wadly and M. Muttaqin, “Implementasi Platform As A Service (Paas) Pada Database E-Commerce Berbasis Cloud Computing,” vol. 3, no. April, pp. 45–58, 2023.

[8] P. T. Informatika, U. M. Sukabumi, A. Engine, and C. Run, “Analisis Perbandingan Serverless Computing Pada Google Cloud Platform,” vol. 7, no. 2, pp. 169–187, 2021.

[9] M. H. Isron and R. E. Putra, “Implementasi Virtual Server Berbasis Container Pada Sistem Informasi Geografis Cagar Budaya Mojokerto,” vol. 03, pp. 143–154, 2021.

[10] Y. P. Herawati, P. Sokibi, and L. Norhan, “Rancang Bangun Sistem Penjualan Kue Online Berbasis Web Dengan Pendekatan Feature Driven Development (Fdd) Elluffi Home Bakery,” vol. 10, no. 3, pp. 209–216, 2025.

[11] P. Keputusan, S. Informasi, and M. Dakwah, “Jurnal Pendidikan dan Konseling,” vol. 5, pp. 4314–4320, 2023.

[12] J. M. Arasy, A. L. Prasasti, and A. Novianty, “Pengembangan Backend untuk Efisiensi Pengelolaan dan Penyebaran Informasi di Himpunan Mahasiswa Teknik Komputer,” vol. 2, no. 2, pp. 35–42, 2024.

[13] T. Grance, S. M. Chen, Y. Zhang, and V. C. M. Leung, “Pengelolaan Arsip Digital menggunakan Teknologi Cloud Computing,” 2010.
[14] S. Informasi, U. Amikom, and K. Kunci, “Private Cloud Computing Infrastructure As A Service Dengan Owncloud Di Smk Al-Islam Joresan Kabupaten Ponorogo,” vol. 4, no. 1, 2022.
[15] R. Rakhman, N. A. Khalifah, and W. Antoni, “Madani : Jurnal Ilmiah Multidisipline Teknik Kontainerisasi dan Virtualisasi Sebagai Solusi Untuk Meningkatkan Portabilitas dan Skalabilitas Sistem Operasi Ubuntu Madani : Jurnal Ilmiah Multidisipline,” vol. 2, no. 8, pp. 81–86, 2024.
[16] D. R. Fauzi, “Implementasi Arsitektur Serverless Aws Pada Aplikasi Sistem Absensi Ksu Postra,” Universitas Komputer Indonesia, 2024. [Online]. Available: https://elibrary.unikom.ac.id/id/eprint/10778/
[17] D. T. Komputer, F. Teknologi, E. Dan, and I. Cerdas, *(Cloud Computing , Mobile Development , dan Machine Learning)*. 2024.
[18] F. A. Slamet, *Model Penelitian Pengembangan (R N D)*. Malang: Institut Agama Islam Sunan Kalijogo Malang, 2022.