PERANCANGAN ARSITEKTUR SISTEM SMART CAMPUS BERBASIS CLOUD DI KAMPUS XYZ

Andre Yonathan Sukhoco[1]
Ferdinand Lanvino[2]

[1,2]Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI Bandung

[1]andre@likmi.ac.id
[2]ferdinand@likmi.ac.id

ABSTRAK

Kampus XYZ sebagai salah satu institusi pendidikan tinggi swasta telah menggunakan teknologi informasi untuk mendukung penyampaian pelayanan akademiknya. Infrastruktur TI yang menjadi landasan sistem rupanya telah digunakan sejak lama dan banyak pembaruan yang kurang efektif ketika diintegrasikan, sehingga meningkatkan risiko ketidakstabilan sistem. Pengoperasian sistem pada umumnya pun masih membutuhkan intervensi secara manual. Namun di sisi lain, kampus XYZ memiliki harapan untuk mewujudkan konsep *smart campus* dengan tetap mengoptimalkan biaya operasional sistem.

Salah satu cara untuk mencapai harapan tersebut adalah dengan mengadopsi *cloud computing*. Artikel ini memaparkan perencanaan dan perancangan arsitektur sistem baru bagi kampus XYZ dengan landasan *cloud*. Arsitektur sistem dioptimalkan dari sisi teknis, biaya, hingga kebutuhan-kebutuhan kampus XYZ untuk mencapai konsep *smart campus*. Orientasi perancangan arsitektur di *cloud* yang digunakan dalam penelitian ini adalah Google Cloud Platform (GCP) sebagai salah satu penyedia jasa layanan *cloud* yang memiliki jangkauan secara global dan beragam akan layanan-layanan yang disediakan bagi organisasi.

Kata kunci: *cloud computing*, Google Cloud Platform, *smart campus*, *system architecture*

1. PENDAHULUAN

Perguruan tinggi memiliki kewajiban untuk mewujudkan Tridharma Perguruan Tinggi, yaitu mendidik, meneliti, dan memberikan pengabdian kepada masyarakat. Untuk mendukung hal tersebut, perguruan tinggi memanfaatkan teknologi informasi dan membangun sistem informasi di atasnya. Kampus XYZ sebagai salah satu institusi pendidikan tinggi swasta telah menerapkan teknologi informasi sejak lama. Namun hingga saat ini, penggunaan infrastruktur TI tersebut diikuti dengan pembaruan yang kurang efektif ketika diintegrasikan akibat ketergantungan terhadap basis sistem utama, sehingga meningkatkan risiko ketidakstabilan sistem. Pengoperasian sistem yang ada pada umumnya masih membutuhkan intervensi secara manual, terutama pada hal-hal yang bersifat administratif dan rutin.

Di sisi lain, kampus XYZ memiliki target untuk memperbarui sistem secara menyeluruh dengan mewujudkan konsep *smart campus*, yaitu suatu tahapan lanjut dari sistem informasi untuk pendidikan. Biaya operasional sistem yang optimal juga menjadi harapan dari pihak kampus untuk membangun sistem *smart campus* tersebut.

Dengan kondisi sistem saat ini, perlu mencari langkah-langkah yang tepat untuk mewujudkan harapan dari pihak kampus XYZ tersebut. Salah satu cara yang bisa dilakukan adalah mengadopsi *cloud computing*. Berdasarkan statistik di tahun 2021, sudah semakin banyak organisasi yang memusatkan beban kerjanya di layanan *cloud* dan disebut sudah

menjadi hal *mainstream*. Dukungan layanan *cloud computing* dimanfaatkan untuk meringankan beban kerja operasional juga melakukan efisiensi biaya. Namun di sisi lain, salah satu tantangan besar bagi organisasi untuk mengadopsi *cloud* adalah memahami dependensi aplikasi-aplikasinya [1]. Maka dari itu, perlu disusun sebuah perencanaan yang matang dan rancangan arsitektur baru yang mengakomodasi berbagai kebutuhan yang ada.

2. LANDASAN TEORI

2.1. Cloud Computing

Cloud computing menurut National Institute of Standards and Technology (NIST) adalah sebuah model yang memungkinkan sebuah sumber daya komputasi dapat diakses di mana pun sesuai permintaan (*on-demand*) [2].

Karakteristik penting yang harus ada dalam sebuah layanan *cloud computing* adalah:

1. *On-demand self-service*
   Konsumen dapat mengatur sendiri kapabilitas komputasi sesuai kebutuhan tanpa harus berinteraksi dengan penyedia layanan.
2. *Broad network access*
   Untuk mengatur kapabilitas komputasi pada *cloud*, konsumen harus dapat mengaksesnya melalui berbagai platform yang ada seperti laptop, tablet, ponsel.
3. *Resource pooling*
   Sumber daya komputasi dikumpulkan oleh penyedia layanan yang bertujuan untuk melayani banyak konsumen dengan konsep *multi-tenant*, di mana tiap konsumennya memiliki sumber daya komputasi fisik dan *virtual* dengan kebutuhan yang berbeda-beda.
4. *Rapid elasticity*
   Kapabilitas komputasi dapat diadakan maupun dilepas secara fleksibel, dan pada beberapa kasus, kapabilitas komputasinya dapat disesuaikan secara terukur sesuai dengan permintaan.
5. *Measured service*
   Penggunaan sumber daya komputasi dapat dipantau dan dikendalikan; memberikan transparansi terhadap konsumen maupun penyedia layanan terhadap sumber daya yang digunakan.

Menurut NIST terdapat tiga model layanan *cloud computing* yaitu [2]:

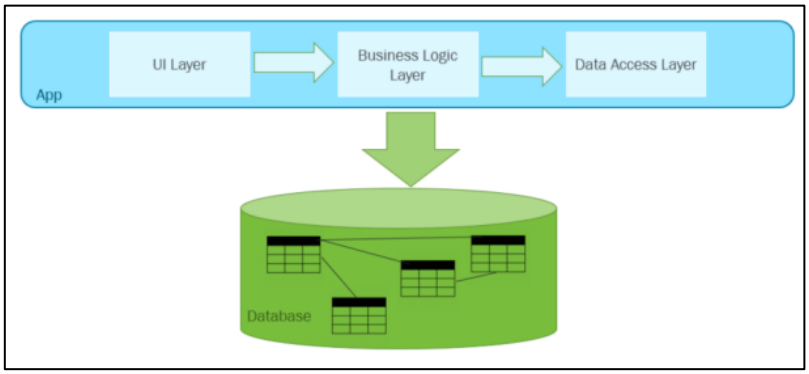
1. *Software as a Service* (SaaS)
   SaaS menyediakan suatu perangkat lunak sebagai layanan *cloud*-nya, di mana pengguna hanya perlu memakai perangkat lunak tersebut dan tidak perlu dipusingkan dengan pengelolaan infrastruktur *cloud*. Layanan SaaS dapat sepenuhnya gratis ataupun berbayar. Pengguna memiliki akses terbatas terhadap fitur-fitur yang diberikan oleh penyedia layanannya saja. Contoh dari SaaS adalah Gmail, Google Docs, Google Drive.
2. *Platform as a Service* (PaaS)
   PaaS menyediakan sebuah platform yang penggunanya dapat melakukan *deployment* ataupun menjalankan aplikasi dengan bahasa pemrograman, layanan, *library*, dan *tools* yang didukung oleh penyedia layanannya. PaaS hanya memberikan kendali yang hanya terbatas pada aplikasi yang di-*deploy* berikut dengan konfigurasinya saja. Contoh dari PaaS adalah App Engine pada GCP.
3. *Infrastructure as a Service* (IaaS)
   Pada IaaS, penyedia layanan memberikan kebebasan pengguna untuk menentukan sendiri komponen *server*, prosesor, penyimpanan, jaringan dan sumber daya komputasi *cloud* lainnya sesuai kebutuhan. IaaS tidak memberikan kontrol penuh

terhadap infrastruktur fisik, namun menawarkan kendali penuh terhadap sistem operasi, penyimpanan, aplikasi yang di-*deploy*, dan beberapa pengaturan jaringan yang berjalan di atasnya. Contoh dari IaaS adalah Compute Engine pada GCP.

2.2. Microservice vs. Monolith

Microservice adalah sebuah aplikasi yang dipecah menjadi beberapa komponen kecil yang dapat berdiri sendiri membentuk satu kesatuan sistem [3]. Arsitektur *microservice* sendiri dapat dipandang sebagai sebuah pendekatan untuk mengembangkan sebuah aplikasi dalam bentuk beberapa komponen atau layanan, yang masing-masing berjalan dalam prosesnya sendiri. Masing-masing komponen tersebut dapat disesuaikan secara terukur, di-*deploy,* maupun diuji secara independen.

Aplikasi atau sistem informasi pada umumnya dikembangkan dengan menggunakan pendekatan arsitektur *monolithic*. Berbeda dengan *microservice*, pada arsitektur *monolithic* semua komponen berada dalam satu *deployable unit* yang saling bergantung satu dengan yang lainnya [4]. Dalam aplikasi *monolith*, UI *layer*, *business logic layer* dan *data layer* ada dalam satu *source code* dan terhubung pada satu *database*. Segala perubahan sekecil apapun mengharuskan aplikasi *monolith* untuk di-*build* dan di-*deploy* ulang. Aplikasi *monolith* tersebut umumnya dikembangkan dengan menggunakan metode *object-oriented*; di mana biasanya berumur panjang dan sangat penting terhadap keberlangsungan proses bisnis.



Gambar 1
Sistem Aplikasi *Monolith*

Seiring dengan berjalannya waktu, kondisi bisnis terus berubah. Hal tersebut memaksa sistem informasi untuk terus beradaptasi, untuk itu sistem informasi harus bersifat dinamis sehingga dapat mengakomodasi perubahan dengan cepat. Pada sistem aplikasi *monolith*, perubahan-perubahan tersebut masih dapat direalisasikan, namun semakin kompleks aplikasinya semakin besar juga waktu, tenaga dan uang yang dikeluarkan. Selain itu, arsitektur *monolith* memerlukan komitmen dalam jangka waktu panjang untuk menggunakan *technology stack* yang sama dengan risiko pemberhentian dukungan. Untuk dapat mengakomodasi sistem informasi yang dinamis, arsitektur *microservice* lebih unggul dan menguntungkan perusahaan dibandingkan arsitektur *monolith* dalam jangka waktu yang panjang.

2.3. Smart Campus

Smart campus adalah sistem informasi pendidikan dengan tingkat yang lebih tinggi, merupakan evolusi dari *smart campus* yang dibangun dan dikembangkan berdasarkan *digital campus* [5]. *Smart campus* merupakan tren yang muncul karena terinspirasi dari konsep *smart city* yang memanfaatkan infrastruktur secara efisien. Tabel 1 menunjukkan perbandingan konsep antara *digital campus* dengan *smart campus* menurut [5].

Tabel 1
Perbandingan konsep *digital campus* dengan *smart campus* [5]

	Digital campus	***Smart campus***
Technical enviroment	Local Area Network Internet	IOT *Cloud computing* *Wireless network* *Mobile terminal* RFID
Application	*Digital teaching resources* *Distance education* *Digital library* *Network administrator*	*Smart sensory system* *Interopability* *Control capabilities*
System management	*Isolated system*	*System sharing* *Intelligent system* *Push notification system*

2.4. Google Cloud Architecture Framework

Google Cloud Architecture Framework merupakan kumpulan rekomendasi rancangan dan *best-practices* yang dapat menjadi panduan bagi arsitek, pengembang, administrator dan praktisi *cloud* lainnya dalam hal merancang dan menjalankan sebuah topologi *cloud* yang aman, efisien, berdaya tahan, berkinerja tinggi dan hemat biaya [6]. *Architecture framework* ini disusun untuk membantu perancangan *deployment* layanan-layanan Google Cloud agar cocok dengan kebutuhan bisnis perusahaan. Google Cloud Architecture Framework diatur dalam beberapa kategori berikut yaitu:

1. *System design*
   System design memberikan petunjuk bagaimana menentukan arsitektur, komponen, modul, antarmuka, dan data pada platform *cloud* untuk memenuhi rencana kebutuhan sistem. Pembahasan pada *system design* mencangkup pemilihan *region* yang tepat, pengelolaan sumber daya *cloud*, pemilihan dan pengelolaan platform komputasi (*compute platform*), perancangan arsitektur jaringan, penentuan dan implementasi strategi penyimpanan data, optimisasi *database*, analisis data, dan implementasi *machine learning*.
2. *Operational excellence*
   Operational excellence memberikan petunjuk bagaimana menjalankan, mengelola, dan memantau sistem yang memberikan nilai bisnis secara efisien; baik dalam bentuk rekomendasi implementasi ataupun produk-produk Google Cloud yang dapat membantu untuk mencapai keunggulan operasional (*operational excellence*). Pembahasan pada *operational excellence* meliputi otomatisasi *deployment*, pemasangan sistem *monitoring*, *alerting*, dan *logging*, pembentukan tim *support* khusus hingga pengelolaan kapasitas dan kuota sumber daya *cloud*.
3. *Security*, *privacy*, dan *compliance*
   Kategori ini memberikan petunjuk bagaimana memaksimalkan keamanan data dan beban kerja pada *cloud*, membuat rancangan privasi, dan menyesuaikan dengan peraturan dan regulasi yang berlaku. Pembahasan pada kategori ini mencakup pengelolaan risiko, pengelolaan aset, pengelolaan identitas dan hak akses, implementasi keamanan pada sumber daya komputasi dan *container*, keamanan

jaringan, implementasi keamanan data, keamanan aplikasi, implementasi *privacy requirement*.

4. *Reliability*
   Reliability memberikan petunjuk bagaimana cara merancang dan mengoperasikan layanan yang andal pada platform Google Cloud. Untuk dapat berjalan secara andal, arsitektur pada platform *cloud* perlu mengikuti hal-hal berikut:
   a. *Reliability goals* yang terukur dan realistis.
   b. Pola perancangan sistem yang berfokus pada skalabilitas, *high availability*, *disaster recovery*, dan manajemen perubahan terotomatisasi.
   c. Komponen dalam sistem dapat melakukan *self-heal* jika memungkinkan dan terdapat instrumentasi untuk melakukan observasi (*monitoring, analytics*, *metrics*) yang sudah tertanam dalam sistem,
   d. Memiliki seminimal mungkin prosedur operasional yang menjalankan layanan secara manual dan membebani kognitif operator.
   e. Memiliki prosedur operasional yang dapat membantu mendeteksi dan mengurangi kegagalan sistem.

   Pembahasan pada kategori ini meliputi pemahaman prinsip utama dari *reliability* pada sistem, penentuan *reliability goals*, pembangunan dan implementasi fungsi observasi pada infrastruktur dan aplikasi, perancangan layanan yang *scalable* dan *high availability*, pengunaan *tools* untuk mendukung proses operasional, pembangunan sistem peringatan (*alerts*) yang efisien, dan pembangunan manajemen insiden (*management incident process*) secara kolaboratif.
5. *Cost optimization*
   Cost optimizatiojn memberikan petunjuk untuk melakukan optimalisasi biaya sehingga dapat memaksimalkan nilai bisnis (*business value*) pada platform *cloud*.
6. *Performance optimization*
   Performance optimization memberikan petunjuk untuk melakukan optimalisasi performa pada platform *cloud*. Strategi yang dibahas adalah evaluasi kebutuhan performa sistem dan menerapkan *design pattern* sistem yang bersifat terukur.

3. METODOLOGI

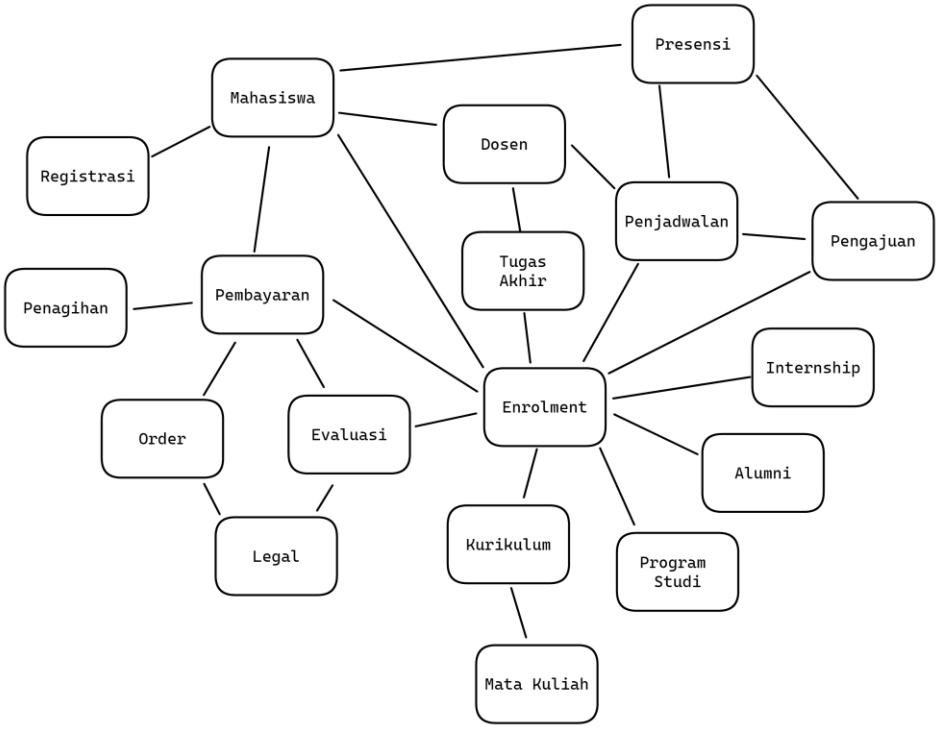
Tahapan perancangan arsitektur sistem yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. Mengumpulkan informasi
   Tahap pertama adalah mengumpulkan informasi dari para pihak yang berhubungan langsung dengan sistem saat ini. Informasi yang dikumpulkan berupa karakteristik individu, pengalaman individu terhadap penggunaan sistem, hambatan yang dialami, dan masukan-masukan untuk meningkatkan pengalaman penggunaan sistem. Selain itu, observasi terhadap sistem saat ini juga dilakukan untuk mendapatkan gambaran tentang proses bisnis apa saja yang didukung oleh sistem.
2. Menganalisis kebutuhan fungsional dan teknis
   Setelah mendapatkan informasi dari para pihak terkait, disusunlah daftar kebutuhan (*requirements*) fungsional secara deskriptif. Dari daftar kebutuhan fungsional tersebut, dipetakanlah kebutuhan-kebutuhan teknis sebelum memilih jenis layanan dari *cloud*.
3. Merancang sistem
   Setelah mendapatkan gambaran besar dari daftar kebutuhan, dibuatlah rancangan sistem baru berdasarkan kelompok-kelompok (*domain*) dan konteks layanan menurut konsep *microservice*.

4. Memilih layanan spesifik berdasarkan rancangan sistem
   Rancangan sistem yang dibuat dan daftar kebutuhan menjadi dasar untuk memilih model layanan *cloud* secara spesifik. Pilihan model layanan *cloud* dalam penelitian ini menggunakan Google Cloud Platform (GCP).
5. Memodelkan arsitektur sistem
   Pilihan layanan *cloud* yang disesuaikan dengan rancangan sistem kemudian dimodelkan menjadi diagram visual arsitektur sistem.
6. Menyusun estimasi biaya berdasarkan pilihan layanan *cloud*
   Daftar layanan *cloud* yang ditetapkan menjadi dasar untuk membuat estimasi biaya yang akan keluar.

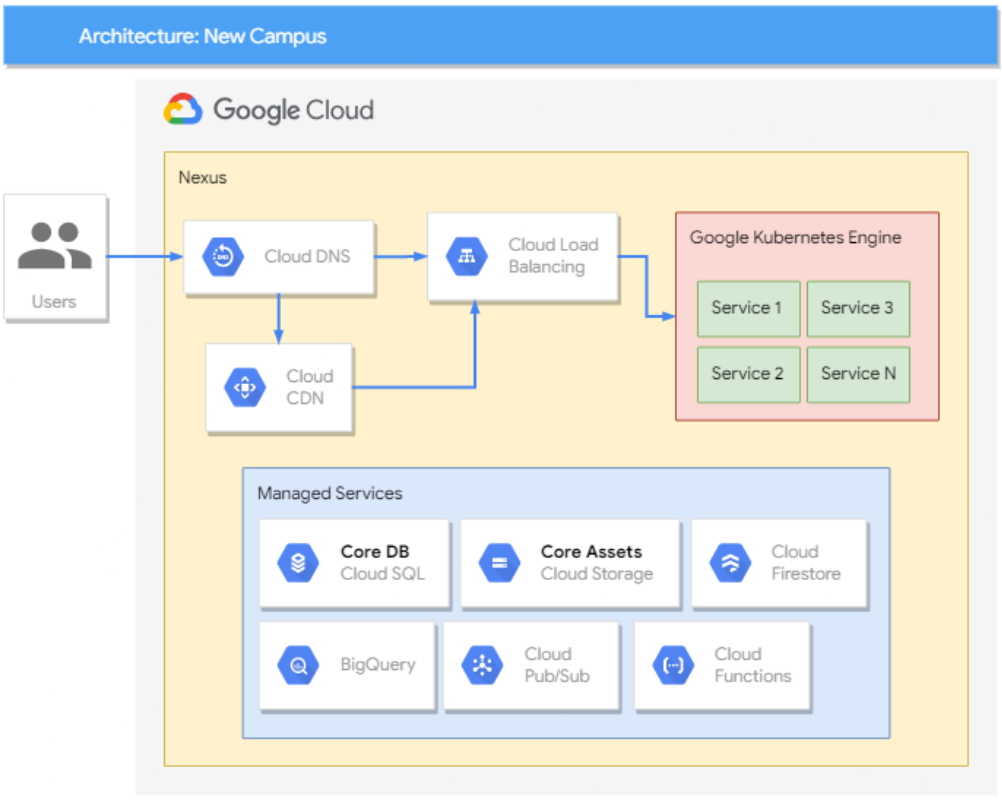
4. RANCANGAN ARSITEKTUR SISTEM BARU

Gambar 2 menunjukkan cakupan layanan-layanan yang akan disediakan oleh sistem yang baru di *cloud* berdasarkan rumusan kebutuhan fungsional dari kampus XYZ.



Gambar 2
Diagram *microservice* dan keterkaitan layanan untuk sistem utama yang baru

Setelah mendapatkan gambaran besar dari diagram *microservice*, disusunlah model arsitektur sistem yang baru sesuai dengan pemilihan jenis layanan yang cocok dari *cloud*, dalam hal ini menggunakan layanan GCP.



Gambar 3
Diagram arsitektur sistem yang baru di *cloud*

Berikut adalah penjelasan dari diagram pada Gambar 3:

1. Semua pengguna akan mengakses sistem melalui domain kampus XYZ yang didukung oleh layanan Cloud DNS. Cloud DNS merupakan layanan pustaka alamat jaringan suatu *domain* yang dapat diakses oleh siapa pun di internet.
2. *Traffic* dari pengguna akan diarahkan menuju layanan Cloud CDN untuk mempercepat akses atau langsung menuju pintu utama sistem melalui alamat IP yang disediakan oleh layanan Cloud Load Balancing (LB). Cloud LB merupakan layanan penyeimbang lalu lintas jaringan otomatis dan cocok bagi sistem yang bersifat *stateless* seperti *microservice*.
3. Selanjutnya pengguna akan diarahkan secara otomatis dan berimbang ke dalam sistem *microservice* yang dikelola oleh layanan Google Kubernetes Engine (GKE). GKE menyediakan panel kontrol dan pemantauan sumber daya sistem *microservice* yang berada di dalamnya.
4. Masing-masing *microservice* di dalam GKE dapat mengakses penyimpanan data dan memanfaatkan fasilitas *cloud* yang lain melalui layanan terkelola (*managed services*) secara independen.

Berdasarkan model layanan *cloud* yang sudah dipilih, disusunlah estimasi biaya operasional sistem yang akan dikeluarkan oleh kampus XYZ setiap bulannya. Dalam menentukan estimasi biaya, penulis menggunakan asumsi sebagai berikut:

1. Beban *network bandwidth*, baik yang masuk (*ingress*) dan keluar (*egress*) masing-masing sebesar 80 GiB disesuaikan dengan temuan.
2. Ukuran aplikasi kurang dari 10 GiB, tetapi syarat minimum ukuran *disk* di GCP adalah 10 GiB.
3. 1 zona/domain diakses kurang lebih hingga 400.000 kali dalam sebulan.

4. Kampus XYZ menghendaki biaya yang optimal, maka dapat menggunakan layanan *commitment use* untuk mendapatkan diskon biaya sumber daya komputasi GCP hingga 3 tahun.
5. Aplikasi ditempatkan di *data center* di wilayah Jakarta.
6. Biaya per bulan dapat sangat bervariasi tergantung pada besaran *network ingress* dan *egress*. Selain itu, estimasi biaya mengikuti skema harga dari GCP.

Tabel 2
Estimasi biaya layanan *cloud*

Layanan	Spesifikasi	Biaya
Google Kubernetes Engine (GKE)	*Region*: Jakarta Komitmen 3 tahun Tipe mesin: e2-medium *Container-optimized OS*	Rp212.676,29
	Persistent Disk *Zonal* SSD PD 10 GiB	Rp31.757,70
Cloud DNS	1 *managed zone* (1 domain) 400.000 kueri	Rp5.173,20
Cloud Load Balancing	*Forwarding rules*: 1 *Network ingress*: 80 GiB	Rp412.419,00
Network Internet Egress	*Source*: Jakarta *Destination*: Asia-Pacific (APAC) Premium Tier 80 GiB	Rp218.424,00
Cloud SQL (opsi penyimpanan data)	Tipe *instance*: db-lightweight-1 Komitmen 3 tahun *Region*: Jakarta SSD 10 GiB Backup 5 GiB	Rp481.283.99
Estimasi biaya per bulan sebelum PPN 11%		Rp1.361.734.18
Estimasi biaya per bulan setelah PPN 11%		Rp1.511.524,94

5. KESIMPULAN

Perancangan arsitektur sistem baru menjadi langkah awal dari rangkaian proses migrasi sistem di kampus XYZ. Perencanaan yang tepat dan berorientasi pada layanan *cloud* yang membuka peluang baru bagi pengembangan *smart campus*, dapat menghasilkan rancangan arsitektur baru yang sesuai dengan kebutuhan institusi. Hasil rancangan sistem dapat menjadi acuan bagi para pengembang di kampus XYZ dalam mengembangkan masing-masing layanan dan memilih teknologi yang tepat sesuai kebutuhan. Bagi institusi, rancangan sistem dapat dijadikan acuan untuk mengalokasikan sumber daya dan menyusun anggaran secara tepat dan terukur.

6. DAFTAR PUSTAKA

[1] Flexera, “State of the Cloud Report:2021,” 2021.
[2] P. M. Mell dan T. Grance, “The NIST definition of cloud computing,” Gaithersburg, MD, 2011. doi: 10.6028/NIST.SP.800-145.
[3] R. V. O’Connor, P. Elger, dan P. M. Clarke, “Continuous software engineering—A microservices architecture perspective,” *J. Softw. Evol. Process*, vol. 29, no. 11, hal. 1–12, 2017, doi: 10.1002/smr.1866.
[4] A. Wu, “Taking the Cloud-Native Approach with Microservices,” 2017. https://cloud.google.com/files/Cloud-native-approach-with-microservices.pdf (diakses Jan 17, 2022).

[5] X. Nie, "Constructing Smart Campus Based on the Cloud Computing Platform and the Internet of Things," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, 2013, vol. 1, no. ICCSEE, hal. 1576–1578, doi: 10.2991/iccsee.2013.395.
[6] Google, "Google Cloud Architecture Framework." https://cloud.google.com/architecture/framework (diakses Jan 21, 2022).