IMPLEMENTASI ARSITEKTUR MICROSERVICE PADA APLIKASI WEB PENGAJARAN AGAMA ISLAM HOME PESANTREN

Tugas Akhir

Diajukan untuk Memenuhi Salah Satu Persyaratan Mencapai Derajat Sarjana Komputer

Dibuat Oleh:

YURI CHANDRA TRI PUTRA

160709045

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA
2020

HALAMAN PENGESAHAN

Tugas Akhir Berjudul

IMPLEMENTASI ARSITEKTUR MICROSERVICE PADA APLIKASI WEB PENGAJARAN AGAMA ISLAM HOME PESANTREN

yang disusun oleh

YURI CHANDRA TRI PUTRA

160709045

dinyatakan telah memenuhi syarat pada tanggal 14 Juli 2020

		Keterangan
Dosen Pembimbing 1	: Thomas Adi Purnomo Sidhi, ST., MT.	Telah menyetujui
Dosen Pembimbing 2	: Joseph Eric Samodra, S.Kom, MIT.	Telah menyetujui
Tim Penguji		
Penguji 1	: Thomas Adi Purnomo Sidhi, ST., MT.	Telah menyetujui
Penguji 2	: Yulius Harjoseputro, ST., MT.	Telah menyetujui
Penguji 3	: Eddy Julianto, ST., MT.	Telah menyetujui

Yogyakarta, 14 Juli 2020

Universitas Atma Jaya Yogyakarta

Fakultas Teknologi Industri

Dekan

ttd

Dr. A. Teguh Siswantoro, M.Sc

PERNYATAAN ORISINALITAS & PUBLIKASI ILMIAH

Saya yang bertanda tangan di bawah ini:

Nama Lengkap	: **Yuri Chandra Tri Putra**
NPM	: **160709045**
Program Studi	: Informatika
Fakultas	: Teknologi Industri
Judul Penelitian	: **Implementasi Arsitektur Microservice Pada Aplikasi Web Pengajaran Agama Islam Home Pesantren**

Menyatakan dengan ini:

1. Tugas Akhir ini adalah benar tidak merupakan salinan sebagian atau keseluruhan dari karya penelitian lain.
2. Memberikan kepada Universitas Atma Jaya Yogyakarta atas penelitian ini, berupa Hak untuk menyimpan, mengelola, mendistribusikan, dan menampilkan hasil penelitian selama tetap mencantumkan nama penulis.
3. Bersedia menanggung secara pribadi segala bentuk tuntutan hukum atas pelanggaran Hak Cipta dalam pembuatan Tugas Akhir ini.

Demikianlah pernyataan ini dibuat dan dapat dipergunakan sebagaimana mestinya.

Jakarta, 14 Juli 2020

Yang menyatakan,

Yuri Chandra Tri Putra

160709045

HALAMAN PERSEMBAHAN

"Tugas akhir ini saya persembahkan kepada keluarga, teman, kerabat beserta dosen Universitas Atma Jaya Yogyakarta yang telah menyemangati dan memberikan ilmu bagi saya dalam empat tahun melewati proses perkuliahan."

KATA PENGANTAR

Puji dan syukur penulis haturkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya penulis dapat menyelesaikan pembuatan tugas akhir "Implementasi Arsitektur Microservice Pada Aplikasi Web Pengajaran Agama Islam Home Pesantren" ini dengan baik.

Penulisan tugas akhir ini bertujuan untuk memenuhi salah satu syarat untuk mencapai derajat Sarjana Komputer dari Program Studi Informatika, Fakultas Teknologi Industri di Universitas Atma Jaya Yogyakarta.

Penulis menyadari bahwa dalam pembuatan tugas akhir ini penulis telah mendapatkan bantuan, bimbingan, dan dorongan dari banyak pihak. Untuk itu, pada kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa atas berkat dan rahmat-Nya penulis dapat memiliki semangat untuk dapat menyelesaikan tugas akhir ini.
2. Bapak Dr. A. Teguh Siswantoro, selaku Dekan Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta.
3. Bapak Thomas Adi Purnomo Sidhi S.T., M.T. selaku dosen pembimbing I yang telah membimbing dan memberikan masukan serta motivasi kepada penulis untuk menyelesaikan tugas akhir ini.
4. Bapak Joseph Eric S, S.Kom., MIT. selaku dosen pembimbing II yang telah membimbing dan memberikan masukan serta motivasi kepada penulis untuk menyelesaikan tugas akhir ini.

Demikian laporan tugas akhir ini dibuat, dan penulis mengucapkan terima kasih kepada semua pihak. Semoga laporan ini dapat bermanfaat bagi pembaca.

Jakarta, 14 Juli 2020

Yuri Chandra Tri Putra

160709045

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

INTISARI

IMPLEMENTASI ARSITEKTUR MICROSERVICE PADA APLIKASI WEB PENGAJARAN AGAMA ISLAM HOME PESANTREN

Intisari

Yuri Chandra Tri Putra
160709045

Arsitektur perangkat lunak merupakan sebuah hal penentu dari keberlangsungan sebuah hidup perangkat lunak. Pembangunan aplikasi dengan arsitektur yang tepat dapat membantu *delivery* produk yang cepat, tanggap, baik dan berkualitas.

Aplikasi web pengajaran agama Islam Home Pesantren adalah aplikasi berbasis web untuk mendapatkan konten pengajaran agama Islam. Pengembangan aplikasi Home Pesantren akan dikembangkan dengan arsitektur *microservice*. Penerapan arsitektur ini akan didukung dua jenis bahasa pada *server side*, yaitu Golang dan PHP. Penerapan arsitektur *microservice* pada aplikasi web Home Pesantren juga akan menerapkan satu basis data untuk masing-masing *microservice* dengan tujuan untuk enkapsulasi dan meningkatkan *single responsibility* dari setiap microservice.

Hasil dari penelitian ini diharapkan dapat membuat aplikasi web Home Pesantren mendapatkan manfaat-manfaat dari penerapan arsitektur *microservice* seperti kemudahan skalabilitas, modularitas setiap layanan serta meningkatkan *reliability* aplikasi.

Kata Kunci: microservice, arsitektur, aplikasi web, docker, skalabilitas

Dosen Pembimbing I : Thomas Adi Purnomo Sidhi S.T., M.T.
Dosen Pembimbing II : Joseph Eric Samodra, S. Kom., MIT
Jadwal Sidang Tugas Akhir : Selasa, 14 Juli 2020

BAB I. PENDAHULUAN

1.1. Latar Belakang

Codebase dapat terus bertumbuh dan menjadi besar ketika penambahan fitur terus terjadi secara terus-menerus [1]. Pertumbuhan *codebase* yang terjadi secara terus-menerus, dapat membuat pengembang menghadapi kesulitan untuk mengetahui dimanakah letak perubahan akan dilakukan ketika *codebase* sudah bertumbuh besar. *Codebase* yang besar dapat terjadi ketika pengembangan perangkat lunak dilakukan dengan menggunakan arsitektur *monolith*. Arsitektur *monolith* membuat semua layanan atau layanan berada pada satu *codebase* yang sama [2].

Home Pesantren (Hompes) merupakan aplikasi pengajaran agama Islam berbasis web yang menggunakan arsitektur *monolith* pada tahap pengembangan. Seiring dengan pengembangan Hompes, *codebase* yang ada pada Hompes juga bertumbuh. *Codebase* yang bertumbuh dengan pesat melahirkan *dependency* antar modul yang membuat pengembangan untuk fitur berikutnya menjadi semakin sulit. Kesulitan lain yang dihadapi saat pengembangan Home Pesantren (Hompes) adalah waktu yang cukup lama untuk melakukan pengembangan. Waktu yang cukup lama ini mencakup proses pemahaman *codebase* yang sudah ada sehingga produktivitas dalam *delivery* produk akhir menjadi berkurang. Selain itu, aplikasi Home Pesantren yang dikembangkan dengan arsitektur *monolith* akan sulit dan tidak efisien dalam hal *resource* untuk *scale* jika pertumbuhan pengguna menjadi tinggi, ini disebabkan *scale* pada arsitektur *monolith* mengharuskan keseluruhan sistem harus di*scale* secara bersamaan, padahal hanya beberapa komponen atau bagian saja yang memerlukan *scale.*

Seiring dengan berkembangnya arsitektur perangkat lunak, kemunculan arsitektur *microservice* menjadi sebuah tren yang digunakan oleh pengembang dalam beberapa tahun terakhir. *Microservice* memberikan keleluasaan bagi pengembang untuk mengembangkan perangkat lunak dalam waktu yang cepat. Berkembangnya arsitektur *microservice* juga didukung oleh kurang handalnya arsitektur *monolith* dalam menangani *system failure*. Karena dalam satu aplikasi

memiliki satu *codebase* yang sama, merupakan hal yang pasti bahwa arsitektur *monolith* juga menjadi *single point of failure* yang akan terjadi jika salah satu layanan di *codebase* yang sama mengalami malfungsi atau error[2]. Sementara itu, arsitektur *microservice* memungkinkan sebuah sistem dibangun oleh *codebase* yang berbeda-beda atau layanan yang terpisah sehingga dapat menawarkan sistem yang resilien dan tetap menjamin *availability* ketika salah satu layanan mengalami malfungsi[3]. Keunggulan arsitektur *microservice* lainnya adalah perubahan yang terjadi pada satu layanan dalam frekuensi yang tinggi akan tetap membuat sistem secara keseluruhan bekerja dengan baik tanpa adanya perubahan yang signifikan[4]. Keunggulan arsitektur *microservice* juga ada pada poin *scale*. Dengan menggunakan arsitektur *microservice*, *scale* dapat dilakukan pada sistem atau komponen yang membutuhkan *scale* saja, tanpa harus *scale* pada keseluruhan sistem.

Di Indonesia, jumlah penduduk yang menganut agama Islam berada di angka 209,12 juta jiwa atau sekitar 87, 17% dari total 239,89 juta jiwa pada tahun 2010. Angka tersebut diperkirakan akan meningkat pada tahun 2020 menjadi 263,92 juta jiwa[5]. Dengan jumlah penduduk yang beragama Islam cukup tinggi, kualitas pengajaran Agama Islam menjadi salah satu hal yang harus diperhatikan. Pengajaran Agama Islam saat ini masih dianggap keliru karena agama dimaknai sebagai ritual dan bacaan-bacaan semata tanpa memperhatikan hakikat pengajaran agama yang seutuhnya yaitu berkaitan dengan tingkah laku manusia di dalam hidup dengan mendasarkan iman kepada Tuhan dan pertanggungjawaban di kemudian hari[6]. Pernyataan tersebut didukung dengan proses pembelajaran agama Islam yang masih memiliki paradigma bahwa pengajaran agama Islam hanya bersumber dari sekolah/madrasah ataupun melalui guru yang memiliki pedoman dasar berupa kurikulum. Paradigma inilah yang harus diselesaikan dengan penggunaan teknologi yang mendukung adanya penggunaan sumber daya berupa teknologi yang secara maksimal uintuk mendukung pengajaran agama Islam.

Penerapan arsitektur *microservice* diharapkan dapat meningkatkan ketersediaan, resiliensi, dan menjamin malfungsi yang terjadi pada satu komponen dalam sistem tidak dapat mempengaruhi bagian sistem lainnya. Selain itu penerapan

arsitektur *microservice* diharapkan dapat memaksimalkan penggunaan sumber daya pada perangkat lunak Home Pesantren (Hompes).

1.2. Rumusan Masalah

Berdasarkan latar belakang masalah diatas, permasalahan yang muncul diantaranya yaitu:

a) Bagaimana melakukan pemisahan arsitektur *monolith* dan mengubahnya menjadi arsitektur *microservice*?

b) Bagaimana setiap layanan akan berkomunikasi untuk menjadi sebuah kesatuan?

1.3. Batasan Masalah

Agar pembahasan menjadi lebih terarah, maka diperlukan batasan masalah, batasan-batasan masalah yaitu:

a) Implementasi arsitektur *microservice* hanya akan dilakukan pada aplikasi Home Pesantren (Hompes).

b) Implementasi arsitektur *microservice* hanya akan diterapkan untuk memecah atau memisahkan *codebase* menjadi beberapa layanan terpisah. Layanan yang terpisah mencakup satu *database* untuk tiap layanan.

1.4. Tujuan Penelitian

Dari latar belakang, rumusan masalah serta batasan masalah yang disebutkan sebelumnya, maka tujuan penelitian ini adalah:

a) Melakukan pemisahan arsitektur *monolith* dan mengubahnya menjadi arsitektur *microservice*

b) Merancang sistem komunikasi antar layanan *microservice* agar dapat menjadi sebuah kesatuan

1.5. Metode Penelitian

Metode penelitian yang diterapkan dalam melaksanakan penelitian ini adalah sebagai berikut:

1.5.1. Kajian Pustaka

Pada tahap ini, penulis mencari bahan pustaka untuk dijadikan referensi dalam pengembangan aplikasi. Referensi yang dicari akan secara khusus membahas pengembangan aplikasi dengan arsitektur *microservice*. Pada tahapan ini juga penulis berharap mendapatkan gambaran tentang arsitektur *microservice* serta cara untuk memisahkan aplikasi dengan arsitektur *monolith* menjadi arsitektur *microservice*.

1.5.2. Analisis

Pada tahap ini, penulis akan melakukan analisa terhadap sistem yang sudah berjalan. Pada tahap ini pula, penulis akan menentukan *boundary context* yang digunakan untuk memecah layanan menjadi beberapa layanan terpisah. Hasil dari analisis ini adalah dokumen Spesifikasi Perangkat Lunak (SKPL).

1.5.3. Perancangan

Pada tahap ini, penulis akan melakukan perancangan sistem dengan arsitektur *microservice*. Pada tahap ini, perancangan akan dilakukan dengan merancang basis data untuk tiap layanan terpisah dan menentukan kebutuhan – kebutuhan tiap layanan dalam proses pengembangan. Tahap ini menjadi bagian penting dalam pengembangan sistem. Hasil dari perancangan adalah dokumen Deskripsi Perancangan Perangkat Lunak (DPPL).

1.5.4. Implementasi

Pada tahap ini, penulis akan memulai menerapkan arsitektur *microservice* pada *backend* aplikasi Home Pesantren (Hompes). Penerapan akan dimulai dari satu layanan yang tidak memiliki ketergantungan satu sama lain sehingga tahap implementasi dimulai dari layanan yang paling mudah.

1.5.5. Pengujian

Pada tahap terakhir ini, pengujian akan dilakukan pada tiap layanan yang dibuat untuk memastikan bahwa tiap layanan memiliki *availability* serta fungsionalitas yang baik untuk membentuk satu kesatuan sistem. Pengujian yang akan dilakukan berupa *unit testing* di setiap layanan.

1.5.6. Pelaporan

Pada metode ini dilakukan untuk pembuatan laporan tugas akhir dimana tujuan dari pelaporan ini adalah untuk mengetahui perkembangan atau keterangan dalam pembuatan program yang nantinya pelaporan ini dapat digunakan untuk bahan evaluasi dalam pembuatan program selanjutnya.

1.6. Sistematika Penulisan

Untuk mempermudah memahami laporan ini, maka penulisan laporan dibagi dalam beberapa bagian atau bab dengan pembagian sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang masalah, rumusan masalah, Batasan masalah, tujuan penelitian, metode yang digunakan selama penelitian, serta sistematika penulisan yang digunakan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi penelitian-penelitian terdahulu yang menyangkut dengan penelitian yang dilakukan. Terdapat juga tabel perbandingan antara penelitian yang dilakukan dengan penelitian-penelitian terdahulu.

BAB III LANDASAN TEORI

Bab ini membahas mengenai uraian dasar teori yang akan digunakan dalam melakukan pengembangan dan pembuatan program meliputi referensi teknik pengembagan web dan referensi mengenai pengelolaan barang.

BAB IV ANALISIS DAN PERANCANGAN

Bab ini berisi penjelasan mengenai tahap-tahap perancangan perangkat lunak yang akan dibuat, serta desain sistem yang akan digunakan dalam proses pengembangan.

BAB V IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini berisi tentang implementasi antarmuka dari system yang dikembangkan, pengujian fungsionalitas perangkat lunak, serta hasil yang dilakukan terhadap pengguna.

BAB II. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Tujuan pengembangan aplikasi menggunakan arsitektur *microservice* adalah membantu *engineer* untuk menghantarkan sebuah produk dengan cepat, aman dan berkualitas tinggi [7]. Dengan memisahkan layanan menjadi satuan yang kecil membuat pengembangan tiap layanan dapat menjadi lebih cepat dan tanpa memiliki dampak yang besar bagi sistem secara keseluruhan. Penerapan arsitektur *microservice* pada platform Medium dimulai akibat *bottleneck* yang terjadi pada sistem yang sudah berjalan. Beberapa tugas yang dijalankan di *platform* Medium memiliki beban komputasi serta ketergantungan I/O yang tinggi yang dianggap tidak sesuai dengan teknologi yang digunakan saat itu, yaitu Node.js. Hal berikutnya yang menjadi alasan mengapa Medium mengubah arsitektur menjadi *microservice* adalah membuat pengembangan aplikasi menjadi lama akibat banyaknya *engineer* yang mengembangkan aplikasi dalam satu *codebase* yang sama sehingga menyebabkan *coupling* yang tinggi. Arsitektur *monolith* juga tidak memberikan ruang bagi *engineer* di Medium untuk melakukan perubahan yang besar terhadap *codebase*. Karena *codebase* di-*deploy* menjadi satu kesatuan, ketika terjadi perubahan yang besar dan terjadi malfungsi pada perubahan yang dilakukan, maka dapat dipastikan sistem akan mengalami malfungsi secara keseluruhan. Arsitektur *microservice* memberikan keleluasaan bagi *engineer* di Medium untuk mengembangkan layanan yang tidak terikat dengan layanan lain. Penyebab lain dari perubahan arsitektur adalah kesulitan untuk melakukan *scale* terhadap sistem. Karena *codebase* menjadi satu, sulit untuk melakukan isolasi terhadap tugas atau pekerjaan kepada *resource* tertentu. Sehingga penggunaan *resource* menjadi tidak hemat karena tugas yang ringan juga menggunakan satu *resource* yang sama. Hal terakhir adalah keleluasaan yang diberikan arsitektur *microservice* untuk memilih teknologi tertentu dalam memproses tugas tertentu. Hasil penelitian dan penerapan yang dilakukan di Medium menghasilkan peningkatan produktifitas pengembang serta mendukung penggunaan teknologi yang berbeda secara lebih aman.

Penelitian yang melibatkan arsitektur *microservice* juga dilakukan pada *refactoring* Sistem Manajemen Anggota AISINDO (Asosiasi Profesi Sistem Informasi Indonesia). Penelitian tersebut bertujuan untuk meningkatkan resiliensi untuk kebutuhan kualitas perangkat lunak [3]. *Refactoring* dilakukan dengan mengubah arsitektur perangkat lunak serta penggunaan *tools* seperti Docker untuk pembangunan arsitektur *microservice*. Hasil dari penelitian ini dapat diamati melalui peningkatan aspek resiliensi sistem yang memampukan ketangguhan sistem ketika salah satu layanan mengalami gangguan. Penelitian ini mendukung penggunaan Eureka dan Zuul yang masing – masing merupakan *service discovery* dan *api gateway* yang memiliki fungsi untuk mencari layanan yang tersedia dan menjadi layanan untuk memetakan *request* kepada salah satu layanan.

Penelitian berikutnya adalah penerapan arsitektur *microservice* untuk membangun platform *open source* SmartCity [8]. Tujuan penelitian tersebut adalah membuktikan bahwa dampak yang diberikan dengan pembangunan arsitektur *microservice* dapat berdampak pada *scalability* dan *evolveabilty* dari sistem yang dikembangkan. Hasil dari penelitian ini adalah memang benar bahwa arsitektur *microservice* dapat meningkatkan *scalability* dalam sebuah sistem atau perangkat lunak. Ini dibuktikan dengan *scaling* yang dilakukan dapat meningkatkan rata – rata *response time* dari perangkat lunak yang sebelumnya berkisar 1720 *milisecond* menjadi 320 *milisecond.*

Penelitian terakhir yang melibatkan penerapan arsitektur microservice adalah pengembangan *Student Information System (RosarioSIS)* [9], yang merupakan *open source legacy* yang sebelumnya menerapkan arsitektur *monolith*. Tujuan utama dari penerapan arsitektur ini adalah pemanfaatan *single responsibility* dan proses *independent deployment*. Penerapan arsitektur microservice pada penelitian ini juga didukung dengan penggunaan Amazon Web Service (AWS) sebagai *cloud environment* untuk kebutuhan *deployment.* Hasil dari penelitian ini adalah arsitektur *microservice* mendukung penerapan *single responsibility* dan *independent deployment* dengan hasil penerapan tiap layanan di-*deploy* pada satu instance pada platform *Amazon Web Service*.

Tabel 2.1 Tabel Pembanding

No	Unsur Pembanding	**Xiao Ma**[7]	**H. Suryotrisongko**[3]	**A. D. M. Del Esposte**[8]	**S. Habibullah**[9]	**Y. C. Tri Putra (2020) ***
1	Konten	*Microservice Architecture at Medium*	Arsitektur Microservice untuk Resiliensi Sistem Informasi	*InterSCity: A scalable microservice-based open source platform for smart cities*	*Reviving Legacy Enterprise Systems with Micro service-Based Architecture with in Cloud Environments*	Implementasi Arsitektur Microservice pada Aplikasi Web Pengajaran Agama Islam Home Pesantren
2	Bahasa Pemrograman	Node.js, Go	Java	-	PHP, Javascript	PHP, Golang, Javascript
3	Basis Data	Amazon RDS, Amazon DynamoDB	NoSQL Cassandra	PostgreSQL	MySQL	PostgreSQL
4	Cloud Platform	Amazon Web Services	Amazon Web Services	Digital Ocean	Amazon Web Services	Digital Ocean

5	Containerized Tools	Kubernetes, Amazon Elastic Container Service	Docker Swarm	Docker	Amazon Elastic Container Service	Docker
6	Hasil Penelitian	Sistem yang meningkatkan produktivitas pengembang dalam jumlah yang banyak dan kebebasan dalam pemilihan teknologi dalam tahap pengembangan	Sistem yang mendukung resiliensi saat salah satu layanan terjadi malfungsi.	Sistem yang mendukung skalabilitas serta mendukung performa	Sistem yang meningkatkan *single responsibility* serta mendukung *independent deployment* pada platform Amazon Web Service	Sistem yang dibangun dengan penerapan arsitektur *microservice* berbasis aplikasi web

Tabel 2.1 berisi perbedaan antara penelitian yang dilakukan saat ini dengan penelitian terdahulu.

BAB III. LANDASAN TEORI

3.1. Microservice

Microservice adalah sebuah arsitektur perangkat lunak yang kecil, independen yang saling bekerja sama untuk membentuk satu kesatuan perangkat lunak. Karakteristik dari arsitektur *microservice* adalah kecil dan hanya memiliki satu tugas. Pendekatan karakteristik dari arsitektur *microservice* diawali dari prinsip pengembangan perangkat lunak yang dipopulerkan oleh Robert. C. Martin yang mengatakan bahwa semua modul yang berubah akibat alasan yang sama harus diletakan pada satu modul yang sama, dan pisahkan modul yang mengalami perubahan yang disebabkan oleh alasan yang berbeda [1].

Pendekatan yang dimiliki oleh arsitektur *microservice* sama dengan prinsip yang dipopulerkan Robert. C. Martin, modul yang sama diletakan pada satu layanan yang dibatasi oleh kebutuhan bisnis. Dengan pembatasan tersebut, pencegahan terhadap pertumbuhan *codebase* yang begitu besar dapat dilakukan [1]. Karakteristik berikutnya dari arsitektur *microservice* adalah *autonomous.* Artinya arsitektur ini mendukung bahwa satu layanan merupakan sebuah entitas yang terpisah dengan layanan lainnya. Artinya, satu layanan bertanggung jawab terhadap proses *deployment*. Proses komunikasi antar *microservice* dapat dilakukan dalam du acara, yaitu *asynchronous* dan *synchronous* [10]. Bentuk komunikasi dalam model *asynchronous* dilakukan dengan menggunakan *pub / sub* dengan model arsitektur *event-driven communication*. Lalu, untuk *synchronous* dilakukan dalam bentuk HTTP atau gRPC [10].

Keuntungan arsitektur *microservice* adalah setiap layanan juga dapat memilih teknologi yang sekiranya cocok untuk diaplikasikan pada layanan yang bersangkutan, sehingga pengembangan tidak bergantung pada penggunaan satu teknologi atau bahasa pemrograman [11].

Untuk memisahkan arsitektur *monolith* menjadi sebuah arsitektur *microservice* dapat dilakukan melalui beberapa tahap. Tahap yang pertama adalah menentukan *boundary context* yang terdapat pada sistem secara keseluruhan. Cara yang paling mudah adalah dengan mengetahui *seam* dari sebuah sistem. Istilah *seam* merujuk kepada blok *code* yang dapat dipisahkan secara terisolir dan

perubahannya tidak akan berdampak besar terhadap sistem secara keseluruhan. Dengan menentukan *seam,* melakukan pemisahan menjadi sebuah layanan yang kecil menjadi lebih mudah. Tahap berikutnya adalah melakukan *splitting* pada database. Pada tahap ini sering terjadi kesulitan mengingat kemungkinan terjadinya sebuah ketergantungan antar sebuah tabel [1].

3.2. Aplikasi Web

Aplikasi web adalah sebuah aplikasi yang menggunakan teknologi web browser sebagai *client side* dan menggunakan *network* sebagai komponen komunikasi. Penggunaan web browser sebagai *client side,* membuat pengembangan aplikasi web harus menggunakan teknologi yang mendukung penggunaan browser seperti HTML dan Javascript [12].

Aplikasi web dirancang untuk penggunaan yang cukup luas dan dapat digunakan oleh siapapun. Aplikasi web saat ini banyak digunakan untuk pembangunan platform *webmail* sampai yang popular saat ini *e-commerce.*

Penggunaan aplikasi web dapat dikategorikan cukup mudah, pengguna tidak perlu mengunduh aplikasi tersebut, karena dapat diakses melalui *network.* Pengguna saat ini dapat menggunakan aplikasi browser seperti *Chrome, Firefox* dan *Safari* untuk mengakses halaman web atau aplikasi web. Agar aplikasi web dapat bekerja dibutuhkan *server* yang dapat menyimpan *web server, application server* dan *database* untuk penyimpanan data [13]. Proses yang dibutuhkan aplikasi web untuk dapat bekerja dimulai ketika pengguna melakukan *request* ke *web server* untuk mengakses aplikasi tertentu. Lalu, *web server* akan mengarahkan *request* menuju *application server* yang nantinya akan melakukan akses ke *database* jika memang diperlukan.

Keuntungan yang diberikan oleh aplikasi web dibandingkan aplikasi melalui platform lainnya adalah tidak butuh instalasi. Mengingat aplikasi web dapat diakses selama pengguna terhubung dan memiliki sebuah jaringan, maka pengguna tidak perlu memasang aplikasi web di *device* yang melakukan akses, sehingga dapat menghemat penggunaan *storage* pada perangkat pengguna. Keuntungan berikutnya adalah aplikasi web dapat diakses pada beberapa platform yang berbeda, sehingga

memberikan kemudahan bagi pengguna untuk mengakses aplikasi web dengan piranti atau perangkat apapun

3.3. Pengajaran Agama Islam

Permasalahan dalam sebuah pendidikan adalah mutu serta perluasan kesempatan belajar [6]. Pengunaan teknologi menjadi salah satu upaya yang baik untuk meningkatkan kualitas mutu pendidikan serta peluang untuk memiliki kesempatan belajar yang semakin baik. Selain itu, permasalahan tidak hanya sekedar pada pendidikan itu sendiri, namun juga pada pengajaran pendidikan agama islam yang mencakup salah mendefinisikan agama, paradigma, serta tujuan pembelajaran. Pengajaran agama islam yang terjadi selama ini hanya dimaknai sebagai ritual dan baca-bacaan saja tanpa memaknainya dalam kehidupan sehari-hari. Permasalahan yang disebutkan sebelumnya, didukung oleh ketergantungan pembelajaran yang cukup tinggi dengan fasilitas pendukung pendidikan, kondisi sekolah, keluarga serta pandangan pengajar terhadap kurikulum [6]. Dukungan teknologi sendiri dapat diharapkan memecahkan masalah tersebut melalui beberapa tahap, yang pertama yaitu penggunaan teknologi yang memungkinkan pengajaran agama islam dapat lebih luas sehingga paradigma pengajaran agama islam dapat berubah. Yang kedua, mewujudkan integrasi antara agama islam dengan pendidikan sains. Yang ketiga, menggunakan teknologi secara positif untuk mengkaji ilmu pendidikan dengan dasar pengajaran agama islam [6].

3.4. Docker

Docker adalah *platform as a service* yang mendukung virtualisasi pada *level operating system* untuk membungkus dan mengantarkan perangkat lunak dalam bentuk *container* [14]. Penggunaan docker dipercaya dapat memudahkan *developer* untuk *delivery* produk perangkat lunak tanpa harus memikirkan kompatibilitas terhadap komputer server [15].

3.5. Docker Swarm

Docker swarm adalah *container orchestration tools* yang diciptakan oleh Docker untuk mendukung pengguna agar dapat mengelola docker *containers* yang terletak pada beberapa mesin komputer yang berbeda. Keuntungan penggunaan

docker swarm adalah menggabungkan beberapa *machine* yang memiliki docker engine untuk dapat digabungkan menjadi satu *cluster*. Keuntungan ini dapat diaplikasikan pada penerapan arsitektur *microservice*, yang dimana beberapa *microservice* dapat berada pada *machine* yang berbeda. Dengan komparasi dengan *tools* sejenis seperti Kubernetes, Docker Swarm memiliki kemudahan dalam hal konfigurasi serta penggunaan umum yang mudah[16]. Penggunaan dapat dibilang mudah karena pengaturan *cluster* dapat dilakukan dengan Docker engine CLI tanpa harus mengintegrasikan terhadap *tools* lainnya[17].

BAB VI. PENUTUP

6.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi serta pengujian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Pemisahan sistem dengan arsitektur *monolith* ke arsitektur *microservice* dilakukan dengan menentukan *boundary context*. *Boundary context* bertujuan untuk memberikan batasan dari *domain model* tertentu. Dalam kasus Home Pesantren, *boundary context* yang ditentukan ada empat, yaitu *user* yang berkaitan dengan pengolahan data pengguna beserta autorisasi, berikutnya adalah *transaction* yang berkaitan dengan pengolahan data transaksi yang terjadi didalam sistem. Berikutnya adalah *content* yang berkaitan dengan pengolahan data konten buku, penulis, penerbit dan paket buku. Yang terakhir adalah *file* yang berkaitan dengan pengolahan *file* pada web Home Pesantren. Selanjutnya, penerapan dapat dilakukan dengan melakukan pengkodean sistem dengan menerapkan basis data yang terpisah antar *microservice.*
2. Penerapan komunikasi antar *microservice* sehingga dapat menjadi sebuah kesatuan adalah dengan menerapkan dua teknik komunikasi yang berbeda, yaitu REST API untuk mendukung *synchronous communication* dan RabbitMQ untuk mendukung *asynchronous communication*. Implementasi REST API *Call* diterapkan pada pemanggilan *microservice content-service* yang dilakukan oleh *microservice transaction-service*. Implementasi RabbitMQ untuk mendukung *asynchronous communication* dilakukan saat pengguna melakukan konfirmasi transaksi dan pengolahan data penulis.

6.2. Saran

Saran yang dapat diberikan untuk pengembangan tahap selanjutnya adalah penggunaan layanan seperti Grafana untuk melakukan *monitoring* terhadap *microservices*. Selain itu, penggunaan *node* alangkah baiknya untuk ditambahkan

pada pengembangan berikutnya, agar penempatan atau distribusi *container* pada *microservice* semakin baik. Yang terakhir, komunikasi antar *microservice* dapat ditingkatkan dengan menerapkan gRPC *(Remote Procedure Call)*. Penggunaan gRPC dapat bermanfaat untuk mendukung performa yang lebih baik dibandingkan REST API.

DAFTAR PUSTAKA

[1] [1] S. Newman, *Building Microservices*, First. California: O`Reilly, 2015.

[2] M. Villamizar *et al.*, "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," *2015 10th Colomb. Comput. Conf. 10CCC 2015*, no. July 2018, pp. 583–590, 2015.

[3] H. Suryotrisongko, "Arsitektur Microservice untuk Resiliensi Sistem Informasi," *Sisfo*, vol. 06, no. 02, pp. 231–246, 2017.

[4] R. V. O'Connor, P. Elger, and P. M. Clarke, "Continuous software engineering—A microservices architecture perspective," *J. Softw. Evol. Process*, vol. 29, no. 11, 2017.

[5] V. B. Kusnandar, "Berapa Jumlah Penduduk Muslim Indonesia? | Databoks," *Katadata.Co.Id*, p. 2050, 2019.

[6] A. Nurdin, "Inovasi Pembelajaran Pendidikan Agama Islam Di Era Information and Communication Technology," *TADRIS J. Pendidik. Islam*, vol. 11, no. 1, p. 49, 2016.

[7] Xiao Ma, "Microservice Architecture at Medium," pp. 1–14, 2019.

[8] A. D. M. Del Esposte, F. Kon, F. M. Costa, and N. Lago, "InterSCity: A scalable microservice-based open source platform for smart cities," *SMARTGREENS 2017 - Proc. 6th Int. Conf. Smart Cities Green ICT Syst.*, no. Smartgreens, pp. 35–46, 2017.

[9] S. Habibullah, X. Liu, Z. Tan, Y. Zhang, and Q. Liu, "Reviving Legacy Enterprise Systems with Micro service-Based Architecture with in Cloud Environments," pp. 173–186, 2019.

[10] C. Synchronous and D. Delivery, "Designing interservice communication for microservices," pp. 1–9, 2020.

[11] Microsoft Corporation, "Building microservices on Azure," pp. 1–5, 2019.

[12] S. Al-Fedaghi, "Developing web applications," *Int. J. Softw. Eng. its Appl.*, vol. 5, no. 2, pp. 57–68, 2011.

[13] M. Rouse, “Web Application (Web App),” pp. 1–5, 2019.

[14] P. Software and S. Units, “What is a Container ? A standardized unit of software Package Software into Standardized Units for Everywhere : Linux ,” pp. 1–5, 2020.

[15] M. Sharma, “Docker — Basics & Architecture How does Docker solves this problem ?,” pp. 3–7, 2020.

[16] J. Ruostemaa, “Docker Swarm vs. Kubernetes: Comparison of the Two Giants,” 2016.

[17] D. Engines *et al.*, “Swarm mode overview Swarm mode key concepts and tutorial Swarm mode key concepts,” pp. 1–2, 2020.