BAB 2
TINJAUAN PUSTAKA

Bab tinjauan pustaka dijelaskan terkait tinjauan pustaka dan studi literatur yang digunakan pada penelitian mengenai rancang bangun manajemen Tugas Akhir berbasis *web* dengan pendekatan *Microservice* di Program Studi Informatika ITK. Tujuannya untuk mempelajari literatur yang berkaitan dengan penelitian. Teori yang dibahas antara lain Tugas Akhir, Sistem Informasi Tugas Akhir, Program Studi Informatika ITK, pendekatan *Microservice*, *Web Service*, *UML*, *User story*, Laravel, *REST API*, *Database MySql*, *Testing Jmeter Apache*, JSON dan beberapa penelitian terdahulu.

2.1 Tugas Akhir

Tugas Akhir atau bisa dikatakan sebagai skripsi merupakan salah satu syarat kelulusan yang ditetapkan di masing-masing program studi/fakultas ataupun universitas. Hal ini juga menjadi tuntutan dari sebuah formulasi kurikulum yang diberlakukan. Tugas akhir juga menjadi suatu syarat yang harus dipenuhi oleh calon sarjana untuk bisa mendapatkan gelar sarjana. Oleh karena itu, mahasiswa harus menetapkan tujuan tertentu dalam mempertanggungjawabkan skripsinya kepada dosen pembimbing dan dosen penguji (Machmud, 2016). Penyelesaian skripsi merupakan pengerjaan suatu karangan ilmiah yang wajib ditulis oleh seorang mahasiswa tingkat akhir sebagai salah satu persyaratan untuk menyelesaikan masa pendidikannya. Hal ini juga menjadi salah satu ajang tanda kemampuan akademik mahasiswa. Selain sebagai persyaratan akhir pendidikan skripsi juga menjadi salah satu syarat untuk mendapatkan gelar sarjana (Seto, Wondo dan Mei, 2020).

2.2 Sistem Informasi Tugas Akhir

Sistem merupakan himpunan dari sesuatu benda nyata maupun abstrak yang terdiri dari bagian-bagian atau komponen-komponen yang saling berkaitan, berhubungan, yang tentunya saling mendukung, yang secara keseluruhan bersatu

dalam satu kesatuan untuk mencapai sebuah tujuan tertentu secara efisien dan efektif. Untuk pengertian Informasi sendiri merupakan sebuah data yang telah diolah sesuai dengan keperluan tertentu (Suwita, 2020). Maka dari itu dapat disimpulkan bahwa sistem informasi merupakan suatu himpunan maupun komponen-komponen yang di dalamnya berupa data yang sudah diolah menjadi suatu informasi yang terintegrasi dengan tujuan dibuatnya sistem tersebut (Wibawa dan F., 2017). Dengan adanya sistem informasi yang dapat memanajemen Tugas Akhir dalam suatu instansi Pendidikan perguruan tinggi akan membuat *monitoring* pelaksanaan dan pengelolaan tugas akhir dapat dilakukan dengan cepat, tepat dan akurat, serta mampu meningkatkan efisiensi dan efektivitas kerja pihak-pihak yang terkait. Selain itu juga dapat mengurangi penggunaan kertas, sehingga perguruan tinggi dapat menekan biaya pengadaan kertas (Simatupang dan Muhammad, 2019).

2.3 Program Studi Informatika ITK

Program studi ITK merupakan salah satu perguruan tinggi yang berada di Kalimantan Timur, tepatnya berada di Jalan Soekarno Hatta KM.15, Kota Balikpapan. Program Studi Informatika ITK atau yang biasa disingkat dengan ITK adalah perguruan tinggi negeri yang berdiri sejak tahun 2012 dan disahkan melalui Peraturan Presiden No. 125 Tahun 2014. (Lembaga Penelitian dan Pengabdian Kepada Masyarakat, 2019). Salah satu Program Studi yang ada di ITK ialah Program Studi Informatika. Program Studi Informatika berdiri pada tahun 2016 bersamaan dengan Program Studi Teknik Industri dan Teknik Lingkungan. Kehadiran program studi Informatika di Institut Teknologi Kalimantan diharapkan dapat berpartisipasi aktif dalam mendukung Buku Putih Indonesia 2005-2025 di bidang Teknologi Informasi dan Komunikasi. Demi mencapai hal tersebut, maka diperlukan visi, misi, dan tujuan yang jelas. Visi, misi dan tujuan Program Studi Informatika ITK (Kalimantan,2020).

- Visi

Rumusan visi institusi Program Studi Informatika adalah *"Menjadi program studi unggul di bidang Informatika yang inovatif dan kreatif dalam poros Kalimantan pada tahun 2025"*.

- Misi

Sebagai upaya dalam mewujudkan visi tersebut, maka misi Program Studi Informatika ITK adalah:

1. Menyelenggarakan sistem pendidikan yang efektif, efisien, dan berkelanjutan dalam rangka menghasilkan lulusan sarjana Informatika
2. Menghasilkan lulusan yang memiliki kompetensi di bidang Informatika, berjiwa wirausaha (*entrepreneur*) dan dapat berperan positif di tingkat nasional dan internasional (*world class*)
3. Meningkatkan kontribusi dan kolaborasi dengan berbagai pihak dalam masyarakat dengan mengembangkan produk dan layanan dalam bidang Informatika di tingkat regional, nasional maupun internasional

- Tujuan

Dari keseluruhan visi dan misi program studi Informatika ITK tersebut, maka didapatkan beberapa tujuan yang ingin dicapai:

1. Menghasilkan lulusan yang memiliki kompetensi di bidang Informatika, berjiwa wirausaha (*entrepreneur*) dan dapat dipercaya sehingga mampu bekerja sama dan memberikan kontribusi di tingkat nasional dan internasional (*world class*), melalui kurikulum yang disusun dengan mempertimbangkan model kurikulum Informatika pada tingkat nasional dan internasional.
2. Menjalankan sistem pendidikan dengan penjaminan mutu sesuai standar nasional dan internasional.
3. Melibatkan *sivitas academica* Prodi Informatika dalam penelitian yang dapat memperkaya keilmuan di bidang komputasi dalam rangka mengisi dan menunjang pembangunan regional maupun nasional.
4. Melibatkan *sivitas academica* Prodi Informatika dalam pengabdian masyarakat dalam bentuk pembinaan, bimbingan dan konsultasi dalam rangka meningkatkan peran serta masyarakat dalam pembangunan potensi.
5. Meningkatkan kontribusi dan kolaborasi dengan berbagai pihak dalam masyarakat dengan mengembangkan produk dan layanan hasil inovasi dan

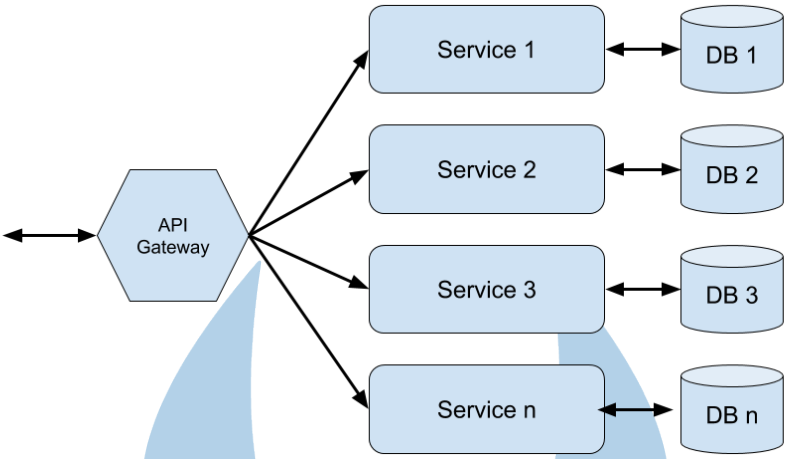
kreasi dalam bidang Informatika di tingkat regional, nasional maupun internasional.

6. Mengembangkan sertifikasi kompetensi di bidang Informatika di tingkat regional, nasional maupun internasional.

2.4 Pendekatan *Microservice*

Pendekatan *Microservice* merupakan salah satu pola arsitektur yang muncul saat ini, yang dibangun di sekitar konsep inti unit yang dikerahkan secara terpisah, komponen layanan, dan memiliki sistem yang bersifat terdistribusi. Pola arsitektur *Microservice* adalah pengembangan sistem dari *Service Oriented Architecture* (SOA) dengan sekelompok komponen layanan independen (otonom) yang terbentuk dari suatu aplikasi. Keuntungan terbesar dari penerapan pola arsitektur ini adalah komponen yang mudah diterapkan dan diuji serta kemampuan untuk menerapkan skala secara horizontal. Dapat dimisalkan jika kita mengumpulkan data dari sumber independen dan memprosesnya (atau setidaknya menggabungkannya) adalah salah satu pilihan yang baik adalah menerapkan pendekatan ini yang bertujuan untuk pemrosesan data waktu nyata dan memvisualisasikan data tersebut (Damyanov, 2019).

Microservice juga merupakan tren baru yang meningkat pesat, karena meningkatkan fleksibilitas untuk menggabungkan teknologi yang berbeda, mengurangi kompleksitas dengan menggunakan layanan yang ringan dan modular, dan meningkatkan skalabilitas dan ketahanan sistem secara keseluruhan. Dalam definisi gaya arsitektur *Microservice* adalah pendekatan untuk mengembangkan aplikasi tunggal sebagai *suite* layanan kecil, masing-masing berjalan dalam prosesnya sendiri dan berkomunikasi dengan mekanisme ringan (misalnya, *HTTP resource* API). Aplikasi kemudian tersusun dari sejumlah layanan (*service-based application*) yang bekerja secara kohesif untuk menyediakan fungsionalitas yang kompleks. Karena keunggulan arsitektur *Microservice*, banyak pengembang berniat untuk mengubah aplikasi monolitik tradisional menjadi *service-based application* (Hu, de Laat dan Zhao, 2019).



Gambar 2. 1 *Arsitektur Microservice*

Berdasarkan Gambar 2. 1 dapat dilihat Gambaran terkait arsitektur *Microservice*, yang di mana penggunaan *Microservice* juga didasarkan pada pemenuhan kebutuhan beberapa sistem yang memiliki *server-side*, yang dimana memungkinkan dimiliki oleh beberapa *user* dan mungkin perlu mengekspos API untuk pengembangan sistem tersebut. *Server-side* di sini dimaksud jika pengembangan sistem lanjutannya bisa menggunakan API yang sudah dibuat pada sistem sebelumnya (Ortiz *et al.*, 2019).

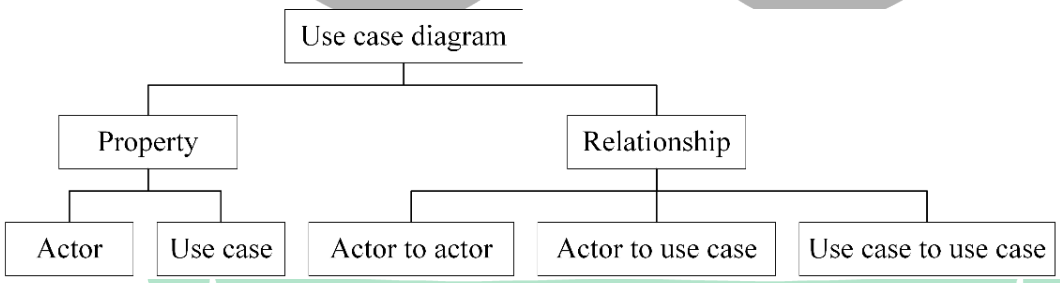
2.5 *Web Service*

Web service merupakan sebuah entitas komputasi yang dapat diakses melalui jaringan internet maupun intranet melalui standar protokol tertentu dalam platform dan antarmuka bahasa pemrograman yang independen. Biasanya digunakan sebagai layanan yang disediakan oleh sebuah situs web untuk memberikan layanan (dalam bentuk data) kepada sistem lain untuk memungkinkan sistem lain berinteraksi dengan sistem melalui layanan (layanan) yang disediakan oleh sistem yang menyediakan layanan web (Maulidiansyah, Rakhman dan Ramdhani, 2017). *Web Service* adalah aplikasi komputasi terdistribusi melalui Internet yang dapat diakses melalui satu set XML homogen antarmuka. W3C (*World Wide Web Consortium*) mendefinisikan layanan Web sebagai "perangkat lunak" sistem yang dirancang untuk mendukung interoperabilitas interaksi mesin-ke-mesin melalui jaringan. Sebuah *Web Service* terdiri dari satu set komputasi atau aktivitas fisik dengan sejumlah sumber daya untuk memenuhi kebutuhan penggunanya (Tun dan Onn,

2020). Atau dapat diartikan juga sebagai kumpulan protokol komunikasi korespondensi yang berkembang dan ditetapkan yang terdiri dari *Extensible Markup Language* (XML), *Simple Object Application Protocol* (SOAP), *Universal Description Discovery and Integration* (UDDI) dan *Web Services Description Language* (WSDL) melalui *Hypertext Transfer Protokol* (*HTTP*) (Hammoudeh dan Al-Ajlan, 2020).

2.6 *Unified Modeling Language*

Unified Modelling Language (UML) merupakan suatu struktur dan cara untuk pembuatan model desain program berorientasi projek (OOP) dalam pembuatan sebuah *software*. Bisa dikatakan sebagai metodologi dalam pengembangan OOP dan kumpulan perangkat untuk mendukung pengembangan sistem tersebut. Singkatnya UML merupakan suatu bahasa yang biasa digunakan developer untuk mengembangkan, menentukan, memvisualisasikan serta mendokumentasikan suatu sistem informasi (Pakaya, Tapate dan Suleman, 2020), (Lorenzo *et al.*, 2017). Demikian halnya dalam memvisualisasikan sebuah perancangan dalam pembuatan sebuah sistem dibutuhkan sebuah diagram yang dapat memudahkan *developer*. Dalam UML sendiri diagram yang biasa digunakan adalah *Use Case Diagram*, contohnya dapat dilihat pada Gambar 2. 2.



Gambar 2. 2 *Use Case Diagram Component*

Berdasarkan Gambar 2. 2 dapat dilihat gambaran dari *use case diagram component.* Mulai dari *property* sampai dengan beberapa jenis hubungan yang terdapat *use case diagram. Use case diagram* juga dapat digunakan untuk memudahkan developer untuk memahami bagaimana sistem seharusnya bekerja. *Use case diagram* digunakan untuk menggambarkan interaksi apa saja yang terjadi

antara pengguna dalam sistem atau biasa disebut sebagai aktor dengan *use case* sejalan dengan skenario tertentu (Fauzan *et al.*, 2021). Atau dalam kata lain *use case* merupakan ilustrasi setiap fungsi *requirement* yang diberikan oleh *stakeholder* (pemangku kepentingan). Aktor dalam *use case diagram* ini bisa berbentuk seseorang, perangkat maupun sebuah sistem lainnya. Pembuatan *use case diagram* yang benar dapat mendukung implementasi sistem dan eksekusi kebutuhan sistem yang efektif dan tepat (Sabharwal, Kaur dan Sibal, 2017).

2.7 *User story*

User story merupakan suatu pernyataan singkat/sederhana yang dapat menjelaskan sesuatu yang diperlukan pengguna dalam pembangunan sistem tersebut. Suatu *tools* yang bertujuan untuk mendefinisikan perilaku sistem dengan cara yang dapat mudah dimengerti oleh *developer* maupun *user* nantinya. *User story* memfokuskan pada *value* yang akan pengguna dapatkan daripada struktur fungsional yang *detail*. Dengan memiliki tujuan untuk memberikan pendekatan yang ringan dan efektif untuk mengelola *requirements* pada sistem yang akan dikembangkan. Contohnya "pengguna membutuhkan laporan untuk pemakaian energinya untuk setiap hari" (Leffingwell dan Behrens, 2010) (Andipradana dan Dwi Hartomo, 2021).

Maka dari itu, sebelum pengembangan suatu sistem diperlukan kegiatan *interview* atau setidaknya pengembang sudah memiliki beberapa *statement* yang berisi beberapa kebutuhan *user* dalam pengembangan sistem tersebut. Suatu pernyataan deklaratif yang ringkas dan dibentuk dengan baik tentang aspek bisnis yang dapat dinyatakan dalam suatu *statement*, menggunakan bahasa sederhana yang tidak ambigu dan dapat diakses oleh semua pihak yang berkepentingan dapat disebut sebagai *business rule*. *Business rule* dapat digunakan untuk menentukan dengan tepat terkait hak akses terhadap data maupun fungsi yang akan dibuat pada sistem tersebut. Contohnya menentukan fungsi utama atau *core function* yang ada pada sistem yang akan dikembangkan (Engelenburg, Janssen dan Klievink, 2019).

2.8 Laravel

Laravel merupakan *framework* bahasa PHP yang paling dapat digunakan untuk *programmer* pemula dan *amateur*. Ini dapat mengurangi waktu pengembangan aplikasi web dan pemasaran dengan metode PHP berorientasi objek modern. *Sintaks* ekspresif dan fungsi modernnya menarik bagi pengembang yang ingin membuat aplikasi yang kuat. Menggunakan *framework* yang dapat memudahkan proses pengembangan karena menyediakan beberapa modul dengan koneksi yang bisa dikerjakan bersamaan. Laravel menyediakan fitur-fitur canggih seperti lapisan abstrak *database* ekspresif dan injeksi ketergantungan (Subecz, 2021).

Framework Laravel adalah *framework* CLI (*Command-Line Interface*), migrasi dan artisan CLI yang menawarkan seperangkat alat dan arsitektur aplikasi yang menggabungkan banyak fitur terbaik dari *framework* seperti *Codeigniter*, *Yii*, ASP.NET MVC, dll. Laravel juga menganut konsep MVC, yang di mana berarti membuat program terstruktur dengan memisahkan logika menjadi 3 bagian, yaitu *model*, *view* dan *controller* (Santoso, Sinaga dan Zuhdi, 2021). Sering dikatakan bahwa Laravel adalah pengembangan web berbasis MVP yang ditulis dalam bahasa PHP yang bertujuan untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan awal dan biaya pemeliharaan. Hal ini juga dapat meningkatkan pengalaman bekerja dengan pengembangan aplikasi dengan membuat ekspresif, disediakan *sintaks* yang jelas, dan tentunya untuk menghemat waktu juga (Hermanto, Yusman dan Nagara, 2019). Namun untuk struktur pola MVC dalam laravel sedikit berbeda pada struktur pola MVC pada biasanya. *Framework* ini terdapat *routing* yang memberikan koneksi antara *request* dari *user* dan *controller*. Jadi *controller* tidak langsung menerima *request* tersebut (Purnama Sari dan Wijanarko, 2020).

2.9 *REST API*

REST API merupakan sebuah implementasi dari API (*Application Programming Interface*). *REST* (*Representional State Transfer*) adalah suatu arsitektur metode komunikasi yang menggunakan protokol *HTTP* untuk pertukaran

data pada sistemnya. *REST API* diakses menggunakan protokol *HTTP* (*Hypertext Transfer Protocol*), Dikarenakan hal tersebut, maka diperlukan penamaan dan struktur URL yang baik dan mudah dimengerti dalam penggunannya. *Endpoint* dalam pemanggilannya yang biasa dikenal sebagai URL API. Contoh pemanggilan URL yang baik adalah seperti berikut: *users*, *users/123*, *users/123/photos* dan seterusnya (Perdana, 2018). Gaya arsitektur *REST* semakin populer tetapi ada banyak perdebatan dan kekhawatiran yang berkembang tentang pemodelan layanan *web REST (REST API)*. *REST API* menjelaskan satu set sumber daya, dan satu set operasi yang dapat dipanggil pada sumber daya tersebut.

Operasi di *REST API* dapat dipanggil dari klien *HTTP* apa pun, termasuk kode *JavaScript* sisi klien yang berjalan di *browser web*. Klien *HTTP* menggunakan jalur relatif terhadap jalur dasar yang mengidentifikasi sumber daya di *REST API* yang diakses klien. Jalur ke sumber daya dapat bersifat hierarkis, dan struktur jalur yang dirancang dengan baik dapat membantu konsumen *REST API* memahami sumber daya yang tersedia di dalam *REST API* tersebut. Setiap sumber daya di *REST API* memiliki serangkaian operasi yang dapat dipanggil oleh klien *HTTP*. Operasi di *REST API* memiliki nama dan metode *HTTP* (seperti *GET*, *POST*, atau *DELETE*) yang dapat direpresentasikan pada beberapa bahasa pemrograman seperti XML atau JSON. Nama operasi harus bersifat *unique* di semua sumber daya di *REST API* tersebut. Selain itu, satu sumber daya hanya dapat memiliki satu operasi yang ditentukan untuk metode *HTTP* tertentu (Kusuma, 2021).

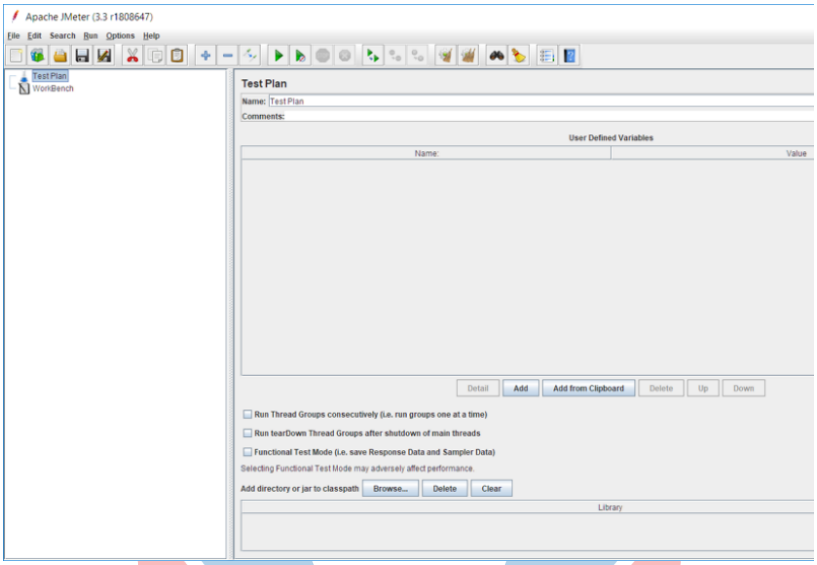
2.10 *Database* MySql

Sebuah sistem yang dirancang untuk mengorganisasi, menyimpan dan menarik data dengan mudah dapat dikatakan sebagai *database*. *Database* terdiri dari kumpulan data yang terorganisir untuk 1 atau lebih penggunaan yang disusun dalam bentuk digital. Selain kegunaan tersebut, *database* juga dapat mempermudah identifikasi data dengan cara pengelompokkan data dan dapat mempermudah penggunaan data, baik itu *input* maupun *output*. (Ramadhan dan Mukhaiyar, 2020). Maka dari itu, platform *MySql* yang digunakan untuk pengoperasian SQL pada penelitian ini.

MySql merupakan salah satu jenis *database server* yang sangat terkenal. *MySql* menggunakan bahasa SQL untuk mengakses *database* nya. Berdasarkan *Dataconomy*, versi terbaru *MySql* ialah salah satu *database* terpopuler di dunia. Hal tersebut dikarenakan *MySql* ialah basis data yang andal dan kompatibel dengan semua penyedia *hosting* utama. *MySql* juga hemat biaya penanganan dan juga mudah untuk dikelola (Ohyver *et al.*, 2019). Selain daripada keuntungan tersebut, *MySql* memiliki operator dan fungsi secara penuh yang mendukung perintah *SELECT* dan *WHERE* pada barisan *query*, penggunaan fungsi *CREATE, UPDATE*, dan *DELETE* juga sering kali digunakan untuk pengembangan sebuah sistem informasi. Dalam hal batas kemampuan, *MySql* terbukti mampu menangani *database* dalam skala cukup besar, dengan jumlah record lebih dari 50 juta dan 60 ribu Tabel serta 5 miliar baris. Selain itu, batas indeks yang dapat ditampung mencapai 32 indeks pada setiap Tabelnya (Sahara, 2019).

2.11 *Testing* Jmeter Apache

JMeter merupakan aplikasi dalam bentuk *desktop application* yang digunakan untuk menguji maupun mengukur kinerja atau perilaku fungsional aplikasi klien dan servernya (Sandra *et al.*, 2016). Sampai saat ini aplikasi ini digunakan dalam pengujian open-source yang paling banyak digunakan dan didistribusikan secara bebas yang dapat ditawarkan oleh Net. Ini murni berbasis java dan cocok dikembangkan melalui API (*Application Programming Interface*). Digunakan untuk menguji kinerja baik pada sumber daya statis maupun dinamis (SOAP / REST), Web bahasa dinamis - PHP, Java, ASP.NET, File, dll -, Java Objects, FTP Server dll) (Studi *et al.*, 2022). Halaman *menu* Jmeter Apache untuk pengelolaan *system testing* pada penelitian ini yang dapat dilihat pada Gambar 2. 3.



Gambar 2. 3 Tampilan awal *Jmeter Apache*

Berdasarkan Gambar 2. 3 dapat dilihat tampilan awal dari Jmeter Apache. Aplikasi ini dapat digunakan untuk menguji dengan menambahkan beban berat pada server, grup server, jaringan atau objek untuk menguji kekuatan ataupun menganalisis kinerja secara keseluruhan di bawah jenis beban yang berbeda. (Permatasari, 2020).

2.12 JSON

JSON (JavaScript Object Notation) adalah format data semi terstruktur ringan yang populer berdasarkan tipe data bahasa pemrograman JavaScript. Ini telah menjadi format pertukaran data utama di World Wide Web dalam beberapa tahun terakhir. JSON juga mendapatkan popularitas dalam penelitian komunitas basis data. Sebagai format data semi terstruktur, JSON tidak hanya dapat diintegrasikan dalam sistem *database* tradisional, tetapi juga banyak digunakan dalam sistem *database* NoSQL. Model data menggambarkan struktur data dasar dan semantik dari data yang mendasarinya, sehingga merupakan aspek fundamental dan kunci untuk format data apa pun termasuk data JSON. Karena model data juga merupakan kunci dan fondasi untuk teknologi manajemen data lainnya, seperti pengindeksan data, kueri data, pencarian data, pemetaan data, integrasi data, dan penambangan data, bagaimana merancang model yang efisien dan kuat untuk data JSON diperlukan dan penting untuk topik penelitian terkait lainnya, seperti integrasi data,

pencarian dan kueri data, kualitas dan evaluasi data, dll (Lv, Yan and He, 2018). Keuntungan dari pemakaian JSON antara lain penggunaan Bahasa yang sederhana dan banyak dipakai oleh pengembang, tipe sistem yang sederhana, dan Sebagian besar penggunaan API sekarang menggunakan data JSON. Atau dalam kata lain JSON merupakan format teks untuk pemberian *endpoint* supata data terstruktur. Itu berasal dari literal objek JavaScript. JavaScript, JSON dapat mewakili empat tipe primitif (string, angka, boolean, dan null) dan dua tipe terstruktur seperti objek maupun array (Piech and Marcjan, 2018).

2.13 *Data Flow Diagram*

Penggambaran bagaimana pengembangan sistem yang berinteraksi dengan lingkungannya dalam bentuk data, baik itu masuk ke dalam sistem maupun keluar dari sistem. DFD dapat digunakan untuk menggali apa yang dibutuhkan pengguna pada sistem yang akan dikembangkan dengan berfokus kepada struktur dan proses kerjanya (Simatupang dan Nafisah, 2020). Arus data yang coba digambarkan dengan sejumlah *symbol* tertentu yang di mana bertujuan untuk menerangkan perpindahan suatu data yang terjadi pada proses bisnis di suatu sistem manajemen yang sedang dikembangkan (Muliadi, Andriani dan Irawan, 2020). Diagram ini juga sering dimanfaatkan untuk penggambaran sistem yang telah ada begitu juga dengan sistem yang baru akan dikembangkan.

Dalam DFD terdapat 4 simbol yaitu *terminator*, proses, data store dan alur data. *Terminator* mewakili entitas eksternal yang menjalin komunikasi dengan sistem yang ingin dikembangkan. Pada simbol proses merupakan alat yang menggambarkan bagian dari sistem yang mengubah *input* menjadi suatu *output*. Kemudian untuk *data store* digunakan untuk pembuatan *model* sekumpulan sebuah kelompok data dan diberi nama dengan kata jamak. Simbol ini biasanya berhubungan dengan penyimpanan *database* sebagai pembacaan atau pengaksesan suatu paket tunggal data. *Data flow* yang biasa ditunjukkan dengan sebuah simbol anak panah. Anak panah tersebut menunjukkan arah menuju ke maupun dari suatu proses. Alur data ini penggambaran perpindahan data atau paket data dari suatu sistem ke suatu bagian sistem lainnya (Herlambang dan Setyawati, 2015).

2.14 Penelitian Terdahulu

Rangkuman hasil penelitian terdahulu yang telah dikaji oleh beberapa penulis sebagai bahan referensi pada penelitian ini yang ditampilkan pada Tabel 2. 1.

.

Tabel 2. 1 Penelitian Terdahulu

No.	Judul Penelitian	Peneliti Dan Tahun	Metode	Hasil Penelitian
1.	Sistem Aplikasi Pengelolaan Tugas Akhir Berbasis Mobile	(Simatupang dan Muhammad, 2019)	*Sistem Development Life Cycle (SDLC)*	Pengembangan Sistem Aplikasi Pengelolaan TA pada penelitian ini menggunakan pengujian *blackbox*, yang bertujuan untuk menguji fitur-fitur yang telah dikerjakan berjalan baik atau tidak.
2.	Pengembangan Sistem Informasi Tugas Akhir dan Skripsi (SIMITA) di Universitas Komputer Indonesia (UNIKOM)	(Suwita, 2020)	*Prototype*	Hasil dari penelitian ini telah berhasil menerapkan metode *prototype* ke dalam sistem informasi TA yang telah dikerjakan. SIMITA ini telah dikembangkan dengan konsep berbasis konten (CMS) yang dapat memudahkan pengelolaan atribut kelengkapan peserta skripsi.
3.	Implementasi Arsitektur *Microservices* Pada Rancang Bangun Aplikasi Marketplace Berbasis Web	(Sinambela dan Coastera, 2021)	*Microservice*	Aplikasi *marketplace* menggunakan arsitektur *Microservices* telah berhasil dibangun dengan hasil persentase pengujian *black box* 100% dari 25 aktivitas berhasil dan hasil persentase pengujian *endpoint* api 100% dari 29 aktivitas berhasil. Pada penelitian ini memiliki 3 aktor yaitu Admin, Penjual dan Pembeli.
4.	Implementasi Arsitektur *Microservice* Pada Aplikasi *Web* Pengajaran Agama Islam *Home* Pesantren	(Yuri Chandra Tri Putra, Thomas Adi Purnomo Sidi dan Joseph Eric Samodra, 2020)	*Microservice*	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*. Pada penelitian ini memiliki 3 aktor yaitu pengurus pesantren, admin dan anak-anak pesantren.

No.	Judul Penelitian	Peneliti Dan Tahun	Metode	Hasil Penelitian
5.	Analisis dan Desain Arsitektur *Microservice* dengan GraphQL Sebagai *API Gateway* untuk Sistem Informasi Akademik AIS UIN Jakarta	(Radhiyan, 2020)	*Microservice* dan GraphQL	Analisis dan desain arsitektur *Microservices* dengan GraphQL sebagai API *Gateway* untuk sistem informasi akademik AIS UIN Jakarta dilakukan dengan metode *decomposition by business capabilities*, artinya metode tersebut dapat digunakan untuk analisis dan desain arsitektur *Microservices* untuk sistem informasi akademik AIS UIN Jakarta. Pada penelitian ini memiliki 3 aktor yaitu admin,Dosen dan mahasiswa.
6.	Aplikasi Apotek Berbasis Web Menggunakan Arsitektur *Microservices* (Studi Kasus Apotek Glen, Kab.Toba)	(*Microservices et al.*, 2021)	*Microservices*	Aplikasi menggunakan arsitektur *Microservices* merupakan keputusan yang tepat apabila ingin membangun aplikasi dalam skala yang besar dikarenakan aplikasi dibangun terdiri dari beberapa *service* yang masing - masing memiliki satu tanggung jawab. Berbeda dengan arsitektur monolitik yang semua layanan berada dalam satu komponen sehingga satu program akan menangani semua layanan dalam aplikasi.
7.	Desain Sistem Terfederasi Dengan Pendekatan *Microservice Architecture* Pada Kasus Studi Sistem Pelaporan Pajak	(Darmayantie, 2020)	*Microservice*	Teknologi *web service* berbasis *REST*-API digunakan untuk menjamin interoperabilitas data dan informasi pada masing-masing sistem.
8.	Rancang Bangun *Back-End* "Siap": Sistem Informasi Aspirasi Dan Pengaduan Masyarakat Berbasis Web Dengan Menggunakan Metode *Microservice Springboot*	(Jayanto, 2017)	*Microservice Springboot*	Dalam membangun aplikasi SIAP yang berbentuk *Microservice* maka kebutuhan fungsional harus dipecah menjadi beberapa *Microservice* yang saling berhubungan sehingga membentuk satu kesatuan

No.	Judul Penelitian	Peneliti Dan Tahun	Metode	Hasil Penelitian
				aplikasi sesuai dengan proses bisnis yang telah ditentukan.
9.	Aplikasi *Taking Order* (*Front End*) Menggunakan *Microservice* Pada Pt. Xyz	(Rachman, 2018)	*Microservice* dan *DSRM*	Pengembangan sistem aplikasi *taking* order sudah dilakukan dan menghasilkan aplikasi *taking order* (*front end*) menggunakan *Microservice* pada PT XYZ untuk diimplementasikan di PT XYZ
10.	Pengembangan Sistem Informasi Penjadwalan dan Manajemen Keuangan Kegiatan Seminar dan Sidang Skripsi/Tugas Akhir (Studi Kasus Program Studi Sistem Informasi UNIKOM)	(Wibawa dan F., 2017)	*Prototype*	Pengembangan sistem pada penelitian ini menggunakan metode *prototype*, dikarenakan menurut penulis pada metode ini pengembang dan *user* dapat saling berinteraksi selama proses pengembangan sistem.