NARATIF : Jurnal Ilmiah Nasional Riset Aplikasi dan Teknik Informatika
Vol. 05 No. 01 Juni 2023
P-ISSN: 2656-7377 || E-ISSN: 2714-8467

PEMANFAATAN *WEB SERVICE* DALAM SISTEM LAYANAN GEREJA KATOLIK PAROKI MRPD

Glovenia Sanca Frisca[1], Sandy Kosasi[2], Tony Wijaya[3], Robertus Laipaka[4], David[5]
Program Studi Teknik Informatika[1,2,3,4,5]
STMIK PONTIANAK[1,2,3,4,5]
glovenia1111@gmail.com[1], sandykosasi@stmikpontianak.ac.id[2], tony_wijaya@stmikpontianak.ac.id[3],
robertus.laipaka@stmikpontianak.ac.id[4], david@stmikpontianak.ac.id[5]

Abstrak

Teknologi bagi Gereja akan selalu berkembang sesuai kebutuhan umat dalam sistem layanan informasi maupun administrasi. Namun, terdapat beberapa Gereja yang masih beroperasi menggunakan sistem layanan yang sederhana dan perlu dikembangkan, termasuk Paroki MRPD Pancasila Pontianak. Tujuan penelitian adalah menghasilkan sebuah sistem layanan berteknologi *Web Service* dengan *RESTful API*. Pemanfaatan *Web service* pada *Website* Gereja mengoptimalisasi kinerja pada sistem yang ada sebelumnya dalam mengintegrasikan data pada Paroki MRPD Pancasila Pontianak. Pengembangan sistem layanan dilakukan pada *Framework Codeigniter* dengan konsep Model *View Controller* (MVC). *Library Rest* yang digunakan untuk integrasi antara *Framework Codeigniter* dengan *web service* adalah *Chriskacergius*. Pendekatan yang digunakan ialah metode *Action Research* (AR). Metode pengembangan perangkat lunak yang digunakan adalah *Exetreme Programming* (XP). Hasil akhir dari penelitian adalah sistem layanan *Website* Gereja Katolik Paroki MRPD Pancasila Pontianak dengan teknologi *Restful API*. Adanya Notifikasi *e-mail* dari penambahan *library PHPmailer* serta *hyperlink Instagram*, *Youtube*, *Facebook*, dan *Whatsapp*. Sistem memungkinkan umat untuk melakukan transaksi *online* dalam persembahan. Pengujian sistem dilakukan menggunakan metode *White-Box*. Pengujian teknologi *API* pada sistem menggunakan *Postman*. Hasil pengujian pada tahap *cyclomatic complexity* dan *independent path* menghasilkan nilai yang sama yaitu 2, menunjukkan bahwa alur dan *source code* pemrograman *websit*e MRPD sesuai logika sehingga dapat melakuakan sistem layanan kepada umat.
Kata kunci: Sistem Layanan, *Web Service*, *RESTful API*, *PHPmailer*, *Postman*

Abstract

Technology for the Church will continuously develop according to the needs of the people in information and administrative service systems. However, several churches still operate using a simple service system and need to be developed, including the MRPD Pancasila Pontianak Parish. This research aims to produce a service system with Web Service technology with RESTful API. Utilizing the Web service on the Church's Website optimizes the existing system's performance in integrating data at the MRPD Pancasila Pontianak Parish. Service system development is carried out on the CodeIgniter Framework with the Model View Controller (MVC) concept. The Rest library to integrate the Codeigniter Framework and web services is Chriskacergius. The approach used is the Action Research (AR) method. The software development method used is Extreme Programming (XP). The final result of the research is the service system of the Pontianak Pancasila MRPD Parish Catholic Church Website service system with Restful API technology. There is an e-mail notification from the addition of the PHPmailer library and Instagram, Youtube, Facebook, and Whatsapp hyperlinks. The system allows devotees to make online transactions in offerings. System testing is carried out using the White-Box method. API technology testing on the system using Postman. The test results at the cyclomatic complexity and independent path stages produce the same value, namely 2, indicating that the flow and source code of the MRPD website programming is logical so that they can carry out a service system for the people.
Keywords: Service System, Web Service, RESTful API, PHPmailer, Postman

I. PENDAHULUAN

Di lingkungan Gereja Katolik, pemanfaatan teknologi informasi dapat diwujudkan dalam suatu *website* guna mempermudah sistem pelayanan dan pendataan. Sistem layanan dapat membantu dan mendukung kinerja suatu instansi atau organisasi dalam pengolahan data untuk menghasilkan informasi yang relevan, akurat dan berguna [1]. Berdasarkan keperluan dan keinginan dari objek yang perlu diatasi seperti penjelasan di atas, maka Gereja membutuhkan sebuah sistem berbasis *web* yang dapat menunjang proses layanan administrasi dalam sekretariat, sistem penyampaian pengumuman yang produktif serta penjadwalan petugas dan ibadah yang tertata. Kualitas *website* biasanya dinilai dari segi kepuasan pengguna perangkat lunak terhadap *website* yang digunakan [2]. *Restful* dinilai lebih mudah untuk diimplementasikan dalam hal pemrograman, selain itu navigasi yang ada didalam aplikasi *restful web service* mudah untuk dilakukan dengan mengikuti *hyperlink* dengan menggunakan *HTTP*. Serta, dalam pertukaran data restful dinilai memiliki kecepatan yang lebih cepat [3]. Gaya *Representational State Transfer* (REST) adalah abstraksi dari elemen arsitektur dalam sistem *hypermedia* terdistribusi [4]. Oleh sebab itu, dalam sistem layanan Gereja Katolik Paroki MRPD mentransformasi model pada tahap perancangan agar menghasilkan *Interface* dalam bentuk *Web Service* yang berdasarkan pada prinsip-prinsip *Service Oriented Architecture* (SOA). Tanpa *Web service*, komunikasi *peer-to-peer* khusus menjadi rumit dan khusus *platform* [5]. Dengan mengimplementasikan *restful web service* dalam SOA akan memudahkan pengembangan aplikasi perangkat lunak di luar sistem maupun dengan bahasa pemrograman atau *platform* berbeda [6]. Tujuan penelitian adalah mengembangkan sebuah *web service* yang dibentuk dari *Rest Server* dan *Rest Client* dalam *Framework CodeIgniter* dengan arsitektur *Model View Controller* (MVC). Dalam hal pengembangan ini dilakukan perbaikan sistem administrasi

Gereja dan penambahan fitur layanan formulir, fitur persembahan, *API Whatsapp*, Notifikasi beserta *e-mail* dengan *PHPmailer, hyperlink* media sosial Gereja sebagai media layanan lainnya yaitu *Whatsapp*, *Youtube*, *Instagram* dan *Facebook.*

II. TINJAUAN PUSTAKA

Rekayasa perangkat lunak mencangkup proses, metode, dan alat – alat yang memungkinkan sistem berbasis komputer yang kompleks dibangun secara tepat waktu dan berkualitas. *Framework* proses yang umum bagi rekayasa perangkat lunak terdiri atas lima aktivitas berikut ini [7]:

1. Komunikasi, sebelum pekerjaan teknis apa pun dapat dimulai, sangatlah penting untuk berkomunikasi dan berkolaborasi dengan pelanggan (dan para pemangku kepentingan *stakeholder* yang lain).
2. Perancangan, alur yang rumit dapat disederhanakan jika terdapat suatu peta. Suatu proyek perangkat lunak pada dasarnya merupakan suatu alur yang rumit, dan kegiatan perencanaan perangkat lunak tersebut menciptakan suatu "peta" yang membantu membimbing tim perangkat lunak ketika mereka melakukan suatu perjalanan.
3. Permodelan, seorang rekayasawan perangkat lunak membuat model-model untuk memahami kebutuhan perangkat lunak maupun rancangan-rancangan yang akan memenuhi kebutuhan tersebut.
4. Konstruksi, kegiatan ini menggabungkan pembentukan kode (*code generation*) dan pengujian yang sangat dibutuhkan untuk menemukan kekeliruan-kekeliruan/kesalahan-kesalahan dalam kode program komputer yang dihasilkan sebelumnya.

Service-Oriented Architecture (SOA) merupakan arsitektur dalam mengembangkan sistem informasi yang berorientasi pada *service*. SOA dapat meningkatkan efisiensi dan efektivitas dalam mengakses informasi. Kemampuan SOA dapat mengidentifikasi *client* yang mengakses *service* [8]. Sistem *database* terdistribusi merupakan bagian integral dari sebagian besar bisnis dan sebagian besar aplikasi perangkat lunak. Aplikasi ini menyediakan logika dan antar pengguna Antarmuka, sementara sistem *database* menjaga integritas data, konsistensi, dan redundansi [9]. Dapat disimpulkan penerapan Data Terdistribusi dalam pengembangan *Web Service* pada sistem layanan Gereja adalah tidak kesusahan dalam pengelolahan data, data terintegrasi dengan bai kantar Umat dan manajemen data Gereja dari Admin dan Sekretariat, akses penerimaan informasi juga menjadi lebih cepat, terutama ketika terjadi lalu lintas data yang padat kemudian informasi dan data yang sifatnya local akan lebih mudah diakses serta membuat pendataan Umat oleh Staff dapat bekerja lebih akurat dan efektif. *Web service* adalah mekanisme komunikasi yang didefinisikan antara berbagai sistem komputer. Tanpa *Web service*, komunikasi *peer-to-peer* khusus menjadi rumit dan khusus *platform*. *Web* perlu memahami dan menafsirkan ratusan hal berbeda dalam bentuk protokol. Jika sistem komputer dapat menyelaraskan dengan protokol yang *web* dapat memahami dengan mudah, itu sangat membantu. *Web service* adalah sistem perangkat lunak yang dirancang untuk mendukung mesin-ke-mesin yang dapat dioperasikan interaksi melalui jaringan. Ada banyak jenis *Web service* yang telah berkembang dari waktu ke waktu. Beberapa lagi yang menonjol adalah sebagai berikut [5]:

1. *Simple Object Access Protocol* (SOAP)
2. *Universal Description, Discovery, and Integration* (UDDI)
3. *Web Services Description Language* (WSDL)
4. *Representational State Transfer* (REST)

REST API memungkinkan sistem yang berbeda untuk berkomunikasi dan mengirim/menerima data dengan cara yang sangat cara sederhana. Setiap panggilan *REST API* memiliki hubungan antara kata kerja HTTP dan URL. Sumber daya dalam database dalam aplikasi dapat dipetakan dengan titik akhir *API* (*Application Programming Interface*) dalam arsitektur *REST*. *REST* adalah arsitektur *stateless, cacheable*, dan sederhana yang bukan merupakan protokol, tetapi sebuah pola. Pola ini memungkinkan titik akhir yang berbeda untuk berkomunikasi satu sama lain melalui HTTP. Alasan pengembang mengembangkan *REST* dalam *Web Service* pada Sistem layanan Gereja MRPD adalah properti utama yang membuat *REST* sederhana dan unik dibandingkan dengan yang terdahulu [5]:

1. *Client-server based architecture*: Arsitektur ini paling penting untuk *web* modern untuk berkomunikasi melalui HTTP. Satu klien-server mungkin terlihat naif awalnya, tetapi banyak arsitektur *hibrida* berkembang. Kami akan membahas lebih lanjut tentang ini segera.
2. *Stateless*: Ini adalah karakteristik paling penting dari layanan *REST*. Permintaan *HTTP REST* terdiri dari semua data yang dibutuhkan oleh *server* untuk memahami dan kembali tanggapannya. Setelah permintaan dilayani, *server* tidak ingat apakah permintaan itu tiba setelah beberapa saat. Jadi, operasinya akan menjadi *stateless* satu.
3. *Cacheable*: Untuk menskalakan aplikasi dengan baik, kita perlu *cache* tertentu tanggapan. Layanan *REST* dapat di-*cache* untuk *throughput* yang lebih baik. Representasi sumber daya: *REST API* menyediakan antarmuka yang seragam untuk berbicara dengan. Ini menggunakan *Uniform Resource Identifier* (URI) untuk memetakan sumber daya (data). Ini juga memiliki keuntungan meminta data tertentu
4. *Representation of resources*: REST hanyalah sebuah mekanisme untuk mendefinisikan layanan *web*. Ini adalah gaya arsitektur yang dapat diimplementasikan dalam berbagai cara. Karena fleksibilitas ini, dapat membuat layanan *REST* dengan cara yang diinginkan. Selama itu mengikuti prinsip *REST*, dapat memiliki kebebasan untuk memilih *platform* atau teknologi untuk *server*.

Berikut yang digunakan peneliti dalam Pengembangan *Web Service* sehingga dapat menggunakan *JSON* [10]:

1. Digunakan saat menulis aplikasi berbasis *JavaScript* yang mencakup ekstensi *browser* dan situs *web*.
2. Format *JSON* digunakan untuk serialisasi dan transmisi terstruktur data melalui koneksi jaringan. Hal ini terutama digunakan untuk mengirimkan data antara server dan aplikasi *web*.
3. Layanan *web* dan API menggunakan format *JSON* untuk menyediakan data *public*.

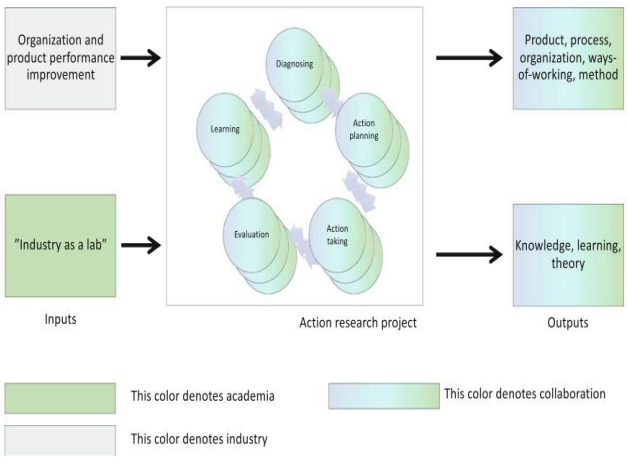
Pada penelitian sebelumnya tentang Penerapan *Web Service* dalam sistem informasi untuk Integrasi Data Simperpus dan SIAK menggunakan sistem *Rest Client* dan *Rest Server* [11]. Penelitian lainnya, *Web Service* untuk Integrasi Data dalam Pengelolaan Data Potensi Prestasi Mahasiswa STMIK Amikom Purwokerto berbasis *Website* dengan sistem *Client-Server* [12]. Perancangan *RESTful Web Service* pada Sistem Informasi Terintegrasi Menggunakan *Framework CodeIgniter* dalam uji dan simulasi penerapan *REST web service* pada Sistem Informasi perguruan tinggi dengan melakukan integrasi antar sistem [13].

Berdasarkan 3 (Tiga) penelitian di atas memiliki persamaan dengan penelitian ini yaitu penerapan *Web Service* dalam sistem yang dikembangkan. Perbedaan penelitian ini dengan sebelumnya ialah dikhususkan pengembangan *web service* sistem layanan pada Gereja Katolik Paroki MRPD dengan menerapkan *RESTful API* yang diintegrasikan dengan *framework CodeIgniter 3* dengan konsep MVC. Dalam penerapan *REST* pada *Codeigniter* ini diperlukan beberapa *library* tambahan yang tidak disediakan secara *default* pada *Codeigniter*, salah satu *library* yang digunakan adalah *library Chriskacerguis* untuk *Rest Server*. *Website* sistem layanan dan *Website* Admin bekerja sama dengan *library Guzzle*. Secara umum konsep MVC merupakan suatu metode dalam pemrograman dengan memisahkan komponen utama yang membangun aplikasi, yaitu memanipulasi data, *user interface*, dan juga bagian yang mengontrol aplikasi [14]. *CodeIgniter* memiliki ukuran yang kecil dibandingkan dengan *framework* lain [15]. *RESTful API* mewakili aplikasi *client* dan *server* dimana *client* dapat mengirimkan permintaan kepada *server* dan *server* akan memproses permintaan tersebut dan memberikan respon [16]. Integrasi data antar modul dalam menghindari terjadinya redudansi data dengan sistem terdistribusi membantu dalam penyebaran informasi dan pembagian sumber daya yang efektif, luas, efisien serta memiliki banyak macam teknologi [17]. Sistem *database* terdistribusi menyediakan logika dan antar pengguna Antarmuka, sementara sistem *database* menjaga integritas data, konsistensi, dan redundansi [18].

III. ANALISIS DAN PEMBAHASAN

1. Metode Penelitian

Penelitian ini menggunakan bentuk studi kasus (*case study*) dengan metode penelitian *Action Research* (Penelitian Tindakan). *Action Research* adalah semua tentang umpan balik, dan menggunakan sistem percobaan perangkat lunak, dapat memperluas kegiatan penelitian dengan melibatkan pengguna perangkat lunak. Dengan ini pengembang bisa dengan mudah tahu tentang fitur mana yang terbaik atau diperlukan dari sudut pandang pengguna dan melihat langsung apa yang mereka sukai. Berikut 5 (lima) tahapan dalam metode penelitian *Action Research* yaitu *Diagnosing*, *Action Planning*, *Action Taking*, *Evaluation*, dan *Learning* [19] (Gambar 1).



Gambar 1. Tahapan *Action Research* (Penelitian Tindakan)

Pada fase mendiagnosis, pengembang melakukan identifikasi masalah-masalah pokok yang ada, guna menjadi dasar objek sehingga terjadi perubahan. Pada fase *Action Planning*, pengembang memecahkan masalah menjadi bagian-bagian yang dapat dikelola, harus melakukan apa dan persiapan apa. Fase *Action Taking* didedikasikan untuk membuat perubahan dalam konteks intervensi. Pada tahap evaluasi, tim tindakan menganalisis data yang dikumpulkan dari fase sebelumnya. Bagian akhir dari siklus penelitian tindakan adalah spesifikasi pembelajaran (*Learning*). Metode perancangan perangkat lunak yang digunakan ialah *Extreme Programming* (XP). Ada pun data dalam penelitian ini terbagi menjadi 2 jenis data, yaitu data primer dan data sekunder. Teknik pengumpulan data dengan observasi dan

wawancara memperoleh data primer berupa buku Paroki, data DPP dan Komunitas serta dokumentasi untuk data sekunder berupa 4 formulir pendataan, foto – foto Gereja dan *record* wawancara. Pemodelan sistem yang digunakan untuk merancang *website* dalam penelitian ini adalah menggunakan *Unified Modeling Language* yaitu *Use Case Diagram*, *Activity Diagram*, *Sequence Diagram*, dan *Class Diagram*. Metode pengujian perangkat lunak menggunakan metode *White-Box Testing* dengan teknik *Basis Path Testing*.

2. Metode Pengumpulan Data

Metode pengumpulan data yang digunakan peneliti di dalam penelitian ini ialah Studi Dokumentasi yang terbagi menjadi 2 jenis data, yaitu :

a. *Data primer*; diperoleh peneliti dari wawancara kepada Pastor Paroki, beberapa staf Paroki terutama anggota DPP, perwakilan OMK, serta beberapa umat dan observasi langsung mengenai lingkup Paroki dan Data Pengurus serta Umat Gereja. Dari wawancara didapatkan jawaban dati pertanyaan – pertanyaan terkait sistem pengolahan data dan penyampaian informasi di Paroki MRPD sebelumnya, kendala yang dihadapi Paroki MRPD, sistem yang dibutuhkan untuk pengolahan data dan penyampaian informasi pada Paroki MRPD, alasan Paroki memerlukan sebuah perangkat lunak pengolahan data dan penyampaian informasi, apa saja data yang diambil dari umat di berbagai Formulir, serta fitur yang diharapkan ada dalam perangkat lunak Gereja Paroki MRPD tersebut. Dari Observasi, peneliti mengamati kegiatan/aktivitas dalam Paroki berjalan selaku anggota OMK Gereja Paroki, baik dari pendataan dan pengumuman/penyampaian informasi kepada umat serta bagaimana kendala yang dihadapi pihak pengurus maupun Umat Paroki secara langsung. Sekretariat juga memberikan dokumen kepengurusan DPP, contoh coretan penjadwalan untuk petugas ibadah dari Bidang Liturgi, contoh renungan harian, prognosis KK Paroki Per lingkungan beserta ketua stasi, Formulir Krisma, Formulir Komuni Pertama, Formulir Katekumen, dan Formulir Permandian. Pastor Paroki yang aktif RD.Fidelis Sajimin, RD.Saut Maruli Tua, RD.Pio Kristi Djelani dan RD.Alexander Mardalis.
b. *Data sekunder*; Dalam pengumpulan data sekunder, peneliti menggunakan studi dokumentasi. Peneliti memperoleh data dan informasi dengan membaca dan mempelajari dokumen – dokumen yang diberikan oleh Ibu Sarinah, selaku staf sekretariat Paroki atas izin dari Pastor RD. Fidelis Sajimin, selaku Ketua Umum DPP Paroki MRPD. Dokumen – dokumen tersebut antara lain Wilayah cakupan Paroki, Jumlah Lingkungan/Stasi, Data Ketua Stasi, Data jumlah KK pada setiap Stasi, Data kepengurusan DPP terbaru, Data Jumlah umat yang tercatat di Dekanat terakhir rekap dan Jumlah Umat Paroki di sekretariat yang tercatat sekarang di BIDUK KAP, serta data lainnya yang akan membantu dalam pengembangan perangkat lunak Gereja Paroki.

Laporan penelitian dapat disusun apabila data-data yang dibutuhkan dalam penelitian tersebut sudah terkumpul dan terpenuhi. Pengumpulan data dilakukan secara langsung dilakukan ke alamat Gereja MRPD tepanya di Gereja Paroki Maria Ratu Pencinta Damai Pancasila yang beralamatkan di Jl. Gusti Hamzah Gg.Pancasila V No. 1, Kelurahan Sungai Bangkong, Kota Pontianak adalah Paroki yang berada di kawasan Pontianak Kota dengan jumah umat 3028 jiwa dan 881 Kepala Keluarga (rekapan terakhir kali oleh ibu Sarinah selaku pegurus Sekretariat). Gereja Paroki Maria Ratu Pencinta Damai Pancasila merupakan salah satu Paroki yang berdiri sejak tahun 1986. Data-data yang dibutuhkan tersebut dikumpulkan dengan teknik tertentu yang disebut pengumpulan data. Kemudian data-data yang telah didapat itu dianalisis dan disimpulkan. Adapun teknik pengumpulan data dalam penelitian ini terdiri atas:

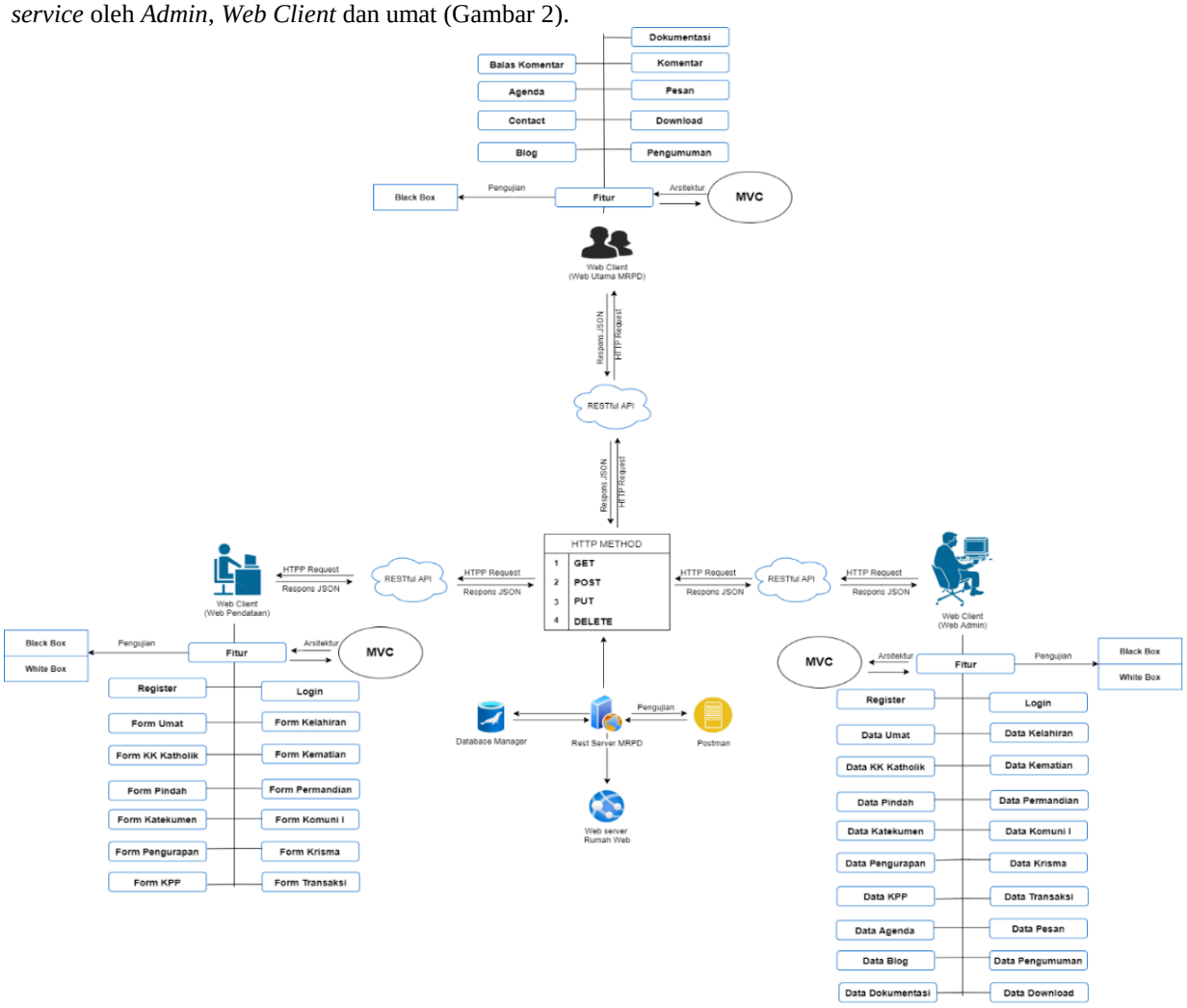
a. *Observasi*; pada penelitian ini dilakukan bertahap agar data yang akan diolah benar – benar valid. Dilakukan observasi awal agar peneliti yakin untuk melakukan penelitian terkait tema yang tersebut. Kemudian mencocokkan data-data dan dokumen yang telah terkumpul dengan realitas yang ada di lapangan disertai dengan wawancara dengan pihak-pihak yang berkaitan dalam pengurus Paroki. Dalam hal ini peneliti mendapatkan data informasi Gereja beserta pengurusnya dan data – data yang akan membantu dalam pengembangan perangkat lunak Gereja Paroki MRPD. Dari observasi peneliti mendapatkan satu buah buku Paroki MRPD Dari Masa ke Masa yang berisikan awal mula terbentuk Paroki hingga sekarang, Daftar hadir pembekalan Anggota DPP MRPD 2021-2023 yang berisikan Jabatan, Nama dan Nomor Telepon semua anggota DPP, Prognosis KK Paroki Per Lingkungan/Stasi/Kring dari rekapan BIDUK KAP.
b. *Wawancara*; dalam penelitian ini, peneliti melakukan tanya jawab langsung dengan narasumber RD. Fidelis Sajimin selaku Ketua Umum DPP Paroki MRPD, Bapak Paulus Purba selaku Sekretaris I DPP Paroki MRPD Pancasila Pontianak, Ibu Sarinah selaku Staf Sekretariat Paroki MRPD Pancasila Pontianak, Fransiskus Febri selaku Ketua Komunitas Orang Muda Katolik di Paroki MRPD Pancasila Pontianak dan beberapa umat Paroki. Pedoman wawancara yang digunakan hanya berupa garis – garis besar permasalahan yang akan ditanyakan. Adapun yang diperoleh adalah penambahan fitur formulir, Pihak Paroki memberikan 4 Formulir fisik untuk pencatatan data Umat yang kerap digunakan, yaitu Formulir Permandian, Formulir Katekumen, Formulir Krisma dan Formulir Komuni Pertama. Selain itu peneliti juga membicarakan sistem penjadwalan khusus petugas – petugas dalam liturgi yang berjalan di Gereja saat ini. Kemudian mengulas tentang renungan harian serta peyampaian kepada umat yang diharapkan pada sistem.
c. *Dokumentasi*; pada panelitian ini dilakukan peneliti menggunakan media yang dapat untuk merekam data, baik itu data visual/gambar, lisan, maupun tulisan. Media yang digunakan yaitu Kamera, *Recorder Handphone* dan

Lapptop. Alat tersebut digunakan untuk memudahkan untuk penyusunan laporan penelitian dan untuk mencocokan atau verifikasi data penelitian.

3. Proses Pengembangan *Web Service*

Mengembangkan aplikasi ini digunakan metode pengembangan sistem *extreme programming* (XP) dalam metode penelitian *Action Research*. Maka dari itu, pengembangan *Web Service* pada Sistem Layanan Gereja Katolik Paroki MRPD Pancasila Pontianak dilakukan secara bertahap sesuai dengan tahapan Metode penelitian dan metode pengembangan yang digunakan. Dimulai dengan melakukan diagnosis masalah – masalah yang ada pada Paroki. Melakukan identifikasi masalah-masalah pokok yang ada, guna menjadi dasar objek disini yaitu Gereja Katolik Paroki MRPD sehingga terjadi perubahan. Berikut masalah – masalah pokok yang dimaksud, yaitu kurang memuaskan dalam melakukan pelayanan untuk semua umat di Paroki dikarenakan jumlah umat yang banyak sedangkan jumlah pengurus kurang dari yang seharusnya, terutama saat perayaan hari raya. Banyaknya umat komplain dengan pelayanan yang kurang memuaskan. Jumlah kepengurusan dan data pengurus dalam Paroki tidak konsisten karena anggota yang keluar masuk paroki atau tidak terdata dengan baik. Kesulitan mendata dengan akurat dikarenakan data yang diambil dari umat dilakukan dengan cara pengisian Formulir. Ada pula keluhan umat karena adanya kesalahan data terutama dalam penulisan. Pendataan dan arsip juga dilakukan dengan pembukuan, sehingga staf kesulitan dalamm mencari data atau melakukan perubahan data umat yang sudah ada. Susahnya mendapatkan informasi dan pengumuman yang akurat atau terlambat mendapatkan informasi dikarenakan penyampaian secara lisan atau hanya dipapan pengumuman.

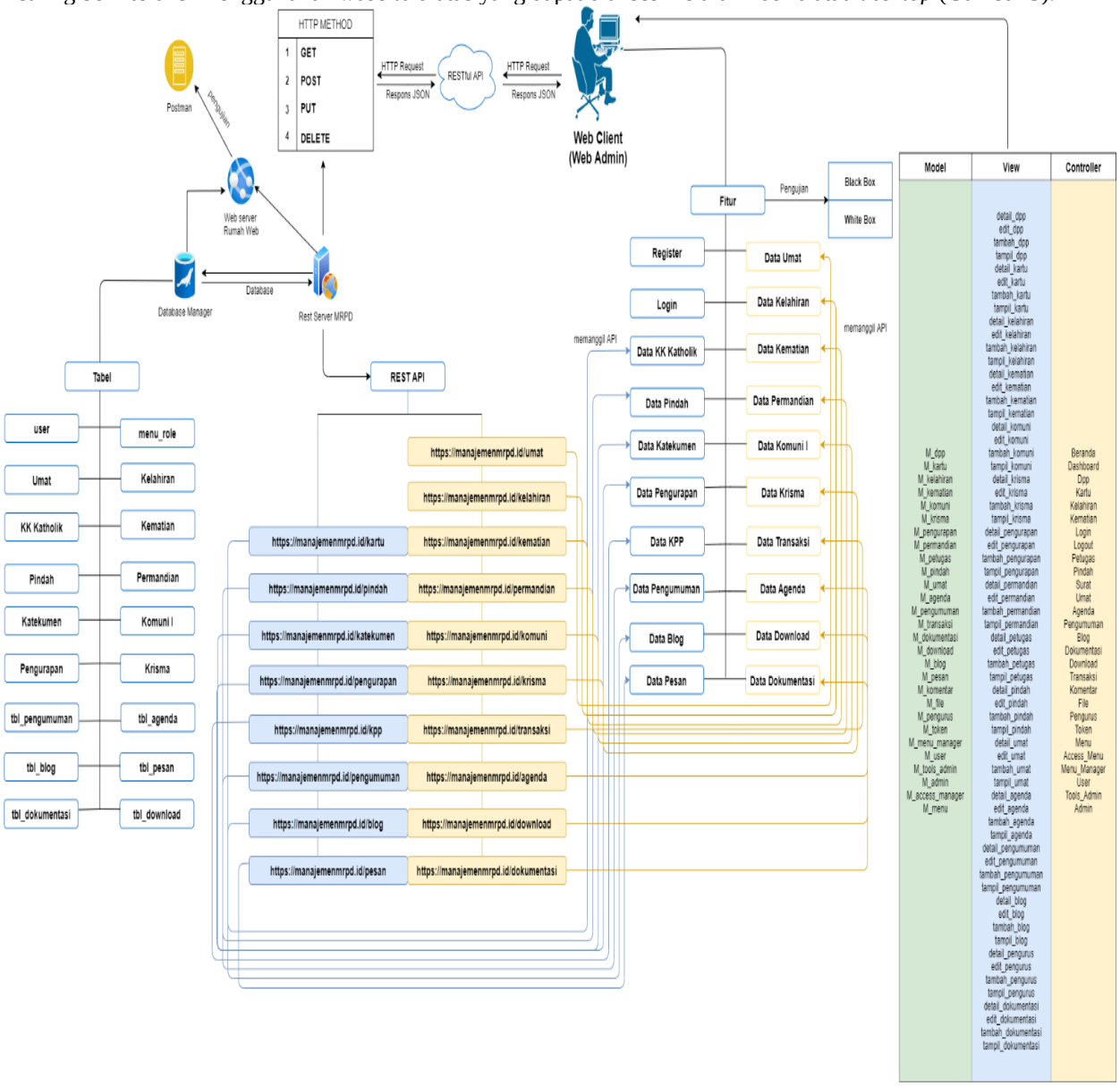
Perancangan arsitektur sistem dibuat juga dimaksudkan untuk memberikan gambaran dari kebutuhan *hardware* dan pengguna yang terlibat serta model dari arsitektur yang mendukung dalam pengembangan *web service* pada sistem layanan Gereja secara lengkap dan terperinci. Arsitektur pengambilan data menggunakan *RESTful API* dalam *web service* oleh *Admin*, *Web Client* dan umat (Gambar 2).



Gambar 2. Arsitektur Sistem *WEB SERVICE pada Sistem Layanan Gereja*

Perancangan arsitektur sistem ini dimaksudkan untuk memberikan gambaran yang mendukung dalam paparan atau penjabaran dari sistem *REST API* pada sistem layanan Gereja Paroki MRPD secara terperinci. *Admin* disini berperan sebagai pengelola data yang utama. Jika umat dan Sekretariat melakukan permintaan/*request* maka *REST API* akan

melakukan proses permintaan ke *Database*. *Request* tersebut akan diterima umat dan sekretariat dalam bentuk *JSON* dan diuraikan oleh perangkat lunak sehingga data hasil dari proses tersebut bisa digunakan atau ditampilkan. *Website* yang akan dikembangkan, dibangun menggunakan *framework Codeigniter*. Dalam *framework Codeigniter*, terdapat arsitektur MVC. Setiap bagian arsitektur memiliki perannya masing – masing. Ada beberapa hal yang harus dilakukan dalam *framework CodeIgniter* agar sistem integrasi *API* bisa berjalan, seperti menambahkan *library Rest* dalam *CodeIgniter*. Penerapan *REST* pada *Codeigniter* ini diperlukan beberapa *library* tambahan yang tidak disediakan secara *default* pada *Codeigniter*, salah satu *library* yang digunakan adalah *library* dari *Chriskacerguis*. Arsitektur sistem *website* MRPD yang memiliki model interaksi antara *Admin* dengan umat (*Client*) dimana kedua pengguna ini saling berinteraksi menggunakan *website* diatas yang dapat diakses melalui *mobile* atau *desktop* (Gambar 3).

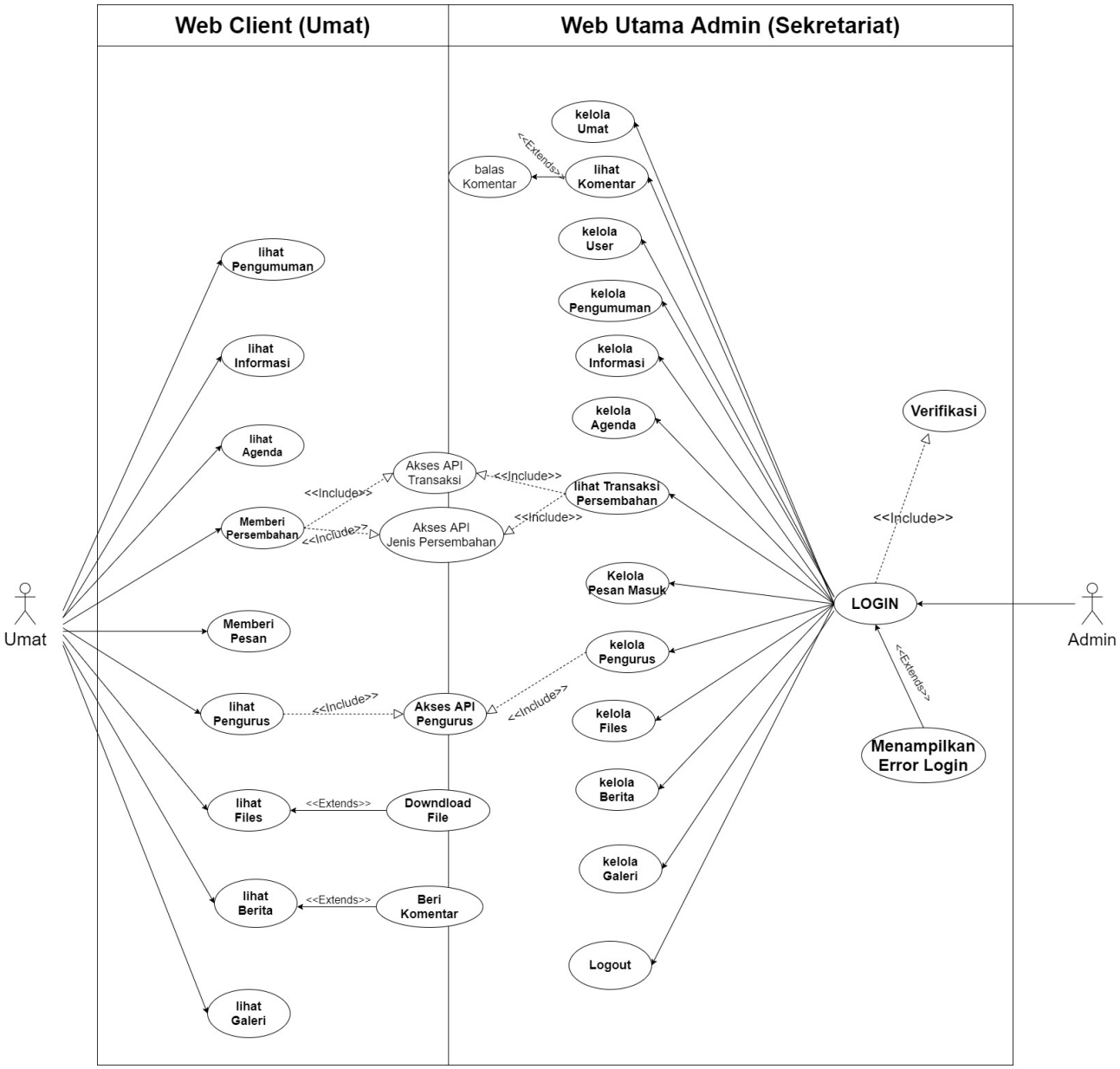


Gambar 3. Arsitektur Sistem *Admin*

Menjalankan kedua pengguna ini, semua perangkat harus terhubung dengan koneksi internet. Internet menghubungkan kedua *user* dengan *database* untuk pertukaran data melalui *Rest Server* MRPD (*web service*). Gambaran dari arsitektur ini dimulai dari *Admin* (*staff*) melakukan *login* ke sistem manajemen data MRPD. Sedangkan dalam *website* pendataan, setelah berhasil *login user* dapat membuka fitur atau fungsi sesuai dengan akses masing – masing yang telah di *Acc* oleh *Admin*. Pada *website* pendataan, semua formulir bisa diisi langsung oleh umat, sesuai dengan data yang diperlukan di dalam setiap formulir. Umat hanya perlu *login* untuk masuk ke *website* pendataan dan mengisi *form* sesuai kebutuhan dan bisa membuka fitur sesuai dengan akses yang diberikan *Admin*. Pada *website* pendataan, semua *form* bisa diisi langsung oleh Umat, sesuai dengan data yang diperlukan di dalam setiap formulir. arsitektur sistem *website* MRPD utama yang digunakan untuk sistem informasi umum Paroki MRPD kepada Umat MRPD maupun Umat umum serta beberapa penginputan beberapa data yang akan disampaikan kepada Umat MRPD.

Umat hanya perlu *login* untuk masuk ke *website* pendataan dan mengisi *form* sesuai kebutuhan dan bisa membuka fitur sesuai dengan akses yang diberikan *Admin*. *Client* yang menggunakan PC atau Laptop dapat mengakses *website* melalui *browser* dengan adanya jaringan internet, *browser* melakukan *request* melalui alamat *url* yang diakses dan Sistem *Middleware* (*CI, PHP, API, Apache, PhpMyAdmin)* didalamnya ada *framework CI* akan melakukan response dengan menampilkan halaman beranda *website* yang datanya hasil *response* dari *RESTful API* yang ada pada *backend framework CI* dan *response* tersebut akan diteruskan ke bagian *Model View Controller* (MVC) untuk diproses lebih lanjut ke bagian model untuk melakukan *request server* dan *request* ke *Database* (*MariaDB*), data hasil *request* tersebut akan dikembalikan dalam bentuk *response* oleh *controller* kemudian ditampilkan dalam bentuk *view*. *Middleware* dan *Database* menggunakan satu *Hosting*.

Pengembang memodelkan sistem menggunakan UML (*Unified Modeling Language*). Pemodelan sistem UML menggunakan 4 (empat) diagram, yaitu *Use Case Diagram*, *Activity Diagram*, *Sequence Diagram*, dan *Class Diagram*. *Use case diagram* berguna untuk mendeskripsikan tindakan sistem dari sudut pandang pengguna, sebagai deskripsi fungsional dari sebuah sistem dan proses utamanya, serta menjelaskan siapa saja yang terlibat sebagai aktor dalam menggunakan sistem berikut interaksinya. *Use case diagram* untuk sistem layanan Gereja Katolik Paroki MRPD digambarkan seperti berikut (Gambar 4):

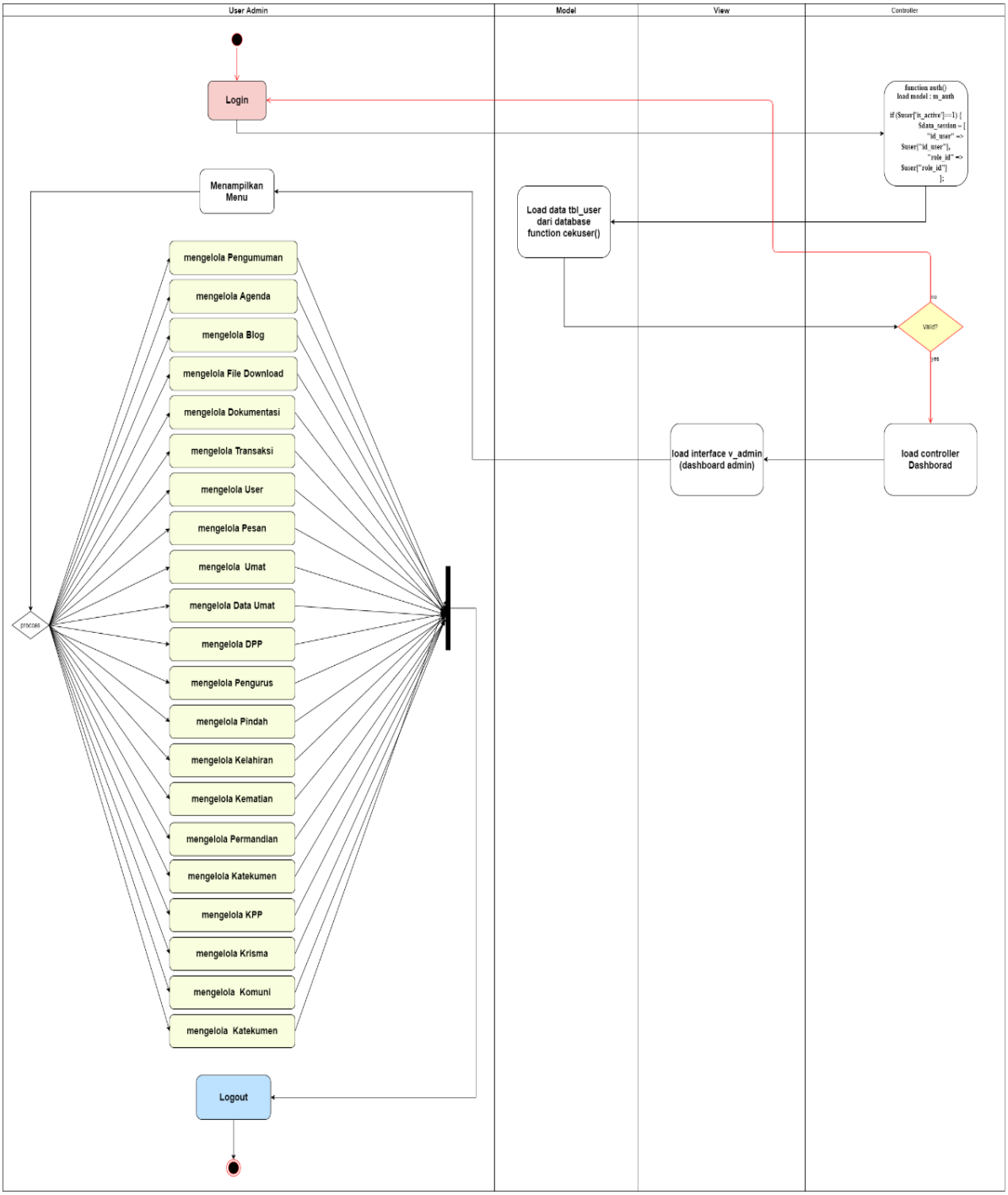


Gambar 4. *Use Case Diagram WEB SERVICE pada Sistem Layanan Gereja*

REST server menyediakan API yang berisi data umat, data kartu keluarga katolik, data sakramen permandian, sakramen pertama, sakramen krisma, data transaksi, data konseling, sakramen KPP, katekumen, pindah, peristiwan

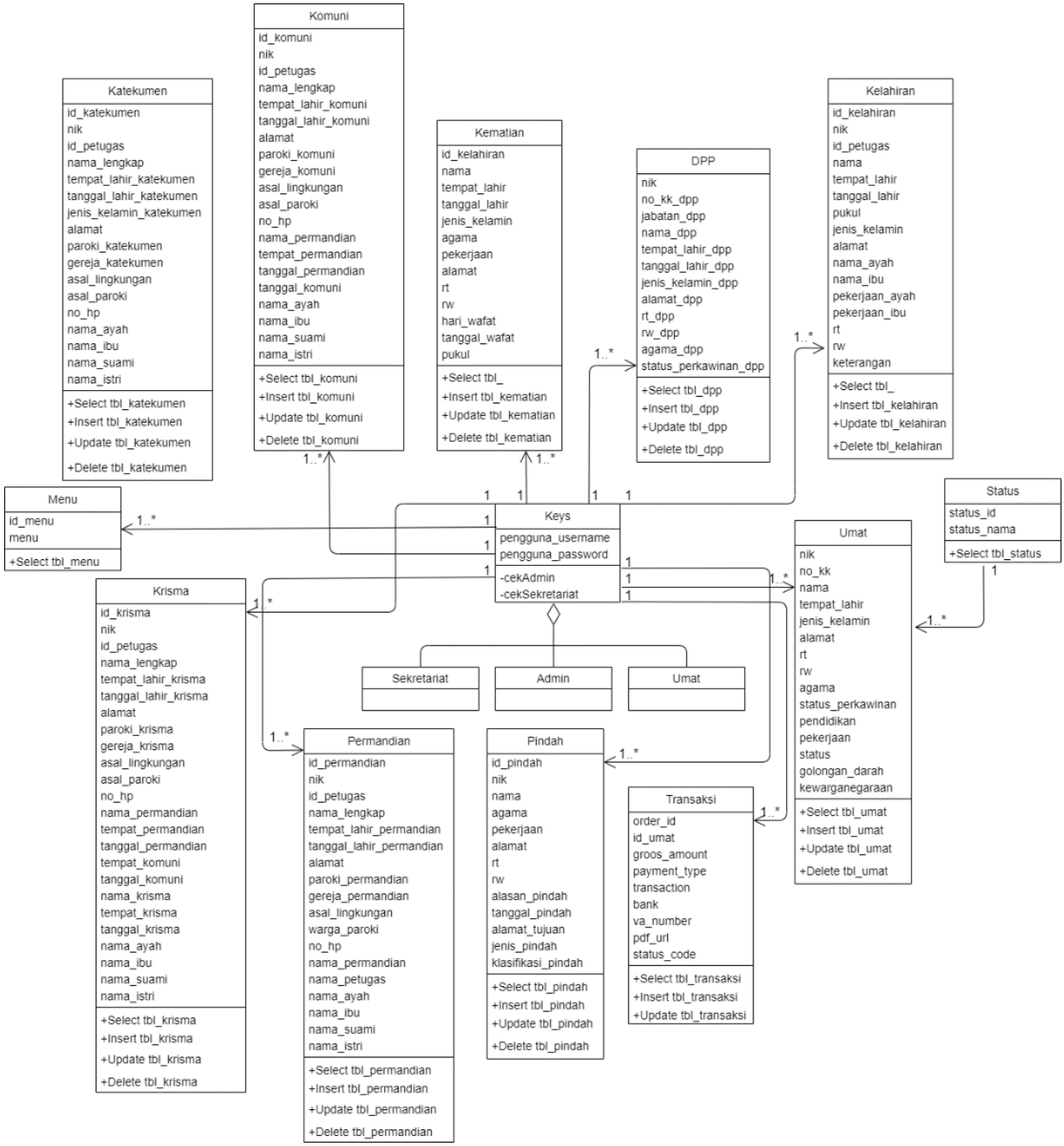
kelahiran, dan peristiwa kematian. *REST client* dapat mengakses dan mengelola data tersebut dengan *login* terlebih dahulu. Apabila *login* dianggap *valid*, maka *client* dapat mengakses data. *User* umat dapat mengakses dan melakukan *Get* dan *Put* data umat serta *Post* dan *Put* data transaksi. User pengurus mengakses dan melakukan *Get*, *Post*, *Put*, dan *Delete* pada data umat, data kartu keluarga katolik, data sakramen permandian, sakramen pertama, sakramen krisma, *get* data transaksi, data konseling, sakramen KPP, katekumen, pindah, peristiwan kelahiran, dan peristiwa kematian. *Model* dalam *Rest Client* akan memanggil dan menggunakan *composer Library Guzzle*. *Model Rest Client* menginput *base_uri* dan *auth* untuk akses *Guzzle* untuk dapat tersambung dengan *Rest API* (*Rest Server* MRPD). Jika dinyatakan valid, maka *model Rest client* dapat melakukan *request Restful API* (*Get/Post/Put/Delete*) data dari *API*. *Controller Rest Server* akan melakukan *load Model Rest Server* untuk *request Restful API* (*Get/Post/Put/Delete*) data dari *database* kemudian mengembalikannya ke *Controller Rest Server*. *Controller Rest Server* mengirimkan *response* ke *Model Rest client* dengan *response* format data *JSON*. *Model Rest client* melakukan *json decode* pada data yang telah diambil sehingga dapat dilihat dan diakses oleh *user*.

Integrasi data yang terjadi di *web wervice* memungkinkan *Model* dalam *Rest Client* memanggil dan menggunakan *composer Library Guzzle*. *Model Rest Client* menginput *base_uri* dan *auth* untuk akses *Guzzle* untuk dapat tersambung dengan *Rest API* (*Rest Server* MRPD). Jika dinyatakan valid, maka *model Rest client* dapat melakukan *request Restful API* (*get/post/put/delete*) data dari *API*. *Controller Rest Server* akan melakukan *load Model Rest Server* untuk *request Restful API* (get/post/put/delete) data dari *database* kemudian mengembalikannya ke *Controller Rest Server*. *Controller Rest Server* mengirimkan *response* ke *Model Rest client* dengan *response* format data *JSON*. *Model Rest client* melakukan *json decode* pada data yang telah diambil sehingga dapat dilihat dan diakses oleh *user*. *Activity diagram* yang menggambarkan aktivitas *user* dari proses *login* hingga *logout* dengan sistem *model*, *view*, *controller*, dari *framework codeigniter*. *Start* proses dengan *login* yaitu penginputan *e-mail* dan *password*, maka *Controller* akan memanggil *Model* untuk mengambil data (*query* (*SELECT*)) dari tabel *user* di *database local*. Kemudian *controller* akan mencocokan hak akses dari *e-mail* dan *password* diinputkan *user*. jika sesuai maka sistem akan memanggil *Controller Dashboard* dan menampilkan *dashboard user* untuk *interface* (Gambar 5).



Gambar 5. *Activity Diagram Web Admin*

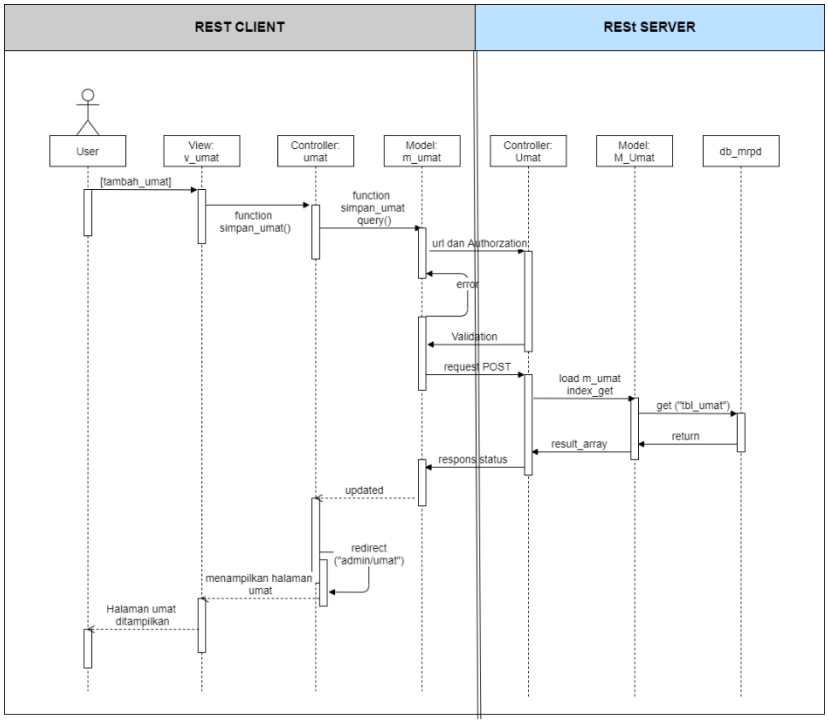
Class Diagram yang menjelaskan dari *User* yang harus *Login* dan mengelola Tabel Pengumuman, Pesan, Berita, Agenda, Menu, Balasan Komentar, Komentar, Kelahiran, Kematian, Kartu Keluarga Katolik, Pindah dan Umat. *Class Diagram* tersebut akan digunakan sebagai data yang akan disimpan di *database* yang di-*handle* oleh *Models*. *Class* kematian, DPP, transaksi, komuni, katekumen, krisma, menu, status, permandian, *user*, pindah, kelahiran, kematian, dan umat memiliki relasi dengan *Models* karena data tersimpan di *database* sehingga terhubung semua ke dalam *class keys* (Gambar 6).



Gambar 6. *Class Diagram WEB SERVICE pada Sistem Layanan Gereja*

Pada Gambar 6 di atas merupakan *class diagram* dari struktur sistem dan hubungan dari setiap *class* yang ada dalam pendataan *web service* MRPD Pancasila Pontianak. *Class* Umat memiliki relasi kardinalitas *one-to-one* dengan *class* kelahiran dan *class* kematian artinya satu umat hanya dapat memiliki satu kelahiran atau satu kematian. *Class* umat memiliki relasi kardinalitas *one-to-one* dengan *class* komuni, *class* krisma, *class* permandian, *class* katekumen, artinya satu umat dapat memiliki satu komuni, satu krisma satu permandian atau satu katekumen.

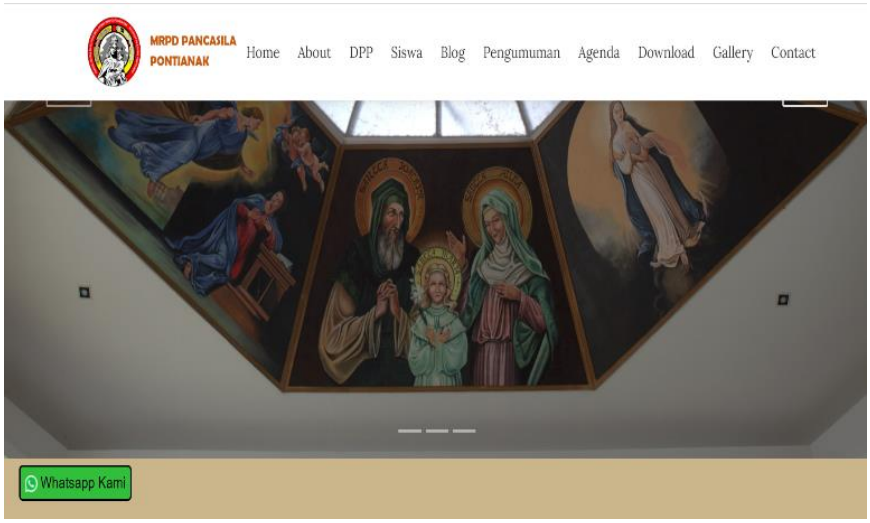
REST *client* melakukan *GET* data umat pada *REST server* dengan arsitektur MVC pada *Codeigniter*. *User* membuka halaman *View:v*_umat. *Controller*:umat akan menjalankan *function index()* yang kemudian memanggil *function* tampil_umat() pada *Model*:m_umat. *Model*:m_umat akan melakukan *request GET* data umat dari *API* dengan memasukkan *base_url* dan *auth API*. *Controller*:umat pada *REST server* melakukan *load Model*:m_umat untuk *GET* dari *database* tbl_umat kemudian mengembalikkannya ke *Controlle*r:umat dan *Controller* :umat mengirimkan *response* ke *REST Client* dengan format data *JSON*. *Model*:m_umat pada *REST Client* melakukan *json_decode* pada data yang telah diambil dan mengirimkannya ke *Controller*:umat yang kemudian dikembalikan ke halaman *View:*v_umat sehingga dapat di akses oleh *user* (Gambar 7).



Gambar 7. *Sequence Diagram* Data Umat

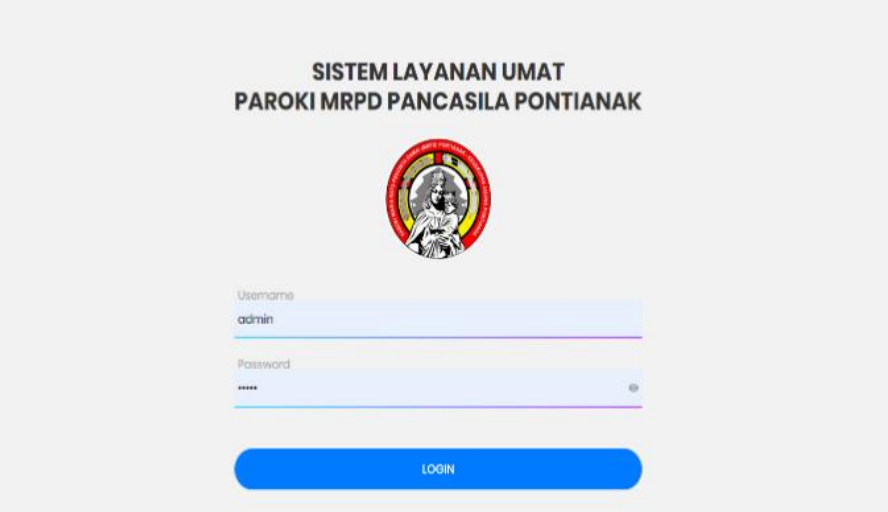
Rest Server MRPD dibangun menggunakan *framework CodeIgniter 3* menggunakan *library Chriskacergius*. Berikut ini ialah *Script* penggunaan *library Chriskacergius* pada *Controller* dalam *Rest Server CI*. Dalam pembuatan *Rest Server* ini agar tidak bisa sembarang akses maka dibuatlah *Authentication* berupa *API Key* yang terdiri dari dalam file *rest.php* pada *config*. Nama *key* yang digunakan untuk *Authentication* dalam *Rest Server* MRPD ialah "*MRPD-API-KEY*". Nama *key* tersebut yang nanti akan digunakan untuk pemanggilan *API* dari *Rest Server* MRPD. Nama *key* dan *key API* yang ditentukan ini sangat penting untuk pengujian *API* dalam tahap pengujian. Sebelum itu, diperlukan *autoloader* komposer (*require* '*vendor/autoload.php*') dilanjukan dengan mendeklarasikan *Client* (*use GuzzleHttp\Client*). *Client* membuat *requests*, mengirim *requests* dan mengatur *respons* pada objek *requests*. *Client* membuat "URL dasar" yang diinputkan pada "*base_uri*" dengan *http://localhost/mrpd_rest_server/* dan *valid login Rest* (*auth*) dengan nama "*mrpdrest*" serta *password* "110101". Nama *key* yang digunakan untuk *Authentication* dalam *Rest Server* MRPD ialah "*MRPD-API-KEY*". Nama *key* tersebut yang nanti akan digunakan untuk pemanggilan *API* dari *Rest Server* MRPD.

Rest Server, yang digunakan hanya fungsi dari *Model* Dan *Controller* karena tidak akan menampilkan *view*. *Website* sistem layanan Paroki MRPD Pancasila Pontianak dan *website Admin* mendapatkan data- data dari *RESTful Web Service*, sehingga menggunakan perintah atau *function HTTP method* yang terdiri dari *GET, PUT, POST,* dan *DELETE*. *User website* utama adalah umat atau umum (Gambar 8).



Gambar 8. *Home Website* Utama

User Website Pendataan adalah staf Gereja dan *Admin*. *Form Login* untuk akses yang ada dalam *Website* Sistem Layanan (Pendataan). *User* harus memasukkan *Username* dan *Password* sesuai *user* data yang sudah ada dalam *database user*. Jika data tersebut sesuai makan *Username* dan *Password* benar dan *user* bisa mengakses menu – menu pendataan (Gambar 9).

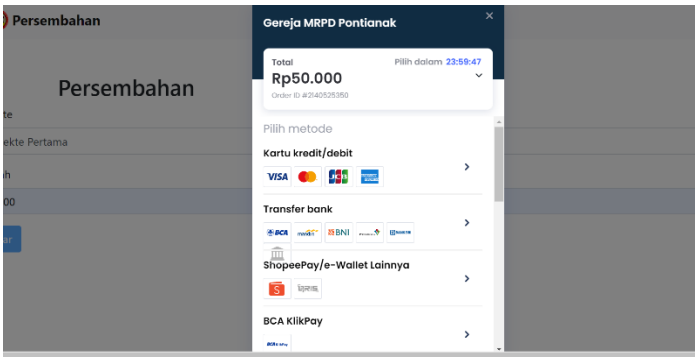


Gambar 9. *Form Login* Pendataan

Menu – menu pendataan dalam *Website* Sistem Layanan (Pendataan) antara lain, data Umat, data DPP, data Kelahiran, data Kematian, data KK Katolik, data Pindah, data Permandian, data Katekumen, data Pernikahan, data Komuni Pertama, data Krisma, data Pengurapan Orang Sakit, dan data Persembahan. Umat menginput data, jika berhasil maka sistem akan memberi notifikasi "Terima Kasih! Data anda berhasil diinput". Saat Data berhasil diinput maka Data umat di sistem *Admin* betambah. Dalam proses ini, penginputan diproses bagian *Controlers admin* dan dikelola dalam Database *Models Admin* kemudian proses diteruskan ke *Views Admin* untuk ditampilkan pada Halaman Data Umat Paroki. Jika Data yang di-*input* umat gagal atau tidak berhasil di-*input* (ada salah satu *Form* data ada yang kosong atau tidak diisi) maka sistem secara otomatis akan mengirim notifikasi "Maaf! Data Anda Gagal di-*input*." Dan tampil tulisan merah pada *Form* yang tidak diisi. Dalam sistem *admin*, bisa mengakses fungsi tambah, jika ada umat yang meminta bantuan untuk inputkan data. Dalam sistem sama seperti halnya penginputan oleh umat. Dan saat menjalankan fungsi hapus, maka sistem memberi notifikasi "Apakah anda yakin menghapus data?". Jika *Admin* klik iya, maka sistem memberikan notifikasi "Data berhasil di hapus!" dan data yang dihapus tadi tidak ada lagi di *database*. Jika *admin* menjalankan fungsi *edit*, admin memilih data yang akan di *edit*. Sistem membuka data yang dipilih oleh *admin* dalam bentuk *Form Edit*. *Admin* melakukan perubahan dan klik simpan. Sistem memberikan notifikasi "Data berhasil di *Update*!" kemudian sistem meng-*update* data dalam *database*. Data tersebut akan diintegrasi menggunakan *function Restful API* dari *Rest Server* MRPD yang dikembangkan (Gambar 10).

Gambar 10 Menu Pendataan

Midtrans merupakan perusahaan finansial berbasis teknologi atau biasa disebut dengan *Payment Gateway* membantu dalam proses transaksi persembahan dalam *Website* Gereja Katolik Paroki MRPD Pancasila Pontianak. Dalam sistem *client-server*, pada *midtrans* yang menjadi patokan ialah *client key* dan *server key* dari layanan *midtrans* (Gambar 11).



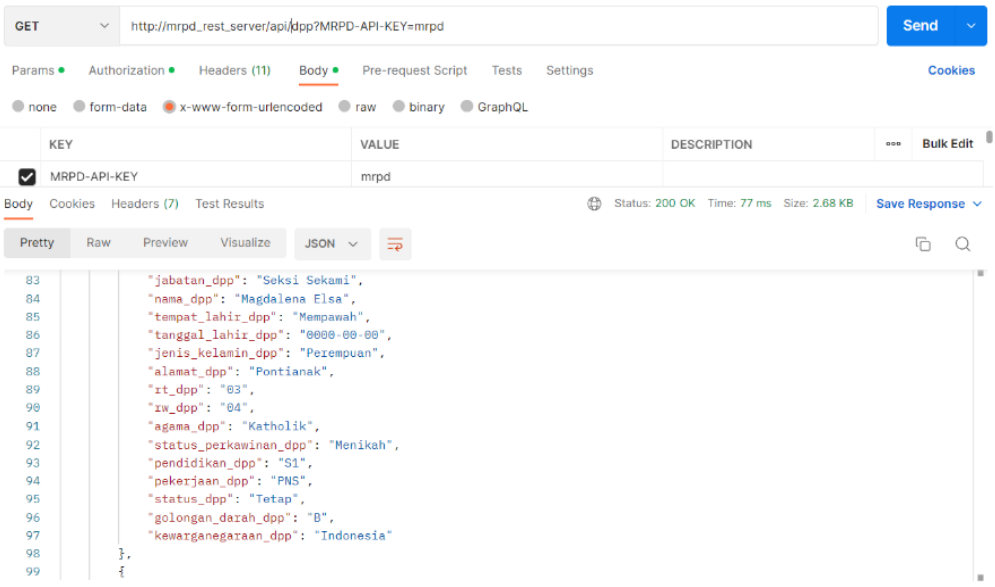
Gambar 11. Persembahan dengan Teknologi *Paymant Gateway*

Script function Get DPP dakan diuji untuk mendapatkan data DPP. Data DPP tersebut diperoleh dari *DPP_model* oleh *Rest Server*.

```
public function index_get()
{
    $this->response([
        'status' => 200,
        'data' => $this->Dpp_model->getDpp()
    ], RestController::HTTP_OK);
}
```

Gambar 12. *Script function Get* DPP

Berdasarkan hasil pengujian pengujian *script function Get* DPP yang dicantumkan sebelumnya, status “200 OK” menunjukan pengujian berhasil sehingga menampilkan data DPP yang ada dalam format data *JSON*. Jika status “500” maka *error* atau *failed*!, dan status lainnya (Gambar 13).



Gambar 13. Hasil Pengujian *Script function GET* DPP pada *Postman*

Test case function Get seperti pada tabel 1, merupakan hasil *Postman test* dari *REST Server configuration* yang digunakan. Status “200 OK” jika inputan Benar, yaitu *key* “*MRPD-API-KEY*” dan *Value* “mrpd” dalam *Params*, *Type* “*Basic Auth*”, *Username* “*mrpdrest*” dan *Password* “110101” dalam *Authorization*. Status “403 *Forbidden*” jika *Rest Key* tidak ada atau tidak sesuai (*Empty value*).

TABEL 1
TEST CASE GET RESTFUL WEB SERVICE MRPD

Case	***HTTP Methods***	***URL***	***Params***		***Authorization (Basic Auth)***		***Status***
			Key	***Value***	***Username***	***Password***	
Kelahiran	*GET*	http://mrpdpancasilapontianak.id/api/kelahiran	MRPD-API-KEY	mrpd	*mrpdrest*	110101	200 OK
	GET	http://mrpdpancasilapontianak.id/api/kelahiran	*Empty value*		*mrpdrest*	110101	*403 Forbidden*
	GET	http://mrpdpancasilapontianak.id/api/kelahiran	MRPD-API-KEY	mrpd	*No Auth*		*401 Unauthorized*
Kartu	*GET*	http://mrpdpancasilapontianak.id/api/kartu	MRPD-API-KEY	mrpd	*mrpdrest*	110101	*200 OK*
	GET	http://mrpdpancasilapontianak.id/api/kartu	Empty value		*mrpdrest*	110101	*403 Forbidden*
	GET	http://mrpdpancasilapontianak.id/api/kartu	MRPD-API-KEY	mrpd	*No Auth*		*401 Unauthorized*

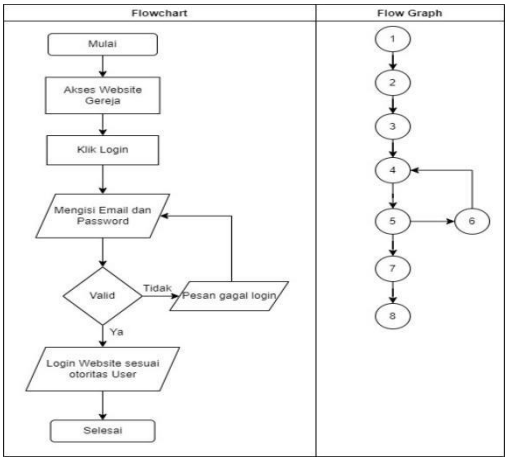
Status "*401 Unauthorized*" jika *authorization* tidak dimasukan atau tidak sesuai (*No Auth*). Diatas hanya pengujian *GET* dan URL pada *case* Kelahiran dan Kartu Keluarga Katholik. Status *valid* jika fungsi sistem berjalan dengan baik. Di pengujian ini teknik yang digunakan adalah *basis path* untuk mendefinisikan himpunan basis dari jalur eksekusi. Dengan kata lain *test case* yang dibuat akan mengeksekusi setiap *statement* yang ada pada program. Pengujian *white box* ini dilakukan dengan melihat pure kode tanpa melihat tampilan *interface* dari halaman aplikasi.

a. Pemetaan *source code* halaman *login*
Pada tahap ini di petakan bagian-bagian dari *source code auth.php* sebagai berikut:

```
public function login()
{
    $email      = $this->input->post("email");
    $password   = $this->input->post("password");
    $user = $this->db->get_where("tbl_user",["email" => $email])->row_array();
    if ($user) {
        if (password_verify($password, $user['password'])) {
            if ($user['is_active']==1) {
                $data_session = [
                    "id_user" => $user["id_user"],
                    "role_id" => $user["role_id"]
                ];
                switch ($user["role_id"]) {
                    case 1:
                        $this->session->set_userdata($data_session);
                        redirect('admin');
                        break;
                    case 2:
                        $this->session->set_userdata($data_session);
                        $this->session->set_flashdata('info','Selamat datang '.$user["username"]);
                        redirect('home');
                        break;
                    case 3:
                        $this->session->set_userdata($data_session);
                        $this->session->set_flashdata('info','Selamat datang '.$user["username"]);
                        redirect('home');
                        break;
                    default:
                        $this->session->set_flashdata('error','Oops , Role anda tidak ditemukan');
                        redirect('login');
                        break;
                }
            }else{
                $this->session->set_flashdata('warning','Akun anda belum di aktivasi');
                redirect('login');
            }
        }else{
            $this->session->set_flashdata('error','Password salah');
            redirect('login');
        }
    }else{
        $this->session->set_flashdata('error','Email tidak ditemukan');
        redirect('login');
    }
}
```

Gambar 14. *Source Code Login*

b. *Membuat flowgraph dari pemetaan source code*
Terdapat 1 verifikasi pada nomor 5, dan pada nomor 7 yang artinya berhasil Login dan berlanjut ke menu *Admin*. Dan jika gagal *Login* akan kembali ke nomor 4 yaitu *form* verifikasi *Email* dan *password* (Gambar 15).



Gambar 15. *Flowgraph Login*

c. *Cyclomatic Complexity*

Cyclomatic complexity adalah *software* yang menyediakan acuan kuantitatif kompleksitas suatu logika dalam program. Pada gambar 5.92 terdapat beberapa *nodes*, *edges*, dan *predicated nodes* untuk menghitung *Cyclomatic Complexity* berikut:

V(G) = E – N + 2
E = jumlah *edge flowgraph*
N = jumlah *simpul flowgraph*
Sehingga kompleksitas pada *flowgraph* data adalah:
V(G) = 8 – 8 + 2 = 2

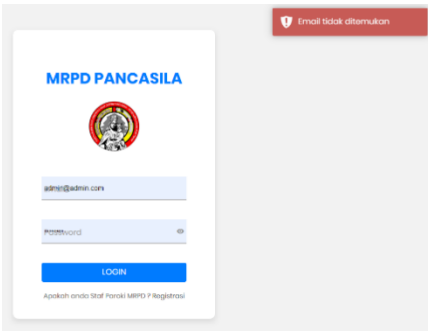
d. Independent *Path*

Dari hasil penghitungan *Cyclomatic Complexity* terpadat 2 (dua) *path* berikut:
Jalur R1 = 1-2-3-4-5-7-8
Jalur R2 = 1-2-3-4-5-6-4-5-7-8

e. Uji *Skenario*

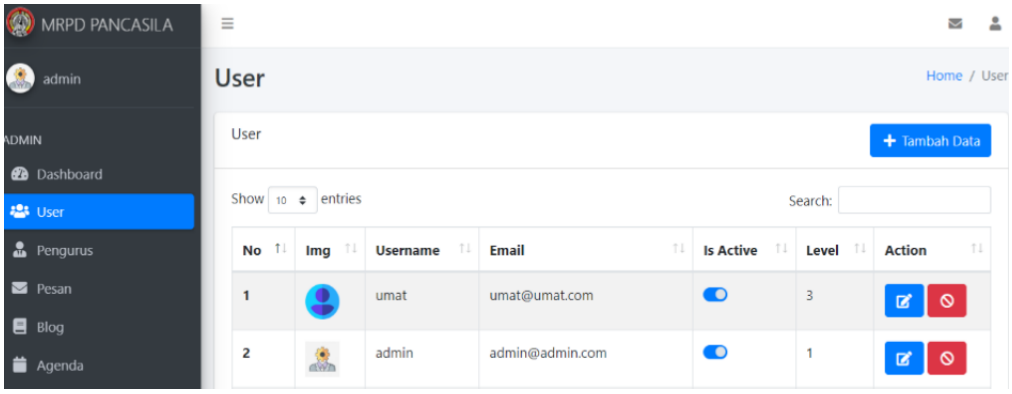
Pada uji skenario ini melakukan uji coba halaman login disimpulkan bahwa pada halaman tersebut terdapat *looping*/percabangan pada sistem *login* didapat dari hasil *Cyclomatic Complexity* dan *Graph Matiks* mendapatkan nilai 2 (dua) hal ini dapat diamati dari hasil pengujian dengan jalur *path* yang diterima dengan hasil *Cyclomatic Complexity*. Hasil ini menunjukan bahwa pengujian pada halaman *login* sudah benar dan berfungsi dengan baik.

Langkah pertama dengan memasukan *email* administrator dan *password* administrator yang belum terdaftarkan maka muncul suatu pesan *Email* tidak di temukan (Gambar 16) dan kemudian memasukan data yang sudah terdaftar dengan benar.



Gambar 16. Menu *Login* dengan *Email* Tidak Ditemukan

Langkah kedua dengan memasukan *email* dan *password* yang sudah terdaftar sebelumnya, maka akan tampil halaman Administrator (Gambar 17).



Gambar 17. Halaman Administrator

Dari uji skenario di atas dapat di simpulkan bahwa saat melakukan beberapa tes dengan memasukan *email* "admin@admin.com" dan *password* "adminaja" yang belum terdaftar maka muncul suatu pesan "*Email* tidak ditemukan". Sedangkan apabila memasukan *username* "glovenia1111@gmail.com" dan *password* "adminaja" yang sebelumnya sudah terdaftar, maka kemunculan pesan *login* gagal akan di lewati dan masuk ke halaman administrator.

f. Kesimpulan *Pengujian*

Tahapan yang ada pada pengujian ini, terdapat beberapa tahapan yang seperti tahap pemetaan *source code*, pembuatan *flow graph*, penghitungan *cyclomatic complexity*, *independent path*, pembuatan *graph matriks* serta melakukan skenario uji sesuai dengan yang sudah ditentukan. Pada tahap *cyclomatic complexity* dan *independent path* menghasilkan nilai yang sama yaitu 2 (dua). Hal ini menunjukkan bahwa alur dan *source code* pemrograman *website* MRPD sesuai logika sehingga sistem dapat melakuakan sistem layanan kepada umat.

IV. KESIMPULAN

Hasil dari penelitian adalah sebuah sistem yang dikembangkan menjadi sistem layanan berteknologi *Web Service* berbasis *website*. *Website* ini dilengkapi interaksi sistem dengan umat berupa pesan *Email* menggunakan *PHPmailer* dalam pengisian data oleh umat, membuat pendataan Komunitas agar lebih terstruktur sehingga sistem informasi tersampaikan secara efektif ke para umat. Jadi dapat disimpulkan bahwa, pengembangan *Web Service* pada Sistem Layanan Gereja MRPD dapat mengatasi masalah yang dihadapi, terutama dalam sistem layanan dan administrasi umat. Adanya teknologi *payment gateway* sebagai sistem persembahan Gereja mempermudah umat untuk melakukan transaksi persembahan dan sumbangan secara *online*. Kelemahan dari sistem *web service* ini ialah masih belum ada fitur transaksi dari *web service* langsung. Sehingga transaksi lain menggunakan teknologi dari *midtrans*. Kekurangan dari *website* yang dikembangkan adalah tidak memiliki fitur Penjadwalan Pastor dan tidak ada modul Ketua Lingkungan. Selain pendataan yang tertera di bab – bab sebelumnya, terdapat pendataan – pendataan yang tidak bisa dilepas tangankan oleh pihak Gereja atas privasi pihak tertentu.

REFERENSI

[1] Wahab, S.R., Dan Sihombing, E.D.C., Penerapan *Framework Model-View Controller* (MVC) Pada Sistem Informasi Manajemen Data Jemaat Berbasis *Web* (Studi Kasus Gki Maranatha Kampung Harapan), *Journal Of Information System, Applied, Management, Accounting And Research*, 2021, Vol. 5, No. 1, Pp. 152-160.
[2] Rosa, A.S., Dan Shalahuddin, M., Rekayasa Perangkat Lunak Terstruktur Dan Berorientasi Objek, Edisi Revisi, Informatika, Bandung, 2018.
[3] Sari, D. W., Kosasi, S., Gat, G., David, D., Dan Yuliani, I. D. A. E., Pemanfaatan *Restful Web Services* Pada Perangkat Lunak Penyewaan Lapangan Badminton, *Infosys* (*Information System*) *Journal*, 2022, Vol. 6, No. 2, Pp. 103-114.
[4] Wiley., *Torus 1–Toward An Open Resource Using Services Cloud Computing For Environmental Data*, London, 2020.
[5] Yellavula, N., *Hands-On Restful Web Services With Go Second Edition Developelegant Restful Apis With Golang For Microservices And The Cloud, Packt Publishing* Ltd, Amerika, 2020.
[6] Halim, Y., Kosasi, S., Wijaya, T., Dan Kuway, S. M., *Self-Service Technology* Berbasis *Android* Menggunakan *Restful Web Service* Pada Bisnis Restoran, *Journal Of Applied Computer Science And Technology*, 2021, Vol. 2, No. 2, Pp. 73-82.
[7] Pressman, R.S., dan Maxim, B.R., 2020, *Software Engineering: A Practitioner's Approach.Ninth Edition, McGraw-Hill Education, New York.*
[8] Wijaya, T., 2017, Penerapan *Service-Oriented Architecture* Pada Sistem Informasi Eksepdisi, *Eksplora Informatika*, Vol. 6, No. 2, pp. 190-197.
[9] Nugroho, S., Dan Primadewi, A., Penerapan *Web Service* Untuk Integrasi Data Simperpus Dan Siak, Jurnal Komtika (Komputasi Dan Informatika), 2020, Vol. 4, No. 2, Pp. 71-81.
[10] Patni, S., 2017, *Pro RESTful APIs Design, Build and Integrate with REST, JSON, XML and JAX-RS, Apress*: California.
[11] Wijanarko, A., Dan Ngafifuddin, D., *Web Service* Untuk Integrasi Data Dalam Pengelolaan Data Potensi Prestasi Mahasiswa Stmik Amikom Purwokerto, Teknikom, 2017, Vol. 1, No. 2, Pp. 51-58.
[12] Petrov, A., *Database Internals A Deep Dive Into How Distributed Data Systems Work*. O'reilly Media, Inc., Amerika, 2019.
[13] Utomo, S.P., Primadewi, A., Hanafi, M., Sani, Z.A., Dan Alfiyah, N.H., Perancangan *Restful Web Service* Pada Sistem Informasi Terintegrasi Menggunakan *Framework Codeigniter*. Seminar Nasional Dinamika Informatika 2020 Universitas Pgri Yogyakarta, 2020, Vol. 4, No. 1, Pp. 124-128.
[14] Alip, A., Kosasi, S., Yuliani, I. D. A. E., Syarifudin, G., Dan David, D., Implementasi Arsitektur *Model View Controller* Pada *Website* Toko *Online*, Jurnal *Bumigora Information Technology* (Bite), 2021, Vol. 3, No. 2, Pp. 135-150.

[15] Raharjo, B., Belajar Otodidak *Framework Codeigniter*, Informatika Bandung, 2018.
[16] Gunawan, E., Dan Kosasi, S., Perancangan Perangkat Lunak Persediaan Berbasis *Web* Menggunakan Django Pada Toko Sumber Baru, E-Jurnal Jusiti: Jurnal Sistem Informasi Dan Teknologi Informasi, 2022, Vol. 11, No. 1, Pp. 13-23.
[17] Tohir, A.S., Pemodelan Sistem Data Terdistribusi Untuk Mengintegrasikan Data Akademik Dan Keuangan, Jurnal Intensif, 2017, Vol. 1, No. 1, Pp. 44-52.
[18] Petrov, A., *Database Internals A Deep Dive Into How Distributed Data Systems Work*. O'reilly Media, Inc., Amerika, 2019.
[19] Staron, M., *Action Research In Software Engineering Theory And Applications, Springer Nature Switzerland* Ag, Amerika, 2020.