Penerapan *Web Service* untuk Integrasi Data Simperpus dan SIAK

Setiya Nugroho[1*], Ardhin Primadewi[2]
[1,2]Teknik Informatika, Universitas Muhammadiyah Magelang
*email: setiya@ummgl.ac.id

DOI: https://doi.org/10.31603/komtika.v4i2.4225

Received: 23-11-2020, Revised: 03-01- 2021, Accepted:21-01- 2021

ABSTRACT

Library Information Systems and Academic Information Systems that are not integrated with each other have resulted in duplication and inconsistency of data. This study aims to integrate database by implementing web service in both systems. Simperpus uses the Rest Client and SIAK uses the Rest Server. Exchange of data between the server utilizes JSON. This research uses UML as a system design modeling with several diagrams, namely use case diagrams, class diagrams, sequence diagrams, activity diagrams, component diagrams, and deployment diagrams. Writing programming language in the study is Codeigniter framework with MySQL database. The test results showed that the process of sending data in JSON format from SIAK to Simperpus obtained a response time of 137 ms.

Keywords: *Web Service, Rest API, Rest Server, Rest Client, JSON, framework CodeIgniter, Postman*

ABSTRAK

Sistem Informasi Perpustakaan dan Sistem Informaasi Akademik yang tidak saling terintegrasi menyebabkan adanya duplikasi dan ketidakkonsistensian data. Penelitian ini bertujuan melakukan integrasi database dengan menerapkan web serice pada kedua sistem. Simperpus menggunakan Rest Client dan SIAK memakai Rest Server. Pertukaran data antar server memanfaatkan JSON. Perancangan sistem menggunakan UML dengan beberapa diagram, yaitu use case diagram, class diagram, sequence diagram, activity diagram, component diagram, dan deployment diagram. Penulisan bahasa pemrograman menggunakan framework Codeigniter dengan database MySQL. Hasil pengujian menunjukkan proses pengiriman data dengan format JSON dari SIAK ke Simperpus diperoleh respon time 137 ms.

Kata-kata kunci: Web Service, Rest API, Rest Server, Rest Client, JSON, framework CodeIgniter, Postman

PENDAHULUAN

Saat ini kebutuhan sistem informasi sangat penting bagi sebuah instansi. Suatu sistem informasi yang makin besar dan kompleks maka diperlukan perhatian yang besar terhadap kebutuhan pengolahan dan integrasi data. Proses bisnis yang berubah seiring kebutuhan organisasi sehingga diperlukan pengembangan sistem dan aplikasi yang ada[1]. Kebutuhan sistem informasi setiap instansi ada kalanya lebih dari satu tergantung dari jumlah unit pada instansi tersebut. Suatu intansi yang tidak memiliki banyak departemen tetapi membuat lebih dari satu sistem informasi disebabkan kebutuhan proses bisnisnya.

Sistem informasi yang berbeda pada suatu instansi akan memiliki kondisi lingkungan sistem yang berbeda. Perbedaan kondisi tersebut diantaranya adalah pada sistem operasi, basis data, dan bahasa pemrograman yang digunakan. Perbedaan kondisi itu akan menjadi masalah ketika sistem informasi yang satu membutuhkan data dari sistem informasi yang lain. Data yang sama tetapi disimpan pada dua atau lebih sistem informasi juga akan menimbulkan masalah karena akan muncul duplikasi data dengan *record* yang berbeda atau akan menjadi sebab ketidakkonsistensian sebuah data.

Terdapat permasalahan pada sistem informasi perpustakaan (Simperpus) Universitas Muhammadiyah Magelang. Petugas melakukan input data mahasiswa sebagai anggota ketika yang bersangkutan pertama kali meminjam buku. Hal ini akan menimbulkan duplikasi data mahasiswa pada skala universitas karena sudah ada data mahasiswa pada Sistem Informasi Akademik (SIAK). Duplikasi data mahasiswa ini akan menyebabkan ketidakkonsistensian data mahasiswa seperti nama lengkap dan nomor pokok mahasiswa.

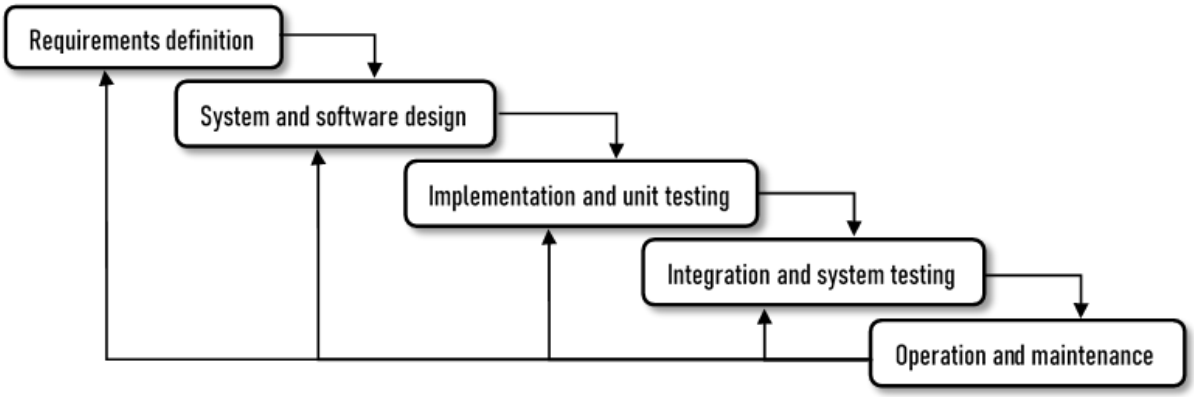
Untuk mengatasi permasalahan tersebut, pada penelitian ini dikembangkan sistem informasi yang terintegrasi antara Simperpus dan SIAK dengan menerapkan *web service*. Implementasi w*eb service* memberikan kemudahan dalam perpaduan fungsi untuk pengembangan program aplikasi tanpa bergantung pada sistem operasi, bahasa pemrograman, atau basis data yang sama karena *web service* berkomunikasi menggunakan sebuah standar format data yang *universal* yaitu *XML* dan *JSON* [2].

Terdapat tiga metode *web service* yang telah distandarisasi konsorsium web dalam melakukan pertukaran informasi yaitu *Extensible Markup Language Remote Procedure Call* (XML-RPC), *Simple Object Access Protocol* (SOAP), dan *Representational State Transfer* (REST). Pada beberapa kondisi pengujian, REST mendapat hasil nilai *latency* yang paling kecil [3]. Format pertukaran data pada *web service* yang sering digunakan ada dua yaitu XML dan *Javascript Object Notation* (JSON). Keunggulan JSON dibanding XML diantaranya ukuran karakter yang dibutuhkan JSON lebih kecil, proses *browser parsing* JSON lebih sederhana, dan pertukaran data JSON lebih unggul dibandingkan XML [4].

Model interoperabilitas antara sistem perpustakaan dan sistem akademik menggunakan web services menggunakan teknologi *SOAP*, *XML*, *IIS* dan *ASP.NET*. telah dilakukan[5]. Selain itu, juga telah dilakukan teknologi *XML-RPC* pada sistem perpustakaan dan sistem akademik tetapi tidak dijelaskan pertukaran data antar server[6]. Java dan *SOAP* telah diimplementasikan di sisi server, sedangkan di client menggunakan bahasa C#[7]. Teknologi *RESTful*, *MongoDB*, dan *Node.js*, juga telah diterapkan namun tidak ada pembahasan pemrograman pada *backend* [8]. Aspx.net, SQL Server, IIS Server, dan web service juga telah diterapkan pada SIMKA dan SIATMA, tetapi tidak ada pembahasan proses pertukaran datanya[9].Penelitian juga telah dilakukan pada rest server pada perpustakaan dan rest client pada Akademik dengan JSON sebagai pertukaran data juga telah dilakukan[10] serta penerapan web service pada sistem informasi perpustakaan menggunakan teknologi web service nya [11]. Selain itu beberapa penelitian yang menerapkan web service pada server perpustakaan juga telah dilakukan namun tidak ada pembahasan pertukaran data yang *universal* antar server.[7]–[9], [11]–[14] . Dalam penelitian ini dilakukan penerapan Rest server di SIAK dan Rest client di simperpus dengan format pertukaran datanya adalah JSON.

METODE

Penelitian ini menggunakan metode *Waterfall* sebagai pemodelan untuk merancang aplikasi Sistem Informasi Terintegrasi antara Sistem Perpustakaan dan Sistem Akademik. Tahapan dalam metode *Waterfall* untuk model pengembangan sistem informasi disajikan secara sistematik dan sekuensial seperti pada Gambar 1 [15].



Gambar 1. Metode Waterfall

Pada gambar 1 terlihat terdapat lima tahapan pada metode *Waterfall*. Pada tahap awal dari metode ini adalah *requirements definition* yang merupakan tahap identifikasi kebutuhan sistem, kendala, dan tujuan yang selanjutnya didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem. Tahap kedua adalah *system and software design* yang merupakan tahap perancangan sistem baik untuk alokasi kebutuhan perangkat keras maupun perangkat lunak yang dirancang secara keseluruhan dalam suatu arsitektur. Tahap ketiga adalah *implementation and unit testing* yang merupakan tahap implementasi ke dalam serangkaian program. Pada tahap pengujian dilakukan verifikasi setiap unit untuk memenuhi spesifikasi yang telah dirancang. Tahap keempat adalah *integration and system testing* yang merupakan integrasi program untuk diuji sebagai sebuah sistem yang lengkap. Tahap kelima adalah *Operation and maintenance* yaitu tahap saat sistem dipasang dan digunakan secara nyata. *Maintenance* merupakan kegiatan pembetulan kesalahan yang tidak ditemukan pada tahapan-tahapan sebelumnya untuk meningkatkan layanan sistem. [15]

Pada tahapan model waterfall, alur perangkat lunak digambarkan menggunakan *tools Unified Modelling Language* (*UML*). *UML* telah menjadi standar untuk pemodelan perangkat lunak. *UML* sering digunakan untuk memvisualisasikan, mengkomunikasikan dan memahami struktur dan perilaku suatu sistem[16]. Melalui *UML* maka dapat diperoleh informasi mengenai ilustrasi dan pemetaan secara visual serta model desain dan struktur sistem perangkat lunak yang disajikan melalui diagram. [17]

HASIL DAN PEMBAHASAN

Requirements definition

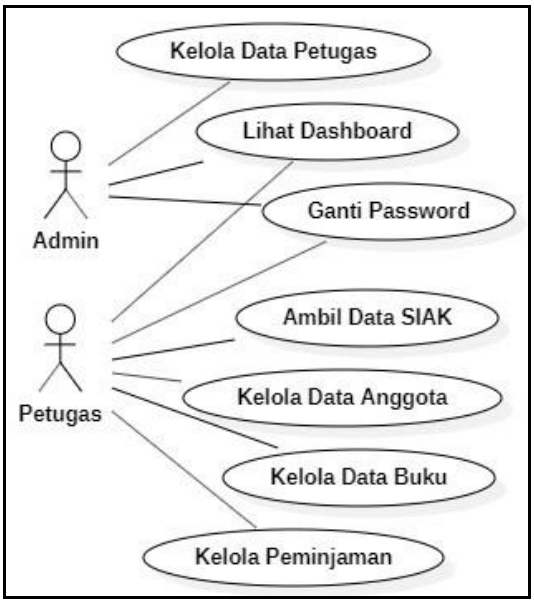
Terdapat permasalahan yang terjadi pada Simperpus yaitu petugas yang melakukan input data mahasiswa sebagai anggota pada Simperpus secara manual menyebabkan adanya duplikasi data mahasiswa pada skala universitas karena sudah ada data mahasiswa di SIAK. Duplikasi data mahasiswa ini akan menyebabkan ketidakkonsistensian data mahasiswa seperti nama lengkap dan nomor pokok mahasiswa.

Data yang dibutuhkan dalam desain Simperpus ini diantaranya adalah data mahasiswa, petugas perpustakaan, administrator, buku, peminjaman buku. Kebutuhan fungsional pada

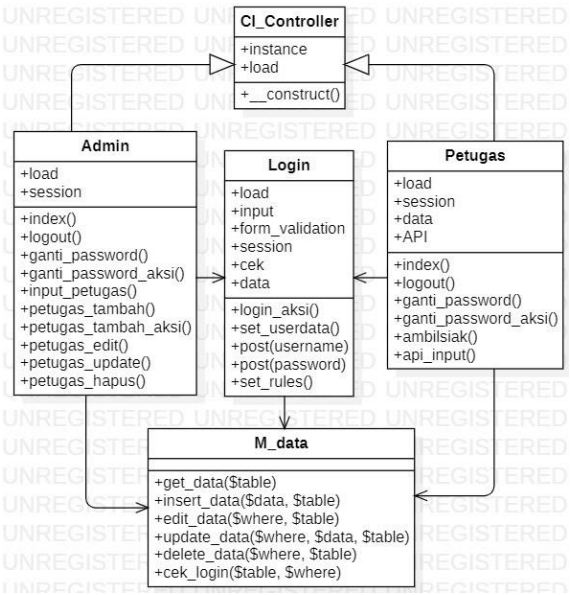
sistem ini diantaranya adalah proses login untuk admin dan petugas, ganti password admin, pengelolaan data petugas, pengambilan data mahasiswa dari SIAK,

System and software design

Penelitian ini menggunakan beberapa diagram diantaranya *use case diagram*, *class diagram*, *sequence diagram*, *activity diagram*, *component diagram*, dan *deployment diagram*. *Use case diagram* adalah metodologi yang digunakan dalam analisis sistem untuk mengidentifikasi, memperjelas, dan mengatur persyaratan sistem.[18]. *Use case diagram* pada penelitian ini menggunakan dua aktor yaitu Admin dan Petugas. Aktor Admin mempunyai *use case* Kelola Data Petugas, Lihat Dashboard, dan Ganti Password. Akor Petugas memiliki *use case* Ambil Data SIAK, Kelola Data Anggota, Kelola Data Buku, dan Kelola Peminjaman. *Use case diagram* pada penelitian ini terlihat seperti pada Gambar 2.



Gambar 2. *Use Case* Diagram Aplikasi Simperpus



Gambar 3. *Class Diagram* Aplikasi Simperpus

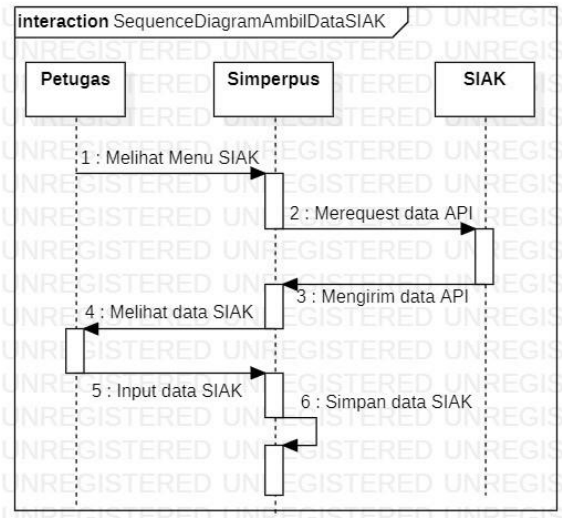
Class diagrams UML menunjukkan kelas-kelas dari suatu sistem, keterkaitannya, serta operasi dan atribut kelas[19]. Dalam penelitian ini menggunakan empat *class*, yaitu *class* Admin, CI_Controller, Login, Petugas, dan M_data. *Class* Admin dan Petugas merupakan turunan dari *class* CI_Controller. *Class* Admin, Petugas, dan Login melakukan hubungan asosiasi secara langsung ke *class M_data*. *Class diagram* aplikasi Simperpus terlihat pada Gambar 3.

Setiap *class* memiliki atribut dan operasi masing-masing. *Class* Login memiliki atribut *load*, *input*, *form_validation*, *session*, *cek*, *data*. *Class* Login juga mempunyai operasi *post (username)*, *set_userdata()*, *post(password)*, *login_aksi()*, *set_rules()*. Atribut dan operasi pada class yang lain bisa dilihat pada gambar 3.

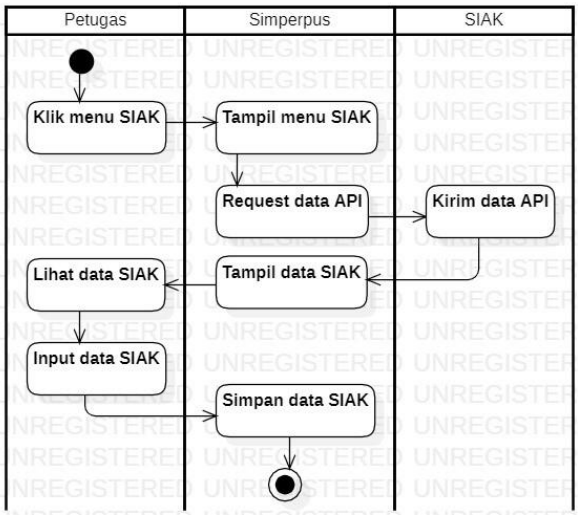
Sequence diagram menjelaskan perilaku dalam sistem, mengilustrasikan bagaimana objek berinteraksi dengan objek lainnya. Untuk satu *use case* hanya diperlukan satu *sequence* diagram[20]. Pada penelitian ini, *sequence diagram* yang dibuat hanya Ambil Data SIAK karena fokus penelitian pada proses integrasi data dari sistem SIAK ke sistem perpustakaan. *Sequence diagram* untuk Ambil Data SIAK dimulai dari Petugas melihat menu SIAK pada halaman Simperpus. Simperpus kemudian melakukan request data *API* ke SIAK yang selanjutnya mengirim data *API* hasil request ke Simperpus. Petugas dapat melihat data SIAK yang telah dikirim oleh SIAK dan dapat menginputkan data yang berelasi dengan data *API* ke

SIAK. Terakhir Simperpus akan menyimpan data hasil input petugas ke *database*. *Sequence diagram* pada penelitian ini terlihat seperti pada Gambar 4.

Activity Diagram pada dasarnya adalah diagram alir untuk merepresentasikan arus dari satu aktivitas ke aktivitas lainnya[19]. *Activity Diagram* untuk Ambil Data SIAK dimulai dari Petugas melakukan Klik menu SIAK, kemudian diikuti Simperpus melakukan Tampil menu SIAK. Berikutnya Simperpus melakukan r*equest* data *API* untuk meminta data *API* dari SIAK dan setelah menerima request data *API* akan melakukan Kirim data *API* ke *server* Simperpus. Simperpus melakukan Tampil data SIAK setelah mendapat data *API*. Petugas dapat melihat data SIAK kemudian melakukan input data ke *server* SIAK dengan data yang ada relasi dengan data SIAK. Terakhir Simperpus akan menyimpan data hasil input petugas ke *server* SIAK. *Activity diagram* pada penelitian ini terlihat seperti pada Gambar 5.



Gambar 4. *Sequence Diagram* Ambil Data SIAK

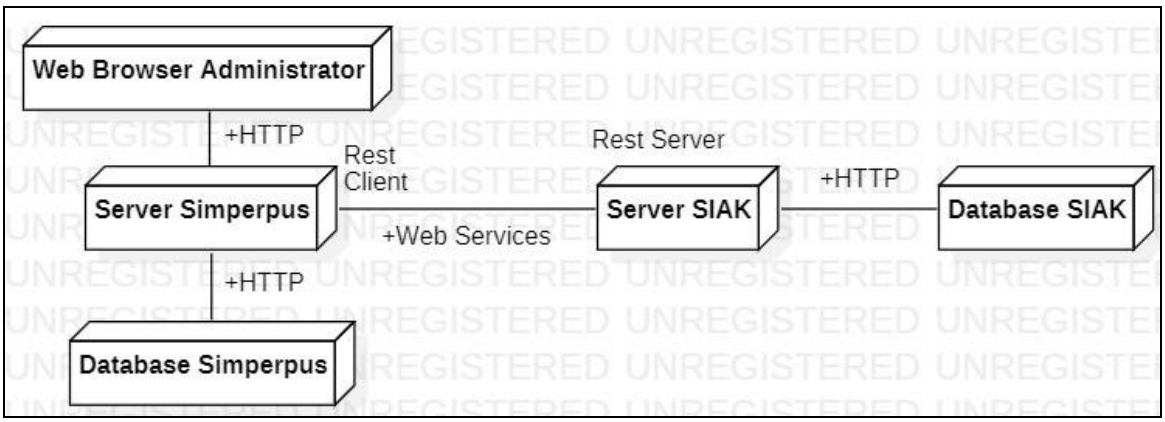


Gambar 5. *Activity Diagram* Ambil Data SIAK

Implementation and unit testing

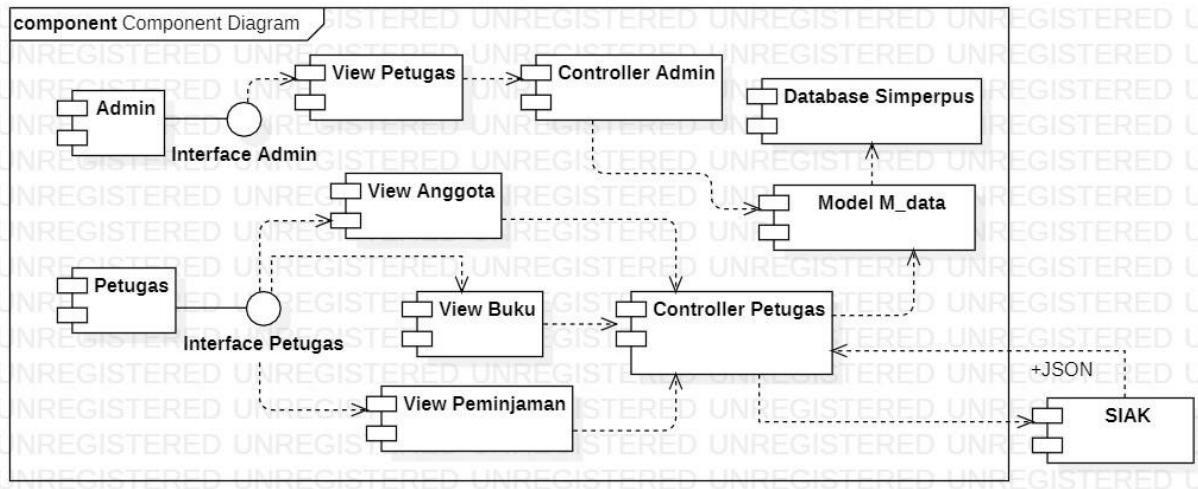
Implementasi aplikasi Simmperpus pada penelitian ini dibuat berdasar *Component diagram* dan *Deployment diagram*. *Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan[19]. *Deployment diagram* menunjukkan perangkat keras untuk sistem, perangkat lunak yang diinstal pada perangkat keras tersebut dan *middleware* yang digunakan untuk menghubungkan mesin yang berbeda satu sama lain. [19].

Deployment diagram pada penelitian ini seperti terlihat pada Gambar 6. Pengguna dapat mengakses server Simperpus menggunakan web browser pada sistem Simperpus halaman Administrator. Server Simperpus terkoneksi ke *database* Simperpus untuk menyimpan dan mengambil data. *Server* Simperpus juga dapat melakukan koneksi ke *server* SIAK melalui *web services* menggunakan teknologi *Rest API*. Simperpus dan SIAK menggunakan *database server MySQL* dan *web server Apache*.



Gambar 6. *Deployment* D*iagram* Aplikasi Simperpus

Component diagram pada penelitian ini terlihat pada Gambar 7. Pada bagian server Simperpus terdapat aktor Admin dan Petugas yang berinteraksi menggunakan masing-masing interface. Admin melalui interface Admin yaitu *View* Petugas dapat berinteraksi dengan sistem. *View* Petugas akan dikontrol oleh *Controller* Admin. Petugas memiliki *Interface* Petugas sebagai penghubung dengan *View* Anggota, View Buku, dan *View* Peminjaman yang dikontrol oleh *Controller* Petugas. Untuk mengakses *Database* Simperpus, *Controller* Admin dan *Controller* Petugas mengaturnya melalui *Model* M_data. *Controller* Petugas mengatur koneksi dengan *server* SIAK dengan cara bertukar data menggunakan format data *JSON*. SIAK akan merubah data dari *database* SIAK menjadi format *JSON* sebelum dikirim ke *server* Simperpus.



Gambar 7. *Component* D*iagram* Aplikasi Simperpus

Gambar 8 dan 9 adalah potongan *script* untuk *Rest API server* pada aplikasi Simperpus dengan *class* Biodata. Pada Gambar 8 menunjukkan *class* Biodata menggunakan *Restserver* pada *libraries* di file *REST_Controller*. *Class* Biodata merupakan turunan yang mewarisi *class REST_Controller*. Fungsi *__construct* pada *class* Biodata akan menyimpan data pada variabel *config* dan memberi nilai *rest*. Fungsi *__construct* juga akan memanggil model dengan nama *classs* Biodata_model dengan nilai biodata. Selama fungsi __desctruct belum ada maka nilai dari fungsi dan atribut akan tetap ada.

```
4   use Restserver\Libraries\REST_Controller;
5
6   class Biodata extends REST_Controller {
7       function __construct($config = 'rest') {
8           parent::__construct($config);
9           $this->load->model('Biodata_model','biodata');
10      }
```

Gambar 8. Potongan S*cript* C*lass* Biodata pada *Rest Server*

Gambar 9 menunjukkan fungsi *index_get* pada *class Biodata*. Variabel *id* menyimpan data dari fungsi *get* dengan nilai *id*. Nilai *id* yang dimaksud adalah sebuah kolom pada tabel *user* yang ada di *database* SIAK. Sintak ini dihubungkan dengan *class Biodata_model* dengan fungsi *getBiodata*. Jika variabel *id* bernilai *null* atau kosong yang berarti tidak ada data pada kolom *id* maka tidak ada data yang disimpan pada variabel *biodata*. Selain itu, jika ada data maka *class Biodata* akan menggunakan fungsi *getBiodata* pada *class Biodata_model* dengan nilai id yang datanya akan disimpan pada variabel *biodata*. Data pada variabel *biodata* inilah yang akan dikirimkan oleh *Rest_Controller* ke server lain yang menggunakan *rest client*.

```
12      //Menampilkan data anggota
13      function index_get() {
14          $id = $this->get('id');
15          if ($id === null) {
16              $biodata = $this->biodata->getBiodata();
17          } else {
18              $biodata = $this->biodata->getBiodata($id);
19          }
20          $this->response($biodata, REST_Controller::HTTP_OK);
```

Gambar 9. Potongan S*cript* Fungsi *index_get* pada *Rest Server*

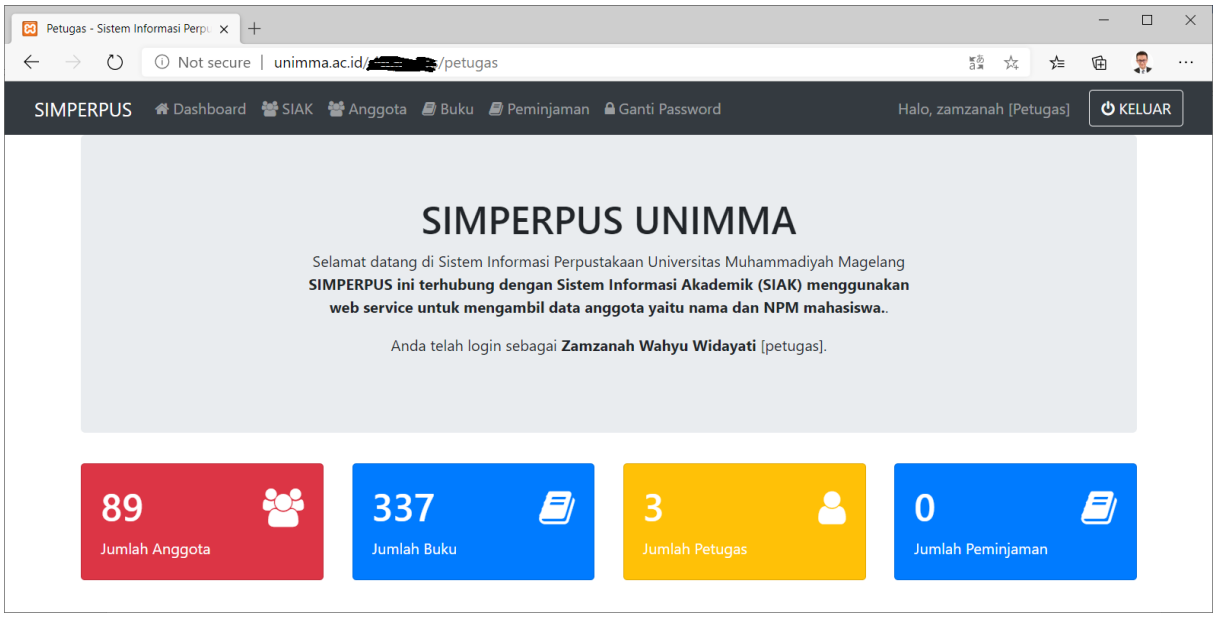
Gambar 10 menunjukkan potongan *script* untuk *Rest API client* pada aplikasi Simperpus dengan *class* Petugas. Pada baris 120 didefinisikan sebuah variabel dengan nama *API*. *Class* Petugas mempunyai fungsi dengan nama ambilsiak. Fungsi ambilsiak memiliki *atribut API* dengan nilai alamat server SIAK. Fungsi ambilsiak juga mengambil *library* pada *curl*. Data *API* yang dikirim oleh server SIAK dalam bentuk *json* telah melalui proses *encode*. Untuk membaca data tersebut maka fungsi ambilsiak akan menggunakan *json_decode*, kemudian datanya disimpan pada variabel data dengan nilai databiodata. Data pada variabel data ini akan dikirim ke *view* dengan nama *v_siak*. Data yang berasal dari database SIAK tersebut bisa memiliki lebih dari satu *record*, sehingga v_siak akan menggunkan perulangan untuk menampilkan datanya. Perulangan yang digunakan *v_siak* adalah *foreach* dengan beberapa tabel yang diperlukan untuk ditampilkan.

```
119     // Rest API Client
120     var $API ="";
121     function ambilsiak(){
122         $this->API="http://unimma.ac.id/[redacted]/index.php/";
123         $this->load->library('curl');
124         $data['databiodata'] = json_decode($this->curl->simple_get($this->API.'/biodata'));
```

Gambar 10. Potongan *script Rest API Client*

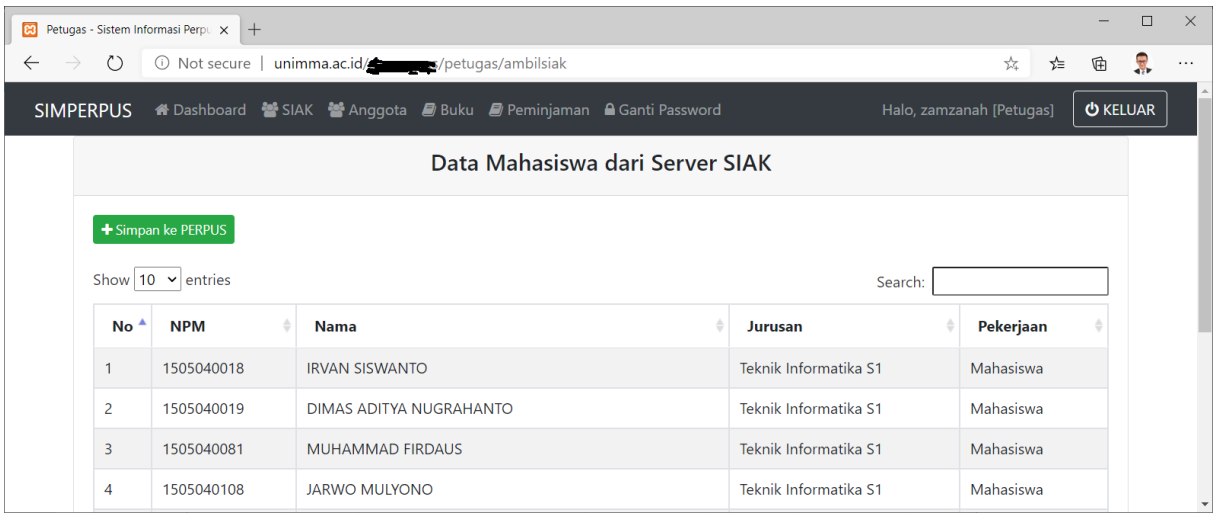
Integration and system testing

Halaman *dashboard* aplikasi Simperpus untuk petugas terlihat seperti pada Gambar 11. Pada halaman *dashboard* ini terdapat informasi jumlah orang yang telah terdaftar sebagai anggota di perpustakaan, jumlah buku yang dapat dipinjam anggota, jumlah petugas yang akan melayani peminjaman, serta jumlah transaksi peminjaman. Aplikasi ini dibuat menggunakan *framework CodeIgniter* versi 3.11.



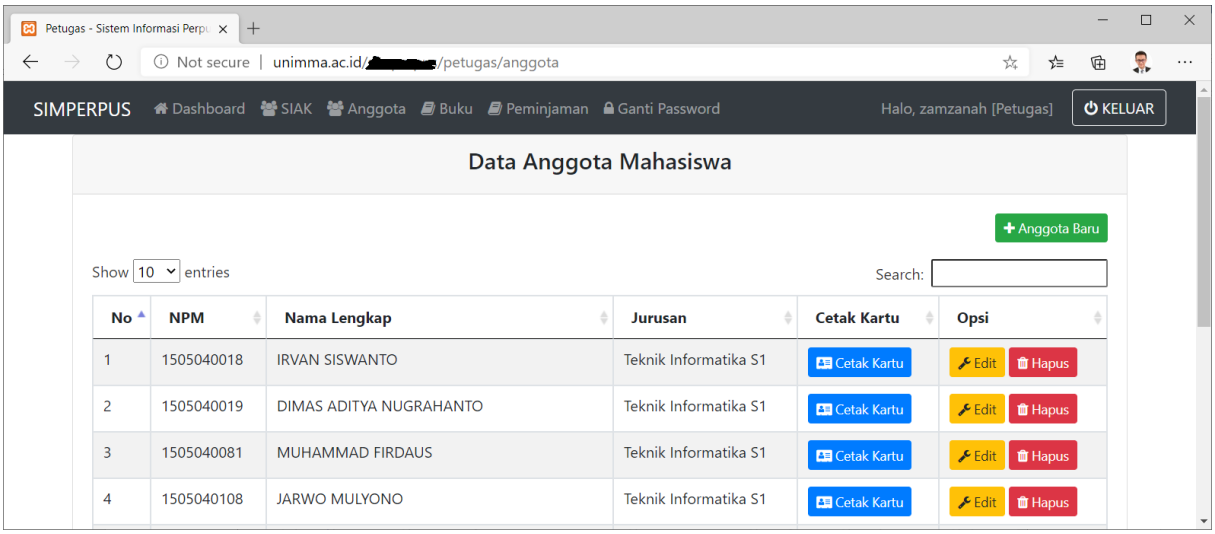
Gambar 11. Halaman *dashboard* Simperpus

Tampilan v_siak adalah seperti pada gambar 12. Data tampilan v_siak berasal dari *database* pada *server* SIAK yang dikirim menggunakan *Rest API* dalam bentuk file *json*. Data yang diminta oleh *v_siak* hanya nama, npm, jurusan dan perkerjaan dari mahasiswa. Data tersebut ditampilkan menggunakan perulangan sehingga semua data pada tabel yang bersangkutan akan ditampilkan semua. Data yang ditampilkan tersebut bisa disimpan ke *server* Simperpus dengan tombol Simpan ke PERPUS pada tampilan *v_siak*.



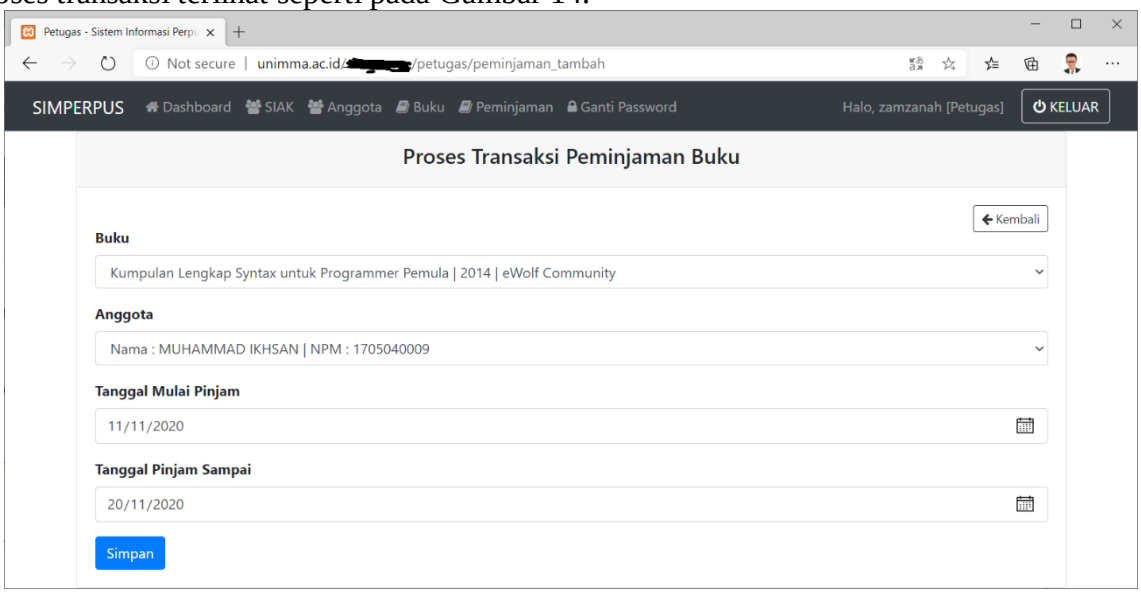
Gambar 12. Tampilan Data Mahasiswa dari *server* SIAK

Gambar 13 menunjukkan tampilan data dari mahasiswa yang telah disimpan di *server* Simperpus. Data tersebut diperoleh setelah petugas menekan tombol simpan ke PERPUS pada tampilan *v_siak*. Proses ini dikendalikan oleh *class* Petugas dengan fungsi *api_iput*. Fungsi *api_input* memanggil *library* pada *curl* yang digunakan untuk melakukan proses *decode* file *json* yang dikirimkan. Fungsi api_input memanggil *trans_begin* kemudian melakukan perulangan menggunakan *foreach* untuk menyimpan data ke *database* pada *server* Simperpus. Data tersebut kemudian ditampilan oleh v_anggota.



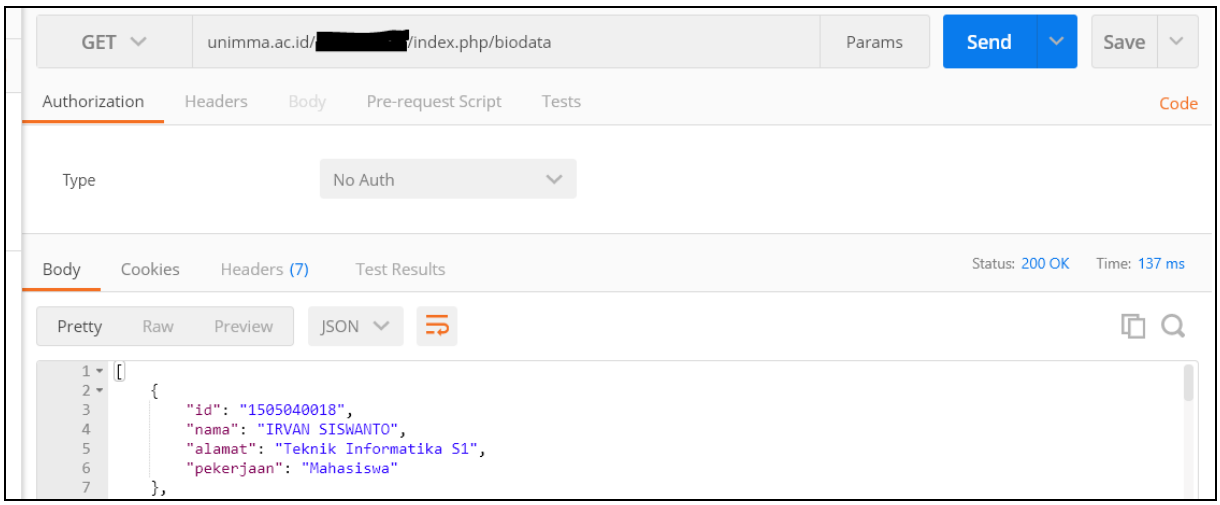
Gambar 13. Tampilan Data Mahasiswa dari *Server* Simperpus

Data mahasiswa yang diperoleh dari *server* SIAK, digunakan untuk proses transaksi pada Simperpus. Proses transaksi tersebut dikerjakan oleh fungsi peminjaman_tambah yang terdapat pada *class* Petugas. Fungsi tersebut memanggil *model* M_data yang digunakan untuk melakukan koneksi ke *database* dengan tabel *m_biodata_mhs*, dan *m_data_buku*. Pada proses transaksi terdapat *form dropdown list* data buku yang mengambil data dari m_*data_buku* dan, *form dropdown list* data mahasiswa yang mengambi data dari *M_biodata_mhs*. Tampilan proses transaksi terlihat seperti pada Gambar 14.



Gambar 14. Tampilan proses transaksi peminjaman pada Simperpus

Pengujian *Rest API* pada penelitian ini menggunakan aplikasi *Postman*. *Postman* dapat mendukung beberapa *parameter*, diantaranya *get*, *post*, *put*, *delete*. *Parameter* digunakan pada penelitian ini adalah *get*. *Parameter get* ini berfungsi untuk menampilkan beberapa data mahasiswa dari *database SIAK*. Jika data tersebut ditampilkan pada aplikasi *Postman* dengan format *json* maka *Rest API Sever* yang dibangun pada server SIAK sudah berhasil. Jika dilihat pada gambar 15, *respon time* yang didapat untuk menampilkan data adalah 137 ms.



Gambar 15. Pengujian *Rest API Server* mengggunakan *Postman*

Operation and maintenance

Operasionalisasi dirancang agar dapat digunakan di perpustakaan dan biro akademik Universitas Muhammadiyah Magelang, sedangkan pemeliharaan sistem dirancang dengan dilakukannya pemeriksaan periodik terhadap data pada aplikasi.

KESIMPULAN

Intergrasi data yang dibangun menggunakan *Rest API* didapat hasil pengujian respon time pertukaran data sebesar 137ms. Format pertukaran data yang digunakan adalah *JSON*. Penelitian ini menerapkan *Rest Server* pada SIAK, dan *Rest Client* pada Simperpus. Sistem lain dapat menggunakan data mahasiswa pada SIAK dengan memanfaatkan *API* yang terdapat pada *Rest Server*.

DAFTAR PUSTAKA

[1] M. A. Arianto, S. Munir, and K. Khotimah, "Analisis dan Perancangan Representational State Transfer (REST) Web Service Sistem Informasi Akademik STT Terpadu Nurul Fikri Menggunakan YII Framework," *J. Teknol. Terpadu*, vol. 2, no. 2, 2016.

[2] A. AyuningTyas and A. Ashari, "Pemanfaatan Teknologi Web Service Untuk Integrasi Sistem Layanan Materi Pelajaran Terdistribusi," *J. ANGKASA*, vol. VII, no. 2, 2016.

[3] Y. Yogiswara, W. Wijono, and H. Dahlan, "Kinerja Web Service Pada Proses Integrasi Data," *J. EECCIS*, vol. 1, no. 1, 2014.

[4] R. Somya and T. M. E. Nathanael, "Pengembangan Sistem Informasi Pelatihan Berbasis Web Menggunakan Teknologi Web Service Dan Framework Laravel," *J. Techno Nusa Mandiri*, vol. 16, no. 1, 2019, doi: 10.33480/techno.v16i1.164.

[5] N. Hermanto and W. W. Winarno, "Model Interoperabilitas Antara Sistem Akademik Dan Sistem Perpustakaan Menggunakan Web Services (Studi Kasus: STMIK Amikom Purwokerto)," vol. 10, no. 1, pp. 108–116, 2017.

[6] M. Nasir, "Sinkronisasi Data User Antara Sistem Informasi Perpustakaan Dengan Sistem Informasi Akademik," in *semnasIF*, 2012, pp. 168–174.

[7] U. S. Ratulangi, B. A. Kasaedja, R. Sengkey, and O. A. Lantang, "Rancang Bangun Web Service Perpustakaan Universitas Sam Ratulangi," *J. Tek. Elektro dan Komput.*, vol. 3, no. 3, pp. 38–50, 2014, doi: 10.35793/jtek.3.3.2014.5332.

[8] A. Firdaus, S. Widodo, A. Sutrisman, S. G. F. Nasution, and R. Mardiana, "Rancang Bangun Sistem Informasi Perpustakaan Menggunakan Web Service Pada Jurusan Teknik Komputer Polsri," *J. Inform.*, vol. 5, no. 2407–1730, p. 83, 2019.

[9] A. T. S. Christanto and R. Kurniawati, "Penerapan Service Oriented Architecture Menggunakan Web Service Pada Aplikasi Perpustakaan Berbasis Android," *J. Buana Inform.*, vol. 7, no. 1, pp. 75–82, 2016.

[10] R. Rizal and A. Rahmatulloh, "RESTful Web Service untuk Integrasi Sistem Akademik dan Perpustakaan Universitas Perjuangan," *J. Ilm. Inform.*, vol. 7, no. 1, pp. 54–59, 2019.

[11] Norhikmah, "Penerapan Webservice Pada Sistem Informasi Perpustakaan," *J. Teknol. Inf.*, vol. XIII, no. 3, pp. 17–24, 2018.

[12] J. M. Putera, M. A. Irwansyah, and A. S. Sukamto, "Rancang Bangun Aplikasi Berbasis Android Dengan Penerapan Web Service Pada Sistem Informasi Perpustakaan," *J. Sist. dan Teknol. Inf.*, 2017.

[13] E. Yudhistira, A. Purwinarko, and I. U. Wusqo, "Implementasi Restful Web Service Menggunakan AsyncTask pada Aplikasi Library Automation Berbasis Android," no. Snik, pp. 286–292, 2016.

[14] S. Sibagariang, "Penerapan Web Service Pada Perpustakaan Berbasis Android," *J. Maharjana Inf.*, vol. 1, no. 1, pp. 8–11, 2016.

[15] I. Sommerville, *Software Engineering 10th Edition*. Pearson India, 2018.

[16] T. Ahmad, J. Iqbal, A. Ashraf, D. Truscan, and I. Porres, "Model-based testing using UML activity diagrams: A systematic mapping study," *Computer Science Review*. 2019, doi: 10.1016/j.cosrev.2019.07.001.

[17] S. Lee, "Unified Modeling Language (UML) for Database Systems and Computer Applications," *Int. J. Database Theory Appl.*, vol. 5, no. 1, pp. 157–164, 2012.

[18] A. Y. Aleryani, "Comparative Study between Data Flow Diagram and Use Case Diagram," *Int. J. Sci. Res. Publ.*, 2016.

[19] S. W. Ambler, "UML Activity Diagrams," in *The Elements of UML™ 2.0 Style*, Cambridge University Press, 2005, pp. 113–131.

[20] I. K. Raharjana and A. Justitia, "Pembuatan Model Sequence Diagram Dengan Reverse Engineering Aplikasi Basis Data Pada Smartphone Untuk Menjaga Konsistensi Desain Perangkat Lunak," *JUTI J. Ilm. Teknol. Inf.*, 2015.