IMPLEMENTASI *REST*API*WEBSERVICE* DALAM MEMBANGUN APLIKASI *MULTIPLATFORM* UNTUK MEMAKSIMALKAN PEMESANAN TUKANG

Romi Choirudin[1], Ahmat Adil[2]
[1,2]STMIK Bumigora Mataram
[1]romi@stmikbumigora.ac.id, [2]adilahmat@gmail.com

Abstract

Aplikasi *multiplatform* (lebih dari satu *platform*) dapat memberikan kemudahan kepada pengguna dalam memilih *platform* yang akan digunakan. Karena itu sistem yang dibangun akan menggunakan arsitektur REST Web *Service*. Tukang merupakan orang atau kelompok yang mempunyai kepandaian dalam suatu pekerjaan tangan (dengan alat atau bahan yang tertentu). Seorang tukang dalam mempromosikan jasa secara konvensional membuat kurang efektif dalam meningkatkan pendapatan. Sedangkan bagi para calon pemesan jasa tukang juga sering mengalami masalah dalam melakukan pencarian jasa tukang.. Untuk mengoptimalkan pemesanan jasa tukang,dibutuhkan system yang dapat digunakan untuk media promosi tukang serta dapat berinteraksi antara calon pemesan jasa tukang dengan tukang tersebut. Perancangan system ini dilakukan dengan metodologi penelitian *waterfall*, yaitu metode pengembangan system yang diawali dengan tahap analisa, kemudian dilakukan perancangan system yang dilanjutkan dengan membangun API Web *Service* yang akan menangani *request* dan *respon* data antar *client* dengan *server* sehingga data tetap terintegrasi. Hasil yang ingin dicapai dalam penelitian ini adalah terciptanya sebuah system pemesanan jasa tukang dengan arsitektur REST web *service* serta mengimplementasikannya pada sebuah *multiplatform* yang terintegrasi. *Platform* yang akan dihasilkanhanya berupa web *client* dan android *client*.

Kata kunci : *Internet,REST API Web Service, Aplikasi Multiplatform*

I. PENDAHULUAN

Teknologi informasi dan komunikasi berkembang semakin pesat. Hal ini disebabkan oleh manusia yang terus berinovasi untuk memenuhi kebutuhannya. Teknologi informasi adalah seperangkat alat yang membantu Anda bekerja dengan informasi dan melakukan tugas-tugas yang berhubungan dengan pemrosesan informasi[1].

Mataram merupakan salah satu kota yang berada di Provinsi Nusa Tenggara Barat tepatnya di Pulau Lombok. Bidang ekonomi, pariwisata, pendidikan masih menjadi faktor utama yang menyebabkan banyak pendatang berkunjung ke kota Mataram. Bahkan tidak terkecuali beberapa pendatang dari luar pulau. Berbagai macam profesi juga terdapat di kota Mataram salah satunya adalah jasa tukang. Tukang merupakan orang atau kelompok yang mempunyai kepandaian dalam suatu pekerjaan tangan (dengan alat atau bahan yang tertentu)[2].

Pada umumnya para calon pemesan jasa tukang sering mengalami kesulitan dalam mencari tukang, khususnya bagi para pendatang. Para pendatang yang tidak mengetahui wilayah sekitar yang menyediakan jasa tukang, tidak mengetahui tentang harga tukang. Tentunya membuat kendala bagi calon pemesan untuk melakukan proses pemesanan tukang. Untuk mengatasi masalah tersebut, para calon pemesan jasa tukang biasanya melakukan *survey* terhadap beberapa tukang. Cara survey yang mencari di daerah sekitar seperti ini dapat membutuhkan waktu, biaya dan tenaga yang terkadang tidak sedikit.

Tukang juga melakukan promosi secara konvensional, seperti menggunakan spanduk, pamflet, kartu nama serta dengan komunikasi seseorang membuat langkah promosi seperti ini kurang efektif dalam meningkatkan pendapatan, karena selain membutuhkan biaya, waktu tidak sedikit, cakupan luas area promosi yang terbatas. Untuk mengoptimalkan pemesanan jasa tukang,

dibutuhkan sistem yang dapat digunakan untuk media promosi tukang serta dapat berinteraksi antara calon pemesan jasa tukang dengan tukang tersebut.

Aplikasi *multiplatform* (lebih dari satu *platform*) dapat memberikan kemudahan pengguna dalam memilih *platform* yang akan digunakan. Pengguna tidak akan merasa dituntut untuk menggunakan jenis komputer tertentu, baik itu desktop ataupun mobile. Arsitektur *web service* adalah sebuah entitas komputasi yang dapat di akses melalui jaringan internet maupun intranet dengan standar protokol tertentu dalam *platform* dan antarmuka bahasa pemrograman yang independen[3]. Tujuan pengembangannya adalah untuk "menjembatani komunikasi antar program", sehingga aplikasi yang satu dan aplikasi yang lain yang terdapat pada suatu jaringan yang sama atau pada jaringan yang berbeda dapat saling berkomunikasi asalkan menggunakan standar protokol yang di tetapkan oleh *web service*. Dengan menggunakan arsitektur *web service* sistem dapat terintegrasi walaupun dengan *platform* lain.

Dengan adanya aplikasi *mulitplatfrom* selain pengguna dapat terintegrasi meskipun menggunakan *platform* yang berbeda, tukang maupun calon pemesan tukang dapat saling berinteraksi dengan sistem sehingga proses pemesanan jasa tukang serta pencarian dapat menjadi lebih mudah.

Arsitektur Web Service

Gambar 1. Komponen Web Service

Web *Service* disusun oleh tiga komponen standar [4], yaitu:

a. *Simple Object Access Protocol* (SOAP), yaitu protokol yang bertanggung jawab dalam pertukaran informasi dalam lingkungan jaringan terditribusi.
b. *Web ServiceDefinition Lmanguage* (WSDL), dokumen standar yang dituliskan dalam format XML, dan mendefinisikan kehadiran *web service* dalam suatu jaringan.
c. *Universal Deskription, Discovery, and Integration* (UDDI), yaitu suatu lokasi direktori yang berisikan *service* (layanan) dan bersifat bebas *platfrom* (*platform independent*), dituliskan berbasis XML dan dapat diakses oleh entitas yang berada di dalam jaringan.

RESTful Web Service

REST adalah salah satu jenis *web service* yang menerapkan konsep perpindahan antar *state. State* disini dapat digambarkan seperti jika *browser* meminta suatu halaman web, maka server akan mengirim *state* halaman web yang sekarang ke *browser*[5]. Bernavigasi melalui *link-link* yang disediakan sama halnya dengan mengganti *state* dari halaman web. Begitu pula REST bekerja. Dengan bernavigasi melalui *link-link* HTTP untuk melakukan aktivitas tertentu, seakan-akan terjadi perpindahan *state* satu sama lain.

Perintah HTTP yang bisa digunakan adalah fungsi GET, POST, PUT, UPDATE atau DELETE. Kemudian balasan yang dikirimkan oleh API *Server* adalah dalam bentuk JSON, sehingga informasi yang diterima lebih mudah dibaca dan di-*parsing* disisi *client*.

REST merupakan teknologi yang bekerja berdasarkan *resource* untuk membuat sistem terdistribusi. REST (disebut juga RESTful *services*) adalah perangkat lunak yang didesain dengan penekanan pada kesederhanaan, skalabilitas, serta kegunaan. Dalam pengaplikasiannya, REST lebih banyak digunakan untuk *web service* yang berorientasi pada *resource*. Maksud orientasi pada *resource* adalah orientasi yang menyediakan *resource-resource* sebagai layanannya dan bukan kumpulan dari aktifitas yang mengolah *resource* itu. Alasan mengapa REST digunakan dalam penelitian ini karena pada aplikasi web *client* dan android *client* akan mengolah *resource-resource* tersebut.

Metode REST didasari oleh empat prinsip utama teknologi yaitu:

1. *Resource identifier* melalui *Uniform Resource Identifier* (URI), REST *Web service* mencari sekumpulan sumber daya yang mengindetifikasi interaksi antar *client*.
2. *Uniform interface*, sumber daya yang dimanipulasi CRUD (*Create, Read, Update, Delete*) menggunakan operasi PUT, GET, POST dan DELETE.
3. *Self-descriptive messages*, sumber daya informasi tidak terikat, sehingga dapat mengakses berbagai format konten (HTML, XML, PDF, JPEG, *PlainText* dan lainnya). Metadata pun dapat digunakan.
4. *Stateful interactions* melalui *hyperlink*, setiap interaksi dengan suatu sumber daya bersifat *stateless*, yaitu *requestmessages* tergantung jenis kontennya.

Protokol HTTP memiliki beberapa HTTP *Method* yang dimana masing-masing *method* mendeklarasikan suatu aksi. Berikat penjelasan aksi dari masing-masing method.

Tabel 1.HTTP *Method*

HTTP *Method*	Aksi
POST	Membuat *resource* baru
GET	Membaca *resource*
HEAD	Membaca metadata dari *resource*
PUT	Meng-update *resource*
PATCH	Meng-update beberapa bagian dari *resource*
DELETE	Menghapus *resource*
OPTIONS	Menampilkan HTTP *method* yang tersedia dari sebuah URL
TRACE	Menampilkan kembali data *request*
CONNECT	Menghubungkan antara TCP/IP (jarang diimplementasikan)

1. Arsitektur REST

REST merupakan penyederhaan dari HTTP. Dengan memanfaatkan *noun*, bukan *verb* pada URI HTTP[5]. Para pengembang aplikasi web cenderung melihat hal-hal seperti *verb* HTTP dan kode status respon sebagai sesuatu yang *incidental* untuk aplikasi, atau sebagai suatu hal tidak penting yang akan ditangani jika waktu masih mengijinkan. Penggunaan HTTP sebagaimana yang diharapkan, sering terlihat sebagai sesuatu yang tidak diperlukan atau menyulitkan. Namun dalam beberapa tahun belakangan ini, dengan hadirnya kembali prinsip-prinsip REST telah mengindikasikan bahwasanya HTTP telah lebih dari cukup baik diatas segalanya. REST merupakan cara baru berpikir tentang arsitektur jaringan berdasarkan pengamatan atas bagaimana jaringan bekerja.

2. PHP

PHP adalah salah satu bahasa pemrograman skrip yang dirancang untuk membangun aplikasi web. PHP singkatan dari PHP *Hypertext Processor* yang digunakan sebagai bahasa *scriptserver-side* dalam pengembangan Web yang disisipkan pada dokumen HTML[6].

PHP bisa melakukan apa saja yang dapat dilakukan oleh CGI, seperti mengumpulkan data dari *form*, menghasilkan isi halaman web dinamis, dan kemampuan mengirim serta menerima *cookies*, bahkan lebih daripada kemampuan CGI [3].

III. METODOLOGI

Dalam melakukan penelitian, metode pengembangan sistem yang digunakan penulis adalah metode *waterfall*. Metode *waterfall* merupakan metode pengembangan perangkat lunak yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem sampai pada analisis, desain, kode (Implementasi), test (Pengujian), dan pemeliharaan[7]. Tetapi di dalam penelitian ini, penulis hanya sampai pada tahap pengujian.

Berikut tahapan-tahapan dalam *waterfall*.

1. Analisa kebutuhan.
   Pada tahap ini penulis melakukan analisa kebutuhan yaitu meliputi analisa data dan informasi, serta indentifikasi masalah. Penulis melakukan analisa melalui wawancara. Analisa dilakukan dengan tujuan menetapkan alternatif solusi dan spesifikasi kebutuhan sistem.
2. Desain sistem

Pada tahap ini, penulis melakukan perancangan sistem terhadap solusi dari permasalahan yang ada dengan menggunakan perangkat pemodelan sistem seperti relasi tabel, *Class Diagram, Activity Diagram, Use Case Diagram, Sequence Diagram dan Desain Interface*.

3. Implementasi
   Pada tahap impelementasi, penulis membangun sistem baru dengan mengimplementasi hasil desain sistem yang telah di lakukan. Penulis membangun sistem baru berupa *API Webservice* dengan bahasa pemrograman php, dan *web client* dengan javascript serta *android client* dengan java native.
4. Uji coba
   Pada tahap ini, penulis melakukan pengujian terhadap sistem baru yang telah dibangun. Proses pengujian dilakukan dengan metode *blackbox*. Metode ini melibatkan pengguna dengan kuisioner sebagai alat ukur untuk menilai apakah sistem yang baru sudah dapat memenuhi kebutuhan pengguna atau belum.

IV.HASIL DAN PEMBAHASAN

3.1Analisa kebutuhan

a. Analisis Data dan Informasi

Sebelum melakukan perancangan sistem baru, diperlukan in*form*asi yang jelas mengenai sistem lama yang biasa dilakukan oleh tukang, untuk memperoleh gambaran umum mengenai prosedur pemesanan jasa tukang. Hal ini dilakukan untuk mengetahui bagian mana dari prosedur tersebut yang perlu ditingkatkan atau diganti untuk menghasilkan sistem yang dapat berjalan pada *multiplatform*.

Teknik pengumpulan data dilakukan dengan melakukan wawancara terhadap beberapa orang dengan jasa tukang. Berikut ini adalah rincian hasil pengumpulan data dan in*form*asi yang dilakukan dengan teknik tersebut.

b. Indetifikasi Masalah

Berdasarkan hasil analisa yang telah dijabarkan di atas, ditemukan beberapa hal yang dianggap sebagai kendala dari sistem yaitu, antara lain:

- Media Promosi
  Seorang tukang yang melakukan cara konvensional dalam menyebarkan informasi mengenai jasa hanya dengan menggunakan spanduk, kartu nama, serta dari komunikasi dengan orang lain.
- Kecepatan Pencarian
  Seorang pemesan jasa tukang mengalami kesulitan dalam pencarian jasa tukang. Para pemesan melakukan pencarian dengan berkeliling dengan melihat spanduk ataupun dengan bertanya dengan masyarakat sekitar.
- Pemilihan Tukang
  Seorang pemesan jasa tidak dapat mengetahui kelayakan seorang tukang yang akan mereka pesan jasanya secara objektif.
- Aplikasi Yang Dapat Membantu
  Tidak terdapat sistem yang dapat membantu pemesanan jasa tukang. Khususnya yang dapat membantu interaksi secara tak langsung. Serta mendapatkan informasi jasa tukang yang berada di area tertentu.
- Pemilihan Platform Pengguna
  Pengguna yang tidak diketahui secara pasti *device*/jenis perangkat yang digunakan dalam pengembangan sistem. Maka diperlukan sistem yang dapat dikembangkan pada *multiplatform* (lebih dari satu *platfrom*).

c. Alternatif Solusi

Berdasarkan hasil identifikasi masalah yang telah dijelaskan di atas, maka solusi yang dapat dilakukan untuk mengatasi masing-masing permasalahan antara lain:

- Diperlukan sebuah sistem yang dapat digunakan sebagai media promosi yang mudah dan murah untuk tukang.
- Diperlukan sebuah sistem yang dapat melakukan pencarian berdasarkan lokasi penyedia jasa tukang.
- Diperlukan sebuah sistem yang dapat melakukan interaksi secara tak langsung dengan pemesan dan jasa tukang.
- Diperlukan pembuatan sistem dengan arsitektur REST API *Web Service* sehingga *platform* lain dapat dibangun.
- Mengimplementasikan API *Web Service* dalam *platform* web untuk memberikan user interface yang lebih baik pada web *client*.

- Mengimplementasikan API *Web Service* dalam *platform* android, khususnya aplikasi native yang dapat memberikan *user interface* serta memiliki kecepatan akses yang cepat.

d. Analisa Kebutuhan Sistem

Analisis kebutuhan sistem dilakukan untuk perkiraan kebutuhan sistem baru. Adapun analisis kebutuhan sistem meliputi:

- Perangkat Keras

  Untuk menjalankan sistem ini diperlukan beberapa perangkat keras yaitu:
  - Komputer dengan RAM minimal 2GB.
  - Komputer dengan *Hard disk* minimal 500GB.
  - Komputer dengan CPU minimal *Core* 2 Duo.
  - Smartphone dengan sistem operasi android.
  - Smartphone dengan RAM minimal 512 MB.
  - Smartphone dengan penyimpanan *internal* minimal 2GB.
- Perangkat Lunak

  Untuk menjalankan sistem ini diperlukan beberapa perangkat lunak yaitu:
  - Sistem Operasi Linux/Windows sebagai*server* API *Web service* dan *server* aplikasi *WebClient*.
  - Xampp sebagai *WebServer*.
  - MySQL sebagai DBMS.
  - *Web* Browser untuk mengakses aplikasi *WebClient*.
  - Sistem operasi Android dengan versi minimal 4.1 (*JellyBean*).
- Pengguna

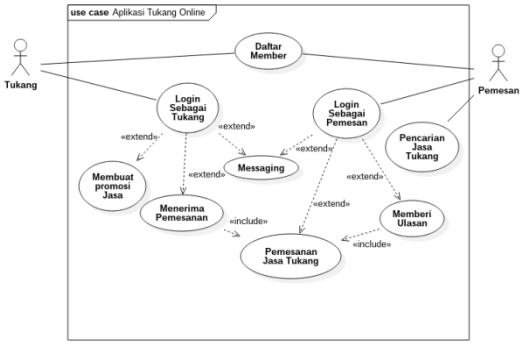
  Agar perangkat lunak yang dibangun dapat berjalan dengan baik, maka diperlukan beberapa pengguna antara lain:
  - *Admin*, yaitu pengguna yang menangani kebutuhan aplikasi *server control*. Kemampuan yang harus dimiliki *admin* yaitu dapat mengoperasikan komputer dan perangkat lainnya dengan baik, dapat bekerja pada sistem operasi Windows maupun Linux, dan dapat mengoprasikan aplikasi *server control* dengan baik dan benar.
  - Tukang dan Pemesan, yaitu pengguna yang akan menggunakan aplikasi ini. Kemampuan yang harus dimiliki antara lain, dapat mengoperasikan komputer dengan baik, dan dapat mengoperasikan perangkat lunak *web browser* dengan baik bagi pengguna web *client*, dapat mengoperasikan smartphone dengan baik, bagi pengguna android *client*.

3.2Desain sistem

Desain sistem dibuat dengan menggunakan UML, meliputi: *Use Case Diagram, Class Diagram, Activity Digram, dan Sequence Diagram. Unfied Modeling Language*(UML) adalah keluarga notasi grafis yang didukung oleh model-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek[7].

a. Use case diagram

Use case diagram digunakan untuk menggambarkan fungsionalitas aplikasi. *Use Case Diagram* terdiri dari *actor-actor* dan fungsi-fungsi apa saja yang bisa dilakukan di dalam sistem.

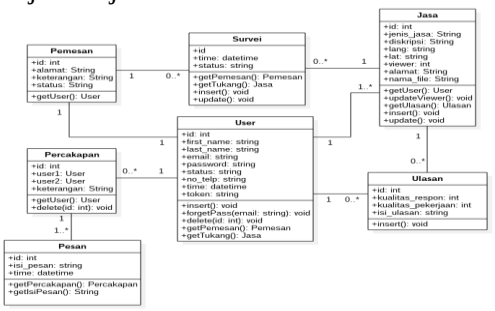


Gambar 2. Usecase diagram

Use Case Diagram di atas menggambarkan kegiatan apa saja yang bisa dilakukan oleh Tukang dan Pemesan di dalam sistem. Tukang dan Pemesan harus melakukan *login* untuk bisa mengakses sistem. Tukang dapat membuat promosi jasa, menerima pemesanan serta melakukan *messaging* dengan pemesan. Semua action ini dapat dilakukan setelah melakukan action *login*. Pemesan dapat melakukan pencarian jasa tukang, *messaging*, melakukan pemesanan jasa tukang, serta

memberi ulasan jika telah melakukan pemesanan.

b. Class Diagram
Class Diagram digunakan untuk menampilkan kelas-kelas yang jika diinisiasikan akan menjadi objek di dalam sistem.

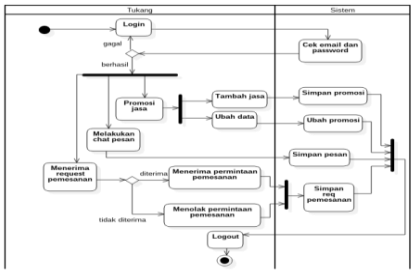


Gambar 3. Class diagram

Pada *Class* diagram di atas, Class Pemesan dan class Jasa meng-*extend* class *User*, sehingga ini menjelaskan *User* dapat menjadi salah satu Pemesan atau jasa Tukang. Class Survei merupakan objek yang menggambarkan antara transaksi pada Pemesan dan Tukang. Class Survei meng-*extend* class Pemesan dan Jasa. Class ulasan meng-*extend* dari class jasa, sehingga dalam satu objek jasa tukang memiliki banyak ulasan. Class yang terakhir adalah class Percakapan yang meng-*extend* dari class user yang menjelaskan pengirim dan penerima, serta meng-*extend* class Pesan yang menjelaskan isi pesan percakapan antara *User*.

c. Activity
activity diagram digunakan untuk menggambarkan urutan aktivitas yang terjadi di dalam aplikasi**.**

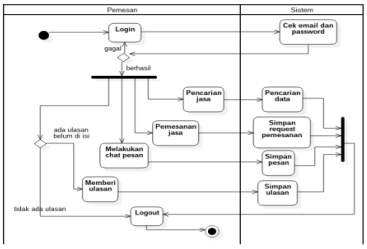
- *Activity Diagram* untuk tukang



Gambar 4. Activity diagram pada tukang

Activity Diagram untuk Tukang, menggambarkan urutan aktivitas yang dilakukan tukang di dalam sistem. Untuk mengakses sistem, tukang harus melakukan *login* terlebih dahulu. Beberapa aktifitas yang bisa dilakukan tukang di dalam sistem adalah melakukan melakukan promosi jasa, menerima pemesanan jasa, serta berkirim pesan.

- *Activity Diagram* untuk pemesan

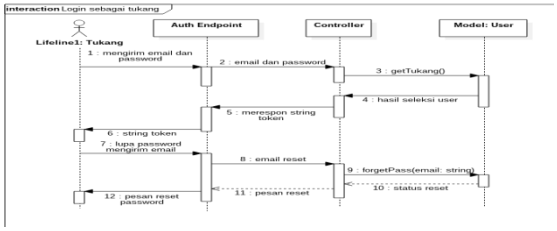


Gambar 5. Activity diagram pada pemesan

Activity Diagram untuk Pemesan, menggambarkan urutan aktivitas yang dilakukan pemesan di dalam sistem. Untuk mengakses sistem, pemesan harus melakukan *login* terlebih dahulu. Beberapa aktifitas yang bisa dilakukan pemesan di dalam sistem adalah pencarian jasa tukang, melakukan pemesanan jasa, memberi ulasan jika terdapat ulaan yang harus diisi serta berkirim pesan.

d. Sequence Diagram
Sequence diagram digunakan untuk menggambarkan interaksi antara sejumlah objek dalam urutan waktu.

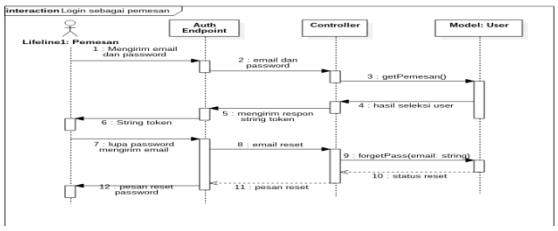
- *Sequence diagram login*sebagai tukang



Gambar 6. Sequence diagram proses login sebagai tukang

Sequence Diagram yang menggambarkan proses login tukang di dalam sistem.

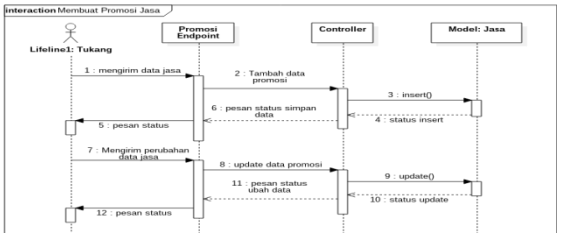
- *Sequence diagram login*sebagai pemesan



Gambar 7. Sequence diagram proses login sebagai *pemesan*

Sequence Diagram yang menggambarkan proses login pemesan di dalam sistem.

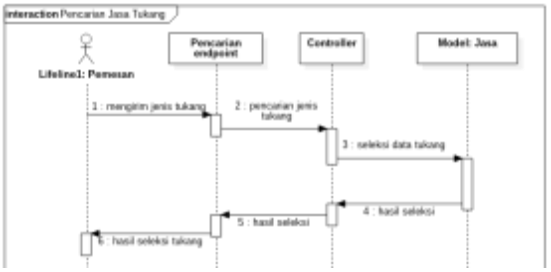
- *Sequence diagram* proses upload promosi jasa



Gambar 8. Sequence diagram proses upload promosi jasa

Sequence Diagram yang menggambarkan proses meng*upload* data promosi jasa ke dalam sistem.

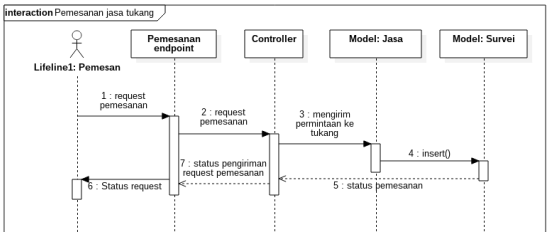
- *Sequence diagram* proses pencarian tukang



Gambar 9. Sequence diagram proses pencarian tukang

Sequence Diagram yang menggambarkan proses pencarian tukang di dalam sistem, baik dengan kata kunci maupun dengan lokasi tertentu.

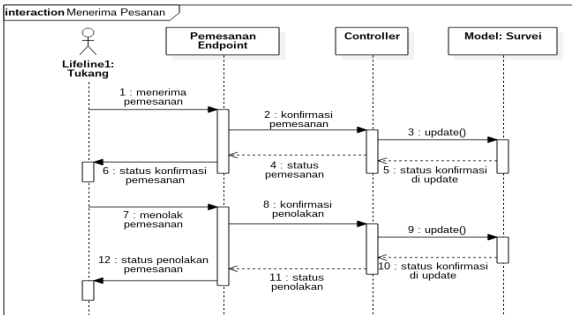
- *Sequence diagram* proses pemesanan jasa



Gambar 10. Sequence diagram proses pemesanan jasa

Sequence Diagram yang menggambarkan proses pemesanan jasa tukang.

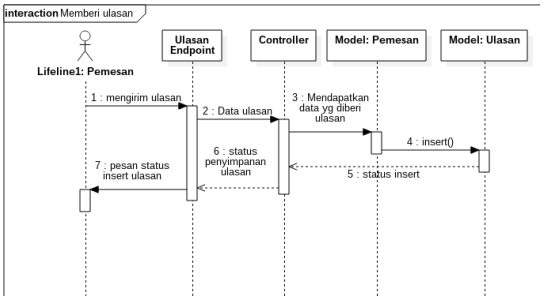
- *Sequence diagram* proses menerima pemesanan jasa



Gambar 11. Sequence diagram proses menerima pemesanan jasa

Sequence Diagram yang menggambarkan proses penerimaan pemesanan jasa tukang.

- *Sequence diagram* proses memberi ulasan



Gambar 12. Sequence diagram proses memberi ulasan

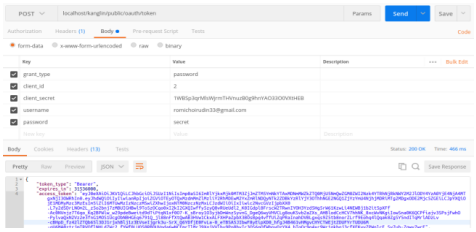
Sequence Diagram yang menggambarkan proses upload ulasan ke sistem.

3.3 Implementasi

Tahap impelementasi dimulai dengan pembuatan *database* di *local server* menggunakan *migration* yang merupakan salah satu fitur Laravel. Kemudian untuk membuat tabel-tabel *database* yang telah dirancang menggunakan fitur *models* pada framework Laravel. Setelah *database* selesai dibuat barulah dilakukan pembuatan *API* Web*service* yang digunakan untuk melayani *request client* terhadap *server* dan respon *server* terhadap *client* yang berupa data JSON yang dapat diolah kembali oleh *client*. Bahasa pemrograman yang digunakan penulis untuk membangun *API Webservice* ini adalah PHP dan framework Laravel. Setelah *API* Web*service* selesai dibuat barulah dilakukan pembuatan tampilan antarmuka aplikasi berbasis

web menggunakan bahasa pemrograman *Javascript* dengan framework *VueJS* serta aplikasi berbasis android dengan menggunakan java *native*. Aplikasi berbasis *web*dan android inilah sebagai tampilan antarmuka yang mengakses data-data dari *endpointAPI Webservice*. Berikut adalah tampilan pengujian API yang dibangun:

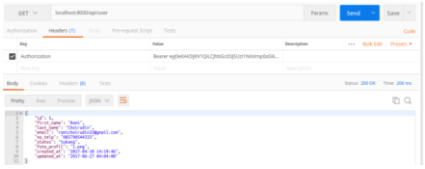
a. URI /oauth/token (POST)



Gambar 13. URI /oauth/token (POST)

Merupakan hasil dari pengujian pada URI /oauth/token. Pada URI ini menggunakan method POST serta menghasilkan respon berupa string token, yang akan digunakan sebagai header dalam mengakses setiap URI yang membutuhkan login user.

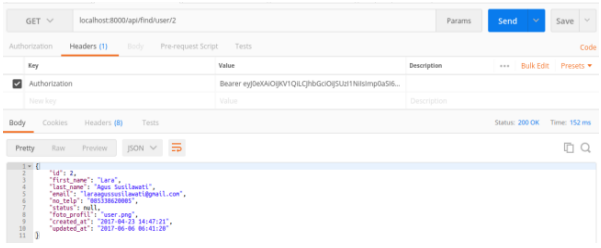
b. URI /api/user (GET)



Gambar 14. URI /api/user (GET)

Berikut merupakan hasil uji coba pada URI /api/user dengan method GET. Pada URI ini menghasilkan respon berupa data json user yang sedang login.

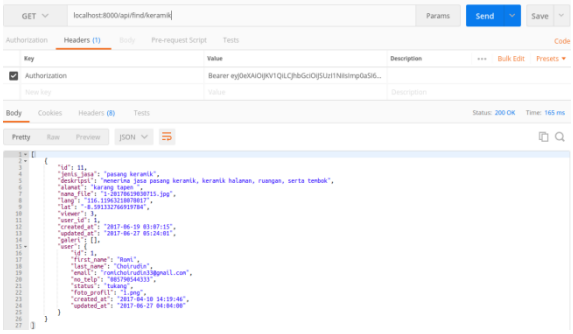
c. URI /api/find/user/{id} (GET)



Gambar 15. URI /api/find/user (GET)

Berikut merupakan hasil uji coba pada URI /api/find/user/{id} dengan method GET. Pada URI ini menghasilkan respon berupa data json user dengan menggunakan parameter id user.

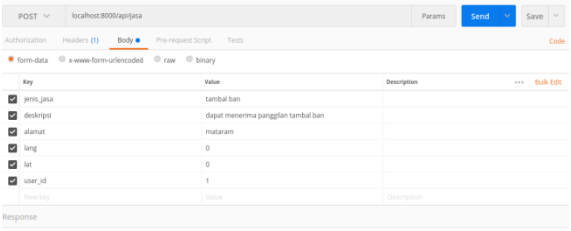
d. URI /api/find/{id}



Gambar 16. URI /api/find/{id} (GET)

Berikut merupakan hasil uji coba pada URI /api/find /{id} dengan method GET. Pada URI ini menghasilkan respon berupa data json jasa dengan parameter data pencarian.

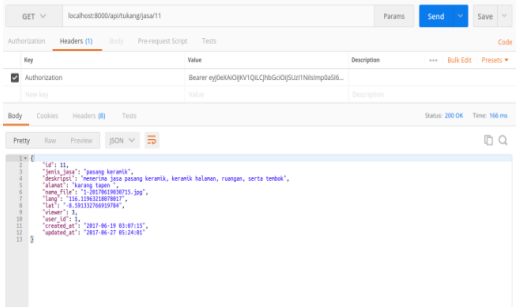
e. URI /api/jasa
(POST)



Gambar 1. URI /api /jasa (POST)

Berikut merupakan hasil uji coba pada URI /api/jasa dengan method POST. Pada URI ini membutuhkan form-data berupa field-field yang akan di simpan di database serta menghasilkan respon json berupa jasa yang baru diposting.

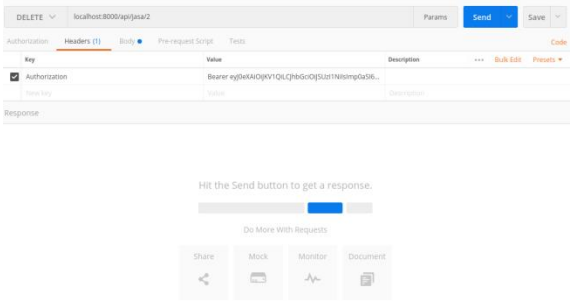
f. URI /api/tukang/jasa/{id} (GET)



Gambar 2. URI /api /tukang/jasa/{id} (GET)

Berikut merupakan hasil uji coba pada URI /api/tukang/jasa/{id} dengan method GET. Pada URI ini menghasilkan respon berupa data json jasa dengan parameter id jasa.

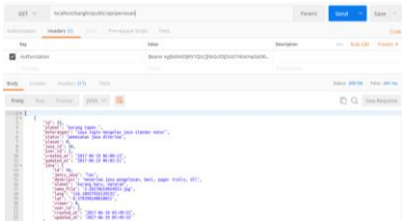
g. URI /api/jasa/{id} (DELETE)



Gambar 3. URI /api/jasa/{id} (DELETE)

Berikut merupakan hasil uji coba pada URI /api/jasa/{id} dengan method DELETE. Pada URI ini digunakan untuk menghapus data jasa pada database menggunakan id jasa.

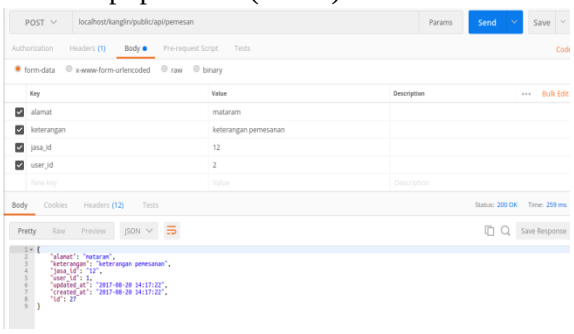
h. URI /api/pemesan (GET)



Gambar 4. URI /api/pemesan (GET)

Berikut merupakan hasil uji coba pada URI /api/pemesan dengan method GET. Pada URI ini menghasilkan respon berupa data json pemesanan.

i. URI /api/pemesan (POST)

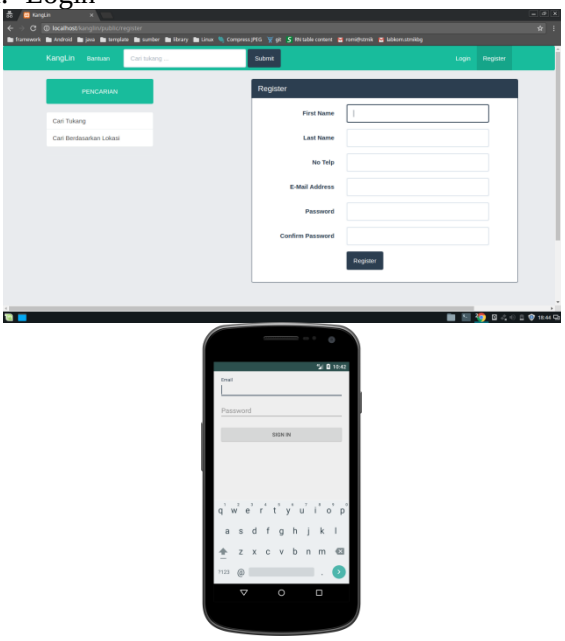


Gambar 21. URI /api/pemesan (POST)

Berikut merupakan hasil uji coba pada URI /api/pemesan dengan method POST. Pada URI ini membutuhkan form-data berupa field-field yang akan di simpan di database serta menghasilkan respon json berupa data pemesanan yang baru diposting.

Berikut adalah tampilan aplikasi yang dibangun:

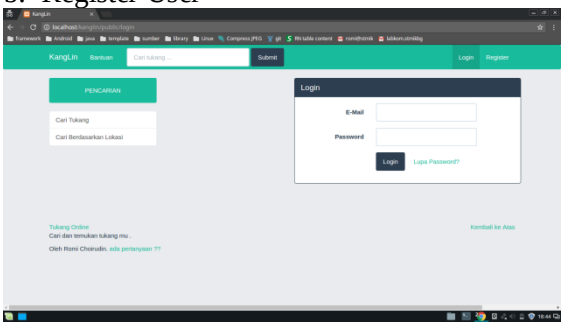
a. Login



Gambar 22. Halaman login web

Pada halaman login sistem menyediakan form untuk memasukkan *username* dan *password.*

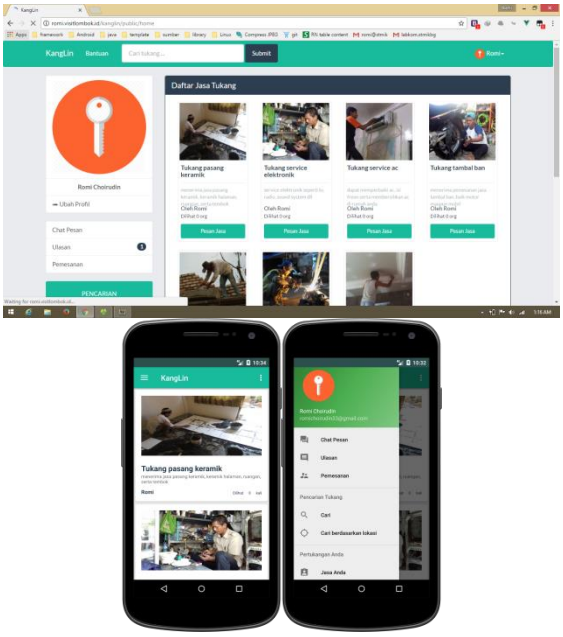
b. Register User



Gambar 23.Halaman register

Halaman Register digunakan untuk melakukan pendaftaran user baru ke dalam aplikasi, pada fitur register aplikasi hanya dibangun di web.

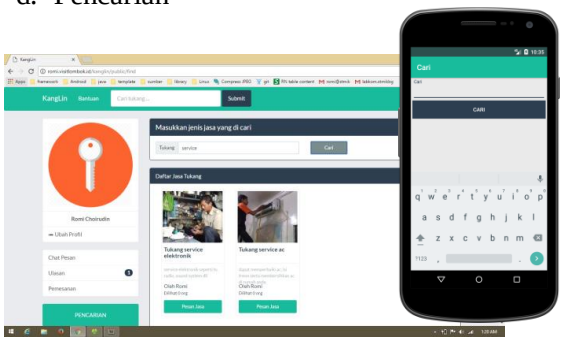
c. Home



Gambar 24. Halaman Home

Halaman homemerupakan halaman yang muncul setelah melakukan proses login.Pada halaman ini secara otomatis sistem akan menampilkan beberapa jasa secara *random*. Jika *user* ingin melakukan beberapa fungsi pada sistem maka user akan menggunakan navigasi yang telah dibuat pada halaman awal

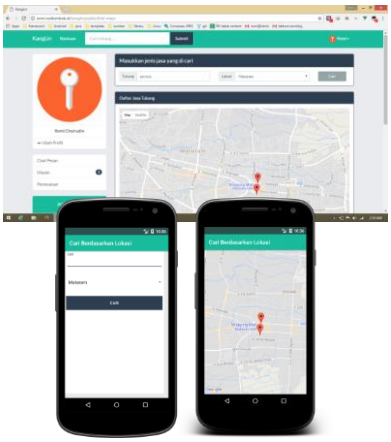
d. Pencarian



Gambar 25. Halaman Pencarian Tukang

Halaman pencarian digunakan untuk melakukan pencarian jasa tukang yang akan dicari oleh Pemesan. Pada halaman ini pemesan dapat memasukkan kata kunci tertentu guna mempermudah pencarian.

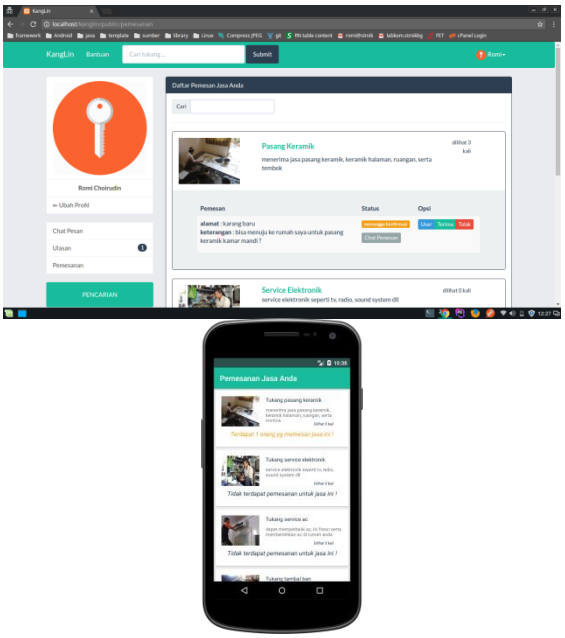
e. Pencarian Tukang Berdasarkan Lokasi



Gambar 26. Halaman Pencarian Tukang Berdasarkan Lokasi

Halaman pencarian tukang berdasarkan lokasi digunakan untuk melakukan pencarian jasa tukang yang akan dicari oleh Pemesan. Pada halaman ini pemesan dapat memasukkan kata kunci serta lokasi tertentu guna mempermudah pencarian.

f. Daftar Pemesan Jasa

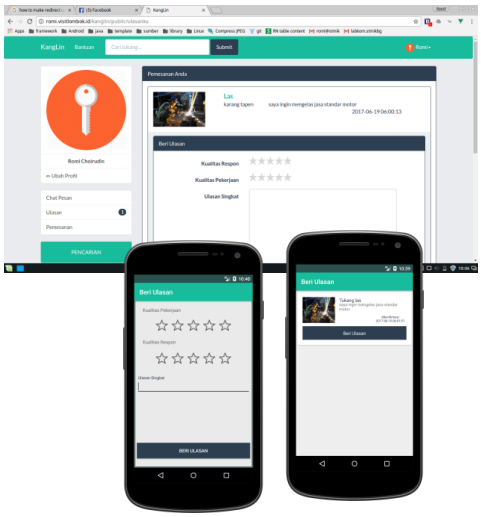


Gambar 27. Halaman Pemesanan Jasa

Halaman berfungsi menampilkan pemesanan yang telah ada yang di *request* oleh

pemesan.Setelah ditampilkan tukang dapat menerima atau menolak pemesan yang di request oleh pemesan.

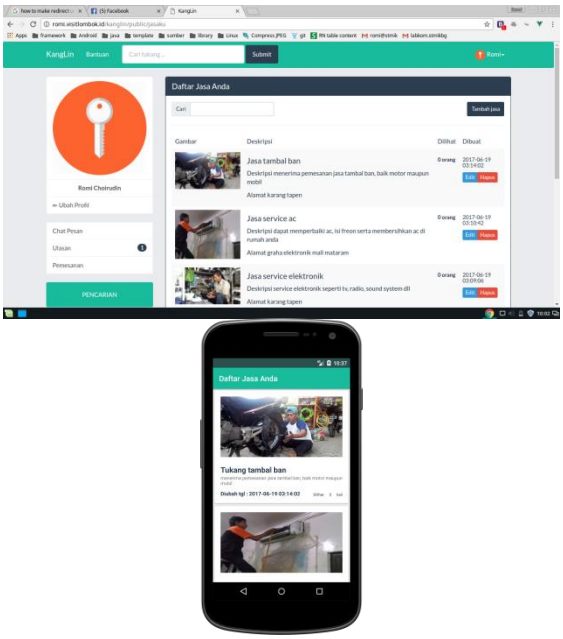
g. Pengisian Ulasan



Gambar 28. Halaman Memberi Ulasan

Halaman ini digunakan untuk melakukan pengisian ulasan pada jasa yang telah di pesan oleh pemesan.

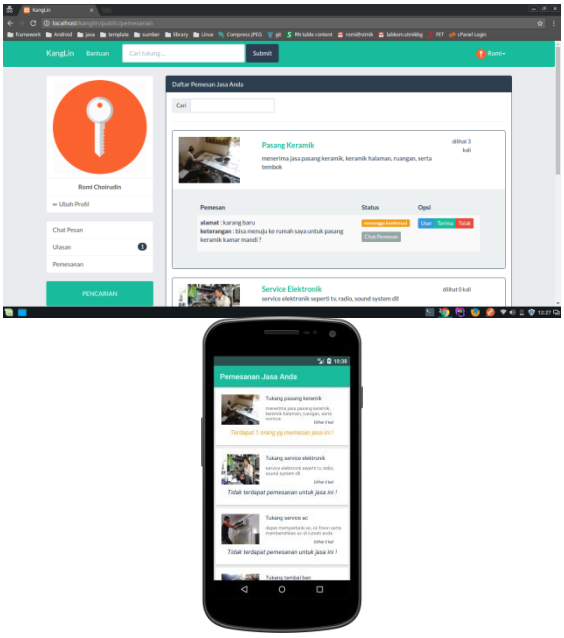
h. Daftar Jasa



Gambar 29. Halaman Daftar Jasa Anda

Halaman ini menampilkan daftar jasa yang telah diupload oleh user tukang.

i. Konfirmasi Pemesanan



Gambar 30. Halaman Konfirmasi Pemesanan

Halaman konfirmasi pemesan merupakan halaman yang berfungsi melakukan konfirmasi pemesan terhadap *request* pemesanan.

3.4 Uji Coba

Uji coba dilakukan dengan menguji aplikasi kepada 20 responden yang terdiri dari 5 orang tukang dan 15 orang pemesan. Berdasarkan hasil pengujian kuesioner, 45,7% mengatakan setuju dan 51% menyatakan sangat setuju bahwa aplikasi ini membantu dalam melakukan promosi jasa serta pencarian jasa tukang.

V. SIMPULAN

5.1Simpulan

Dari pembahasan dapat diambil kesimpulan bahwa *API WebService* dengan arsitektur REST yang telah dibuat berhasil mencapai tujuan dan sasaran dimana dapat digunakan untuk membuat aplikasi *multiplatform* dengan data yang terintegrasi.

Aplikasi yang dibangun telah diuji kepada beberapa tukang. Serta dari hasil pengujian responden tukang memiliki jawaban sangat setuju

sebesar 51.4%, dikarenakan 51.4% merupakan hasil tertinggi pada respon penilaian jawaban maka dapat disimpulkan aplikasi telah memenuhi kebutuhan tukang.

Aplikasi yang dibangun telah diuji kepada beberapa pemesan jasa tukang. Serta dari hasil pengujian responden tukang memiliki jawaban sangat setuju sebesar 46.7%, dikarenakan ini merupakan hasil tertinggi pada respon penilaian jawaban maka dapat disimpulkan aplikasi telah memenuhi kebutuhan

5.2Saran

Sistem memerlukan konfigurasi *web server* secara khusus guna mempercepat akses data. Sistem API yang dapat dikembangkan dengan arsitektur web *socket* untuk membuat sistem dengan *realtime database.* Menambahkan fitur *realtime location* khususnya untuk perangkat *platformmobile* agar dapat melakukan pencarian lebih akurat. Menambahkan fitur *realtime call* agar seorang pemesan dan tukang dapat saling berinteraksi melalui suara secara *online* melalui aplikasi. Sistem dapat dikembangkan dengan menambahkan teknologi sms *gateway* guna mempermudah notifikasi jika terdapat pemesanan baru.

Daftar Pustaka

[1].Haag, & Keen. (1996). *Information Technology. Tomorrow's Advantage Today.* Hammond: Mcgraw-Hill College

[2].*KBBI*. (2016). Retrieved January 11, 2017, from http://kbbi.web.id/tukang: http://kbbi.web.id/tukang.

[3]. Peranginangin, K. (2006). *Aplikasi Web dengan PHP dan MySQL*. Yogyakarta: Andi..

[4]. Siregar, I. M., & Purba, J. (2012). *Membongkar Teknologi Pemrograman Web Service.* Yogyakarta: Gava Media.

[5].Tidwell, D. (2001). *Programming Web Services with SOAP.* Sebastopol: O'Reilly Media.

[6]. Raharjo, B. (2011). *Pemrograman Web dengan PHP + Oracle.* Bandung: Informatika Bandung.

[7].Fowler, M. (2004). *UML Distilled Edisi 3 Panduan Singkat Bahasa Pemodelan Objek Standar*. Yogyakarta: Penerbit ANDI