TUGAS AKHIR – IF184802

RANCANG BANGUN API UNTUK ODOO ERP PADA MODUL CRM (*CUSTOMER RELATIONSHIP MANAGEMENT*)

RAHAJENG DWI PERMATASARI
NRP 05111540000033

Dosen Pembimbing
Nurul Fajrin A., S.Kom., M.Sc.
Abdul Munif, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – IF184802

RANCANG BANGUN API UNTUK ODOO ERP PADA MODUL CRM (*CUSTOMER RELATIONSHIP MANAGEMENT*)

RAHAJENG DWI PERMATASARI
NRP 05111540000033

Dosen Pembimbing
Nurul Fajrin A., S.Kom., M.Sc.
Abdul Munif, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – IF184802

DESIGN AND IMPLEMENTATION API FOR ODOO ERP IN CUSTOMER RELATIONSHIP MANAGEMENT MODULE

RAHAJENG DWI PERMATASARI
NRP 05111540000033

Supervisors
Nurul Fajrin A., S.Kom., M.Sc.
Abdul Munif, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN API UNTUK ODOO ERP PADA MODUL CRM (*CUSTOMER RELATIONSHIP MANAGEMENT*)

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Teknik Informatika
Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
RAHAJENG DWI PERMATASARI
NRP : 05111540000033

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Nurul Fajrin A., S.Kom., M.Sc. .....
   NIP: 19860722 201504 2 003 (Pembimbing 1)

2. Abdul Munif, S.Kom., M.Sc. .....
   NIP: 19860823 201504 1 004 (Pembimbing 2)

SURABAYA
JULI 2019

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN API UNTUK ODOO ERP PADA MODUL CRM (*CUSTOMER RELATIONSHIP MANAGEMENT*)

Nama Mahasiswa	:	Rahajeng Dwi Permatasari
NRP	:	05111540000033
Jurusan	:	Departemen Informatika FTIK-ITS
Dosen Pembimbing 1	:	Nurul Fajrin A., S.Kom., M.Sc.
Dosen Pembimbing 2	:	Abdul Munif., S.Kom., M.Sc.

ABSTRAK

Odoo merupakan salah satu aplikasi ERP terbaik di dunia yang memiliki banyak fitur sebagai kelebihannya dibanding aplikasi ERP yang serupa. Hal ini dikarenakan Odoo mencakup semua jenis operasi bisnis yang dibutuhkan, mulai dari manajemen proyek, hubungan dengan pelanggan, penjualan, hingga penagihan pembayaran. Dengan banyaknya kelebihan yang dimiliki, banyak pengguna yang mengandalkan aplikasi Odoo untuk mengintegrasikan semua data perusahaan dimanapun dan kapanpun.

Modul CRM merupakan salah satu bagian terpenting di Odoo. CRM adalah modul untuk mengelola data atau informasi customer dari menambah data customer, melakukan perubahan data customer, hingga melakukan peluang pada customer di Odoo. Namun saat ini modul CRM di Odoo hanya dapat menjalankan fungsionalitasnya ketika perangkat dalam keadaan online. Jika perangkat sedang tidak dapat mendapatkan akses internet, maka semua proses transaksi data pada modul CRM tidak dapat dijalankan. Keterbatasan tersebut tentunya dapat menghambat pekerjaan jika pengguna ingin melakukan transaksi data namun perangkat sedang tidak terhubung dengan internet.

Dalam mengatasi batasan tersebut, diperlukan pengembangan aplikasi lebih lanjut pada modul CRM. Untuk itu, dalam tugas akhir ini dibuatlah API Odoo pada modul CRM (Customer Relationship Management) agar aplikasi dapat dikembangkan sesuai keinginan pengembang dengan mengimplementasikan Couchbase sebagai offline storage pada Odoo untuk melakukan pertukaran data secara lokal ketika sedang tidak dapat mengakses internet, kemudian melakukan sinkronisasi data setelah mendapatkan akses internet.

Kata kunci: API, Couchbase, ERP, Odoo, Offline Storage

DESIGN AND IMPLEMENTATION API FOR ODOO ERP IN CUSTOMER RELATIONSHIP MANAGEMENT MODULE

Name	:	Rahajeng Dwi Permatasari
NRP	:	05111540000033
Major	:	Informatics Department FTIK-ITS
Supervisor I	:	Nurul Fajrin A., S.Kom., M.Sc.
Supervisor II	:	Abdul Munif., S.Kom., M.Sc.

ABSTRACT

Odoo is one of the best ERP applications in the world that has many features as advantages over similar ERP applications. This is because Odoo covers all types of business operations needed, from project management, customer relationship, sales, to collection of payments. With many advantages, many users rely on the Odoo application to integrate all company data wherever and whenever.

The CRM Module is one of the most important parts in Odoo. CRM is a module for managing customer data or information from adding customer data, making changes to customer data, and making opportunities for customers in Odoo. But now the CRM module in Odoo can only run its functionality when the device is online. If the device is unable to get internet access, then all data transaction processes in the CRM module cannot be executed. These limitations certainly can hinder work if the user wants to do data transactions but the device is not connected to the internet.

To get over this limitations, further application development is needed in the CRM module. For this reason, in this final project, the Odoo API was created in the CRM (Customer Relationship Management) module so that applications can be developed according to the wishes of the developer by implementing Couchbase as offline storage in Odoo to exchange data locally on

mobile devices while unable to access the internet, then synchronize data after getting internet access.

Keywords: API, Couchbase, ERP, Odoo, Offline Storage

KATA PENGANTAR

Alhamdulillahirobbil ‘alamiin, puji syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

RANCANG BANGUN API UNTUK ODOO ERP PADA MODUL CRM (*CUSTOMER RELATIONSHIP MANAGEMENT*)

Melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Allah SWT atas segala nikmat dan rahmat yang telah diberikan selama ini serta senantiasa menemani dan memberi kemudahan kepada penulis dalam menyelesaikan Tugas Akhir.
2. Kedua Orang Tua Bapak Sukam dan Ibu Lilik Setiyoningsih, serta kakak Ila Verdiana tersayang yang telah memberikan doa dan dukungan selama ini.
3. Ibu Nurul Fajrin A., S.Kom., M.Sc selaku dosen pembimbing I yang selalu memberikan motivasi dan membimbing penulis selama pengerjaan Tugas Akhir.
4. Bapak Abdul Munif., S.Kom., M.Sc selaku dosen pembimbing II yang senantiasa memberikan masukan, arahan, dan bantuan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
5. Bapak dan Ibu dosen Departemen Informatika ITS yang telah membina dan memberikan ilmu kepada penulis selama menempuh studi di Departemen Informatika ITS.
6. Byan, Rezky, Ronald, Irvan, dan Ariya yang telah banyak membantu dalam proses pengerjaan Tugas Akhir serta memberikan masukan kepada penulis.
7. Salamah, teman seperjuangan Tugas Akhir yang saling menyemangati, menghibur dan saling menguatkan selama mengerjakan Tugas Akhir.

8. Sahabat Geng Bunda; Bella, Nafi, Daus, Yola, dan Salma yang selalu memberi warna kehidupan penulis selama kuliah di Departemen Informatika ITS.
9. Teman-teman Pengurus Harian HMTC ITS Kreasi 2017/2018 yang telah memberikan kesempatan penulis untuk mendapatkan pengalaman lebih selama 1,5 tahun kepengurusan.
10. Teman-teman Administrator Laboratorium MI yang menjadi keluarga selama penulis menimba ilmu di Departemen Informatika ITS.
11. Teman-teman yang tinggal bersama penulis selama hampir 4 tahun, yang telah membantu meluangkan waktu untuk memberikan masukan dan memberikan dukungan kepada penulis.
12. Teman-teman angkatan 2015 yang telah memberikan motivasi selama penulis berkuliah di Informatika ITS.
13. Serta pihak lain yang namanya tidak dapat penulis sebutkan satu-persatu.

Penulis menyadari sepenuhnya bahwa Tugas Akhir ini masih memiliki kekurangan. Oleh karena itu, dengan tangan terbuka, penulis menerima segala saran dan kritik dari pembaca untuk perbaikan ke depannya.

Surabaya, Juli 2019

Rahajeng Dwi Permatasari

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Pada era digital seperti sekarang ini, teknologi informasi merupakan salah satu alat yang sangat berperan penting pada hampir semua aspek kehidupan manusia. Mulai dari kehidupan sehari-hari hingga urusan bisnis. Dalam dunia perbisnisan, teknologi informasi merupakan suatu hal yang sangat krusial. Dengan adanya perkembangan teknologi informasi seperti saat ini, para pebisnis tidak akan kesulitan lagi untuk mengontrol semua aktifitas bisnis (mulai dari produksi, *customer relationship*, hingga penjualan) yang dijalankan karena semua aktifitas dapat terintegrasi pada satu sistem.

Dalam pengembangan aplikasi bisnis, terdapat *software* bernama Odoo, yaitu sebuah *software open source* yang menyediakan beberapa modul, seperti modul CRM (*Customer Relationship Management*). CRM merupakan bagian terpenting di Odoo karena untuk mengelola data informasi customer. Beberapa penelitian banyak yang memanfaatkan CRM dengan teknologi lainnya, seperti penelitian dengan menggunakan teknologi *Gammu Service* yang berfungsi untuk mengirim informasi dengan cepat melalui SMS ke customer [1]. Selain itu, dalam penelitian sebelumnya, memanfaatkan CRM yang dibangun pada sistem *Service Oriented Architecture* (SOA) untuk menyelesaikan masalah yang ada terkait dengan perkembangan dunia bisnis [2]. Pada penelitian lainnya, CRM juga diterapkan dengan *Fuzzy Comprehensive* yang digunakan sebagai evaluasi kinerja CRM [3]. Dalam keseluruhan penelitian yang memanfaatkan CRM, perlunya mengintegrasikan semua data yang dibutuhkan harus dilakukan

secara online agar semua data dapat ter-*update*, sehingga pada pemanfaatan modul CRM hanya dapat menjalankan fungsionalitasnya ketika aplikasi hanya dalam keadaan online. Hal ini dapat menghambat pekerjaan jika pengguna ingin memasukkan data atau mengubah data namun sedang tidak dapat mengakses koneksi internet.

Oleh karena itu, tugas akhir ini mencoba membawa solusi baru dengan menerapkan *Couchbase* pada modul CRM untuk menyimpan data secara lokal pada perangkat *mobile* ketika sedang tidak dapat mengakses internet, kemudian akan melakukan sinkronisasi data setelah mendapatkan akses internet kembali.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana implementasi pembuatan API untuk mengintegrasikan aplikasi *mobile* dan aplikasi web?
2. Bagaimana implementasi *offline storage* pada modul CRM?

1.3 Batasan Masalah

Batasan masalah pada Tugas Akhir ini antara lain:

1. Aplikasi yang dikembangkan hanya fitur pada modul CRM dan hanya dapat digunakan pada user Administrator.
2. Data yang digunakan adalah data sintetis.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini antara lain:

1. Mengimplementasikan pembuatan API untuk mengintegrasikan aplikasi Android dan aplikasi web.
2. Mengimplementasikan *offline storage* pada modul CRM.

1.5 Manfaat

Tugas Akhir ini diharapkan dapat digunakan dalam menjalankan proses bisnis khususnya pada modul *Customer Relationship Management* pada perangkat Android. Tugas akhir ini

mengimplementasikan *offline storage* menggunakan Couchbase Mobile untuk menyimpan data sementara secara lokal ketika sedang dalam kondisi tidak terhubung dengan internet, nantinya otomatis akan melakukan sinkronisasi data setelah terhubung dengan internet.

1.6 Metodologi Pembuatan Tugas Akhir

Adapun beberapa tahap dalam proses pengerjaan Tugas Akhir ini, yaitu:

1. Penyusunan proposal Tugas Akhir

   Proposal Tugas Akhir ini berisi tentang perencanaan "Rancang Bangun Aplikasi Bisnis untuk Modul *Customer Relationship Management* Berbasis Perangkat Bergerak Android dengan Menggunakan Odoo ERP". Proposal terdiri dari deskripsi pendahuluan yang menjabarkan latar belakang dan rumusan masalah yang mendasari dibangunnya aplikasi ini, batasan masalah dalam pembangunan aplikasi ini, serta tujuan dan manfaat yang diharapkan dapat dicapai dengan dibangunnya aplikasi ini. Selain itu, pada proposal Tugas Akhir ini juga terdapat tinjauan pustaka yang menjelaskan teori-teori yang menjadi dasar pembuatan tugas akhir ini, ringkasan isi tugas akhir yang menggambarkan secara umum aplikasi yang dibangun dan framework yang digunakan, serta bagian metodologi dari penyusunan proposal tugas akhir ini.

2. Studi literatur

   Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan ataupun relevan dalam pembuatan program yaitu mengenai Appplication Programming Interface (API), *framework* Laravel, *Enterprise Resource Planning* (ERP), Odoo, Laradoo, Modul *Customer Relationship Managamenet* (CRM), JSON, Couchbase dan Couchbase Lite.

3. Analisis dan desain perangkat lunak

   Analisis kebutuhan dan perancangan sistem dilakukan untuk merumuskan solusi yang tepat dalam pembuatan aplikasi serta kemungkinan yang dapat dilakukan untuk mengimplementasikan

rancangan tersebut. Tahap desain meliputi arsitektur perangkat lunak yang digunakan, desain antarmuka, serta diagram-diagram yang mendukung pendeskripsian sistem aplikasi.

4. Implementasi

Aplikasi ini diimplementasikan dengan menggunakan:

1. Sistem operasi yang dipakai adalah Android dengan spesifikasi minimal Android 4.0 (Ice Cream Sandwich).
2. Bahasa pemrograman yang digunakan yaitu PHP dan Java.
3. IDE yang digunakan yaitu Android Studio.
4. Postman, sebagai kakas bantu untuk menguji integrasi API dengan Odoo ERP.
5. Sublime, sebagai *text editor* dalam pengerjaan API.

5. Uji coba dan evaluasi

Pengujian dilakukan untuk mengetahui tingkat keberhasilan pada sistem ini. Terdapat dua macam pengujian pada Tugas Akhir ini, yaitu pengujian API dan pengujian pada *offline storage*. Data yang digunakan adalah data sintetis. Setelah aplikasi diujikan, lalu akan dilakukan evaluasi berdasarkan berhasil tidaknya pengguna melakukan fungsionalitas tersebut.

6. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini. Pada tahap ini juga disertakan hasil dari implementasi perangkat lunak yang telah dibuat. Sistematika penulisan buku Tugas Akhir ini secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Masalah
    d. Tujuan
    e. Manfaat
    f. Metodologi Pembuatan Tugas Akhir

g. Sistematika Penulisan
2. Tinjauan Pustaka
3. Analisis dan Perancangan Sistem
4. Implementasi
5. Pengujian dan Evaluasi
6. Kesimpulan dan Saran
7. Daftar Pustaka

1.7 Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detil mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini akan membahas tentang analisis permasalahan, deskripsi umum sistem, lingkungan perancangan perangkat lunak, perancangan API Odoo ERP, serta perancangan offline storage dan sinkronisasi pada Android.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan dan implementasi desain API yang telah dibuat serta

implementasi dari offline storage. Penjelasan berupa kode sumber yang digunakan untuk proses implementasi.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dan evaluasi dari sistem yang telah dibuat dengan melakukan uji fungsionalitas dan pengujian menggunakan skenario.

Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan pada tugas akhir ini. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi diagram dari proses bisnis dari modul CRM.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir ini. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap perangkat lunak yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Penelitian Terkait

Pada pembuatan Tugas Akhir ini merupakan aplikasi manajemen yang memanfaatkan Odoo ERP terutama pada modul CRM untuk mengelola customer serta memanfaatkan Couchbase untuk penyimpanan *offline storage*. Adapun penelitian yang terkait dengan modul CRM ditunjukkan pada Tabel 2.1.

Tabel 2.1 Penelitian Terkait

Judul Penelitian	Teknologi yang Digunakan	ERP yang Digunakan
Customer Relationship Management Untuk Pengelolaan Donor Darah	*Gammu Service* yang berfungsi untuk mengirim SMS	-
Rekayasa Perangkat Lunak Customer Relationship Management (CRM) Reporting dan Sales Force Automation menggunakan metode Service Oriented Architecture (SOA)	*Service Oriented Architecture (SOA)* sebagai penyelesaian masalah yang ada terkait dengan perkembangan dunia bisnis	-
Research on CRM Performance Evaluation Based on Fuzzy Comprehensive Algorithm	*Fuzzy Comprehensive* sebagai evaluasi kinerja CRM	-
Rancang Bangun Api Untuk Odoo ERP pada Modul CRM (Customer Relationship Management)	*Couchbase* sebagai *offline storage* dalam penyimpanan secara lokal di perangkat Android	Odoo ERP

Berikut ini merupakan beberapa contoh aplikasi yang menggunakan Couchbase sebagai *offline storage*.

2.1.1 Doddle

Doddle adalah cara mudah untuk mengirim dan menerima paket. Doddle memudahkan untuk berbelanja online dengan menyediakan tempat yang strategis bagi pelanggan onlinenya dari *retailer* manapun untuk mengambil paket. Solusi pada basis data Doddle sangat mahal dan sulit untuk digunakan terutama pada kenyataan bahwa Doddle harus menyediakan koneksi jaringan yang berkualitas pada setiap lokasi mereka yang ada di Inggris. Doddle akhirnya memutuskan untuk menggunakan Couchbase Mobile karena memenuhi permintaan pelanggan dengan menggunakan perangkat seluler mereka. Ini merupakan solusi yang sempurna karena basis data yang tertanam (Couchbase Lite) menyimpan data secara lokal ynag membuat aplikasi Doddle bekerja untuk pelanggan dan karyawan bahkan ketika sedang *offline* [4].

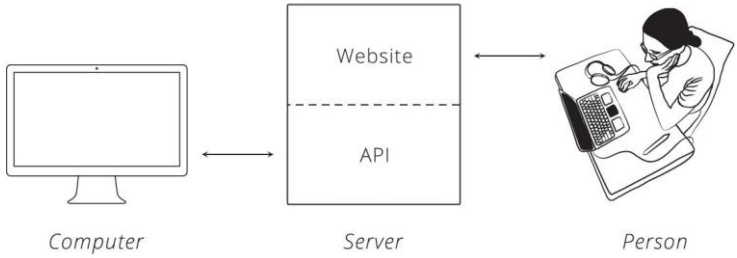
2.1.2 BD

BD adalah perusahaan teknologi medis global yang memajukan dunia kesehatan dengan meningkatkan penemuan medis, diagnostik, dan pemberian perawatan. Sebagai perusahaan teknologi medis terkemuka, BD terus mencari cara baru untuk meningkatkan diagnostik dan pemberian perawatan kesehatan. Contohnya pada kasus diabetes tipe 2, hambatan utama untuk perawatannya adalah kurangnya data dalam hal informasi individual, karena terlalu sulit bagi pasien untuk mengumpulkan data dengan dokter yang tidak punya waktu untuk memilah semua. Solusi BD yaitu membuat aplikasi pasien dan portal klinik yang dibuat dengan Couchbase dan Couchbase Mobile. Solusi ini menggunakan perangkat medis yang terhubung dan aplikasi seluler dengan secara otomatis mencatat data *real-time* dalam hal kadar, aktifitas, makanan, glukosa, dan lokasi pasien. Berdasarkan data, aplikasi seluler akan memberikan peringatan dan rekomendasi khusus kepada pasien. Portal klinik akan mengumpulkan semua data dan menyajikan kepada dokter dalam format yang efisien untuk membuat diagnosis dan menentukan

obatnya. Couchbase menyediakan kemampuan *offline* aplikasi pasien dan sinkronisasi data pasien dari perangkat medis ke cloud [4].

2.2 API

API (Application Programming Interfaces) adalah alat yang membuat data situs web dapat dicerna untuk komputer. Karakteristik yang membuat situs web optimal bagi manusia, namun membuat sulit digunakan oleh komputer sehingga dibutuhkanlah sebuah API. Melalui API, komputer dapat melihat dan mengedit data, sama seperti seseorang yang dapat memuat halaman dan submit formulir. Apa yang mungkin membutuhkan waktu berjam-jam dengan manusia untuk menyelesaikannya, namun dapat memakan waktu beberapa detik dengan komputer melalui API.



Gambar 2.1 Komunikasi Dengan Server [2]

Seperti pada Gambar 2.1, ketika dua sistem terhubung melalui API, dapat dikatakan bahwa dua sistem tersebut terintegrasi. Satu sisi yaitu server dan sisi lain yaitu klien. Dalam sisi server sebenarnya yang menyediakan API. Jika dalam sisi klien, dapat mengetahui data apa yang tersedia melalui API dan dapat memanipulasinya, biasanya atas permintaan dari pengguna. *Request method* nantinya akan memberitahu server tindakan apa yang diinginkan pengguna untuk diambil oleh server [5]. Terdapat empat *method* yang paling sering digunakan di API adalah:

1. GET - Meminta server untuk mengambil data
2. POST - Meminta server untuk *create* data
3. PUT - Meminta server untuk mengedit / memperbarui data
4. DELETE - Meminta server untuk menghapus data

2.3 Laravel

Laravel adalah sebuah framework PHP yang dirilis dibawah lisensi MIT dengan kode sumber yang sudah disediakan oleh Github. Sama seperti framework yang lain, Laravel dibangun dengan konsep MVC yang ditulis dalam PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan awal dan biaya pemeliharaan, serta untuk meningkatkan pengalaman bekerja dengan menyediakan sintaks yang ekspresif, jelas, dan menghemat waktu. Selain itu, Laravel juga dilengkapi dengan *command line tool* yang bernama *Artisan* yang bisa digunakan untuk *packaging bundle* dan *instalasi bundle* melalui command prompt [6].

MVC adalah sebuah pendekatan perangkat lunak yang memisahkan aplikasi berdasarkan komponen-komponen aplikasi, seperti manipulasi data, controller, dan user interface.

1. Model, mewakili suatu struktur data. Biasanya berisi fungsi-fungsi yang membantu dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data, dan lain sebagainya.
2. View, merupakan bagian yang mengatur tampilan untuk pengguna.
3. Controller, merupakan bagian yang menjembatani model dan view.

Beberapa fitur yang dimiliki oleh framework Laravel adalah sebagai berikut:

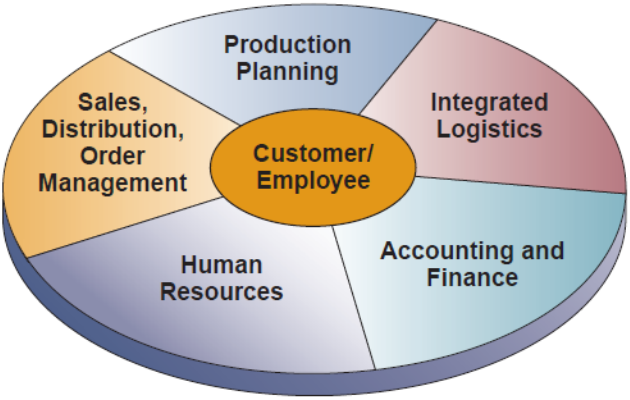
- Bundles, sebuah fitur dengan sistem pengemasan modular dan tersedia untuk digunakan di aplikasi.
- Eloquent ORM, penerapan PHP lanjutan yang menyediakan metode internal dari pola “active record” yang mengatasi masalah pada hubungan objek database.
- Application Logic, bagian dari aplikasi yang dikembangkan, baik menggunakan Controller atau sebagai bagian dari deklarasi Route.

- Reverse Routing, mendefinisikan relasi atau hubungan antara Link dan Route.
- Restful Controllers, memisahkan logika dalam melayani HTTP GET dan POST.
- Class Auto Loading, menyediakan loading otomatis untuk class PHP.
- View Composer, kode unit logikal yang dapat dieksekusi ketika view sedang loading.
- IoC Container, memungkinkan obyek baru dihasillkan dengan pembalikan controller.
- Migration, menyediakan sistem kontrol untuk skema database.
- Unit Testing, mempunyai banyak tes untuk mendeteksi dan mencegah regresi.
- Automatic Pagination, menyederhanakan tugas dari penerapan halaman.

2.4 *Enterprise Resource Planning*

Enterprise Resource Planning (ERP) adalah sebuah sistem yang diperuntukkan bagi perusahaan manufaktur maupun jasa yang berperan mengintegrasikan dan mengotomasikan proses bisnis yang berhubungan dengan aspek operasi, produksi, maupun distribusi di perusahaan yang bersangkutan. ERP sendiri bermula pada kebutuhan bisnis sebuah perusahaan skala *enterprise* dalam menjalankan semua proses bisnisnya.

Beberapa modul *software* ERP digunakan untuk mengelola proses bisnis, menggunakan *database* yang tersentralisasi. Proses bisnis adalah sekumpulan aktivitas yang menerima *input* dan menghasilkan *output*. Umumya dimulai dari memproses data yang masuk, melacak status dari penjualan, inventory, pengiriman barang, dan memperkirakan bahan baku serta kebutuhan sumber daya manusia. Terdapat 5 komponen utama dari sistem ERP [7].



Gambar 2.2 Komponen Utama Sistem ERP [4]

Jadi pada dasarnya, ERP (*Enterprise Resource Planning*) menggabungkan beberapa fungsi manajemen ke dalam sistem yang terintegrasi dan memfasilitasi semua arus informasi pada fungsi manajemen tersebut. ERP dirancang untuk mengotomasikan proses-proses dasar pada seluruh proses bisnis melalui database terpusat dan menghilangkan kebutuhan sistem yang berbeda yang dikelola oleh berbagai unit kerja dalam suatu perusahaan.

2.5 Odoo

Odoo merupakan sebuah *platform open source* yang digunakan untuk keperluan bisnis. Aplikasi atau modul-modul yang terintegrasi dibangun di atas *platform* tersebut, meliputi semua bisnis dari CRM, *sales*, stok, dan lain sebagainya. Odoo dibangun menggunakan bahasa pemrograman Phyton, XML, dan JavaScript.

Odoo termasuk sebagai *software Enterprise Resource Planning* (ERP). Odoo dulunya dikenal sebagai OpenERP. Odoo dibangun secara *open source*, sehingga Odoo mendukung pemanfaatan kembali *library* yang telah ada dan setiap orang dapat terlibat dalam pengembangannya. *Platform* Odoo terdiri dari tiga komponen utama, yaitu *database* PostgreSQL sebagai *database* bawaannya, *application server* Odoo, dan *web server*. *Database* PostgreSQL menampung

semua data yang berhubungan dengan data dan konfigurasi Odoo. Selain sebagai aplikasi, Odoo juga dapat berfungsi sebagai *framework* atau kerangka kerja bagi para *Software Developer* [8]. Odoo menyediakan modul-modul dasar yang mendukung fungsi bisnis, sehingga setiap modul dapat dikostumisasi sesuai dengan kebutuhan [9]. Modul-modul tersebut terdiri dari 7 kelompok aplikasi, antara lain:

1. Aplikasi website: website builder, blog, e-commerce, forum, slides, live chat, appointments
2. Aplikasi sales management: Customer Relationship Management (CRM), point of sale, sales, subscriptions
3. Aplikasi keuangan: accounting, invoicing, expenses
4. Aplikasi operasi: inventory, timesheets, project, purchase, helpdesk, documents
5. Aplikasi human resources: recruitment, employees, fleet, leaves, appraisal
6. Aplikasi marketing: marketing automation, mass mailing, events, survey
7. Aplikasi manufaktur: MRP, PLM, equipment, quality
8. Aplikasi komunikasi: discuss, eSignature.

2.6 Laradoo

Laradoo[1] merupakan sebuah API Odoo ERP yang khusus digunakan pada platform Laravel yang dikembangkan oleh Eduardo Marcos. Dengan menggunakan Laradoo, memudahkan para pengguna untuk membuat API di Laravel dan mengakses langsung Odoo dengan akun miliknya sendiri. Untuk arsitektur dari Laradoo ditunjukkan pada Gambar 2.3. Dengan menggunakan Laradoo ini, aplikasi langsung dapat melakukan transaksi data melewati API yang sudah dibuat di Odoo tanpa harus melakukan login terlebih dahulu. Dikarenakan pada Laradoo sudah terdapat akun Odoo yang sudah dibuat. Untuk langsung dapat mengakses akun Odoo miliknya sendiri, Laradoo memiliki konfigurasi sendiri pada Laravel yang terletak di

[1] Sumber asli: https://github.com/Edujugon/laradoo

laradoo.php. Pada konfigurasi ini nantinya dapat langsung menambahkan nama database yang sudah dibuat saat instalasi Odoo, dan username serta password akun Odoo yang sudah dibuat.

Gambar 2.3 Arsitektur Laradoo

2.7 Modul *Customer Relationship Management* pada Odoo

Customer Relationship Management atau sering disebut CRM merupakan modul dalam platform Odoo. Pada modul ini, dapat melihat semua prospek atau peluang, serta dapat mengelolanya dari satu tahap ke tahap lainnya, dan menganalisis hasil tersebut. Dengan Odoo sendiri dapat membantu penjual dan manajer, seperti mengurangi tenaga kerja, menjadikannya lebih efisien, dan memberikan akses informasi yang mudah. Pada salah satu modul Odoo ini, yaitu CRM memberikan banyak kemudahan dalam segi *customer relationship* [10]. Berikut beberapa fitur penting yang terdapat pada Odoo CRM, yaitu:

1. Lead Entry
   Lead merupakan sebuah peluang atau sering disebut prospek dalam penjualan. Pada Odoo CRM, ada dua kelebihan pada leads, yaitu dapat melihat semua leads dalam satu lokasi, dan dapat menambahkan catatan atau mengubah informasi kontak pada leads yang sudah ada.
2. Lead/Opportunity Management
   Setelah lead ditambahkan ke dalam sistem, langsung dapat ditujukan kepada salesperson. Saat lead melewati proses penjualan, salesperson dapat memindahkan lead dari satu tahap ke tahap lain (New Lead → Qualification → Proposition → Won).
3. Lead Scoring

Dengan fitur ini, dapat memprioritaskan lead dengan memberikan beberapa jumlah bintang. Degan memberi lebih banyak bintang, menunjukkan bahwa diindikasikan memiliki peluang kualitas yang lebih tinggi.

4. Next Activities
   Fitur ini untuk menjadwalkan kegiatan berikutnya, seperti panggilan telepon dan rapat. Selain itu juga dapat langsung mengirim email pada klien dengan integrasi email. Terdapat kalender bawaan Odoo agar dapat mengelola jadwal dengan mudah.
5. Reports
   Dengan Odoo CRM, dapat dengan mudah melihat prospek atau peluang dalam bentuk graph, chart, dll. Dengan adanya ini, memungkinkan untuk menganalisa prospek atau pelung berdasarkan negara, lead *source*, serta dapat menganalisis peluang yang *lost* atau *won*.

Selain fitur-fitur yang disebutkan di atas, Odoo CRM memiliki beberapa menu yang dapat digunakan, diantaranya yaitu:

- Customers, merupakan menu untuk mengelola kontak *customer* dan *company*.
- My Pipeline, merupakan menu untuk mengelola leads dalam satu sales team yang sama.
- Pipeline Analysis, merupakan menu untuk mengelola *leads* dalam semua sales team. Pada menu ini juga terdapat *view chart* untuk mempermudah dalam melihat prospek atau peluang.
- Sales Teams, merupakan menu untuk mengelola sales team berupa nama negara.
- Activity Types, merupakan menu untuk mengelola macam-macam activity untuk penjadwalan.
- Tags, merupakan menu untuk mengelola tag atau label dengan tujuan untuk mengelompokkan sesuai dengan tag masing-masing.

- Lost Reasons, merupakan menu untuk mengelola alasan-alasan saat terjadi *lost*.
- Team Pipelines, merupakan menu untuk melihat jumlah total peluang lead dan jumlah total peluang yang di dapat. Selain itu, pada menu ini juga terdapat *view chart* dalam range tanggal dari *expected closing* pada lead yang telah ditentukan pada data lead.

2.8 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan oleh komputer. Format ini merupakan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang menggunakan gaya bahasa yang umum digunakan oleh *programmer*, seperti bahasa C, C++, C#, Java, JavaScript, Python, dll. JSON terbuat dari dua struktur:

1. Kumpulan pasangan nama atau nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai *object*, *record*, *struct*, *dictionary*, *hash table*, *keyed list*, atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada beberapa bahasa, hal ini dinyatakan sebagai *array*, *vector*, *list*, atau *sequence*.

Struktur data ini disebut sebagai struktur data unviersal, karena format data mudah dipertukarkan dengan semua bahasa pemrograman modern yang juga berdasarkan pada struktur data ini [11]. JSON menggunakan bentuk sebagai berikut:

- Objek, adalah sepasang nama atau nilai yang tidak terurutkan. Penulisan objek dimulai dengan kurung kurawal buka ({) dan di akhir dengan kurung kurawal tutup (}). Lalu setiap nama diikuti dengan titik dua (:), serta setiap pasangan nama atau nilai dipisahkan oleh koma (,).

- Array, adalah sekumpulan nilai yang terurutkan. Penulisan array dimulai dengan kurung siku buka ([ ) dan diakhiri dengan kurung siku tutup ( ]), serta setiap nilai dipisahkan oleh koma (,).
- Nilai (*value*) biasanya berupa string dalam tanda kutip ganda ("), atau integer, juga dapat berupa *true* atau *false* atau *null*, serta dapat berupa sebuah objek atau array.
- String, adalah kumpulan dari beberapa karakter *Unicode* yang berada dalam tanda kutip ganda. String sangat mirip dengan string pada bahasa C atau Java.
- Angka, juga mirip dengan bahasa C atau Java, kecuali format oktal dan heksadesimal yang tidak digunakan.

2.9 Couchbase

Couchbase merupakan sebuah open source, terdistribusi, berorientasi dokumen, database NoSQL dengan arsitektur inti yang mendukung model data JSON yang fleksibel, sinkronisasi mobile, serta keamanan tingkat lanjut. Couchbase mampu memberikan kinerja caching, mengelola data model yang perlu dihilangkan, serta memahami perbedaan bahasa dan API. Couchbase merupakan gabungan dari dua teknologi NoSQL, yaitu Membase yang memberikan teknologi cache dalam kinerja tinggi dan CouchDB yang mempelopori model berorientasi dokumen berdasarkan JSON. Dengan menggunakan model dokumen ini cukup fleksibel sehingga dapat mengubah objek aplikasi tanpa harus *migrate* skema database [12]. Couchbase memiliki dua produk utama, yaitu Couchbase Server dan Couchbase Mobile [13]. Couchbase merupakan salah satu sistem NoSQL yang paling populer digunakan. Namun dengan seiringnya perkembangan zaman, Google menawarkan sebuah solusi untuk mempermudah dalam pengembangan aplikasi mobile maupun web dalam layanan Dbaas (Database as a Service) yang bernama Firebase. Perbandingan antara Couchbase dan Firebase sendiri dijelaskan pada Tabel 2.2.

Tabel 2.2 Perbandingan Couchbase dan Firebase

Couchbase	Firebase
Semua data tersedia secara *offline*	Realtime Database dan secara otomatis menerima pembaruan dengan data terbaru sebagai JSON
Digunakan pada proyek skala besar dengan jumlah data besar (max. 50 MB)	Digunakan untuk proyek berukuran kecil dengan tidak memerlukan operasi data yang rumit (max. 10 MB)
Penyimpanan data sebagai dokumen JSON	Terdapat pembatasan jumlah data dalam penyimpanan

2.9.1 Couchbase Lite

Couchbase Lite adalah *embedded* database yang menangani query dan fungsionalitas manajemen data dari basis data khususnya yang digunakan pada perangkat seluler. Couchbase lite merupakan bagian dari produk Couchbase Mobile yang dibangun untuk aplikasi iOS dan Android atau perangkat seluler. Beberapa fitur penting Couchbase Lite yang membantu membangun sebuah aplikasi adalah Offline Use, memungkinkan aplikasi untuk selalu *always-on*, dapat menjalankan dan menyimpan data pada perangkat seluler hingga jaringan tersedia kembali. Couchbase Lite dirancang dengan kemampuan untuk menyimpan lampiran dokumen seperti gambar, PDF, dll. Dokumen mengambil bentuk objek JSON yaitu sebuha kumpula kunci atau nilai dimana nilai tersebut dapat berupa berbagai jenis tipe data yang berbeda seperti angka, string, array, atau bahkan lainnya. Setiap dokumen diidentifikasi oleh ID dokumen yang dapat dihasilkan secara otomatis (sebagai UUID) atau dapat ditentukan secara programatis, namun kendalanya adalah bahwa ID harus unik di dalam database, dan itu tidak dapat di ubah [12].

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab 3 ini akan dijelaskan mengenai analisis dan perancangan perangkat lunak untuk mencapai tujuan dari Tugas Akhir. Perancangan ini meliputi analisis permasalahan, deskripsi umum sistem, dan perancangan sistem yang akan dibuat.

3.1 Analisis Metode Secara Umum

Tahap analisis dibagi menjadi beberapa bagian, antara lain analisis permasalahan dan deskripsi umum sistem.

3.1.1 Analisis Permasalahan

Sebagai salah satu aplikasi ERP terbaik di dunia, tentunya Odoo memiliki banyak fitur yang menjadi kelebihannya dibanding aplikasi ERP yang serupa. Hal ini dikarenakan Odoo mencakup semua jenis operasi bisnis yang dibutuhkan, mulai dari manajemen proyek, CRM, hingga penjualan. Dengan banyaknya kelebihan yang dimilikinya, tentunya Odoo memiliki banyak pengguna yang mengandalkan aplikasi Odoo untuk mengintegrasikan semua data perusahaan dimanapun dan kapanpun.

Namun saat ini aplikasi Odoo hanya dapat menjalankan fungsionalitasnya ketika perangkat dalam keadaan *online*. Hal ini tentunya dapat menghambat pekerjaan jika pengguna ingin memasukkan data namun sedang tidak dapat mengakses atau mengalami gangguan koneksi internet.

Untuk mengatasi batasan tersebut, diperlukan pengembangan aplikasi lebih lanjut. Untuk itu, dalam tugas akhir ini dibuatlah API Odoo agar aplikasi dapat dikembangkan sesuai keinginan pengembang.

Selain itu, tugas akhir ini juga mencoba bagaimana cara mengimplementasikan Couchbase Mobile pada Odoo untuk menyimpan data secara lokal ketika sedang tidak dapat mengakses internet, kemudian melakukan sinkronisasi data segera setelah mendapatkan akses internet.

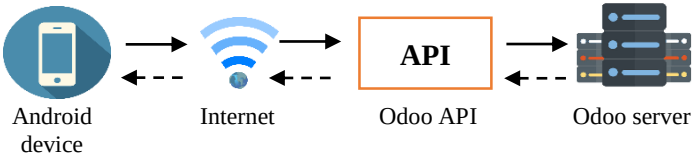
3.1.2 Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibuat sebuah rancang bangun API untuk Odoo ERP pada modul *Customer Relationship Management* agar aplikasi dapat dikembangkan sebagai aplikasi berbasis Android yang dapat menerapkan *offline storage*. Pembuatan API Odoo ini akan diimplementasikan dengan memanfaatkan Laradoo yang merupakan API Odoo ERP untuk Laravel. Secara garis besar, proses bisnis yang ada pada modul CRM dilihat pada Lampiran A.

Pada Lampiran A merupakan proses bisnis pada Modul CRM yang dimulai dari menambah data baru dari customer yang dapat dilakukan pada Android. Setelah menambahkan data, data yang berupa informasi dari customer akan tersimpan pada server. Data customer tersebut dapat kita lanjutkan untuk proses pembuatan pipeline yang dapat dilakukan pada web Odoo. Pada pembuatan pipeline dimana akan membuat sebuah kesempatan sekaligus peluang pada satu *sales teams* yang sama. Peluang ditandai dengan menggunakan jumlah bintang, dimana jika semakin besar jumlah bintang maka peluang tersebut memiliki kualitas yang lebih tinggi. Setelah membuat kesempatan dan peluang, maka proses selanjutnya yaitu menentukan schedule atau penjadwalan yang akan dilakukan berdasarkan *activity types*. Selanjutnya pipeline yang sudah dibuat, dapat diubah statusnya dari *new* hingga *won* (dapat disesuaikan) dan dapat dibentuk sebagai laporan dokumen. Dengan kemampuan *offline storage* yang memanfaatkan Couchbase Mobile, pada aplikasi Android ini akan tetap dapat menjalankan fungsionalitasnya walaupun sedang dalam keadaan *offline* (tanpa ada koneksi internet yang terhubung). Setelah itu, sistem akan otomatis melakukan sinkronisasi ketika perangkat terhubung kembali dengan internet.

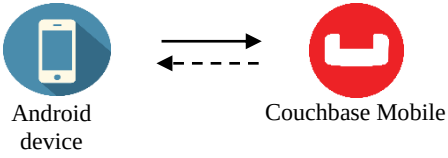
Pada Gambar 3.1 merupakan alur aplikasi saat perangkat dalam keadaan *online*. Untuk mendapatkan atau mengolah data, dapat diperolah dari Odoo API yang langsung terhubung dengan Odoo server. Untuk mempermudah dalam memahami, tanda panah lurus merupakan proses *request* dari *device* ke Odoo server,

sedangkan untuk tanda panah dengan garis putus-putus merupakan proses *response* dari Odoo server ke *device*.



Gambar 3.1 Alur aplikasi dalam keadaan online

Sedangkan pada Gambar 3.2 menjelaskan alur aplikasi ketika perangkat sedang dalam keadaan *offline*. Semua data yang diakses terdapat pada lokal menggunakan Couchbase Mobile. Untuk mempermudah dalam memahami, tanda panah lurus merupakan proses *request* dari *device* ke Couchbase Mobile, sedangkan untuk tanda panah dengan garis putus-putus merupakan proses *response* dari Couchbase Mobile ke *device*.



Gambar 3.2 Alur aplikasi dalam keadaan offline

3.2 Perancangan

Pada subbab ini membahas mengenai tahapan perancangan dari aplikasi Tugas Akhir ini. Subbab ini terdiri dari lingkungan perancangan perangkat lunak, perancangan API untuk Odoo ERP, dan perancangan antarmuka pengguna.

3.2.1 Lingkungan Perancangan Perangkat Lunak

Spesifikasi perangkat keras serta perangkat lunak yang digunakan dalam tahap perancangan perangkat lunak Tugas Akhir ini seperti dijelaskan pada Tabel 3.1.

Tabel 3.1 Lingkungan Perancangan Perangkat Lunak

Perangkat Keras	Komputer	Dell Vostro 14-5468
	Prosesor	Intel® Core™ i7-6500U processor (4M cache, up to 3.5 GHz)
	Memori Primer	8 GB
	Memori Sekunder	1 TB
Perangkat Lunak	Sistem Operasi	Windows 10 Pro 64-bit
	Perangkat Lunak	Odoo 12.0, Android Studio 3.4.1, Sybase PowerDesigner 16.5, Microsoft Word 2016, Sublime Text 3, PostgreSQL, Postman.

3.2.2 Perancangan API Odoo ERP

Pada Laravel nantinya terdapat beberapa API yang dapat diakses. Namun, untuk dapat mengakses API tersebut, perlu dilakukan beberapa instalasi dan konfigurasi terlebih dahulu.

3.2.2.1 Instalasi Odoo

Untuk melakukan instalasi Odoo pada Windows, diperlukan langkah-langkah berikut ini:

1. Unduh **Odoo 12.0**
2. Lakukan penginstalan
3. Buka Odoo 12.0 yang telah terinstal dengan membuka URL https://localhost:8069 pada browser
4. Klik ***sign up*** untuk melakukan pembuatan akun (admin) dan tulis nama database yang di inginkan untuk postgreSQL
5. Verifikasi akun Odoo melalui email yang telah dimasukkan saat pembuatan akun
6. Untuk mendapatkan modul CRM, pilih menu **"Apps"**, cari **"CRM"**, lalu pilih **install**
7. Setelah proses instalasi modul CRM selesai, browser akan *reload* secara otomatis

8. Klik menu **CRM** untuk melihat fitur-fitur yang terdapat pada modul tersebut.

3.2.2.2 Instalasi dan Konfigurasi Laradoo

Dalam Tugas Akhir ini, pembuatan API diimplementasikan dengan memanfaatkan Laradoo. Langkah-langkah yang perlu dilakukan untuk melakukan instalasi Laradoo pada sistem operasi Windows adalah sebagai berikut:

Instalasi Laradoo

1. Buat *project* baru di Laravel
2. Buka **Command Prompt**
3. Masuk pada folder *project* Laravel yang telah dibuat
4. Tambahkan Laradoo pada *project* dengan ketikkan pada console

```
composer require edujugon/laradoo
```

5. Buka *project* menggunakan *text editor*
6. Tambahkan Laradoo dengan menambahkannya ke providers array pada **app.php**

```
'providers' => array(
        ...
        Edujugon\Laradoo\Providers\OdooService
        Provider::class
    )
```

7. Tambahkan juga Alias facade, dengan menambahkannya ke aliases array

```
'aliases' => array(
        ...
        'Odoo'=>
        Edujugon\Laradoo\Facades\Odoo::class,
    )
```

8. Publikasikan file konfigurasi package dengan

```
php artisan vendor:publish --provider=
"Edujugon\Laradoo\Providers\OdooServiceProvider
" --tag="config"
```

9. Setelah publikasikan file konfigurasi package, *base configuration* untuk package Laradoo terletak di file **config/laradoo.php**

Penggunaan Laradoo

1. Instance main Odoo class, dengan
```
$odoo = new \Edujugon\Laradoo\Odoo();
```

2. Hubungkan ke Odoo ERP
```
$odoo = $odoo->connect();
```

3.2.2.3 Perancangan API *Endpoint*

Perancangan API *Endpoint* untuk Modul CRM dijelaskan pada Tabel 3.2.

Tabel 3.2 Perancangan API *Endpoint*

No	Endpoint	Method	Deskripsi
1	/customer	GET	Menampilkan semua data customer
2	/customer/{id}	GET	Menampilkan semua data berdasarkan Id customer
3	/customer	POST	Menambah data customer
4	/customer/{id}	DELETE	Menghapus data customer berdasarkan Id
5	/editcustomer	POST	Mengubah data customer
6	/tags	GET	Menampilkan semua data tags
7	/tags/{id}	GET	Menampilkan semua data berdasarkan Id tags
8	/tags	POST	Menambah data tags
9	/tags/{id}	DELETE	Menghapus data tags berdasarkan Id

10	/edittags	POST	Mengubah data tags
11	/lost	GET	Menampilkan semua data lost reason
12	/lost/{id}	GET	Menampilkan semua data berdasarkan Id lost reason
13	/lost	POST	Menambah data lost reason
14	/lost/{id}	DELETE	Menghapus data lost reason berdasarkan Id
15	/editlost	POST	Mengubah data lost reason
16	/act	GET	Menampilkan semua data activity types
17	/act/{id}	GET	Menampilkan semua data berdasarkan Id activity types
18	/act	POST	Menambah data activity types
19	/act/{id}	DELETE	Menghapus data activity types berdasarkan id
20	/editact	POST	Mengubah data activity types
21	/salestim	GET	Menampilkan semua data sales teams
22	/salestim/{id}	GET	Menampilkan semua data berdasarkan Id sales teams
23	/salestim	POST	Menambah data sales teams
24	/salestim/{id}	DELETE	Menghapus data sales teams berdasarkan Id

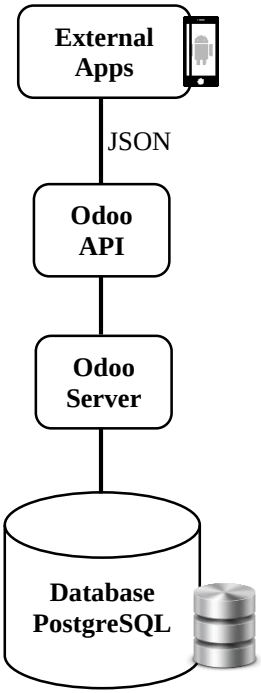
25	/editsalestim	POST	Mengubah data sales teams
26	/pipeline	GET	Menampilkan semua data My Pipeline
27	/pipeline/{id}	GET	Menampilkan semua data berdasarkan Id My Pipeline
28	/pipeline	POST	Menambah data My Pipeline
29	/pipeline/{id}	DELETE	Menghapus data My Pipeline berdasarkan Id
30	/editpipeline	POST	Mengubah data My Pipeline
31	/pplana	GET	Menampilkan semua data Pipeline Analysis
32	/pplana/{id}	GET	Menampilkan semua data berdasarkan Id Pipeline Analysis
33	/pplana	POST	Menambah data Pipeline Analysis
34	/pplana/{id}	DELETE	Menghapus data Pipeline Analysis berdasarkan Id
35	/editpplana	POST	Mengubah data Pipeline Analysis
36	/teamppl	GET	Menampilkan semua data Team Pipelines

3.2.2.4 Alur *Request* Odoo ERP pada Android

Pada subbab ini akan menjelaskan alur pemrosesan *request* pada perangkat Android. Hal yang sama juga terjadi pada penggunaan proses pengujian integrasi API dengan Postman. Seperti ditunjukkan pada Gambar 3.4, ketika seorang pengguna

mengakses fitur dengan melalui perangkat Android, permintaan itu akan langsung menembak ke API pada Laravel melalui Laradoo.

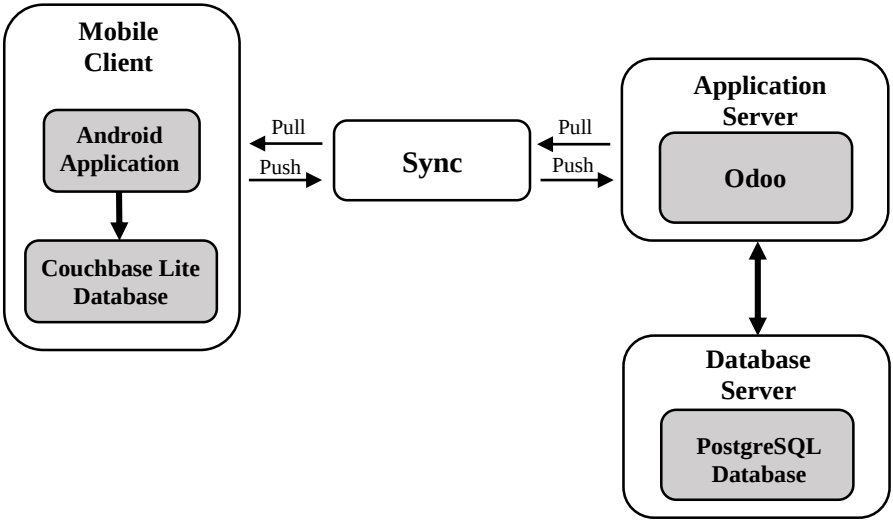
Kemudian Laradoo akan menghubungkan kepada Odoo ERP melalui fungsi **$odoo->connect();** yang terdapat pada file API. Nantinya pada setiap API yang telah relevan atau cocok dengan nama fungsi yang terdapat pada perangkat Android akan meneruskan objek input dan output dari Odoo Server yang telah diteruskan dan diambil pada Database PostgreSQL. Data tersebut selanjutnya akan diteruskan kembali pada perangkat Android, sehingga pengguna akan dapat melihat data yang diminta, dan yang akan ditampilkan dengan benar pada perangkat Android.



Gambar 3.3 Alur *Request* Odoo ERP

3.2.3 Perancangan Offline Storage dan Sinkronisasi pada Android

Pada subbab ini akan dijelaskan tentang *offline storage* yang di implementasikan pada tugas akhir ini. Pada penggunaan *offline storage* ini menggunakan Couchbase Mobile atau yang lebih dikenal dengan Couchbase Lite. Dengan memiliki kelebihan dalam penyimpanan data sebagai dokumen JSON, sehingga dapat meningkatkan hingga terabyte data dalam maksimal ukuran 50MB. Oleh karena itu, Couchbase cocok digunakan dalam proyek skala besar dengan data yang cukup besar, sehingga diperlukan dalam penyimpanan data-data customer pada modul CRM. Couchbase akan bekerja sebagai penyimpanan sementara ketika perangkat sedang dalam keadaan *offline* atau tidak ada koneksi internet dan akan melakukan sinkronisasi dengan mendorong ke server begitu perangkat kembali online. Hal ini akan terlihat pada arsitektur yang dijelaskan pada Gambar 3.5.



Gambar 3.4 Arsitektur *Offline Storage*

Seperti yang dijelaskan pada Gambar 3.5, ketika perangkat dalam keadaan *offline*, pertukaran data dilakukan dengan Couchbase Lite. Sinkronisasi data pada Couchbase Lite dengan server Odoo dapat dilakukan ketika perangkat kembali mendapatkan koneksi internet kembali. Saat sinkronisasi dilakukan, server akan melakukan *pull* data terlebih dahulu untuk mengecek apakah ada perubahan pada data yang tersimpan pada Couchbase Lite. Jika terdapat perubahan data di Couchbase Lite, maka semua data yang berada pada Couchbase Lite akan di *push* ke server Odoo.

Dalam penggunaan Couchbase Lite ini, perlu dilakukan beberapa tambahan *rules* yang diperlukan pada Android. Untuk melakukan tambahan tersebut, diperlukan langkah-langkah berikut ini:

1. Buka **Android Studio**
2. Buka bagian folder **App**, lalu buka file **build.gradle**
3. Tambahkan beberapa *rules* yang diperlukan di Android bagian file **Gradle aplikasi** (app/build.gradle)

```
android {
  packagingOptions {
      exclude 'META-INF/ASL2.0'
      exclude 'META-INF/LICENSE'
      exclude 'META-INF/NOTICE'
  }
}
```

4. Setelah itu, tambahkan Couchbase Lite sebagai *dependency* pada bagian dependensi

```
dependencies {
  // ...
  Compile 'com.couchbase.lite:couchbase-lite-
android:1.3.1'
}
```

5. Di *tool bar* Android Studio, klik ***Sync Project*** agar tambahan yang ditambahkan dapat di sinkronisasi.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Bahasa pemrograan yang digunakan untuk implementasi sistem adalah bahasa pemrograman PHP dan Java.

4.1 Lingkungan Implementasi Perangkat Lunak

Lingungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat Lunak	Sistem Operasi	Windows 10 Pro 64-bit
	Perangkat Lunak	Android Studio 3.4.1, Sybase PowerDesigner 16.5, Microsoft Word 2016, Sublime Text 3, PostgreSQL, Postman.

4.2 Standarisasi Penamaan Resource pada REST Service

Dalam pembuatan *endpoint* dibutuhkan sebuah standar yang konsisten. Standarisasi ini dibutuhkan agar mudah dipahami oleh para pengembang, baik pengembang yang sekarang maupun pengembang yang berikutnya.

4.2.1 Standarisasi Penamaan dan Struktur Endpoint

Terdapat beberapa aturan dalam penulisan endpoint

a. Menggunakan Kata Benda untuk Merepresentasikan *Resource*

   Dalam penamaan endpoint, URL hanya boleh berisi kata benda, bukan kata kerja. Dengan kata lain, harus mengacu ke suatu benda/hal bukan pada tindakan. Hal tersebut karna kata

benda itu punya properti, sedangkan kata kerja tidak. Dalam penulisannya huruf kecil lebih dipilih karena lebih nyaman dibaca oleh pengguna dan lebih kosisten. Sedangkan aktifitas yang dilakukan oleh *endpoint* ditunjukkan oleh HTTP *methods* dari *endpoint* tersebut.

b. Menggunakan Garis Miring (/)
   Karakter garis miring (/) digunakan untuk memisahkan antar kata benda. Pada akhir karakter dalam penulisan *resource* tidak perlu menambahkan garis miring.
c. Menggunakan Tanda Hubung (-)
   Tanda hubung ini digunakan untuk memudahkan pengguna dalam membaca dan menafsirkan alamat *resource*. Penggunaan garis bawah (_) tidak disarankan. Hal tersebut karena, garis bawah dapat tak terlihat pada beberapa *browser*.
d. HTTP *Method*
   Terdapat beberapa HTTP *method* yang penting, yaitu
   1. GET
      GET *method* melakukan *request* data dari *resource*.
   2. POST
      POST *method* melakukan *request* pada server untuk membuat (*create*) sebuah *resource* di basis data.
   3. PUT
      PUT *method* melakukan *request* pada server untuk mengupdate resource atau membuat (*create*) resource apabila *resource* tersebut belum tersedia.
   4. DELETE
      DELETE *method* melakukan *request* bahwa *resource* harus dihapus dari basis data.

Berikut merupakan contoh endpoint yang digunakan pada tugas akhir ini:

- **GET customer** untuk mendapatkan semua data customer.
- **POST customer** untuk menambahkan data customer baru.
- **PUT customer** untuk mengubah data customer yang telah tersimpan.

- **DELETE customer** untuk menghapus data customer yang telah tersimpan.

4.2.2 Respon Kode Status HTTP

Ketika klien mengajukan *request* ke server melalui API, klien harus mengetahui umpan balik (*feedback*), apakah *request* tersebut gagal atau permintaan salah. Kode status HTTP sekelompok kode standar yang memiliki berbagai penjelasan dalam berbagai skenario. Terdapat beberapa kategori kode status HTTP, misalnya 200 apabila sukses, seri 400 apabila terjadi masalah pada sisi klien dan seri 500 untuk masalah pada server. Keterangan mengenai respon kode status HTTP terdapat pada Tabel 4.2.

Tabel 4.2 Respon Kode Status HTTP

Kategori	Kode Status	Keterangan
2xx (Success Category)	200 Ok	Respon HTTP standar yang mewakili kesuksesan untuk GET, PUT, atau POST
	201 Created	Respon HTTP yang menunjukan bahwa data berhasil ditambahkan
	204 No Content	Menunjukkan bahwa permintaan berhasil diproses, tetapi tidak ada hasil yang ditampilkan
3xx (Redirection Category)	304 Not Modified	Menunjukkan bahwa klien memiliki respon pada *cache* sehingga tidak perlu lagi mentransfer data yang sama
4xx (Client Error Category)	400 Bad Request	Menunjukkan bahwa permintaan dari klien tidak diproses, karena server tidak dapat memahami apa yang diminta oleh klien
	401 Unauthorized	Menunjukkan bahwa *request* membutuhkan otentikasi dari pengguna

	403 Forbidden	Menunjukkan bahwa server mengetahui *request* yang diinginkan, tetapi ditolak untuk diotorisasi
	404 Not Found	Menunujukkan bahwa request yang diminta tidak ditemukan / tidak tersedia.
	410 Gone	Menunjukkan bahwa *resource* yang diminta tidak lagi tersedia dan dengan sengaja telah dipindahkan.
5xx (Server Error Category)	500 Internal Server Error	Menunjukkan bahwa *request* tersebut valid, tetapi server
	503 Service Unavailable	Menunjukkan bahwa server sedang down atau tidak dapat memproses *request*. Atau server sedang di *maintenance*

4.3 Implementasi API

Pada implementasi API menggunakan bahasa pemrograman PHP dari Laravel. Pada subbab ini akan menjelaskan dan menampilkan kode yang digunakan dalam pembuatan API pada beberapa menu yang terdapat pada modul CRM. Menu dan fitur terdapat pada modul ditunjukkan pada Tabel 4.3.

Tabel 4.3 Menu dan Fitur Modul CRM

Menu	Fitur
Customer	Mengambil semua data Customer
	Mengambil data Customer berdasarkan Id
	Menambah data Customer
	Menghapus data Customer
	Mengubah data Customer
My Pipeline	Mengambil semua data My Pipeline
	Mengambil data Pipeline berdasarkan Id
	Menambah data Pipeline

	Menghapus data Pipeline
	Mengubah data Pipeline
Pipeline Analysis	Mengambil semua data Pipeline Analysis
	Mengambil data Pipeline Analysis berdasarkan Id
	Menambah data Pipeline Analysis
	Menghapus data Pipeline Analysis
	Mengubah data Pipeline Analysis
Sales Teams	Mengambil semua data Sales Teams
	Mengambil data Sales Teams berdasarkan Id
	Menambah data Sales Teams
	Menghapus data Sales Teams
	Mengubah data Sales Teams
Activity Types	Mengambil semua data Activity Types
	Mengambil data Activity Types berdasarkan Id
	Menambah data Activity Types
	Menghapus data Activity Types
	Mengubah data Activity Types
Tags	Mengambil semua data Tags
	Mengambil data Tags berdasarkan Id
	Menambah data Tags
	Menghapus data Tags
	Mengubah data Tags
Lost Reasons	Mengambil semua data Lost Reasons

	Mengambil data Lost Reasons berdasarkan Id
	Menambah data Lost Reasons
	Menghapus data Lost Reasons
	Mengubah data Lost Reasons
Team Pipeline	Mengambil semua data Team Pipeline

4.3.1 Implementasi API pada Menu Customer

Fitur-fitur yang terdapat pada menu Customer adalah melihat, menambah, menghapus, dan mengubah data customer.

4.3.1.1 Mengambil Semua Data Customer

Kode sumber dari implementasi API melihat data customer ditunjukkan pada Kode Sumber 4.1. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.4.

```
1.  public function read()
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $userId = $odoo -> where('active', true)
8.                      -> where('customer', true)
9.                      -> where('x_softdelete', false)
10.                     -> where('parent_id',
11.                              is_null($odoo))
12.
13.                     -> fields('name', 'category_id',
14.                              'function' , 'city',
15.                              'country_id', 'email',
16.                              'opportunity_count',
17.                              'meeting_count', 'image')
18.                     -> get('res.partner');
19.
20.     return json_encode([
21.         "result"=> $userId
22.     ]);
```

```
23.
24. }
```

Kode Sumber 4.1 Mengambil Data Customer

Tabel 4.4 Penjelasan Kode Sumber 4.1

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 11	Menampilkan data berdasarkan kondisi
13 - 17	Menampilkan bagian kolom data yang ingin ditampilkan
18	Menghubungkan ke model Odoo
20 - 22	Mengembalikan *response* setelah *request*

4.3.1.2 Mengambil Data Customer Berdasarkan Id

Kode sumber dari implementasi API melihat data customer berdasarkan Id ditunjukkan pada Kode Sumber 4.2. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.5.

```
1.  public function read_id($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo -> where('id',$id)
8.                      -> where('active', true)
9.                      -> where('customer', true)
10.                     -> where('x_softdelete', false)
11.                     -> where('parent_id',
12.                                 is_null($odoo))
13.                     -> search('res.partner');
14.
```

```
15.     if (count($result)!=0) {
16.        $upd = $odoo ->where('id',$id)
17.                     ->fields('image', 'name',
18.                                 'opportunity_count',
19.                                 'meeting_count',
20.                                 'active', 'city',
21.                                 'country_id',
22.                                 'contact_address',
23.                                 'function', 'phone',
24.                                 'email', 'website',
25.                                 'lang', 'category_id',
26.                                 'child_ids')
27.                     ->get('res.partner');
28.
29.         return json_encode([
30.             "result"=> $upd
31.         ]);
32.     }
33.     else
34.         return json_encode([
35.             'status' => 500,
36.             'message' => "Internal Server Error"
37.         ]);
38. }
```

Kode Sumber 4.2 Mengambil Data Customer Berdasarkan Id

Tabel 4.5 Penjelasan Kode Sumber 4.2

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
15 - 32	Jika id ada, maka menghubungkan ke model Odoo sekaligus menampilkan bagian kolom data yang ingin ditampilkan

No. Baris	Kegunaan
33 - 37	Jika tidak ada id, maka akan mengembalikan *response* setelah *request* sesuai dengan *status* dan *message*

4.3.1.3 Menambah Data Customer

Kode sumber dari implementasi API menambah data customer ditunjukan pada Kode Sumber 4.3. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.6.

```
1.  public function create(Request $r)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $id = $odoo->create('res.partner',
8.          ['name' => $r->name,
9.           'street' => $r->street,
10.          'city' => $r->city,
11.          'country_id' => $r->country_id,
12.          'function' => $r->function,
13.          'phone' => $r->phone,
14.          'email' => $r->email,
15.          'website' => $r->website,
16.          'active' => $r->active,
17.          'customer' => $r->customer,
18.          'supplier' => $r->supplier,
19.          'employee' => $r->employee,
20.          'image' => $r->image,
21.          'x_softdelete' => false
22.          ]);
23.
24.     if ($id==1 or $id==4)
25.         return json_encode([
26.             'status' => 500,
27.             'message' => "Internal Server Error"
28.         ]);
29.     else
30.         return json_encode([
```

```
31.             'status' => 200,
32.             'message' => "Success"
33.         ]);
34.
35. }
```

Kode Sumber 4.3 Menambah Data Customer

Tabel 4.6 Penjelasan Kode Sumber 4.3

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7	Menghubungkan sekaligus untuk menambahkan data melalui model Odoo
8 - 22	Bagian kolom yang mana saja yang akan digunakan berdasarkan dengan form menambahkan data
24 - 28	Jika sesuai dengan kondisi, maka akan mengembalikan *response* sesuai dengan status dan *message*
39 - 33	Jika tidak, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.1.4 Menghapus Data Customer

Kode sumber dari implementasi API menghapus data customer ditunjukan pada Kode Sumber 4.4. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.7.

```
1.  public function delete($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.      $odoo = $odoo->connect();
5.
6.      $result = $odoo->where('id',$id)
7.                      -> search('res.partner');
8.
```

```
9.     if (count($result)!=0) {
10.       $upd = $odoo->where('id',$id)
11.                           ->update('res.partner',[
12.                               'x_softdelete' => true]);
13.       return json_encode([
14.            'status' => 200,
15.            'message' => $upd
16.       ]);
17.    }
18.    else
19.        return json_encode([
20.            'status' => 500,
21.            'message' => "Internal Server Error"
22.        ]);
23.
24. }
```

Kode Sumber 4.4 Menghapus Data Customer

Tabel 4.7 Penjelasan Kode Sumber 4.4

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
4	Menghubungkan ke Odoo
6 - 7	Menghubungkan sekaligus mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
9 - 17	Jika hasilnya ada, maka “id” yang terdelete akan mengubah *softdelete* menjadi true dengan *response* sesuai dengan status dan *message*
18 - 22	Jika hasilnya tidak ada, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.1.5 Mengubah Data Customer

Kode sumber dari implementasi API mengubah data customer ditunjukan pada Kode Sumber 4.5. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.8.

```
1.  public function update(Request $u)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $updated = $odoo->where('id', $u->id)
8.                      ->update('res.partner',[
9.                          'name' => $u->name,
10.                         'street' => $u->street,
11.                         'city' => $u->city,
12.                         'country_id' =>
13.                           $u->country_id,
14.                         'function' => $u->function,
15.                         'phone' => $u->phone,
16.                         'email' => $u->email,
17.                         'website' => $u->website,
18.                         'active' => $u->active,
19.                         'customer' => $u->customer,
20.                         'supplier' => $u->supplier,
21.                         'employee' => $u->employee,
22.                         'image' => $u->image
23.                         ]);
24.
25.     if ($updated=='true')
26.         return json_encode([
27.             'status' => 200,
28.             'message' => $updated
29.         ]);
30.     else
31.         return json_encode([
32.          'status' => 500,
33.          'message' => "Internal Server Error"
34.         ]);
35.
36. }
```

Kode Sumber 4.5 Mengubah Data Customer

Tabel 4.8 Penjelasan Kode Sumber 4.5

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 23	Mengubah data yang telah tersimpan dalam sistem
9 - 17	Jika hasilnya ada, maka "id" yang terdelete akan mengubah *softdelete* menjadi true dengan *response* sesuai dengan status dan *message*
25 - 29	Mengecek apabila data yang dimasukkan benar, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan
30 - 34	Apabila data yang dimasukkan salah, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan

4.3.2 Implementasi API pada Menu My Pipeline

Fitur-fitur yang terdapat pada menu My Pipeline adalah melihat, menambah, menghapus, dan mengubah data Pipeline.

4.3.2.1 Mengambil Semua Data Pipeline

Kode sumber dari implementasi API melihat data pipeline ditunjukan pada Kode Sumber 4.6. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.9.

```
1.  public function readpipeline()
2.  {
3.          $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.          $odoo = $odoo->connect();
6.
7.          $result = $odoo -> fields('date_open',
8.                                    'name', 'partner_id',
9.                                    'country_id',
10.                                   'activity_date_deadline',
11.                                   'activity_summary',
```

```
12.                                 'stage_id',
13.                                 'planned_revenue',
14.                                 'probability', 'team_id',
15.                                 'user_id')
16.                             -> where('active', true)
17.                             -> where('team_id',1)
18.                             -> where('user_id', 2)
19.                             -
    > where('type','=','opportunity')
20.                             -> get('crm.lead');
21.
22.         return $result;
23.
24. }
```

Kode Sumber 4.6 Mengambil Data Pipeline

Tabel 4.9 Penjelasan Kode Sumber 4.6

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 15	Menampilkan bagian kolom data yang ingin ditampilkan
16- 19	Menampilkan data berdasarkan kondisi
20	Menghubungkan ke model Odoo
22	Mengembalikan *response* ketika sedang *request*

4.3.2.2 Mengambil Data Pipeline Berdasarkan Id

Kode sumber dari implementasi API melihat data pipeline berdasarkan Id ditunjukan pada Kode Sumber 4.7. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.10.

```
1.  public function readpipeline_id($id)
2.  {
```

```
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo ->where('id',$id)
8.                      ->where('active', true)
9.                      ->where('team_id',1)
10.                     ->where('user_id', 2)
11.                     ->where('type','=',
12.                             'opportunity')
13.                     ->search('crm.lead');
14.
15.     if (count($result)!=0) {
16.         $upd = $odoo ->where('id',$id)
17.                      ->fields('name', 'stage_id',
18.                               'planned_revenue',
19.                               'probability',
20.                               'partner_id',
21.                               'email_from',
22.                               'phone', 'user_id',
23.                               'team_id',
24.                               'date_deadline',
25.                               'priority',
26.                               'tag_ids',
27.                               'description',
28.                               'partner_name',
29.                               'street', 'city',
30.                               'zip' ,'country_id',
31.                               'website',
32.                               'campaign_id',
33.                               'medium_id',
34.                               'source_id',
35.                               'contact_name',
36.                               'function', 'mobile')
37.                         ->get('crm.lead');
38.
39.         return json_encode([
40.             "result"=> $upd
41.         ]);
42.     }
43.     else
44.         return json_encode([
```

```
45.             'status' => 500,
46.             'message' => "Internal Server Error"
47.         ]);
48. }
```

Kode Sumber 4.7 Mengambil Data Pipeline Berdasarkan Id

Tabel 4.10 Penjelasan Kode Sumber 4.7

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Mengecek atau mencari apakah "id" yang terpilih tersebut berada pada sistem
15 - 42	Jika id ada, maka menghubungkan ke model Odoo sekaligus menampilkan bagian kolom data yang ingin ditampilkan
43 - 47	Jika tidak ada id, maka akan mengembalikan *response* setelah *request* sesuai dengan *status* dan *message*

4.3.2.3 Menambah Data Pipeline

Kode sumber dari implementasi API menambah data pipeline ditunjukan pada Kode Sumber 4.8. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.11.

```
1.  public function createpipeline(Request $r)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $tag_ids = array();
8.      if ($tag_ids != NULL) {
9.          foreach ($r->tag_ids as $variable ) {
10.             array_push($tag_ids,
11.                 array(4,intval($variable)));
```

```
        }
    }

    $id = $odoo->create('crm.lead',
        ['name' => $r->name,
        'planned_revenue' => $r->planned_revenue,
        'probability' => $r->probability,
        'partner_id' => $r->partner_id,
        'email_from' => $r-> email_from,
        'phone' => $r-> phone,
        'user_id' => $r->user_id,
        'team_id' => $r->team_id,
        'date_deadline' => $r->date_deadline,
        'priority' => $r->priority,
        'tag_ids' => $tag_ids,
        'description' => $r->description,
        'partner_name' => $r->partner_name,
        'street' => $r->street,
        'city' => $r->city,
        'zip' => $r->zip,
        'country_id' => $r->country_id,
        'website' => $r->website,
        'campaign_id' => $r->campaign_id,
        'medium_id' => $r->medium_id,
        'source_id' => $r->source_id,
        'contact_name' => $r->contact_name,
        'function' => $r->function,
        'mobile' => $r->mobile

        ]);

    if ($id==1 or $id==4)
        return json_encode([
            'status' => 500,
            'message' => "Internal Server Error"
        ]);
    else
        return json_encode([
            'status' => 200,
            'message' => "success"
        ]);
```

```
53. }
```

Kode Sumber 4.8 Menambah Data Pipeline

Tabel 4.11 Penjelasan Kode Sumber 4.

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Menambahkan *tag* pada form agar lebih dari 1 *tag* dengan menggunakan array
15- 41	Menghubungkan sekaligus untuk menambahkan data melalui model Odoo
43 - 47	Jika sesuai dengan kondisi, maka akan mengembalikan *response* sesuai dengan status dan *message*
48 -52	Jika tidak, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.2.4 Menghapus Data Pipeline

Kode sumber dari implementasi API menghapus data pipeline ditunjukan pada Kode Sumber 4.9. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.12.

```
1.  public function deletepipeline($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo->where('id',$id)
8.                      -> search('crm.lead');
9.
10.     if (count($result)!=0) {
11.        $upd = $odoo->where('id',$id)
12.                      ->delete('crm.lead');
```

```
13.         return json_encode([
14.             'status' => 200,
15.             'message' => $upd
16.         ]);
17.     }
18.     else
19.         return json_encode([
20.             'status' => 500,
21.             'message' => "Internal Server Error"
22.         ]);
23. }
```

Kode Sumber 4.9 Menghapus Data Pipeline

Tabel 4.12 Penjelasan Kode Sumber 4.9

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
4	Menghubungkan ke Odoo
7 - 8	Menghubungkan sekaligus mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
10 - 17	Jika hasilnya ada, maka “id” akan terdelete dengan *response* sesuai dengan status dan *message*
18 - 22	Jika hasilnya tidak ada, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.2.5 Mengubah Data Pipeline

Kode sumber dari implementasi API mengubah data pipeline ditunjukan pada Kode Sumber 4.10. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.13.

```
1.  public function updatepipeline(Request $u)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
```

```
5.      $odoo = $odoo->connect();
6.
7.      $tag_ids = array();
8.      if ($tag_ids != NULL) {
9.          foreach ($u->tag_ids as $variable ) {
10.             array_push($tag_ids,
11.                 array(4,intval($variable)));
12.         }
13.     }
14.
15.     $updated = $odoo->where('id', $u -> id)
16.         ->update('crm.lead',[
17.             'stage_id' => intval($u ->
18.               stage_id),
19.             'name' => $u->name,
20.             'planned_revenue' => $u->
21.                planned_revenue,
22.             'probability' => $u->
23.                probability,
24.             'partner_id' => $u->partner_id,
25.             'email_from' => $u-> email_from,
26.             'phone' => $u-> phone,
27.             'user_id' => $u->user_id,
28.             'team_id' => $u->team_id,
29.             'date_deadline' => $u->
30.                date_deadline,
31.             'priority' => $u->priority,
32.             'tag_ids' => $tag_ids,
33.             'description' => $u->
34.                description,
35.             'partner_name' => $u->
36.                partner_name,
37.             'street' => $u->street,
38.             'city' => $u->city,
39.             'zip' => $u->zip,
40.             'country_id' => $u->country_id,
41.             'website' => $u->website,
42.             'campaign_id' => $u->campaign_id,
43.             'medium_id' => $u->medium_id,
44.             'source_id' => $u->source_id,
45.             'contact_name' => $u->
46.                contact_name,
47.             'function' => $u->function,
```

```
48.             'mobile' => $u->mobile
49.             ]);
50.
51.     if ($updated=='true')
52.         return json_encode([
53.             'status' => 200,
54.             'message' => $updated
55.         ]);
56.     else
57.         return json_encode([
58.          'status' => 500,
59.          'message' => "Internal Server Error"
60.         ]);
61.
62. }
```

Kode Sumber 4.10 Mengubah Data Pipeline

Tabel 4.13 Penjelasan Kode Sumber 4.10

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Menambahkan *tag* pada form agar lebih dari 1 *tag* dengan menggunakan array
15 - 49	Mengubah data yang telah tersimpan dalam sistem
51 - 55	Mengecek apabila data yang dimasukkan benar, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan
56 - 60	Apabila data yang dimasukkan salah, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan

4.3.3 Implementasi API pada Menu Pipeline Analysis

Fitur-fitur yang terdapat pada menu Pipeline Analysis adalah melihat, menambah, menghapus, dan mengubah data Pipeline Analysis.

4.3.3.1 Mengambil Semua Data Pipeline Analysis

Kode sumber dari implementasi API melihat data pipeline analysis ditunjukan pada Kode Sumber 4.11. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.14.

```
1.  public function readpplana()
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo
8.              ->where('active', true)
9.              ->where('team_id','!=',false)
10.             ->where('type','=','opportunity')
11.
12.             ->fields('create_date', 'name',
13.                      'contact_name', 'city',
14.                      'country_id', 'email_from',
15.                      'phone', 'team_id')
16.
17.             ->get('crm.lead');
18.
19.     return $result;
20. }
```

Kode Sumber 4.11 Mengambil Data Pipeline Analysis

Tabel 4.14 Penjelasan Kode Sumber 4.11

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class

No. Baris	Kegunaan
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 10	Menampilkan data berdasarkan kondisi
12 - 15	Menampilkan bagian kolom data yang ingin ditampilkan
17	Menghubungkan ke model Odoo
19	Mengembalikan *response* ketika sedang *request*

4.3.3.2 Mengambil Data Pipeline Analysis Berdasarkan Id

Kode sumber dari implementasi API melihat data pipeline analysis berdasarkan Id ditunjukan pada Kode Sumber 4.12. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.15.

```
1.  public function readpplana_id($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.      $odoo = $odoo->connect();
5.
6.      $result = $odoo
7.              ->where('id',$id)
8.              ->where('active', true)
9.              ->where('team_id','!=',false)
10.             ->where('type','=','opportunity')
11.             ->search('crm.lead');
12.
13.     if (count($result)!=0) {
14.         $upd = $odoo ->where('id',$id)
15.                      ->fields('name',
16.                                  'stage_id',
17.                                  'partner_name',
18.                                  'street', 'city',
19.                                  'zip',
20.                                  'country_id',
21.                                  'website',
22.                                  'user_id',
```

```
23.                                     'team_id',
24.                                     'contact_name',
25.                                     'email_from',
26.                                     'function',
27.                                     'phone', 'mobile',
28.                                     'priority',
29.                                     'tag_ids',
30.                                     'description')
31.                             ->get('crm.lead');
32.
33.     return json_encode([
34.             "result"=> $upd
35.         ]);
36.     }
37.     else
38.         return json_encode([
39.             'status' => 500,
40.             'message' => "Internal Server Error"
41.         ]);
42. }
```

Kode Sumber 4.12 Mengambil Data Pipeline Analysis Berdasarkan Id

Tabel 4.15 Penjelasan Kode Sumber 4.12

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
4	Menghubungkan ke Odoo
6 - 11	Mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
13 - 36	Jika id ada, maka menghubungkan ke model Odoo sekaligus menampilkan bagian kolom data yang ingin ditampilkan
37 - 41	Jika tidak ada id, maka akan mengembalikan *response* setelah *request* sesuai dengan *status* dan *message*

4.3.3.3 Menambah Data Pipeline Analysis

Kode sumber dari implementasi API menambah data pipeline analysis ditunjukan pada Kode Sumber 4.13. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.16.

```
1.  public function createpplana(Request $a)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $tag_ids = array();
8.      if ($tag_ids != NULL) {
9.          foreach ($a->tag_ids as $variable ) {
10.             array_push($tag_ids,
11.                 array(4,intval($variable)));
12.         }
13.     }
14.
15.     $id = $odoo->create('crm.lead',[
16.         'name' => $a->name,
17.         'partner_name' => $a->partner_name,
18.         'street' => $a->street,
19.         'city' => $a->city,
20.         'zip' => $a->zip,
21.         'country_id' => $a->country_id,
22.         'website' => $a->website,
23.         'user_id' => $a->user_id,
24.         'team_id' => $a->team_id,
25.         'contact_name' => $a->contact_name,
26.         'email_from' => $a->email_from,
27.         'function' => $a->function,
28.         'phone' => $a->phone,
29.         'mobile' => $a->mobile,
30.         'priority' => $a->priority,
31.         'tag_ids' => $tag_ids,
32.         'description' => $a->description
33.        ]);
34.
35.     if ($id==1 or $id==4)
```

```
36.         return json_encode([
37.             'status' => 500,
38.             'message' => "Internal Server Error"
39.         ]);
40.     else
41.         return json_encode([
42.             'status' => 200,
43.             'message' => "success"
44.         ]);
45.
46. }
```

Kode Sumber 4.13 Menambah Data Pipeline Analysis

Tabel 4.16 Penjelasan Kode Sumber 4.13

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Menambahkan *tag* pada form agar lebih dari 1 *tag* dengan menggunakan array
15 - 33	Menghubungkan sekaligus untuk menambahkan data melalui model Odoo
35 - 39	Jika sesuai dengan kondisi, maka akan mengembalikan *response* sesuai dengan status dan *message*
40 - 44	Jika tidak, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.3.4 Menghapus Data Pipeline Analysis

Kode sumber dari implementasi API menghapus data pipeline analysis ditunjukan pada Kode Sumber 4.14. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.17.

```
1.  public function deletepplana($id)
2.  {
```

```
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo ->where('id',$id)
8.                      -> search('crm.lead');
9.
10.     if (count($result)!=0) {
11.
12.        $upd = $odoo -> where('id',$id)
13.                     -> delete('crm.lead');
14.         return json_encode([
15.             'status' => 200,
16.             'message' => $upd
17.         ]);
18.     }
19.     else
20.         return json_encode([
21.             'status' => 500,
22.             'message' => "Internal Server Error"
23.         ]);
24. }
```

Kode Sumber 4.14 Menghapus Data Pipeline Analysis

Tabel 4.17 Penjelasan Kode Sumber 4.14

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
4	Menghubungkan ke Odoo
7 - 8	Menghubungkan sekaligus mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
10 - 18	Jika hasilnya ada, maka “id” akan terdelete dengan *response* sesuai dengan status dan *message*
19 - 22	Jika hasilnya tidak ada, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.3.5 Mengubah Data Pipeline Analysis

Kode sumber dari implementasi API mengubah data pipeline analysis ditunjukan pada Kode Sumber 4.15. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.18.

```
1.  public function updatepplana(Request $a)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $tag_ids = array();
8.      if ($tag_ids != NULL) {
9.          foreach ($a->tag_ids as $variable ) {
10.             array_push($tag_ids,
11.                 array(4,intval($variable)));
12.         }
13.     }
14.
15.     $updated = $odoo->where('id', $a -> id)
16.         ->update('crm.lead',[
17.             'name' => $a->name,
18.             'partner_name' => $a->partner_name,
19.             'street' => $a->street,
20.             'city' => $a->city,
21.             'zip' => $a->zip,
22.             'country_id' => intval($a ->
23.                country_id),
24.             'website' => $a->website,
25.             'user_id' => $a->user_id,
26.             'team_id' => $a->team_id,
27.             'contact_name' => $a->contact_name,
28.             'email_from' => $a->email_from,
29.             'function' => $a->function,
30.             'phone' => $a->phone,
31.             'mobile' => $a->mobile,
32.             'priority' => $a->priority,
33.             'tag_ids' => $tag_ids,
34.             'description' => $a->description
35.         ]);
```

```
36.
37.     if ($updated=='true')
38.         return json_encode([
39.             'status' => 200,
40.             'message' => $updated
41.         ]);
42.     else
43.         return json_encode([
44.          'status' => 500,
45.          'message' => "Internal Server Error"
46.         ]);
47. }
```

Kode Sumber 4.15 Mengubah Data Pipeline Analysis

Tabel 4.18 Penjelasan Kode Sumber 4.15

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Menambahkan *tag* pada form agar lebih dari 1 *tag* dengan menggunakan array
15 - 35	Mengubah data yang telah tersimpan dalam sistem
37 - 41	Mengecek apabila data yang dimasukkan benar, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan
42 - 46	Apabila data yang dimasukkan salah, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan

4.3.4 Implementasi API pada Menu Sales Teams

Fitur-fitur yang terdapat pada menu Sales Teams adalah melihat, menambah, menghapus, dan mengubah data Sales Teams.

4.3.4.1 Mengambil Semua Data Sales Teams

Kode sumber dari implementasi API melihat data sales teams pada Kode Sumber 4.16. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.19.

```
1.  public function readtim()
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo
8.              -> fields('name')
9.
10.             -> get('crm.team');
11.
12.     return $result;
13. }
```

Kode Sumber 4.16 Mengambil Data Sales Teams

Tabel 4.19 Penjelasan Kode Sumber 4.16

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 8	Menampilkan bagian kolom data yang ingin ditampilkan
10	Menghubungkan ke model Odoo
12	Mengembalikan *response* ketika sedang *request*

4.3.4.2 Mengambil Data Sales Teams Berdasarkan Id

Kode sumber dari implementasi API melihat data sales teams berdasarkan Id berdasarkan Id ditunjukan pada Kode Sumber 4.17. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.20.

```
1.  public function readtim_id($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo ->where('id',$id)
8.                      ->search('crm.team');
9.
10.     if (count($result)!=0) {
11.         $upd = $odoo ->where('id',$id)
12.                      ->fields('active',
13.                               'name',
14.                               'user_id',
15.                               'alias_name',
16.                               'dashboard_button_name',
17.                               'dashboard_graph_period',
18.                               'dashboard_graph_group')
19.                          ->get('crm.team');
20.
21.         return json_encode([
22.             "result"=> $upd
23.         ]);
24.     }
25.     else
26.         return json_encode([
27.             'status' => 500,
28.             'message' => "Internal Server Error"
29.         ]);
30. }
```

Kode Sumber 4.17 Mengambil Data Sales Teams Berdasarkan Id

Tabel 4.20 Penjelasan Kode Sumber 4.17

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class

No. Baris	Kegunaan
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 – 8	Mengecek atau mencari apakah "id" yang terpilih tersebut berada pada sistem
10 - 24	Jika id ada, maka menghubungkan ke model Odoo sekaligus menampilkan bagian kolom data yang ingin ditampilkan
25 - 29	Jika tidak ada id, maka akan mengembalikan *response* setelah *request* sesuai dengan *status* dan *message*

4.3.4.3 Menambah Data Sales Teams

Kode sumber dari implementasi API menambah data sales teams ditunjukan pada Kode Sumber 4.18. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.21.

```
1.  public function createtim(Request $a)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $member_ids = array();
8.      if ($member_ids != NULL) {
9.          foreach ($a->member_ids as $variable ) {
10.             array_push($member_ids,
11.                 array(4,intval($variable)));
12.         }
13.     }
14.
15.     $id = $odoo->create('crm.team',[
16.                             'name' => $a->name,
17.                             'user_id' => $a->user_id,
18.                             'alias_name' => $a->alias_name,
19.                             'member_ids' => $member_ids,
20.                             'use_leads' => true,
21.                             'use_opportunities' => true
```

```
                        ]);

    if ($id==1 or $id==4)
        return json_encode([
            'status' => 500,
            'message' => "Internal Server Error"
        ]);
    else
        return json_encode([
            'status' => 200,
            'message' => "success"
        ]);

}
```

Kode Sumber 4.18 Menambah Data Sales Teams

Tabel 4.21 Penjelasan Kode Sumber 4.18

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Menambahkan *member* pada form agar lebih dari 1 *member* dengan menggunakan array
15 - 22	Menghubungkan sekaligus untuk menambahkan data melalui model Odoo
24 - 28	Jika sesuai dengan kondisi, maka akan mengembalikan *response* sesuai dengan status dan *message*
29 - 33	Jika tidak, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.4.4 Menghapus Data Sales Teams

Kode sumber dari implementasi API menghapus data sales teams ditunjukan pada Kode Sumber 4.19. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.22.

```
1.  public function deletetim($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo-> connect();
6.
7.      $result = $odoo-> where('id',$id)
8.                     -> search('crm.team');
9.
10.     if (count($result)!=0) {
11.
12.        $upd = $odoo-> where('id',$id)
13.                     -> delete('crm.team');
14.         return json_encode([
15.             'status' => 200,
16.             'message' => $upd
17.         ]);
18.     }
19.     else
20.         return json_encode([
21.             'status' => 500,
22.             'message' => "Internal Server Error"
23.         ]);
24. }
```

Kode Sumber 4.19 Menghapus Data Sales Teams

Tabel 4.22 Penjelasan Kode Sumber 4.19

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 8	Menghubungkan sekaligus mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
10 - 18	Jika hasilnya ada, maka “id” akan terdelete dengan *response* sesuai dengan status dan *message*

No. Baris	Kegunaan
19 - 23	Jika hasilnya tidak ada, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.4.5 Mengubah Data Sales Teams

Kode sumber dari implementasi API mengubah data sales teams ditunjukan pada Kode Sumber 4.20. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.23.

```
1.  public function updatetim(Request $u)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $member_ids = array();
8.      if ($member_ids != NULL) {
9.          foreach ($u->member_ids as $variable ) {
10.             array_push($member_ids,
11.                 array(4,intval($variable)));
12.         }
13.     }
14.
15.     $updated = $odoo->where('id', $u->id)
16.         ->update('crm.team',[
17.             'name' => $u->name,
18.             'user_id' => $u->user_id,
19.             'alias_name' => $u->alias_name,
20.             'member_ids' => $member_ids,
21.             'use_leads' => true,
22.             'use_opportunities' => true
23.         ]);
24.
25.     if ($updated=='true')
26.         return json_encode([
27.             'status' => 200,
28.             'message' => $updated
29.         ]);
30.     else
```

```
31.         return json_encode([
32.          'status' => 500,
33.          'message' => "Internal Server Error"
34.         ]);
35. }
```

Kode Sumber 4.20 Mengubah Data Sales Teams

Tabel 4.23 Penjelasan Kode Sumber 4.20

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Menambahkan *member* pada form agar lebih dari 1 *member* dengan menggunakan array
15 - 23	Mengubah data yang telah tersimpan dalam sistem
25 - 29	Mengecek apabila data yang dimasukkan benar, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan
30 - 34	Apabila data yang dimasukkan salah, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan

4.3.5 Implementasi API pada Menu Activity Types

Fitur-fitur yang terdapat pada menu Activity Types adalah melihat, menambah, menghapus, dan mengubah data Activity Types.

4.3.5.1 Mengambil Semua Data Activity Types

Kode sumber dari implementasi API melihat data activity types ditunjukan pada Kode Sumber 4.21. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.24.

```
1.  public function readact()
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
```

```
4.      $odoo = $odoo->connect();
5.
6.      $result = $odoo
7.              -> fields('name', 'delay_count',
8.                        'delay_unit',
9.                        'delay_from')
10.             -> get('mail.activity.type');
11.
12.     return $result;
13. }
```

Kode Sumber 4.21 Mengambil Data Activity Types

Tabel 4.24 Penjelasan Kode Sumber 4.21

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
4	Menghubungkan ke Odoo
6 - 9	Menampilkan bagian kolom data yang ingin ditampilkan
10	Menghubungkan ke model Odoo
12	Mengembalikan *response* ketika sedang *request*

4.3.5.2 Mengambil Data Activity Types Berdasarkan Id

Kode sumber dari implementasi API melihat data activity types berdasarkan Id ditunjukan pada Kode Sumber 4.22. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.25.

```
1.  public function readact_id($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo ->where('id',$id)
8.                      ->search('mail.activity.type');
```

```
9.
10.     if (count($result)!=0) {
11.         $upd = $odoo-> where('id',$id)
12.                     -> fields('active', 'name',
13.                                 'category', 'summary',
14.                                 'delay_count',
15.                                 'delay_unit',
16.                                 'delay_from')
17.                     -> get('mail.activity.type');
18.
19.         return json_encode([
20.             "result"=> $upd
21.         ]);
22.     }
23.     else
24.         return json_encode([
25.             'status' => 500,
26.             'message' => "Internal Server Error"
27.         ]);
28. }
```

Kode Sumber 4.22 Mengambil Data Activity Types Berdasarkan Id

Tabel 4.25 Penjelasan Kode Sumber 4.22

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 – 8	Mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
10 - 22	Jika id ada, maka menghubungkan ke model Odoo sekaligus menampilkan bagian kolom data yang ingin ditampilkan
23 - 27	Jika tidak ada id, maka akan mengembalikan *response* setelah *request* sesuai dengan *status* dan *message*

4.3.5.3 Menambah Data Activity Types

Kode sumber dari implementasi API menambah data activity types ditunjukan pada Kode Sumber 4.23. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.26.

```
1.  public function createact(Request $a)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo->create('mail.activity.type',[
8.          'name' => $a->name,
9.          'category' => $a->category,
10.         'summary' => $a-summary,
11.         'delay_count' => $a->delay_count]);
12.
13.     if ($result==1 or $result==4)
14.         return json_encode([
15.             'status' => 500,
16.             'message' => "Internal Server Error"
17.         ]);
18.     else
19.         return json_encode([
20.             'status' => 200,
21.             'message' => "success"
22.         ]);
23.
24. }
```

Kode Sumber 4.23 Menambah Data Activity Types

Tabel 4.26 Penjelasan Kode Sumber 4.23

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo

No. Baris	Kegunaan
5	Menghubungkan ke Odoo
7 - 11	Menghubungkan sekaligus untuk menambahkan data melalui model Odoo
13 - 17	Jika sesuai dengan kondisi, maka akan mengembalikan *response* sesuai dengan status dan *message*
18 - 22	Jika tidak, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.5.4 Menghapus Data Activity Types

Kode sumber dari implementasi API menghapus data activity types ditunjukan pada Kode Sumber 4.24. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.27.

```
1.  public function deleteact($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo->where('id',$id)
8.              -> search('mail.activity.type');
9.
10.     if (count($result)!=0) {
11.
12.        $upd = $odoo->where('id',$id)
13.                    ->delete('mail.activity.type');
14.         return json_encode([
15.             'status' => 200,
16.             'message' => $upd
17.         ]);
18.     }
19.     else
20.         return json_encode([
21.             'status' => 500,
22.             'message' => "Internal Server Error"
23.         ]);
```

```
24. }
```

Kode Sumber 4.24 Menghapus Data Activity Types

Tabel 4.27 Penjelasan Kode Sumber 4.24

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 8	Menghubungkan sekaligus mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
10 - 18	Jika hasilnya ada, maka “id” akan terdelete dengan *response* sesuai dengan status dan *message*
19 - 23	Jika hasilnya tidak ada, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.5.5 Mengubah Data Activity Types

Kode sumber dari implementasi API mengubah data activity types ditunjukan pada Kode Sumber 4.25. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.28.

```
1.  public function updateact(Request $u)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $updated = $odoo->where('id', $u->id)
8.          ->update('mail.activity.type',[
9.              'name' => $u->name,
10.             'category' => $u-> category,
11.             'summary' => $u-> summary,
12.             'delay_count' => $u->delay_count
13.         ]);
14.
```

```
15.     if ($updated=='true')
16.         return json_encode([
17.             'status' => 200,
18.             'message' => $updated
19.         ]);
20.     else
21.         return json_encode([
22.          'status' => 500,
23.          'message' => "Internal Server Error"
24.         ]);
25. }
```

Kode Sumber 4.25 Mengubah Data Activity Types

Tabel 4.28 Penjelasan Kode Sumber 4.25

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 13	Mengubah data yang telah tersimpan dalam sistem
15 - 19	Mengecek apabila data yang dimasukkan benar, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan
20 - 24	Apabila data yang dimasukkan salah, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan

4.3.6 Implementasi API pada Menu Tags

Fitur-fitur yang terdapat pada menu Tags adalah melihat, menambah, menghapus, dan mengubah data Tags.

4.3.6.1 Mengambil Semua Data Tags

Kode sumber dari implementasi API melihat data tag ditunjukan pada Kode Sumber 4.26. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.29.

```
1.  public function readtags()
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo -> fields('name')
8.                      -> get('crm.lead.tag');
9.
10.     return $result;
11. }
```

Kode Sumber 4.26 Mengambil Semua Data Tags

Tabel 4.29 Penjelasan Kode Sumber 4.26

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7	Menampilkan bagian kolom data yang ingin ditampilkan
8	Menghubungkan ke model Odoo
10	Mengembalikan *response* ketika sedang *request*

4.3.6.2 Mengambil Data Tags Berdasarkan Id

Kode sumber dari implementasi API melihat data tags berdasarkan Id ditunjukan pada Kode Sumber 4.27. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.30.

```
1.  public function readtags_id($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
```

```
6.
7.      $result = $odoo ->where('id',$id)
8.                      ->search('crm.team');
9.
10.     if (count($result)!=0) {
11.         $upd = $odoo-> where('id',$id)
12.                     -> fields('name')
13.                     -> get('crm.lead.tag');
14.
15.         return json_encode([
16.             "result"=> $upd
17.         ]);
18.     }
19.     else
20.         return json_encode([
21.             'status' => 500,
22.             'message' => "Internal Server Error"
23.         ]);
24. }
```

Kode Sumber 4.27 Mengambil Data Tags Berdasarkan Id

Tabel 4.30 Penjelasan Kode Sumber 4.27

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 – 8	Mengecek atau mencari apakah “id” yang terpilih tersebut berada pada sistem
10 - 18	Jika id ada, maka menghubungkan ke model Odoo sekaligus menampilkan bagian kolom data yang ingin ditampilkan
19 - 23	Jika tidak ada id, maka akan mengembalikan *response* setelah *request* sesuai dengan *status* dan *message*

4.3.6.3 Menambah Data Tags

Kode sumber dari implementasi API menambah data tags ditunjukan pada Kode Sumber 4.28. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.31.

```
1.  public function createtags(Request $t)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $tag = $odoo->create('crm.lead.tag',
8.                          ['name' => $t->name]);
9.
10.     return json_encode([
11.         'status' => 200,
12.         'message' => $tag
13.     ]);
14. }
```

Kode Sumber 4.28 Menambah Data Tags

Tabel 4.31 Penjelasan Kode Sumber 4.28

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 8	Menghubungkan sekaligus untuk menambahkan data melalui model Odoo
10 - 13	Mengembalikan *response* (id yang di dapat) setelah *request*

4.3.6.4 Menghapus Data Tags

Kode sumber dari implementasi API menghapus data tags ditunjukan pada Kode Sumber 4.29. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.32.

```
1.  public function deletetags($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo-> where('id',$id)
8.                      -> search('crm.lead.tag');
9.
10.     if (count($result)!=0) {
11.
12.        $upd = $odoo-> where('id',$id)
13.                     -> delete('crm.lead.tag');
14.         return json_encode([
15.             'status' => 200,
16.             'message' => $upd
17.         ]);
18.     }
19.     else
20.         return json_encode([
21.             'status' => 500,
22.             'message' => "Internal Server Error"
23.         ]);
24. }
```

Kode Sumber 4.29 Menghapus Data Tags

Tabel 4.32 Penjelasan Kode Sumber 4.29

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo

No. Baris	Kegunaan
5	Menghubungkan ke Odoo
7 - 8	Menghubungkan sekaligus mengecek atau mencari apakah "id" yang terpilih tersebut berada pada sistem
10 - 18	Jika hasilnya ada, maka "id" akan terdelete dengan *response* sesuai dengan status dan *message*
19 - 23	Jika hasilnya tidak ada, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.6.5 Mengubah Data Tags

Kode sumber dari implementasi API mengubah data tags ditunjukan pada Kode Sumber 4.30. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.33.

```
1.  public function updatetags(Request $ut)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $updated = $odoo->where('id', $ut->id)
8.          ->update('crm.lead.tag',[
9.              'name' => $ut->name
10.         ]);
11.
12.     if ($updated=='true')
13.         return json_encode([
14.             'status' => 200,
15.             'message' => $updated
16.         ]);
17.     else
18.         return json_encode([
19.          'status' => 500,
20.          'message' => "Internal Server Error"
21.         ]);
22. }
```

Kode Sumber 4.30 Mengubah Data Tags

Tabel 4.33 Penjelasan Kode Sumber 4.30

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 10	Mengubah data yang telah tersimpan dalam sistem
12 - 16	Mengecek apabila data yang dimasukkan benar, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan
17 - 21	Apabila data yang dimasukkan salah, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan

4.3.7 Implementasi API pada Menu Lost Reasons

Fitur-fitur yang terdapat pada menu Lost Reasons adalah melihat, menambah, menghapus, dan mengubah data Lost Reasons.

4.3.7.1 Mengambil Semua Data Lost Reasons

Kode sumber dari implementasi API melihat data lost reasons ditunjukan pada Kode Sumber 4.31. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.34.

```
1.  public function readlost()
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.      $odoo = $odoo->connect();
5.
6.      $result = $odoo -> fields('name')
7.                      -> get('crm.lost.reason');
8.
9.      return $result;
10. }
```

Kode Sumber 4.31 Mengambil Data Lost Reasons

Tabel 4.34 Penjelasan Kode Sumber 4.31

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
4	Menghubungkan ke Odoo
6	Menampilkan bagian kolom data yang ingin ditampilkan
7	Menghubungkan ke model Odoo
9	Mengembalikan *response* ketika sedang *request*

4.3.7.2 Mengambil Data Lost Reasons Berdasarkan Id

Kode sumber dari implementasi API melihat data lost reasons berdasarkan Id ditunjukan pada Kode Sumber 4.32. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.35.

```
1.  public function readlost_id($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo -> where('id',$id)
8.                      -> search('crm.lost.reason');
9.
10.     if (count($result)!=0) {
11.         $upd = $odoo -> where('id',$id)
12.                      -> fields('name', 'active')
13.                      -> get('crm.lost.reason');
14.
15.         return json_encode([
16.             "result"=> $upd
17.         ]);
18.     }
19.     else
20.         return json_encode([
```

```
            'status' => 500,
            'message' => "Internal Server Error"
        ]);
}
```

Kode Sumber 4.32 Mengambil Data Lost Reasons Berdasarkan id

Tabel 4.35 Penjelasan Kode Sumber 4.32

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 – 8	Mengecek atau mencari apakah "id" yang terpilih tersebut berada pada sistem
10 - 18	Jika id ada, maka menghubungkan ke model Odoo sekaligus menampilkan bagian kolom data yang ingin ditampilkan
19 - 23	Jika tidak ada id, maka akan mengembalikan *response* setelah *request* sesuai dengan *status* dan *message*

4.3.7.3 Menambah Data Lost Reasons

Kode sumber dari implementasi API menambah data lost reasons ditunjukan pada Kode Sumber 4.33. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.36.

```
1.  public function createlost(Request $l)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $lost = $odoo->create('crm.lost.reason',
8.                          ['name' => $l->name]);
9.
10.     return json_encode([
```

```
11.         'status' => 200,
12.         'message' => $lost
13.     ]);
14. }
```

Kode Sumber 4.33 Menambah Data Lost Reasons

Tabel 4.36 Penjelasan Kode Sumber 4.33

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 8	Menghubungkan sekaligus untuk menambahkan data melalui model Odoo
10 - 13	Mengembalikan *response* (id yang di dapat) setelah *request*

4.3.7.4 Menghapus Data Lost Reasons

Kode sumber dari implementasi API menghapus data lost reasons ditunjukan pada Kode Sumber 4.34. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.37.

```
1.  public function deletelost($id)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
5.      $odoo = $odoo->connect();
6.
7.      $result = $odoo-> where('id',$id)
8.                      -> search('crm.lost.reason');
9.
10.     if (count($result)!=0) {
11.
12.         $upd = $odoo->where('id',$id)
13.                     ->delete('crm.lost.reason');
```

```
14.         return json_encode([
15.             'status' => 200,
16.             'message' => $upd
17.         ]);
18.     }
19.     else
20.         return json_encode([
21.             'status' => 500,
22.             'message' => "Internal Server Error"
23.         ]);
24. }
```

Kode Sumber 4.34 Menghapus Data Lost Reasons

Tabel 4.37 Penjelasan Kode Sumber 4.34

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 8	Menghubungkan sekaligus mengecek atau mencari apakah "id" yang terpilih tersebut berada pada sistem
10 - 18	Jika hasilnya ada, maka "id" akan terdelete dengan *response* sesuai dengan status dan *message*
19 - 23	Jika hasilnya tidak ada, maka akan mengembalikan *response* sesuai dengan status dan *message*

4.3.7.5 Mengubah Data Lost Reasons

Kode sumber dari implementasi API mengubah data lost reasons ditunjukan pada Kode Sumber 4.35. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.38.

```
1.  public function updatelost(Request $ut)
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.
```

```
5.      $odoo = $odoo->connect();
6.
7.      $updated = $odoo->where('id', $ut->id)
8.          ->update('crm.lost.reason',[
9.              'name' => $ut->name
10.         ]);
11.
12.     if ($updated=='true')
13.         return json_encode([
14.             'status' => 200,
15.             'message' => $updated
16.         ]);
17.     else
18.         return json_encode([
19.          'status' => 500,
20.          'message' => "Internal Server Error"
21.         ]);
22. }
23.
```

Kode Sumber 4.35 Mengubah Data Lost Reasons

Tabel 4.38 Penjelasan Kode Sumber 4.35

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
5	Menghubungkan ke Odoo
7 - 10	Mengubah data yang telah tersimpan dalam sistem
12 - 16	Mengecek apabila data yang dimasukkan benar, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan
17 - 21	Apabila data yang dimasukkan salah, maka sistem mengembalikan *response* sesuai dengan status dan *message* yang telah ditentukan

4.3.8 Implementasi API pada Menu Team Pipelines

Fitur-fitur yang terdapat pada menu Team Pipelines adalah melihat data Team Pipelines.

4.3.8.1 Mengambil Semua Data Team Pipelines

Kode sumber dari implementasi API melihat data team pipelines ditunjukan pada Kode Sumber 4.36. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.39.

```
1.  public function readteamppl()
2.  {
3.      $odoo = new \Edujugon\Laradoo\Odoo();
4.      $odoo = $odoo->connect();
5.
6.      $result = $odoo -> where('id', 1)
7.              -> fields('name',
8.                        'opportunities_count',
9.                        'opportunities_amount')
10.             -> get('crm.team');
11.
12.     return $result;
13. }
```

Kode Sumber 4.36 Mengambil Data Team Pipeline

Tabel 4.39 Penjelasan Kode Sumber 4.36

No. Baris	Kegunaan
1	Mendeklarasikan method public, agar bisa diakses dari luar class
3	Instance main Odoo class karena menggunakan Laradoo
4	Menghubungkan ke Odoo
6	Menampilkan data berdasarkan kondisi
7 - 9	Menampilkan bagian kolom data yang ingin ditampilkan
10	Menghubungkan ke model Odoo
12	Mengembalikan *response* ketika sedang *request*

4.4 Implementasi Offline Storage

Pada tugas akhir ini, pembuatan *offline storage* pada perangkat Android diimplementasikan menggunakan Couchbase Lite yang merupakan produk Couchbase yang khusus untuk mobile, sehingga bahasa yang digunakan yaitu bahasa pemrograman Java. Dalam subbab ini akan menjelaskan dan menampilkan kode yang digunakan dalam pengimplementasian *offline storage* dengan menggunakan Couchbase Lite dan Android Studio. Terdapat lima fitur yang dibuat, yaitu mengambil, menambah, mengubah, menghapus, dengan kondisi perangkat dalam keadaan *offline* serta sinkronisasi data antara Couchbase Lite dan server Odoo yang dilakukan saat perangkat kembali mendapatkan koneksi internet.

Dalam implementasi *offline storage*, dalam tugas akhir ini menggunakan menu Customer yang dikembangkan dengan menambahkan *field* x_softdelete pada model res.partner. Penambahan *field* ini bertujuan agar fitur menghapus data secara *offline* pada perangkat Android dapat diimplementasikan. *Field* x_softdelete ini memiliki tipe data boolean. Value dari *field* ini sendiri akan bernilai *false* ketika data belum dihapus dan akan bernilai *true* ketika data telah dihapus melalui perangkat Android. Hal inilah yang menyebabkan perbedaan data yang ditampilkan antara di aplikasi web dan aplikasi Android. Kode sumber dari implementasi *offline storage* ditunjukan pada Kode Sumber 4.37. Untuk penjelasan kode sumber ditunjukkan pada Tabel 4.40.

```
1.  Manager manager = null;
2.  try {
3.      manager = new Manager(new AndroidContext(
4.          getApplicationContext()),
5.          Manager.DEFAULT_OPTIONS);
6.  } catch (IOException e) {
7.      e.printStackTrace();
8.  }
9.
10. Database database = null;
```

```
11. try {
12.     database = manager.getDatabase(app);
13. } catch (CouchbaseLiteException e) {
14.     e.printStackTrace();
15. }
16.
17. MapString, Object properties =
18.     new HashMapString, Object();
19. properties.put(title, Couchbase Mobile);
20. properties.put(sdk, Android);
21.
22. Document document = database.createDocument();
23.
24. try {
25.     document.putProperties(properties);
26. } catch (CouchbaseLiteException e) {
27.     e.printStackTrace();
28. }
29.
30. Log.d(app, String.format(Document ID :: %s,
31.     document.getId()));
32. Log.d(app, String.format(Learning %s with %s,
33.     (String) document.getProperty(title),
34.     (String) document.getProperty(sdk)));
```

Kode Sumber 4.37 Implementasi *Offline Storage*

Tabel 4.40 Penjelasan Kode Sumber 4.37

No. Baris	Kegunaan
1 - 8	Membuat objek bernama "manager" untuk mewakili atau instance dari objek Couchbase Lite sendiri. Pada parameter Manager.DEFAULT_OPTIONS menunjukkan opsi default.
10 - 15	Membuat atau membuka database yang memiliki nama "app"
17 - 20	"properties" yang akan disimpan pada sebuah dokumen
22	Membuat dokumen baru
24 - 28	Menyimpan dokumen ke database

No. Baris	Kegunaan
30 - 34	Mencatat Id dari dokumen yang dihasilkan dari database *offline storage*

4.4.1 Struktur Data Couchbase Lite

Struktur data pada Couchbase memiliki konsep yang mirip dengan struktur data dalam Javascript. Struktur data ini disimpan sebagai dokumen JSON di Couchbase. Couchbase tidak menggunakan tabel dalam penyimpanan datanya, melainkan dalam bentuk dokumen. Setiap dokumen di Couchbase dikaitkan dengan *unique key* yang harus disediakan oleh pengguna saat dokumen dibuat. *Key* juga disebut sebagai ID dokumen.


```
[
  {
    "id": 117,
    "name": "contoh nama",
    "street": "3404  Edgewood Road",
    "city": "Jonesboro",
    "country_id": [
      233,
      "United States"
    ],
    "function": "",
    "phone": "435364363633",
    "email": "email@yahoo.com",
    "website": "http://website.co.id",
    "image": "",
    "active": true,
    "customer": true,
    "supplier": false,
    "employee": false,
    "x_softdelete": false
    "uploaded":"0".
  }
]
```


Gambar 4.1 Struktur Data pada Menu Customer

4.4.2 Mengambil Data saat *Offline*

Kode gambar dari implementasi pengambilan data saat perangkat dalam keadaan *offline* ditunjukkan pada Kode Sumber 4.38.

```
1.  private void readOffline() {
2.      try {
3.          DatabaseConfiguration config = new
4.              DatabaseConfiguration
5.              (getApplicationContext());
6.          Database database = new Database("customer"
    ,
7.              config);
8.
9.          ResultHome docs = new ResultHome();
10.         docs.result = new ArrayList<>();
11.
12.         Query query = QueryBuilder
13.                 .select(SelectResult.expression(Met
    a.id),
14.                         SelectResult.property("nama
    "),
15.                         SelectResult.property("imag
    e"),
16.                         SelectResult.property("uplo
    aded"))
17.                 .from(DataSource.database(database)
    );
18.
19.         try {
20.             ResultSet rs = query.execute();
21.             for (Result result : rs) {
22.                 ReturnRead returnRead = new ReturnR
    ead();
23.                 Log.d(Const.TAG, "readOffline id: "
     +
24.                    result.getString("id"));
25.                 Log.d(Const.TAG, "status upload: "
    +
26.                    result.getString("uploaded"));
27.                 returnRead.setId(result.getString
28.                   ("customerId"));
29.                 returnRead.setNama(result.getString

30.                   ("nama"));
31.                 returnRead.setImage(result.getStrin
    g
```

```
                    ("image"));
                docs.result.add(returnRead);
            }
        } catch (CouchbaseLiteException e) {
            Log.e("Sample", e.getLocalizedMessage()
);
        }

        adapter = new CustomerAdapter(docs.result,

           getApplicationContext());
        recyclerView.setAdapter(adapter);
        adapter.setListener(this);
        recyclerView.setLayoutManager(LayoutUtils.

           createGridLayoutManager(getApplicationCo
ntext(),
           RecyclerView.VERTICAL, 2));
        adapter.notifyDataSetChanged();

    } catch (CouchbaseLiteException e) {
        e.printStackTrace();
    }
}
```

Kode Sumber 4.38 Implementasi Pengambilan Data saat *Offline*

4.4.3 Menambah Data saat *Offline*

Kode gambar dari implementasi penambahan data saat perangkat dalam keadaan *offline* ditunjukkan pada Kode Sumber 4.39.

```
public void onFailure(Call<ReturnCreate> call,
   Throwable t) {
    Toast.makeText(AddCustomerActivity.this, "gagal
",
       Toast.LENGTH_SHORT).show();

    DatabaseConfiguration config = new DatabaseConf
iguration
       (getApplicationContext());
```

```
8.      Database database = null;
9.      try {
10.         database = new Database("customer", config)
    ;
11.     } catch (CouchbaseLiteException e) {
12.         e.printStackTrace();
13.     }
14.     MutableDocument docEdit = new MutableDocument(i
    dCustomer);
15.     docEdit.setString("id", idCustomer);
16.     docEdit.setString("nama", et_customer_name.
17.        getText().toString());
18.     docEdit.setString("street", et_customer_street.

19.        getText().toString());
20.     docEdit.setString("city", et_customer_city.
21.        getText().toString());
22.     docEdit.setString("country_id", list_id);
23.     docEdit.setString("function", et_customer_funct
    ion.
24.        getText().toString());
25.     docEdit.setString("phone", et_customer_phone.
26.        getText().toString());
27.     docEdit.setString("email", et_customer_email.
28.        getText().toString());
29.     docEdit.setString("website", et_customer_websit
    e.
30.        getText().toString());
31.     if(cb_customer_active.isChecked())
32.        docEdit.setString("active", "1");
33.     else
34.        docEdit.setString("active", "0");
35.     if (cb_customer_customer.isChecked())
36.        docEdit.setString("customer", "1");
37.     else
38.        docEdit.setString("customer", "0");
39.     if(cb_customer_supplier.isChecked())
40.        docEdit.setString("supplier", "1");
41.     else
42.        docEdit.setString("supplier", "0");
43.     if(cb_customer_employee.isChecked())
44.        docEdit.setString("employee", "1");
45.     else
```

```
46.         docEdit.setString("employee", "0");
47.     docEdit.setString("image", base64);
48.     docEdit.setString("uploaded", "0");
49.
50.          try {
51.              database.save(docEdit);
52.          } catch (CouchbaseLiteException e) {
53.             e.printStackTrace();
54.          }
55. }
56.
```

Kode Sumber 4.39 Implementasi Penambahan Data saat *Offline*

4.4.4 Mengubah Data saat *Offline*

Kode gambar dari implementasi pengubahan data saat perangkat dalam keadaan *offline* ditunjukkan pada Kode Sumber 4.40.

```
1.  public void onFailure(Call<ReturnCreate> call,
2.     Throwable t) {
3.      Toast.makeText(AddCustomerActivity.this, "gagal
    ",
4.          Toast.LENGTH_SHORT).show();
5.
6.      DatabaseConfiguration config = new DatabaseConf
    iguration
7.          (getApplicationContext());
8.      Database database = null;
9.      try {
10.          database = new Database("customer", config)
    ;
11.     } catch (CouchbaseLiteException e) {
12.          e.printStackTrace();
13.     }
14.     MutableDocument docEdit = new MutableDocument(i
    dCustomer);
15.     docEdit.setString("id", idCustomer);
16.     docEdit.setString("nama", et_customer_name.
17.         getText().toString());
```

```
18.     docEdit.setString("street", et_customer_street.

19.        getText().toString());
20.     docEdit.setString("city", et_customer_city.
21.        getText().toString());
22.     docEdit.setString("country_id", list_id);
23.     docEdit.setString("function", et_customer_funct
    ion.
24.        getText().toString());
25.     docEdit.setString("phone", et_customer_phone.
26.        getText().toString());
27.     docEdit.setString("email", et_customer_email.
28.        getText().toString());
29.     docEdit.setString("website", et_customer_websit
    e.
30.        getText().toString());
31.     if(cb_customer_active.isChecked())
32.        docEdit.setString("active", "1");
33.     else
34.        docEdit.setString("active", "0");
35.     if (cb_customer_customer.isChecked())
36.        docEdit.setString("customer", "1");
37.     else
38.        docEdit.setString("customer", "0");
39.     if(cb_customer_supplier.isChecked())
40.        docEdit.setString("supplier", "1");
41.     else
42.        docEdit.setString("supplier", "0");
43.     if(cb_customer_employee.isChecked())
44.        docEdit.setString("employee", "1");
45.     else
46.        docEdit.setString("employee", "0");
47.     docEdit.setString("image", base64);
48.     docEdit.setString("uploaded", "0");
49.
50.          try {
51.              database.save(docEdit);
52.          } catch (CouchbaseLiteException e) {
53.             e.printStackTrace();
54.          }
55. }
```

Kode Sumber 4.40 Implementasi Pengubahan Data saat *Offline*

4.4.5 Menghapus Data saat *Offline*

Dalam implementasi penghapusan data secara *offline* digunakan penambahan *field* pada server Odoo, yaitu *field* x_softdelete. *Field* x_softdelete ini digunakan untuk menandai data mana yang sudah terhapus pada perangkat Android. X_softdelete akan bernilai *false* jika data belum terhapus pada perangkat Android dan akan bernilai *true* ketika data dihapus melalui perangkat Android. Hal ini berfungsi untuk menginformasikan pada server Odoo bahwa data tersebut telah terhapus secara *offline dari* perangkat Android. Hal ini berbeda ketika tidak dilakukan penambahan *field* x_softdelete karena dari sisi server Odoo tidak mendapatkan informasi data mana yang telah terhapus dari perangkat Android ketika dalam keadaan *offline*. Namun dengan penambahan fitur penghapusan ini dapat terjadi perbedaan data yang ditampilkan pada aplikasi berbasis website dengan aplikasi Android yang dikembangkan.

Kode gambar dari implementasi penghapusan data saat perangkat dalam keadaan *offline* ditunjukkan pada Kode Sumber 4.41.

```
1.  private void deleteOffline(String id) {
2.    Log.d(Const.TAG,
3.      "deleting offline doc with id: " + id);
4.    DatabaseConfiguration config = new
5.      DatabaseConfiguration(getApplicationContext());
6.    Database database = null;
7.    try {
8.        database = new Database("customer", config);

9.    } catch (CouchbaseLiteException e) {
10.       e.printStackTrace();
11.   }
12.   MutableDocument docEdit = new MutableDocument(id)
    ;
13.   docEdit.setString("street", "delete");
14.   docEdit.setString("uploaded", "0");
15.   try {
```

```
16.       database.save(docEdit);
17.       Log.d(Const.TAG, "success deleteOffline
18.         (soft_delete) document with id: " + id);
19.   } catch (CouchbaseLiteException e) {
20.       Log.d(Const.TAG, "failure deleteOffline
21.         (soft_delete): " + e.getMessage());
22.       e.printStackTrace();
23.   }
24.
25.   Toast.makeText(this, "Customer deleted offline",
26.      Toast.LENGTH_SHORT).show();
27. }
```

Kode Sumber 4.41 Implementasi Penghapusan Data saat *Offline*

4.4.6 Sinkronisasi Couchbase Lite dan Server Odoo

Kode gambar dari implementasi sinkronisasi Couchbase Lite dan Server Odoo ditunjukkan pada Kode Sumber 4.42. Untuk penjelasan pada proses sinkronisasi ditunjukkan pada Gambar 4.2.

```
1.  private void syncDataToServer() {
2.      android.text.format.DateFormat df = new
3.         android.text.format.DateFormat();
4.      Date date = new java.util.Date();
5.      Calendar cal = Calendar.getInstance();
6.      cal.setTime(date);
7.      cal.add(Calendar.HOUR, -7);
8.      Date gmtDate = cal.getTime();
9.      String lastSync = df.format("yyyy-MM-dd
10.        HH:mm:ss ",gmtDate ).toString();
11.     Log.d(Const.TAG, "pull: " + lastSync);
12.     BaseSharedPref.saveSPString(this,
13.        "last_sync", lastSync);
14.     pull();
15.     push();
16. }
17.
18. private void pull() {
19.     Client api = Server.builder().create(Client.class);
20.     Call<ResultHome> cari = api.readLastSync(
```

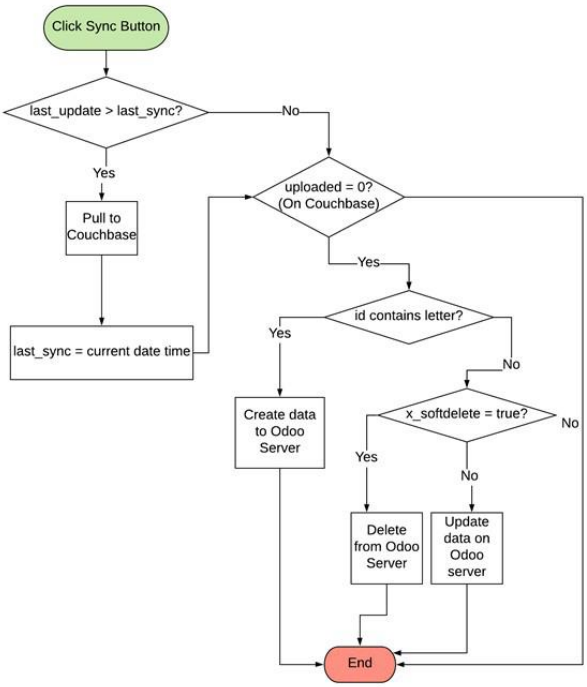
```
21.         BaseSharedPref.getSpString(this, "last_sync"));
22.     cari.enqueue(new Callback<ResultHome>() {
23.         @Override
24.         public void onResponse(Call<ResultHome> call,
25.             Response<ResultHome> response) {
26.              saveOffline(response.body().getResult());
27.         }
28.
29.         @Override
30.         public void onFailure(Call<ResultHome> call,
31.             Throwable t) {
32.              Toast.makeText(CustomerListActivity.this,
33.                 "gagal karena " + t.getMessage(),
34.                 Toast.LENGTH_SHORT).show();
35.         }
36.     });
37. }
38.
39. private void push() {
40.     DatabaseConfiguration config = new
41.         DatabaseConfiguration(getApplicationContext());
42.     Database database = null;
43.     try {
44.         database = new Database("customer", config);
45.     } catch (CouchbaseLiteException e) {
46.         e.printStackTrace();
47.     }
48.
49.     Query query = QueryBuilder
50.             .select(SelectResult.expression(Meta.id),
51.                     SelectResult.property("nama"),
52.                     SelectResult.property("street"),
53.                     SelectResult.property("city"),
54.                     SelectResult.property("country_id"),
55.                     SelectResult.property("function"),
56.                     SelectResult.property("phone"),
57.                     SelectResult.property("email"),
58.                     SelectResult.property("website"),
59.                     SelectResult.property("active"),
60.                     SelectResult.property("customer"),
61.                     SelectResult.property("supplier"),
62.                     SelectResult.property("employee"),
```

```
63.                     SelectResult.property("image"),
64.                     SelectResult.property("uploaded"))
65.             .from(DataSource.database(database))
66.             .where(Expression.property("uploaded").
67.                equalTo(Expression.string("0")));
68.     ResultSet result = null;
69.     try {
70.         result = query.execute();
71.         for (Result temp : result) {
72.             ReturnRead returnRead = new ReturnRead();
73.             uploadToServer(temp);
74.         }
75.     } catch (CouchbaseLiteException e) {
76.         Log.d(Const.TAG, "push error: " + e.toString());
77.         Log.d(Const.TAG, "push error: " +
78.            e.getMessage());
79.         e.printStackTrace();
80.     }
81. }
```

Kode Sumber 4.42 Implementasi Sinkronisasi Couchbase Lite dan Server Odoo

Proses sinkronisasi dimulai saat menekan tombol sync yang terdapat pada aplikasi Android. Pada proses ini dibutuhkan *field* Last Update pada Odoo server dan Last Sync pada aplikasi Android. Last Update merupakan *field* yang dimiliki dari setiap record di server untuk mengetahui tanggal dan jam berapa data melakukan perubahan. Sedangkan Last Sync merupakan *field* tambahan pada aplikasi Android yang tersimpan pada Couchbase untuk mengetahui tanggal dan jam berapa aplikasi Android melakukan sinkronisasi dengan Odoo server. Seperti yang dijelaskan pada Gambar 4.2, pada awal mula akan terjadi pengecekan dimana Last Update yang terdapat pada server lebih besar tanggalnya (lebih update) dari Last Sync yang terdapat pada aplikasi Android. Data Last Update dari server merupakan data dengan tipe data *timestamp* dengan format waktu GMT. Untuk membandingkan dengan waktu pada Last Sync, dilakukan pengambilan data *current date time* dari perangkat Android yang didapatkan dengan pengurangan waktu 7 jam. Jika pengecekan

benar, maka data akan *pull* ke Couchbase dan Last Sync akan di ubah sesuai dengan *current date time*. Jika tidak, makan akan terjadi pengecekan selanjutnya, apakah data ada yang sudah terupload atau belum. Jika data belum ada di server, maka data akan di *create* dengan id yang terdiri dari huruf dan angka (id menggunakan UUID) dan status uploaded = 0. Jika id tidak terdiri dari huruf (data sudah tersimpan di server), maka akan dicek x_softdelete bernilai *true* atau *false*. Jika x_softdelete=*true* maka data tersebut akan terhapus pada Android, namun data masih ada pada database, dikarenakan hanya mengubah status x_softdeletenya. Jika x_softdelete=*false* maka akan terjadi update pada data tersebut. Namun jika data sudah terupload dengan status uploaded = 1 pada Android, maka proses akan berakhir.



Gambar 4.2 Proses Sinkronisasi Couchbase Lite dan Server Odoo

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas tentang pengujian dan evaluasi terhadap sistem yang telah dikembangkan dari implementasi yang telah dilakukan pada bab sebelumnya.

5.1 Lingkungan Pengujian

Pada proses pengujian perangkat lunak, dibutuhkan suatu lingkungan pengujian yang sesuai dengan standar kebutuhan. Dalam Tugas Akhir ini, dilakukan dua macam pengujian yang memerlukan lingkungan pengujian yang berbeda, yaitu pengujian API dan pengujian *offline storage*.

5.1.1 Lingkungan Pengujian API

Lingkungan pengujian sistem pada pembuatan API dilakukan pada lingkungan pengujian sebagai berikut:

Tabel 5.1 Lingkungan Pengujian API

Spesifikasi	Deskripsi
Jenis Perangkat	Komputer
Merek Perangkat	Dell Vostro 14-5468
Sistem Operasi	Windows 10 Pro 64-bit
RAM	8 GB
Tools	Postman

5.1.2 Lingkungan Pengujian *Offline Storage*

Lingkungan pengujian sistem pada pembuatan *offline storage* dilakukan pada lingkungan pengujian sebagai berikut:

Tabel 5.2 Lingkungan Pengujian *Offline Storage*

Spesifikasi	Deskripsi
Jenis Perangkat	*Smartphone*
Merek Perangkat	Samsung S9+
Sistem Operasi	Android Pie (9.0)
RAM	6 GB

5.2 Pengujian

Pada subbab ini akan dijelaskan beberapa pengujian yang dilakukan terhadap sistem yang telah dibangun, yaitu pada pengujian API dan pengujian *offline storage*.

5.2.1 Pengujian API

Pada pengujian API ini akan dilakukan pengujian terhadap semua API yang telah dibuat. Terdapat dua macam pengujian, yaitu pengujian fungsionalitas (tanpa skenario) dan pengujian dengan menggunakan skenario. Pengujian fungsionalitas dilakukan pada *endpoint* yang tidak membutuhkan pengiriman parameter, sedangkan pengujian yang menggunakan skenario dilakukan pada semua *endpoint* yang membutuhkan pengiriman parameter.

5.2.1.1 Pengujian Fungsionalitas

Pada subbab ini akan dilakukan pengujian terhadap pengambilan semua data yang dikirimkan dari setiap menu tanpa memerlukan parameter dalam sistem yang telah dibangun.

5.2.1.1.1 Mengambil Semua Data Customer (PF - 01)

Detil pengujian mengambil semua data customer dijelaskan pada Tabel 5.3. Hasil dari pengujian terdapat pada Gambar 5.1. Respon dari *request endpoint* ini merupakan list yang berisi id, *name, category_id, function, city, country_id, email, opportunity_count, meeting_count* dan *image*.

Tabel 5.3 Pengujian Mengambil Semua Data Customer

Kode Pengujian	**PF – 01**
Nama API	Read customer
Endpoint **(localhost:8000/api)**	/customer
Method	GET
Hasil yang diharapkan	Menampilkan semua data customer
Pengujian	Berhasil

```
{
    "result": [
        {
            "id": 14,
            "name": "Azure Interior",
            "category_id": [
                5
            ],
            "function": false,
            "city": "Jonesboro",
            "country_id": [
                233,
                "United States"
            ],
            "email": "azure.Interior24@example.com",
            "image": "iVBORw0KGgoAAAANSUhEUgAAAFoAAABaCAYAAAA4qEECAAAACXBIWXMAAAsSAAALEgHS3X78AAAFak1EQVR4nO2b7XHjOyGX9
            "opportunity_count": 2,
            "meeting_count": 2
        }
    ]
}
```


Gambar 5.1 Hasil Pengujian Mengambil Semua Data Customer

5.2.1.1.2 Mengambil Semua Data My Pipeline (PF - 02)

Detil pengujian mengambil semua data pipeline dijelaskan pada Tabel 5.4. Hasil dari pengujian terdapat pada Gambar 5.2. Respon dari *request endpoint* ini merupakan list yang berisi id, *date_open*, *name*, *partner_id*, *country_id*, *activity_date_deadline*, *activity_summary*. *stage_id*, *planned_revenue*, *probability*, *team_id*, dan *user_id.*

Tabel 5.4 Pengujian Mengambil Semua Data My Pipeline

Kode Pengujian	**PF – 02**
Nama API	Read pipeline
***Endpoint* (localhost:8000/api)**	/pipeline
Method	GET
Hasil yang diharapkan	Menampilkan semua data pipeline
Pengujian	Berhasil

```
[
    {
        "id": 25,
        "date_open": "2019-04-24 03:45:24",
        "name": "Modern Open Space",
        "partner_id": false,
        "country_id": [
            10,
            "Argentina"
        ],
        "stage_id": [
            3,
            "Proposition"
        ],
        "planned_revenue": 4500,
        "probability": 60,
        "team_id": [
            1,
            "Europe"
        ],
        "user_id": [
            2,
            "Mitchell Admin"
        ],
        "activity_date_deadline": "2019-04-24",
        "activity_summary": "Conf call with technical service"
    },
    {
        "id": 21,
        "date_open": "2019-04-24 03:45:24",
```


Gambar 5.2 Hasil Pengujian Mengambil Semua Data My Pipeline

5.2.1.1.3 Mengambil Semua Data Pipeline Analysis (PF - 03)

Detil pengujian mengambil semua data pipeline analysis dijelaskan pada Tabel 5.5. Hasil dari pengujian terdapat pada Gambar 5.3. Respon dari *request endpoint* ini merupakan list yang berisi id, *create_date*, *name*, *contact_name*, *city*, *country_id*, *email_from*, *phone*, dan *team_id*.

Tabel 5.5 Pengujian Mengambil Semua Data Pipeline Analysis

Kode Pengujian	**PF – 03**
Nama API	Read pipeline analysis

***Endpoint* (localhost:8000/api)**	/pplana
Method	GET
Hasil yang diharapkan	Menampilkan semua data pipeline analysis
Pengujian	Berhasil


```
[
    {
        "id": 27,
        "create_date": "2019-04-23 03:45:24",
        "name": "Interest in your products",
        "contact_name": false,
        "city": "Wavre",
        "country_id": [
            20,
            "Belgium"
        ],
        "email_from": "info@agrolait.com",
        "phone": false,
        "team_id": [
            3,
            "America"
        ]
    },
    {
        "id": 26,
        "create_date": "2019-04-18 03:45:24",
        "name": "Open Space Design",
        "contact_name": false,
        "city": "Wavre",
        "country_id": [
            20,
            "Belgium"
        ],
        "email_from": "info@agrolait.com",
        "phone": false,
```


Gambar 5.3 Hasil Pengujian Mengambil Semua Data Pipeline Analysis

5.2.1.1.4 Mengambil Semua Data Sales Teams (PF - 04)

Detil pengujian mengambil semua data sales teams dijelaskan pada Tabel 5.6. Hasil dari pengujian terdapat pada Gambar 5.4. Respon dari *request endpoint* ini merupakan list yang berisi id dan *name*.

Tabel 5.6 Pengujian Mengambil Semua Data Sales Teams

Kode Pengujian	**PF – 04**
Nama API	Read sales teams
***Endpoint* (localhost:8000/api)**	/salestim
Method	GET
Hasil yang diharapkan	Menampilkan semua data sales teams
Pengujian	Berhasil

Gambar 5.4 Hasil Pengujian Mengambil Semua Data Sales Teams

5.2.1.1.5 Mengambil Semua Data Activity Types (PF - 05)

Detil pengujian mengambil semua data activity types dijelaskan pada Tabel 5.7. Hasil dari pengujian terdapat pada Gambar 5.5. Respon dari *request endpoint* ini merupakan list yang berisi id, *name*, *delay_count*, *delay_unit*, dan *delay_from*.

Tabel 5.7 Pengujian Mengambil Semua Data Activity Types

Kode Pengujian	**PF – 05**
Nama API	Read activity types
Endpoint **(localhost:8000/api)**	/act
Method	GET
Hasil yang diharapkan	Menampilkan semua data activity types
Pengujian	Berhasil


```
[
    {
        "id": 1,
        "name": "Email",
        "delay_count": 0,
        "delay_unit": "days",
        "delay_from": "previous_activity"
    },
    {
        "id": 2,
        "name": "Call",
        "delay_count": 2,
        "delay_unit": "days",
        "delay_from": "previous_activity"
    },
    {
        "id": 3,
        "name": "Meeting",
        "delay_count": 0,
        "delay_unit": "days",
        "delay_from": "previous_activity"
    },
    {
        "id": 6,
        "name": "Follow-up Quote",
        "delay_count": 30,
        "delay_unit": "days",
        "delay_from": "previous_activity"
    },
    {
```


Gambar 5.5 Hasil Pengujian Mengambil Semua Data Activity Types

5.2.1.1.6 Mengambil Semua Data Tags (PF - 06)

Detil pengujian mengambil semua data tags dijelaskan pada Tabel 5.8. Hasil dari pengujian terdapat pada Gambar 5.6. Respon dari *request endpoint* ini merupakan list yang berisi id dan *name*.

Tabel 5.8 Pengujian Mengambil Semua Data Tags

Kode Pengujian	**PF – 06**
Nama API	Read tags
***Endpoint* (localhost:8000/api)**	/tag
Method	GET
Hasil yang diharapkan	Menampilkan semua data tags
Pengujian	Berhasil


```
[
    {
        "id": 1,
        "name": "Product"
    },
    {
        "id": 2,
        "name": "Software"
    },
    {
        "id": 3,
        "name": "Services"
    },
    {
        "id": 4,
        "name": "Information"
    },
    {
        "id": 5,
        "name": "Design"
    },
    {
        "id": 6,
        "name": "Training"
    },
    {
        "id": 7,
        "name": "Consulting"
    },
    {
```

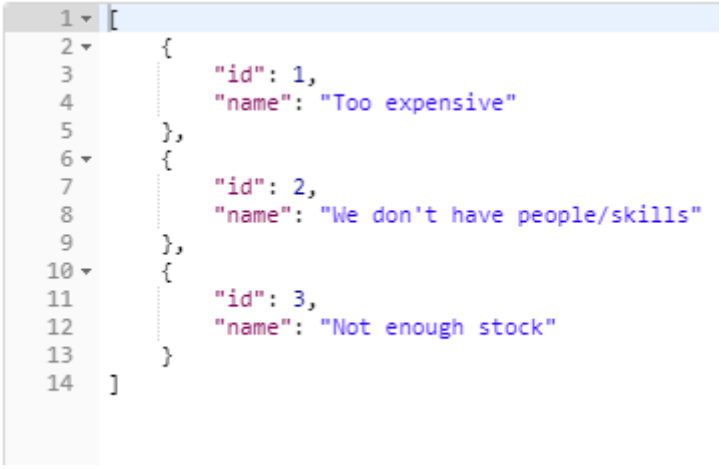

Gambar 5.6 Hasil Pengujian Mengambil Semua Data Tags

5.2.1.1.7 Mengambil Semua Data Lost Reasons (PF - 07)

Detil pengujian mengambil semua data pipeline dijelaskan pada Tabel 5.9. Hasil dari pengujian terdapat pada Gambar 5.7. Respon dari *request endpoint* ini merupakan list yang berisi id dan *name*.

Tabel 5.9 Pengujian Mengambil Semua Data Lost Reasons

Kode Pengujian	**PF – 07**
Nama API	Read lost reasons
***Endpoint* (localhost:8000/api)**	/lost
Method	GET
Hasil yang diharapkan	Menampilkan semua data lost reasons
Pengujian	Berhasil



Gambar 5.7 Hasil Pengujian Mengambil Semua Data Lost Reasons

5.2.1.1.8 Mengambil Semua Data Team Pipelines (PF - 08)

Detil pengujian mengambil semua data team pipelines dijelaskan pada Tabel 5.10. Hasil dari pengujian terdapat pada

Gambar 5.8. Respon dari *request endpoint* ini merupakan list yang berisi id, *name*, *opportunities_count*, dan *opportunities_amount*.

Tabel 5.10 Pengujian Mengambil Semua Data Team Pipelines

Kode Pengujian	**PF – 08**
Nama API	Read team pipelines
***Endpoint* (localhost:8000/api)**	/teamppl
Method	GET
Hasil yang diharapkan	Menampilkan semua data team pipelines
Pengujian	Berhasil

Gambar 5.8 Hasil Pengujian Mengambil Semua Data Team Pipelines

5.2.1.2 Pengujian Menggunakan Skenario

Terdapat dua skenario pengujian yang diterapkan pada *endpoint* yang memerlukan pengiriman parameter, yaitu skenario A dan skenario B. Deskripsi dari setiap skenario dapat dilihat pada Tabel 5.11.

Tabel 5.11 Skenario Pengujian API

Kode Skenario	**Nama Skenario**	**Deskripsi**
A	Parameter Sesuai	Pengujian dilakukan dengan mengirim seluruh parameter dengan benar yang dibutuhkan oleh *endpoint.* Pengujian

		berhasil jika *status code* bernilai 200.
B	Parameter Tidak Sesuai	Pengujian dilakukan dengan mengirim sebagian parameter atau parameter yang salah yang dibutuhkan oleh *endpoint*. Pengujian berhasil jika *status code* bernilai 500.

5.2.1.2.1 Mengambil Data Customer Berdasarkan Id (PS - 01)

Pada pengujian ini, pengguna dapat melihat data customer berdasarkan Id yang dimasukkan. Deskripsi pengujian dapat dilihat pada Tabel 5.12.

Tabel 5.12 Pengujian Mengambil Data Customer

Kode Pengujian			**PS – 01**
Nama *Endpoint*			Get customer
***Endpoint* (localhost:8000/api)**			/customer/{id}
Metode HTTP			GET
Deskripsi Endpoint			Menampilkan semua data customer
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.2 Menambah Data Customer (PS - 02)

Pada pengujian ini, pengguna dapat menambah data customer baru. Deskripsi pengujian dapat dilihat pada Tabel 5.13.

Tabel 5.13 Pengujian Menambah Data Customer

Kode Pengujian	**PS – 02**
Nama *Endpoint*	Create customer
***Endpoint* (localhost:8000/api)**	/customer

Metode HTTP			POST
Deskripsi Endpoint			Menambah data customer
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.3 Menghapus Data Customer (PS - 03)

Pada pengujian ini, pengguna dapat menghapus data customer yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.14.

Tabel 5.14 Pengujian Menghapus Data Customer

Kode Pengujian			**PS – 03**
Nama *Endpoint*			Delete customer
***Endpoint* (localhost:8000/api)**			/customer/{id}
Metode HTTP			DELETE
Deskripsi Endpoint			Menghapus data customer
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.4 Mengubah Data Customer (PS - 04)

Pada pengujian ini, pengguna dapat mengubah data customer yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.15.

Tabel 5.15 Pengujian Mengubah Data Customer

Kode Pengujian			**PS – 04**
Nama *Endpoint*			Update customer
***Endpoint* (localhost:8000/api)**			/editcustomer
Metode HTTP			POST
Deskripsi Endpoint			Mengubah data customer
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.5 Mengambil Data Pipeline Berdasarkan Id (PS - 05)

Pada pengujian ini, pengguna dapat melihat data pipeline berdasarkan Id yang dimasukkan. Deskripsi pengujian dapat dilihat pada Tabel 5.16.

Tabel 5.16 Pengujian Mengambil Data Pipeline

Kode Pengujian			**PS – 05**
Nama *Endpoint*			Get pipeline
***Endpoint* (localhost:8000/api)**			/pipeline/{id}
Metode HTTP			GET
Deskripsi Endpoint			Menampilkan semua data pipeline
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.6 Menambah Data Pipeline (PS - 06)

Pada pengujian ini, pengguna dapat menambah data pipeline baru. Deskripsi pengujian dapat dilihat pada Tabel 5.17.

Tabel 5.17 Pengujian Menambah Data Pipeline

Kode Pengujian			**PS – 06**
Nama *Endpoint*			Create pipeline
***Endpoint* (localhost:8000/api)**			/pipeline
Metode HTTP			POST
Deskripsi Endpoint			Menambah data pipeline
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.7 Menghapus Data Pipeline (PS - 07)

Pada pengujian ini, pengguna dapat menghapus data pipeline yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.18.

Tabel 5.18 Pengujian Menghapus Data Pipeline

Kode Pengujian			**PS – 07**
Nama *Endpoint*			Delete pipeline
***Endpoint* (localhost:8000/api)**			/pipeline/{id}
Metode HTTP			DELETE
Deskripsi Endpoint			Menghapus data pipeline
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.8 Mengubah Data Pipeline (PS - 08)

Pada pengujian ini, pengguna dapat mengubah data pipeline yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.19.

Tabel 5.19 Pengujian Mengubah Data Pipeline

Kode Pengujian			**PS – 08**
Nama *Endpoint*			Update pipeline
***Endpoint* (localhost:8000/api)**			/editpipeline
Metode HTTP			POST
Deskripsi Endpoint			Mengubah data pipeline
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.9 Mengambil Data Pipeline Analysis Berdasarkan Id (PS - 09)

Pada pengujian ini, pengguna dapat melihat data pipeline analysis berdasarkan Id yang dimasukkan. Deskripsi pengujian dapat dilihat pada Tabel 5.20.

Tabel 5.20 Pengujian Mengambil Data Pipeline Analysis

Kode Pengujian	**PS – 09**
Nama *Endpoint*	Get pipeline analysis

***Endpoint* (localhost:8000/api)**			/pplana/{id}
Metode HTTP			GET
Deskripsi Endpoint			Menampilkan semua data pipeline analysis
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.10 Menambah Data Pipeline Analysis (PS - 10)

Pada pengujian ini, pengguna dapat menambah data pipeline analysis baru. Deskripsi pengujian dapat dilihat pada Tabel 5.21.

Tabel 5.21 Pengujian Menambah Data Pipeline Analysis

Kode Pengujian			**PS – 10**
Nama *Endpoint*			Create pipeline analysis
***Endpoint* (localhost:8000/api)**			/pplana
Metode HTTP			POST
Deskripsi Endpoint			Menambah data pipeline analysis
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.11 Menghapus Data Pipeline Analysis (PS - 11)

Pada pengujian ini, pengguna dapat menghapus data pipeline analysis yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.22.

Tabel 5.22 Pengujian Menghapus Data Pipeline Analysis

Kode Pengujian	**PS – 11**
Nama *Endpoint*	Delete pipeline analysis
***Endpoint* (localhost:8000/api)**	/pplana/{id}
Metode HTTP	DELETE

Deskripsi Endpoint			Menghapus data pipeline analysis
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.12 Mengubah Data Pipeline Analysis (PS - 12)

Pada pengujian ini, pengguna dapat mengubah data pipeline analysis yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.23.

Tabel 5.23 Pengujian Mengubah Data Pipeline Analysis

Kode Pengujian			**PS – 12**
Nama *Endpoint*			Update pipeline analysis
***Endpoint* (localhost:8000/api)**			/editpplana
Metode HTTP			POST
Deskripsi Endpoint			Mengubah data pipeline analysis
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.13 Mengambil Data Sales Teams Berdasarkan Id (PS - 13)

Pada pengujian ini, pengguna dapat melihat data sales teams berdasarkan Id yang dimasukkan. Deskripsi pengujian dapat dilihat pada Tabel 5.24.

Tabel 5.24 Pengujian Mengambil Data Sales Teams

Kode Pengujian			**PS – 13**
Nama *Endpoint*			Get sales teams
***Endpoint* (localhost:8000/api)**			/salestim/{id}
Metode HTTP			GET
Deskripsi Endpoint			Menampilkan semua data sales teams
Pengujian	**Skenario**	**A**	Berhasil

		B	Berhasil

5.2.1.2.14 Menambah Data Sales Teams (PS - 14)

Pada pengujian ini, pengguna dapat menambah data sales teams baru. Deskripsi pengujian dapat dilihat pada Tabel 5.25.

Tabel 5.25 Pengujian Menambah Data Sales Teams

Kode Pengujian			**PS – 14**
Nama *Endpoint*			Create sales teams
***Endpoint* (localhost:8000/api)**			/salestim
Metode HTTP			POST
Deskripsi Endpoint			Menambah data sales teams
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.15 Menghapus Data Sales Teams (PS - 15)

Pada pengujian ini, pengguna dapat menghapus data sales teams yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.26.

Tabel 5.26 Pengujian Menghapus Data Sales Teams

Kode Pengujian			**PS – 15**
Nama *Endpoint*			Delete sales teams
***Endpoint* (localhost:8000/api)**			/salestim/{id}
Metode HTTP			DELETE
Deskripsi Endpoint			Menghapus data sales teams
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.16 Mengubah Data Sales Teams (PS - 16)

Pada pengujian ini, pengguna dapat mengubah data sales teams yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.27.

Tabel 5.27 Pengujian Mengubah Data Sales Teams

Kode Pengujian			**PS – 16**
Nama *Endpoint*			Update sales teams
***Endpoint* (localhost:8000/api)**			/editsalestim
Metode HTTP			POST
Deskripsi Endpoint			Mengubah data sales teams
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.17 Mengambil Data Activity Types Berdasarkan Id (PS - 17)

Pada pengujian ini, pengguna dapat melihat data activity types berdasarkan Id yang dimasukkan. Deskripsi pengujian dapat dilihat pada Tabel 5.28.

Tabel 5.28 Pengujian Mengambil Data Activity Types

Kode Pengujian			**PS – 17**
Nama *Endpoint*			Get activity types
***Endpoint* (localhost:8000/api)**			/act/{id}
Metode HTTP			GET
Deskripsi Endpoint			Menampilkan semua data activity types
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.18 Menambah Data Activity Types (PS - 18)

Pada pengujian ini, pengguna dapat menambah data activity types baru. Deskripsi pengujian dapat dilihat pada Tabel 5.29.

Tabel 5.29 Pengujian Menambah Data Activity Types

Kode Pengujian	**PS – 18**
Nama *Endpoint*	Create activity types

***Endpoint* (localhost:8000/api)**			/act
Metode HTTP			POST
Deskripsi Endpoint			Menambah data activity types
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.19 Menghapus Data Activity Types (PS - 19)

Pada pengujian ini, pengguna dapat menghapus data activity types yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.30.

Tabel 5.30 Pengujian Menghapus Data Activity Types

Kode Pengujian			**PS – 19**
Nama *Endpoint*			Delete activity types
***Endpoint* (localhost:8000/api)**			/act/{id}
Metode HTTP			DELETE
Deskripsi Endpoint			Menghapus data activity types
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.20 Mengubah Data Activity Types (PS - 20)

Pada pengujian ini, pengguna dapat mengubah data activity types yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.31.

Tabel 5.31 Pengujian Mengubah Data Activity Types

Kode Pengujian			**PS – 20**
Nama *Endpoint*			Update activity types
***Endpoint* (localhost:8000/api)**			/editact
Metode HTTP			POST
Deskripsi Endpoint			Mengubah data activity types
Pengujian	**Skenario**	**A**	Berhasil

		B	Berhasil

5.2.1.2.21 Mengambil Data Tags Berdasarkan Id (PS - 21)

Pada pengujian ini, pengguna dapat melihat data tags berdasarkan Id yang dimasukkan. Deskripsi pengujian dapat dilihat pada Tabel 5.32.

Tabel 5.32 Pengujian Mengambil Data Tags

Kode Pengujian			**PS – 21**
Nama *Endpoint*			Get tags
***Endpoint* (localhost:8000/api)**			/tags/{id}
Metode HTTP			GET
Deskripsi Endpoint			Menampilkan semua data tags
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.22 Menambah Data Tags (PS - 22)

Pada pengujian ini, pengguna dapat menambah data tags baru. Deskripsi pengujian dapat dilihat pada Tabel 5.33.

Tabel 5.33 Pengujian Menambah Data Tags

Kode Pengujian			**PS – 22**
Nama *Endpoint*			Create tags
***Endpoint* (localhost:8000/api)**			/tags
Metode HTTP			POST
Deskripsi Endpoint			Menambah data tags
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.23 Menghapus Data Tags (PS - 23)

Pada pengujian ini, pengguna dapat menghapus data tags yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.34.

Tabel 5.34 Pengujian Menghapus Data Tags

Kode Pengujian			**PS – 23**
Nama *Endpoint*			Delete tags
***Endpoint* (localhost:8000/api)**			/tags/{id}
Metode HTTP			DELETE
Deskripsi Endpoint			Menghapus data tags
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.24 Mengubah Data Tags (PS - 24)

Pada pengujian ini, pengguna dapat mengubah data tags yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.35.

Tabel 5.35 Pengujian Mengubah Data Tags

Kode Pengujian			**PS – 24**
Nama *Endpoint*			Update tags
***Endpoint* (localhost:8000/api)**			/edittags
Metode HTTP			POST
Deskripsi Endpoint			Mengubah data tags
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.25 Mengambil Data Lost Reasons Berdasarkan Id (PS - 25)

Pada pengujian ini, pengguna dapat melihat data lost reasons berdasarkan Id yang dimasukkan. Deskripsi pengujian dapat dilihat pada Tabel 5.36.

Tabel 5.36 Pengujian Mengambil Data Lost Reasons

Kode Pengujian	**PS – 25**
Nama *Endpoint*	Get lost reasons

***Endpoint* (localhost:8000/api)**			/lost/{id}
Metode HTTP			GET
Deskripsi Endpoint			Menampilkan semua data lost reasons
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.26 Menambah Data Lost Reasons (PS - 26)

Pada pengujian ini, pengguna dapat menambah data lost reasons baru. Deskripsi pengujian dapat dilihat pada Tabel 5.37.

Tabel 5.37 Pengujian Menambah Data Lost Reasons

Kode Pengujian			**PS – 26**
Nama *Endpoint*			Create lost reasons
***Endpoint* (localhost:8000/api)**			/lost
Metode HTTP			POST
Deskripsi Endpoint			Menambah data lost reasons
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.1.2.27 Menghapus Data Lost Reasons (PS - 27)

Pada pengujian ini, pengguna dapat menghapus data lost reasons yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.38.

Tabel 5.38 Pengujian Menghapus Data Lost Reasons

Kode Pengujian			**PS – 27**
Nama *Endpoint*			Delete lost reasons
***Endpoint* (localhost:8000/api)**			/lost/{id}
Metode HTTP			DELETE
Deskripsi Endpoint			Menghapus data lost reasons
Pengujian	**Skenario**	**A**	Berhasil

		B	Berhasil

5.2.1.2.28 Mengubah Data Lost Reasons (PS - 28)

Pada pengujian ini, pengguna dapat mengubah data lost reasons yang telah tersimpan pada sistem. Deskripsi pengujian dapat dilihat pada Tabel 5.39.

Tabel 5.39 Pengujian Mengubah Data Lost Reasons

Kode Pengujian			**PS – 28**
Nama *Endpoint*			Update lost reasons
***Endpoint* (localhost:8000/api)**			/editlost
Metode HTTP			POST
Deskripsi Endpoint			Mengubah data lost reasons
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.2 Pengujian Offline Storage

Pada subbab ini akan dilakukan pengujian terhadap aplikasi berbasis Android yang telah dibuat. Pengujian dilakukan untuk mengetahui keberhasilan *offline storage* yang diimplementasikan menggunakan Couchbase. Terdapat dua skenario pengujian yang diterapkan, yaitu skenario A dan skenario B. Deskripsi dari setiap skenario dapat dilihat pada Tabel 5.40.

Tabel 5.40 Skenario Pengujian *Offline Storage*

Kode Skenario	Nama Skenario	Deskripsi
A	Tidak Terhubung Koneksi Internet	Pengujian dilakukan dengan melakukan semua fitur terhadap data customer dengan *device* tidak terhubung koneksi internet.
B	Sinkronisasi	Pengujian dilakukan dengan melakukan sinkronisasi terhadap perubahan data yang terjadi

		di antara *device* dan Odoo server.

5.2.2.1 Pengujian Pengambilan Data (PO - 01)

Detil pengujian mengambil data saat perangkat dalam keadaan *offline* dijelaskan pada Tabel 5.41.

Tabel 5.41 Pengujian Pengambilan Data saat *Offline*

Kode Pengujian			**PO – 01**
Kasus Penggunaan			Pengambilan data saat *offline.*
Hasil yang Diharapkan			Sistem dapat menampilkan data yang tersimpan dalam Couchbase saat perangkat dalam keadaan *offline.*
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.2.2 Pengujian Penambahan Data (PO - 02)

Detil pengujian menambah data saat perangkat dalam keadaan *offline* dijelaskan pada Tabel 5.42.

Tabel 5.42 Pengujian Penambahan Data saat *Offline*

Kode Pengujian			**PO – 02**
Kasus Penggunaan			Penambahan data saat *offline.*
Hasil yang Diharapkan			Sistem dapat menyimpan data saat perangkat dalam keadaan *offline* ke Couchbase.
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.2.3 Pengujian Pengubahan Data (PO - 03)

Detil pengujian mengubah data saat perangkat dalam keadaan *offline* dijelaskan pada Tabel 5.43.

Tabel 5.43 Pengujian Pengubahan Data saat *Offline*

Kode Pengujian	**PO – 03**
Kasus Penggunaan	Pengubahan data saat *offline.*

Hasil yang Diharapkan			Sistem dapat melakukan pengubahan data yang telah tersimpan saat perangkat dalam keadaan *offline* ke Couchbase.
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.2.4 Pengujian Penghapusan Data (PO - 04)

Detil pengujian menghapus data saat perangkat dalam keadaan *offline* dijelaskan pada Tabel 5.44.

Tabel 5.44 Pengujian Penghapusan Data saat *Offline*

Kode Pengujian			**PO – 04**
Kasus Penggunaan			Penghapusan data saat *offline*.
Hasil yang Diharapkan			Sistem dapat melakukan penghapusan data yang telah tersimpan saat perangkat dalam keadaan *offline* ke Couchbase.
Pengujian	**Skenario**	**A**	Berhasil
		B	Berhasil

5.2.3 Pengujian Sinkronisasi Data

Pada subbab ini akan dilakukan pengujian terhadap sinkronisasi data pada Couchbase dengan data yang ada pada server Odoo. Pengujian ini dilakukan untuk mengetahui apa yang terjadi apabila perangkat A melakukan penghapusan data dan perangkat B melakukan pengubahan pada data yang sama secara *offline* tanpa melakukan sinkronisasi terlebih dahulu. Setelah itu dilakukan sinkronisasi terlebih dahulu di antara dua perangkat tersebut. Oleh karena itu, dibuatlah dua skenario pengujian yang diterapkan, yaitu skenario A dan skenario B. Deskripsi dari setiap skenario dapat dilihat pada Tabel 5.45. Detil pengujian sinkronisasi data pada skenario tersebut dijelaskan pada Tabel 5.46.

Tabel 5.45 Skenario Pengujian Sinkronisasi Data

Kode Skenario	Nama Skenario	Deskripsi
A	Penghapusan data dilakukan terlebih dahulu secara *offline.*	Pengujian dilakukan dengan melakukan penghapusan pada perangkat A terlebih dahulu dibanding pengubahan data yang sama pada perangkat B ketika kedua perangkat dalam keadaan *offline*. Kemudian perangkat A melakukan sinkronisasi terlebih dahulu.
B	Pengubahan data dilakukan terlebih dahulu secara *offline.*	Pengujian dilakukan dengan melakukan pengubahan pada perangkat A terlebih dahulu dibanding penghapusan data yang sama pada perangkat B ketika kedua perangkat dalam keadaan *offline*. Kemudian perangkat A melakukan sinkronisasi terlebih dahulu.

Tabel 5.46 Pengujian Sinkronisasi Data

Kode Pengujian			**PS – 01**
Kasus Penggunaan			Pengujian sinkronisasi data.
Hasil yang Diharapkan			Sistem dapat menampilkan apa yang terjadi menurut skenario yang diujikan.
Pengujian	**Skenario**	**A**	Data berhasil dihapus terlebih dahulu pada perangkat A dan perangkat B menampilkan pesan bahwa data telah

			terhapus oleh perangkat lain.
		B	Data berhasil diubah melalui perangkat A kemudian terhapus melalui perangkat B.

5.3 Evaluasi Pengujian

Pada subbab ini akan diberikan hasil evaluasi dari pengujian yang telah dilakukan. Evaluasi yang dilakukan meliputi evaluasi pengujian fungsionalitas, pengujian menggunakan skenario, dan pengujian *offline storage*.

5.3.1 Evaluasi Pengujian Fungsionalitas

Hasil pengujian fungsionalitas secara keseluruhan dapat dilihat di Tabel 5.47. Data pada tabel tersebut menunjukkan bahwa semua fungsionalitas telah berhasil diimplementasikan dan berjalan sesuai dengan yang telah dirancang. Sehingga dapat ditarik kesimpulan bahwa sistem bekerja sesuai dengan yang diharapkan.

Tabel 5.47 Hasil Pengujian Fungsionalitas

Kode Pengujian Fungsionalitas	Kasus Penggunaan	Hasil
PF – 01	Mengambil semua data Customer	Berhasil
PF – 02	Mengambil semua data My Pipeline	Berhasil
PF – 03	Mengambil semua data Pipeline Analysis	Berhasil
PF – 04	Mengambil semua data Sales Teams	Berhasil
PF – 05	Mengambil semua data Activity Types	Berhasil

PF – 06	Mengambil semua data Tags	Berhasil
PF – 07	Mengambil semua data Lost Reasons	Berhasil
PF – 08	Mengambil semua data Team Pipeline	Berhasil

5.3.2 Evaluasi Pengujian Menggunakan Skenario

Hasil pengujian menggunakan skenario secara keseluruhan dapat dilihat di Tabel 5.48. Berdasarkan pada tabel tersebut, hasil pengujian menunjukkan bahwa semua skenario pengujian untuk setiap *endpoint* berhasil dilakukan. Sehingga dapat ditarik kesimpulan bahwa setiap *endpoint* memiliki *response* yang sesuai.

Tabel 5.48 Hasil Pengujian Menggunakan Skenario

Kode Pengujian	**Nama *Endpoint***	**Hasil Pengujian**	
		Skenario	
		A	**B**
PS – 01	Get customer	Berhasil	Berhasil
PS – 02	Create customer	Berhasil	Berhasil
PS – 03	Delete customer	Berhasil	Berhasil
PS – 04	Update customer	Berhasil	Berhasil
PS – 05	Get pipeline	Berhasil	Berhasil
PS – 06	Create pipeline	Berhasil	Berhasil
PS – 07	Delete pipeline	Berhasil	Berhasil
PS – 08	Update pipeline	Berhasil	Berhasil
PS – 09	Get pipeline analysis	Berhasil	Berhasil
PS – 10	Create pipeline analysis	Berhasil	Berhasil
PS – 11	Delete pipeline analysis	Berhasil	Berhasil
PS – 12	Update pipeline analysis	Berhasil	Berhasil
PS – 13	Get sales teams	Berhasil	Berhasil
PS – 14	Create sales teams	Berhasil	Berhasil

PS – 15	Delete sales teams	Berhasil	Berhasil
PS – 16	Update sales teams	Berhasil	Berhasil
PS – 17	Get activity types	Berhasil	Berhasil
PS – 18	Create activity types	Berhasil	Berhasil
PS – 19	Delete activity types	Berhasil	Berhasil
PS – 20	Update activity types	Berhasil	Berhasil
PS – 21	Get tags	Berhasil	Berhasil
PS – 22	Create tags	Berhasil	Berhasil
PS – 23	Delete tags	Berhasil	Berhasil
PS – 24	Update tags	Berhasil	Berhasil
PS – 25	Get lost reasons	Berhasil	Berhasil
PS – 26	Create lost reasons	Berhasil	Berhasil
PS – 27	Delete lost reasons	Berhasil	Berhasil
PS – 28	Update lost reasons	Berhasil	Berhasil

5.3.3 Evaluasi Pengujian *Offline Storage*

Hasil pengujian *offline storage* secara keseluruhan dapat dilihat di Tabel 5.49. Data pada tabel tersebut menunjukkan bahwa semua fungsionalitas terhadap aplikasi Android yang telah dibuat dengan mengimplementasikan Couchbase telah berhasil diimplementasikan dan berjalan sesuai dengan yang telah dirancang. Sehingga dapat ditarik kesimpulan bahwa aplikasi bekerja sesuai dengan yang diharapkan.

Tabel 5.49 Hasil Pengujian *Offline Storage*

Kode Pengujian	**Kasus Penggunaan**	**Hasil Pengujian**	
		Skenario	
		A	**B**
PO – 01	Pengambilan data saat *offline*	Berhasil	Berhasil
PO – 02	Penambahan data saat *offline*	Berhasil	Berhasil
PO – 03	Pengubahan data saat *offline*	Berhasil	Berhasil

PO – 04	Penghapusan data saat *offline*	Berhasil	Berhasil

5.3.4 Evaluasi Pengujian Sinkronisasi Data

Hasil pengujian sinkronisasi data terhadap dua skenario dapat dilihat pada Tabel 5.46. Data pada tabel tersebut menunjukkan bahwa fitur yang dijalankan terlebih dahulu secara *offline* tidak akan terlebih dahulu dieksekusi, dikarenakan yang terlebih dahulu dieksekusi yaitu perangkat yang terlebih dahulu melakukan sinkronisasi.

Jika dilakukan penghapusan pada perangkat A terlebih dahulu dibanding pengubahan data yang sama pada perangkat B ketika kedua perangkat dalam keadaan *offline,* kemudian perangkat A disinkronkan, maka yang akan dieksekusi terlebih dahulu merupakan fitur penghapusan data yaitu pada perangkat A dan pada perangkat yang melakukan pengubahan akan memunculkan pesan bahwa data terhapus oleh perangkat lain.

Sedangkan jika dilakukan pengubahan pada perangkat A terlebih dahulu dibanding penghapusan data yang sama pada perangkat B ketika kedua perangkat dalam keadaan *offline,* kemudian perangkat A disinkronkan, maka yang akan dieksekusi terlebih dahulu merupakan fitur pengubahan data pada perangkat A lalu dilanjutkan dengan penghapusan data.

Sehingga dapat ditarik kesimpulan bahwa aplikasi akan mengeksekusi perintah sesuai dengan perangkat mana dulu yang melakukan sinkronisasi terlebih dahulu.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan Tugas Akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1 Kesimpulan

Berikut merupakan kesimpulan yang dapat diambil dari proses pengembangan dan hasil uji coba.

1. Pembuatan API Odoo pada modul CRM untuk mengintegrasikan aplikasi *mobile* dan aplikasi web telah berhasil diimplementasikan dengan menggunakan bahasa pemrograman PHP dan memanfaatkan Laradoo. Hal ini dibuktikan dengan keberhasilan seluruh uji coba API yang ditunjukkan pada Tabel 5.47 dan Tabel 5.48.
2. Pada implementasi *offline storage* dengan menggunakan Couchbase berhasil diterapkan pada aplikasi berbasis Android. Implementasi Couchbase dapat diterapkan pada semua fitur dengan penambahan *field* x_softdelete pada model Odoo dengan tujuan agar penghapusan data pada saat *offline* tetap dapat dilakukan. Hal ini tidak mengurangi satupun fungsionalitas yang ada pada aplikasi ini. Hal ini dibuktikan dengan keberhasilan seluruh uji coba fungsionalitas yang ditunjukkan pada Tabel 5.49.
3. Dengan pengeksplorasian *offline storage* menggunakan Couchbase pada tugas akhir ini, yaitu data yang tersimpan saat *offline* hanya dimiliki oleh perangkat mobile yang menggunakan aplikasi tersebut, karena data akan tersimpan secara lokal pada perangkat mobile. Selain itu, data yang dapat disimpan ke dalam *cache* di Couchbase tidak memiliki batasan ukuran. Sehingga sebanyak apapun data yang disimpan ke dalam *cache*, Couchbase tidak mengurangi satupun fungsionalitas yang ada pada aplikasi ini.

6.2 Saran

Saran untuk perbaikan ke depannya adalah berdasarkan dari hasil perancangan, implementasi, dan pengujian yang telah dilakukan, modul CRM dapat dikembangkan lebih lanjut dari API yang sudah dibangun.

DAFTAR PUSTAKA

[1] M. A. Saputra dan B. Setiawan, "Customer Relationship Management Untuk Pengelolaan Donor Darah," *J. Tek. ITS*, vol. 3, no. 2, hlm. A198-A203–A203, Sep 2014.

[2] A. V. Allorerung, "RANCANG BANGUN PERANGKAT LUNAK BERORIENTASI ARSITEKTUR SERVICE (SOA) DENGAN PENDEKATAN WORKFLOW PADA DOMAIN CUSTOMER RELATIONSHIP MANAGEMENT (CRM) UNTUK SISTEM ENTERPRISE RESOURCE PLANNING," *Undergrad. Theses Inform. Eng. RSIf 00512 R 2015*, Jan 2014.

[3] T. Yu, J. Zhou, Y. Zhang, S. Dong, dan W. Wang, "Research on CRM Performance Evaluation Based on Fuzzy Comprehensive Algorithm," dalam *2008 International Conference on Information Management, Innovation Management and Industrial Engineering*, 2008, vol. 1, hlm. 329–334.

[4] "Customers | Couchbase." [Daring]. Tersedia pada: https://www.couchbase.com/customers/. [Diakses: 11-Jul-2019].

[5] G. M. Rama dan A. Kak, "Some structural measures of API usability," *Softw. Pract. Exp.*, vol. 45, no. 1, hlm. 75–110, 2015.

[6] Aminudin, *Cara Efektif Belajar Framework LARAVEL*. CV. LOKOMEDIA, 2015.

[7] J. A. O'Brien dan G. M. Marakas, *Management Information Systems*. McGraw-Hill/Irwin, 2010.

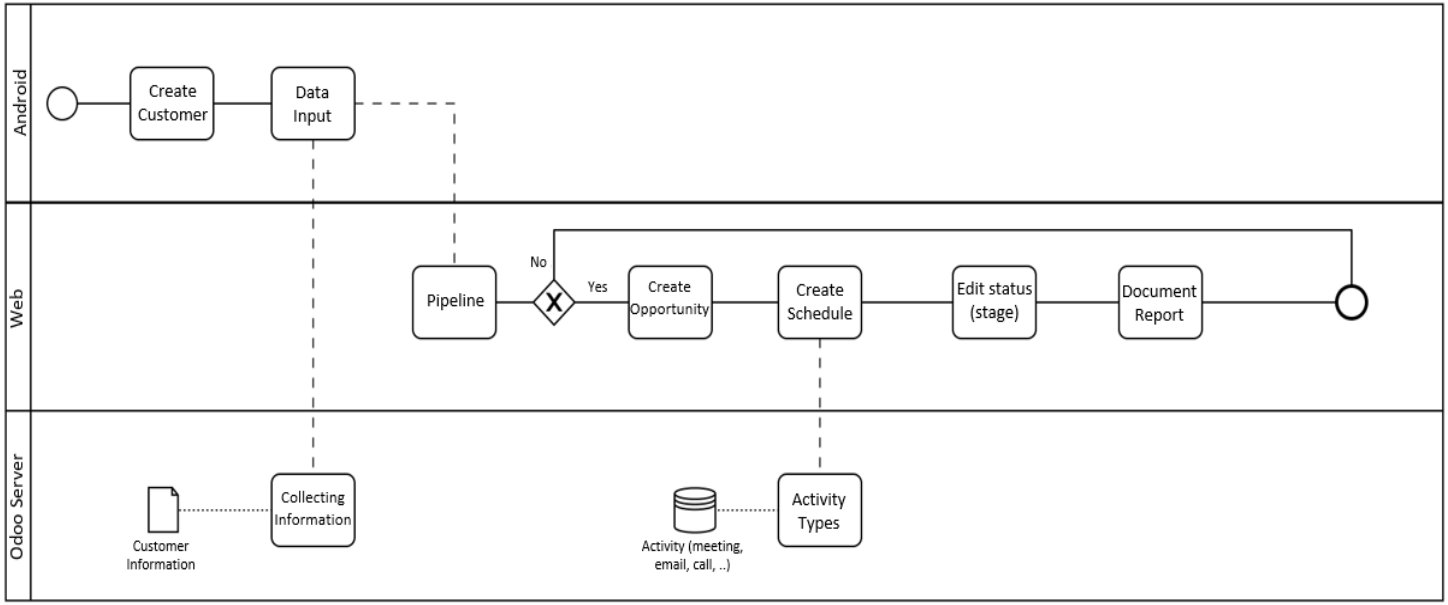
[8] D. Reis, *Odoo Development Essentials*. Packt Publishing Ltd, 2015.

[9] "All Apps - Open Source Business Apps: Odoo," *Odoo S.A.* [Daring]. Tersedia pada: https://www.odoo.com/page/all-apps. [Diakses: 31-Mei-2019].

[10] "Manage Your Customer Pipeline with Odoo CRM Module," *Bista Solutions: Best Odoo Implementation Company | Odoo Gold Partners*, 07-Jan-2019.

[11]"JSON." [Daring]. Tersedia pada: https://www.json.org/. [Diakses: 31-Mei-2019].
[12]"Couchbase Under the Hood An Architectural Overview." Couchbase, 2019.
[13]"Couchbase Mobile – Products | Couchbase." [Daring]. Tersedia pada: https://www.couchbase.com/products/mobile. [Diakses: 28-Apr-2019].

LAMPIRAN A



Gambar A.1 Proses Bisnis Modul CRM pada Odoo

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Rahajeng Dwi Permatasari, anak kedua dari dua bersaudara yang lahir di Probolinggo pada tanggal 4 Januari 1997. Penulis telah menempuh pendidikan formal mulai dari SD Tisnonegaran II Probolinggo (2003-2009), SMP Negeri 1 Kota Probolinggo (2009-2012), SMA Negeri 1 Kota Probolinggo (2012-2015) dan terakhir sebagai mahasiswa Departemen Informatika Institut Teknologi Sepuluh Nopember dengan rumpun mata kuliah Rekayasa Perangkat Lunak (2015-2019).

Selama perkuliahan, penulis aktif dalam organisasi kemahasiswaan, antara lain sebagai Staff EO FTIf Festival 2016, Staff Departemen Kaderisasi dan Pemetaan Himpunan Mahasiswa Teknik Computer-Informatika ITS 2016-2017, Fasilitator Kestari Gerigi ITS 2016, Staff Humas Schematics 2016, Staff REEVA Schematics 2017, Mentor Gerigi ITS 2017, dan Sekretaris Departemen Kaderisasi dan Pemetaan Himpunan Mahasiswa Teknik Computer-Informatika ITS 2017-2018.

Selama kuliah di Departemen Informatika ITS, penulis mengambil bidang minat Rekayasa Perangkat Lunak (RPL) dengan ketertarikan penulis terdapat pada rancang bangun perangkat lunak dan *website*. Penulis dapat dihubungi melalui surel ajeng0482@gmail.com.